

Formula Reference

Spread.NET products provide extensive calculation ability through formulas. With over 450 built-in functions, standard formula operators, and the ability to create custom functions, you can define and perform calculations for a range of data within any of several sheets in a spreadsheet component.

This reference provides an introduction to the use of formulas as well as a complete list of built-in functions. This documentation includes:

- **Formula Overview**
- **Formula Functions**

For more information on using formulas in cells and creating custom functions, refer to the product Developers Guide.

For a complete list of documentation, return to the product documentation page.

1 Table of Contents

	Formula Reference	1
1.	Table of Contents	2-19
	Contacting Us	20
	Getting Technical Support	21
	Formula Overview	22
	What is a Formula?	23
	Sample Formula	24
	Cell References in a Formula	25
	A1 (Letter-Number) Notation	26
	R1C1 (Number-Number) Notation	27
	Relative and Absolute	28
	Scope of Cell References	29
	Sheet References in a Formula	30-31
	Operators in a Formula	32
	Order of Precedence	33
	Using Operators with Dates and Times	34
	Functions in a Formula	35
	Categories of Functions	36
	Database Functions	37
	Date and Time Functions	38
	Engineering Functions	39
	Complex Numbers in Engineering Functions	40
	Financial Functions	41
	Day Count Basis	42
	Information Functions	43
	Logical Functions	44
	Lookup Functions	45
	Math and Trigonometry Functions	46

<u>Statistical Functions</u>	47-48
<u>Text Functions</u>	49
<u>Web Functions</u>	50
<u>Optional Arguments</u>	51
<u>Missing Arguments</u>	52
<u>Volatile Functions</u>	53
<u>Array Formulas</u>	54
<u>Arrays in a Formula</u>	55
<u>Dynamic Array Formulas</u>	56
<u>Data Types in a Formula</u>	57
<u>Custom Functions in Formulas</u>	58
<u>Custom Names in Formulas</u>	59
<u>Resultant Error Values</u>	60
<u>Formula Functions</u>	61-64
<u>Functions A to C</u>	65
<u>ABS</u>	66
<u>ACCRINT</u>	67
<u>ACCRINTM</u>	68
<u>ACOS</u>	69
<u>ACOSH</u>	70
<u>ACOT</u>	71
<u>ACOTH</u>	72
<u>ADDRESS</u>	73
<u>AGGREGATE</u>	74-75
<u>AMORDEGRC</u>	76-77
<u>AMORLINC</u>	78
<u>AND</u>	79
<u>ARABIC</u>	80
<u>AREAS</u>	81
<u>ASC</u>	82

<u>ASIN</u>	83
<u>ASINH</u>	84
<u>ATAN</u>	85
<u>ATAN2</u>	86
<u>ATANH</u>	87
<u>AVEDEV</u>	88
<u>AVERAGE</u>	89
<u>AVERAGEA</u>	90
<u>AVERAGEIF</u>	91
<u>AVERAGEIFS</u>	92
<u>BAHTTEXT</u>	93
<u>BASE</u>	94
<u>BESSELI</u>	95
<u>BESSELJ</u>	96
<u>BESSELK</u>	97
<u>BESSELY</u>	98
<u>BETA.DIST</u>	99
<u>BETA.INV</u>	100
<u>BETADIST</u>	101
<u>BETAINV</u>	102
<u>BIN2DEC</u>	103
<u>BIN2HEX</u>	104
<u>BIN2OCT</u>	105
<u>BINOM.DIST</u>	106-107
<u>BINOM.DIST.RANGE</u>	108
<u>BINOM.INV</u>	109
<u>BINOMDIST</u>	110-111
<u>BITAND</u>	112
<u>BITLSHIFT</u>	113
<u>BITOR</u>	114

<u>BITRSHIFT</u>	115
<u>BITXOR</u>	116
<u>CALL</u>	117
<u>CEILING</u>	118
<u>CEILING.MATH</u>	119
<u>CEILING.PRECISE</u>	120
<u>CELL</u>	121-122
<u>CHAR</u>	123
<u>CHIDIST</u>	124
<u>CHIINV</u>	125
<u>CHISQ.DIST</u>	126
<u>CHISQ.DIST.RT</u>	127
<u>CHISQ.INV</u>	128
<u>CHISQ.INV.RT</u>	129
<u>CHISQ.TEST</u>	130
<u>CHITEST</u>	131
<u>CHOOSE</u>	132
<u>CLEAN</u>	133
<u>CODE</u>	134
<u>COLUMN</u>	135
<u>COLUMNS</u>	136
<u>COMBIN</u>	137
<u>COMBINA</u>	138
<u>COMPLEX</u>	139
<u>CONCAT</u>	140
<u>CONCATENATE</u>	141
<u>CONFIDENCE</u>	142
<u>CONFIDENCE.NORM</u>	143
<u>CONFIDENCE.T</u>	144

<u>CONVERT</u>	145-147
<u>CORREL</u>	148
<u>COS</u>	149
<u>COSH</u>	150
<u>COT</u>	151
<u>COTH</u>	152
<u>COUNT</u>	153
<u>COUNTA</u>	154
<u>COUNTBLANK</u>	155
<u>COUNTIF</u>	156
<u>COUNTIFS</u>	157
<u>COUPDAYBS</u>	158
<u>COUPDAYS</u>	159
<u>COUPDAYSNC</u>	160
<u>COUPNCD</u>	161
<u>COUPNUM</u>	162
<u>COUPPCD</u>	163
<u>COVAR</u>	164
<u>COVARIANCE.P</u>	165
<u>COVARIANCE.S</u>	166
<u>CRITBINOM</u>	167
<u>CSC</u>	168
<u>CSCH</u>	169
<u>Functions D to G</u>	170
<u>DATE</u>	171
<u>DATEDIF</u>	172
<u>DATEVALUE</u>	173
<u>DAVERAGE</u>	174
<u>DAY</u>	175
<u>DAYS</u>	176

<u>DAYS360</u>	177-178
<u>DB</u>	179-180
<u>DBCS</u>	181
<u>DCOUNT</u>	182
<u>DCOUNTA</u>	183
<u>DDB</u>	184
<u>DEC2BIN</u>	185
<u>DEC2HEX</u>	186
<u>DEC2OCT</u>	187
<u>DECIMAL</u>	188
<u>DEGREES</u>	189
<u>DELTA</u>	190
<u>DEVSQ</u>	191
<u>DGET</u>	192
<u>DISC</u>	193
<u>DMAX</u>	194
<u>DMIN</u>	195
<u>DOLLAR</u>	196
<u>DOLLARDE</u>	197
<u>DOLLARFR</u>	198
<u>DPRODUCT</u>	199
<u>DSTDEV</u>	200
<u>DSTDEVP</u>	201
<u>DSUM</u>	202
<u>DURATION</u>	203
<u>DVAR</u>	204
<u>DVARP</u>	205
<u>EDATE</u>	206
<u>EFFECT</u>	207
<u>ENCODEURL</u>	208

<u>EOMONTH</u>	209
<u>ERF</u>	210-211
<u>ERF.PRECISE</u>	212
<u>ERFC</u>	213
<u>ERFC.PRECISE</u>	214
<u>ERROR.TYPE</u>	215
<u>ERRORTYPE</u>	216
<u>EUROCONVERT</u>	217-218
<u>EVEN</u>	219
<u>EXACT</u>	220
<u>EXP</u>	221
<u>EXPON.DIST</u>	222-223
<u>EXPONDIST</u>	224-225
<u>F.DIST</u>	226
<u>F.DIST.RT</u>	227
<u>F.INV</u>	228
<u>F.INV.RT</u>	229
<u>F.TEST</u>	230
<u>FACT</u>	231
<u>FACTDOUBLE</u>	232
<u>FALSE</u>	233
<u>FDIST</u>	234
<u>FILTER</u>	235-236
<u>FILTERXML</u>	237
<u>FIND</u>	238
<u>FINDB</u>	239
<u>FINV</u>	240
<u>FISHER</u>	241
<u>FISHERINV</u>	242

<u>FIXED</u>	243
<u>FLOOR</u>	244
<u>FLOOR.MATH</u>	245
<u>FLOOR.PRECISE</u>	246
<u>FORECAST</u>	247
<u>FORECAST.LINEAR</u>	248
<u>FORMULATEXT</u>	249
<u>FREQUENCY</u>	250
<u>FTEST</u>	251
<u>FV</u>	252
<u>FVSCCHEDULE</u>	253
<u>GAMMA</u>	254
<u>GAMMA.DIST</u>	255
<u>GAMMA.INV</u>	256
<u>GAMMADIST</u>	257
<u>GAMMAINV</u>	258
<u>GAMMALN</u>	259
<u>GAMMALN.PRECISE</u>	260
<u>GAUSS</u>	261
<u>GCD</u>	262
<u>GEOMEAN</u>	263
<u>GESTEP</u>	264
<u>GROWTH</u>	265
<u>Functions H to L</u>	266
<u>HARMEAN</u>	267
<u>HEX2BIN</u>	268
<u>HEX2DEC</u>	269
<u>HEX2OCT</u>	270
<u>HLOOKUP</u>	271
<u>HOUR</u>	272

HYPERLINK	273
HYPGEOM.DIST	274
HYPGEOMDIST	275
IF	276
IFERROR	277
IFNA	278
IFS	279
IMABS	280
IMAGINARY	281
IMARGUMENT	282
IMCONJUGATE	283
IMCOS	284
IMCOSH	285
IMCOT	286
IMCSC	287
IMCSCH	288
IMDIV	289
IMEXP	290
IMLN	291
IMLOG10	292
IMLOG2	293
IMPOWER	294
IMPRODUCT	295
IMREAL	296
IMSEC	297
IMSECH	298
IMSIN	299
IMSINH	300
IMSQRT	301
IMSUB	302

<u>IMSUM</u>	303
<u>IMTAN</u>	304
<u>INDEX</u>	305
<u>INDIRECT</u>	306
<u>INFO</u>	307
<u>INT</u>	308
<u>INTERCEPT</u>	309
<u>INTRATE</u>	310
<u>IPMT</u>	311
<u>IRR</u>	312-313
<u>ISBLANK</u>	314
<u>ISERR</u>	315
<u>ISERROR</u>	316
<u>ISEVEN</u>	317
<u>ISFORMULA</u>	318
<u>ISLOGICAL</u>	319
<u>ISNA</u>	320
<u>ISNONTEXT</u>	321
<u>ISNUMBER</u>	322
<u>ISO.CEILING</u>	323
<u>ISODD</u>	324
<u>ISOWEEKNUM</u>	325
<u>ISPMT</u>	326
<u>ISREF</u>	327
<u>ISTEXT</u>	328
<u>JIS</u>	329
<u>KURT</u>	330
<u>LARGE</u>	331
<u>LCM</u>	332

<u>LEFT</u>	333
<u>LEFTB</u>	334
<u>LEN</u>	335
<u>LENB</u>	336
<u>LINEST</u>	337
<u>LN</u>	338
<u>LOG</u>	339
<u>LOG10</u>	340
<u>LOGEST</u>	341
<u>LOGINV</u>	342
<u>LOGNORM.DIST</u>	343
<u>LOGNORM.INV</u>	344
<u>LOGNORMDIST</u>	345
<u>LOOKUP</u>	346-347
<u>LOWER</u>	348
<u>Functions M to Q</u>	349
<u>MATCH</u>	350
<u>MAX</u>	351
<u>MAXA</u>	352
<u>MAXIFS</u>	353
<u>MDETERM</u>	354
<u>MDURATION</u>	355
<u>MEDIAN</u>	356
<u>MID</u>	357
<u>MIDB</u>	358
<u>MIN</u>	359
<u>MINA</u>	360
<u>MINIFS</u>	361
<u>MINUTE</u>	362
<u>MINVERSE</u>	363

MIRR	364
MMULT	365
MOD	366
MODE	367
MODE.MULT	368
MODE.SNGL	369
MONTH	370
MROUND	371
MULTINOMIAL	372
MUNIT	373
N	374
NA	375
NEGBINOM.DIST	376
NEGBINOMDIST	377
NETWORKDAYS	378
NETWORKDAYS.INTL	379-380
NOMINAL	381
NORM.DIST	382
NORM.INV	383
NORM.S.DIST	384
NORM.S.INV	385
NORMDIST	386
NORMINV	387
NORMSDIST	388
NORMSINV	389
NOT	390
NOW	391
NPER	392
NPV	393-394
NUMBERVALUE	395

<u>OCT2BIN</u>	396
<u>OCT2DEC</u>	397
<u>OCT2HEX</u>	398
<u>ODD</u>	399
<u>ODDFPRICE</u>	400
<u>ODDFYIELD</u>	401
<u>ODDLPRICE</u>	402
<u>ODDLYIELD</u>	403
<u>OFFSET</u>	404
<u>OR</u>	405
<u>PDURATION</u>	406
<u>PEARSON</u>	407
<u>PERCENTILE</u>	408
<u>PERCENTILE.EXC</u>	409
<u>PERCENTILE.INC</u>	410
<u>PERCENTRANK</u>	411
<u>PERCENTRANK.EXC</u>	412
<u>PERCENTRANK.INC</u>	413
<u>PERMUT</u>	414
<u>PERMUTATIONA</u>	415
<u>PHI</u>	416
<u>PHONETIC</u>	417
<u>PI</u>	418
<u>PMT</u>	419
<u>POISSON</u>	420-421
<u>POISSON.DIST</u>	422-423
<u>POWER</u>	424
<u>PPMT</u>	425
<u>PRICE</u>	426

<u>PRICEDISC</u>	427
<u>PRICEMAT</u>	428
<u>PROB</u>	429
<u>PRODUCT</u>	430
<u>PROPER</u>	431
<u>PV</u>	432
<u>QUARTILE</u>	433
<u>QUARTILE.EXC</u>	434
<u>QUARTILE.INC</u>	435
<u>QUOTIENT</u>	436
<u>Functions R to S</u>	437
<u>RADIANS</u>	438
<u>RAND</u>	439
<u>RANDARRAY</u>	440-441
<u>RANDBETWEEN</u>	442
<u>RANK</u>	443
<u>RANK.AVG</u>	444
<u>RANK.EQ</u>	445
<u>RATE</u>	446
<u>RECEIVED</u>	447
<u>REPLACE</u>	448
<u>REPLACEB</u>	449
<u>REPT</u>	450
<u>RIGHT</u>	451
<u>RIGHTB</u>	452
<u>ROMAN</u>	453
<u>ROUND</u>	454
<u>ROUNDDOWN</u>	455
<u>ROUNDUP</u>	456
<u>ROW</u>	457

<u>ROWS</u>	458
<u>RRI</u>	459
<u>RSQ</u>	460
<u>RTD</u>	461
<u>SEARCH</u>	462
<u>SEARCHB</u>	463
<u>SEC</u>	464
<u>SECH</u>	465
<u>SECOND</u>	466
<u>SERIESSUM</u>	467
<u>SEQUENCE</u>	468
<u>SHEET</u>	469
<u>SHEETS</u>	470
<u>SIGN</u>	471
<u>SIN</u>	472
<u>SINH</u>	473
<u>SINGLE</u>	474
<u>SKEW</u>	475
<u>SKEW.P</u>	476
<u>SLN</u>	477
<u>SLOPE</u>	478
<u>SMALL</u>	479
<u>SORT</u>	480-482
<u>SORTBY</u>	483-484
<u>SQRT</u>	485
<u>SQRTPI</u>	486
<u>STANDARDIZE</u>	487
<u>STDEV</u>	488
<u>STDEV.P</u>	489
<u>STDEV.S</u>	490

<u>STDEVA</u>	491
<u>STDEVP</u>	492
<u>STDEVPA</u>	493
<u>STEYX</u>	494
<u>SUBSTITUTE</u>	495
<u>SUBTOTAL</u>	496-497
<u>SUM</u>	498-499
<u>SUMIF</u>	500
<u>SUMIFS</u>	501
<u>SUMPRODUCT</u>	502
<u>SUMSQ</u>	503
<u>SUMX2MY2</u>	504
<u>SUMX2PY2</u>	505
<u>SUMXMY2</u>	506
<u>SWITCH</u>	507
<u>SYD</u>	508
<u>Functions T to Z</u>	509
<u>T</u>	510
<u>T.DIST</u>	511
<u>T.DIST.2T</u>	512
<u>T.DIST.RT</u>	513
<u>T.INV</u>	514
<u>T.INV.2T</u>	515
<u>T.TEST</u>	516
<u>TAN</u>	517
<u>TANH</u>	518
<u>TBILLEQ</u>	519
<u>TBILLPRICE</u>	520
<u>TBILLYIELD</u>	521

TDIST	522
TEXT	523
TEXTJOIN	524
TIME	525
TIMEVALUE	526
TINV	527
TODAY	528
TRANSPOSE	529
TREND	530
TRIM	531
TRIMMEAN	532
TRUE	533
TRUNC	534
TTEST	535
TYPE	536
UNICHAR	537
UNICODE	538
UNIQUE	539-540
UPPER	541
USDOLLAR	542
VALUE	543
VAR	544-545
VAR.P	546
VAR.S	547
VARA	548-549
VARP	550
VARPA	551-552
VDB	553
VLOOKUP	554
WEBSERVICE	555

<u>WEEKDAY</u>	556
<u>WEEKNUM</u>	557
<u>WEIBULL</u>	558
<u>WEIBULL.DIST</u>	559
<u>WORKDAY</u>	560
<u>WORKDAY.INTL</u>	561-562
<u>XIRR</u>	563
<u>XNPV</u>	564
<u>XOR</u>	565
<u>YEAR</u>	566
<u>YEARFRAC</u>	567
<u>YIELD</u>	568
<u>YIELDDISC</u>	569
<u>YIELDMAT</u>	570
<u>Z.TEST</u>	571
<u>ZTEST</u>	572
2. <u>Index</u>	573-585

Contacting Us

If you would like to find out more about our products, contact our Sales department using one of the following methods:

Web site:	https://www.grapecity.com/
E-mail:	us.sales@grapecity.com
Phone:	(800) 858-2739 or (412) 681-4343 outside the U.S.A.
Fax:	(412) 681-4384

Getting Technical Support

If you have a technical question about this product, consult the following sources:

- Help and other documentation files installed with the product.
- Product forum at <https://www.grapecity.com/forums/#spread>
- Videos and other information available on the Web site.

If you cannot find the answer using these sources, please contact Technical Support using one of these methods:

Web site: <https://www.grapecity.com/forums>

E-mail: spread.support@grapecity.com

Phone: (412) 681-4738

Fax: (412) 681-4384

Technical Support is available between the hours of 9:00 a.m. and 5:00 p.m. Eastern time, Monday through Friday.

Formula Overview

Formulas in Spread .NET include operators and functions that follow certain syntax rules and allow you to perform a range of calculations. These topics introduce the concepts you need to make full use of the built-in functions and extensive capability of formulas:

- **What is a Formula?**
- **Cell References in a Formula**
- **Sheet References in a Formula**
- **Operators in a Formula**
- **Functions in a Formula**
- **Array Formulas**
- **Arrays in a Formula**
- **Dynamic Array Formulas**
- **Data Types Using Formulas**
- **Custom Functions in Formulas**
- **Custom Names in Formulas**
- **Resultant Error Values**

For a complete reference of all the built-in functions, refer to **Formula Functions**.

Return to the **Formula Reference**.

What is a Formula?

Formulas can consist of values, operators, and functions. Data can be from other cells, a combination of data in another cell and hard-coded data (for example, $A1 + 2$), or simply hard-coded data (for example, $SUM(4,5)$). Formulas can perform mathematical operations, such as addition and multiplication, on values in other cells or they can compare values in other cells. Formulas can refer to cells in the same sheet by their absolute cell location or relative to the cell with the formula in it; they can refer to individual cells or a range of contiguous cells. If the values in the referenced cells change, then the value of the formula cell changes.

Formulas can be made up of:

- cell references and cell ranges (notation indicating address of cell or cells)
- operators (that act on one or two values)
- built-in functions (predefined formulas) or user-defined functions
- user-defined names (for functions, constants, or cell references)
- constants or array of constants (values you enter that do not change)

See the **Sample Formula**.

Return to the **Formula Overview**.

Sample Formula

Use the SetFormula method in the Column, Row, or Cell class for specifying the formula for a column, row, or individual cell respectively. Use the SetArrayFormula method for an array formula. Returning the value of the Formula property for these classes provides a string containing the written expression of the formula, for example, SUM(A1:B1).

In code the setting of a formula would look something like this in Visual Basic .NET (for illustration purposes only):

```
FpSpread1.ActiveSheet.Cells(2, 0).Formula = "SUM(A1:A10)"
```

or something like this in C#:

```
fpSpread1.ActiveSheet.Cells[2, 0].Formula = "SUM(A1:A10)";
```

and if added in the cell by the end user:

```
=SUM(A1:A10)
```

In this documentation, where examples are shown, the formula appears as:

```
SUM(A1:A10)
```

or

```
SUM(3,4,5) gives the result 12
```

to express that the result of the formula would display the value of 12 in the cell.

Keep these ways of expressing a formula in mind when looking at the examples in this documentation. Refer to the specific product Assembly Reference for more details on the Formula property for that product and the exact code syntax to use. Refer to the Developers Guide for that product to find more examples and discussion of formulas.

Return to the **What is a Formula?**

Return to the **Formula Overview**.

Cell References in a Formula

A formula can refer to constant values or cell references. If a value in any of the referenced cells changes, the result of the formula changes. If you use constant values in the formula instead of references to the cells, the result changes only if you modify the formula (or values in the formula).

If a new row is added right before or after a cell range in a formula then the range does not include the new row.

This topic includes:

- **A1 (Letter-Number) Notation**
- **R1C1 (Number-Number) Notation**
- **Relative and Absolute**

For more information on setting the reference style for a cell, refer to the ReferenceStyle enumeration in the product's Assembly Reference (or help) and the ReferenceStyle property for the specific sheet (SheetView object).

Note: Remember that although most of Spread uses zero-based references to rows and columns, in the creation of formulas you must use one-based references. The column and row numbers start at one (1), not zero (0).

For more information on cell references that include sheet names, refer to **Sheet References in a Formula**.

Return to the **Formula Overview**.

A1 (Letter-Number) Notation

Each cell can be referenced by a combination of its column letter (A through Z, then AA to ZZ, AAA to ZZZ, etc.) and row number (1 and beyond) for a total of 2,147,483,648 rows and columns. For example, D50 refers to the cell at the intersection of column D and row 50. To refer to a range of cells, enter the reference for the cell in the upper-left corner of the range, a colon (:), and then the reference to the cell in the lower-right corner of the range.

See also these topics:

- **R1C1 (Number-Number) Notation**
- **Relative and Absolute**

For more information on setting the reference style for a cell, refer to the ReferenceStyle enumeration in the product's Assembly Reference (or help) and the ReferenceStyle property for the specific sheet (SheetView object).

Return to **Cell References in a Formula**

R1C1 (Number-Number) Notation

Each cell can be referenced by its row and column number by preceding each by the letter "R" for row and the letter "C" for column. For example R1C3 is the cell in the first row and third column.

A1 Cell Ref.	R1C1 Cell Ref.	Description
B12	R12C2	Cell in the second column (column B) and twelfth row (row 12)
D14:D48	R14C4:R48C4	The range of cells in the fourth column (column D) and in rows 14 through 48
E16:H16	R16C5:R16C8	The range of cells in the sixteenth row (row 16) in the fifth through the eighth column (columns E through H)
A25:E70	R25C1:R70C5	The range of cells in the first five columns (column A through E) and rows 25 through 70

See also these topics:

- **A1 (Letter-Number) Notation**
- **Relative and Absolute**

For more information on setting the reference style for a cell, refer to the ReferenceStyle enumeration in the product's Assembly Reference (or help) and the ReferenceStyle property for the specific sheet (SheetView object).

Return to **Cell References in a Formula**

Relative and Absolute

A relative cell reference is a reference to a cell relative to the position of the cell with the formula. An absolute reference is a cell reference that always refers to a cell by its exact location in the sheet and not with reference to the present cell.

Relative references automatically adjust when you copy them and absolute references do not. The FpSpread control can use absolute or relative cell references. You can define the cell reference style for each sheet by using the ReferenceStyle property. The formula also supports range references that contain both absolute and relative row or column references. In other words, the start and end rows in a range reference can be same (both absolute or both relative) or different (one absolute and one relative or vice a versa). The following table contains examples of valid relative cell references in formulas:

Function Description

SUM(A1:A10)	Sums rows 1 through 10 in the first column
PI()*C6	Multiplies pi times the value in cell C6
(A1 + B1) * C1	Adds the values in the first two cells and multiplies the result by the value in the third cell
IF(A1>5, A1*2, A1*3)	Checks if the contents of cell A1 are greater than 5, and if so, multiplies the contents of cell A1 by 2, or else multiplies the contents of cell A1 by 3

For **A1 (Letter-Number) Notation**, use a dollar sign (\$) preceding the row or column (or both) to indicate an absolute reference. For example

\$A\$1	absolute first column, absolute first row
\$A1	absolute first column, relative row plus one
A\$1	relative column plus one, absolute first row
A1	relative column plus one, relative row plus one

For **R1C1 (Number-Number) Notation**, use brackets [] around the row or column number (or both) to indicate a relative reference. For example

R1C1	absolute first row, absolute first column
R1C[1]	absolute first row, relative column plus one
R[1]C1	relative row plus one, absolute first column
R[1]C[1]	relative row plus one, relative column plus one
R[-1]C[-1]	relative row minus one, relative column minus one

In this notation, the number inside the brackets is an offset from the current cell. This number may be a negative or positive integer or zero. Leaving off the offset entirely is short hand way of indicating a zero offset. So,

RC2 is equivalent to R[0]C2

R[3]C is equivalent to R[3]C[0]

See also these topics:

- **A1 (Letter-Number) Notation**
- **R1C1 (Number-Number) Notation**

Return to **Cell References in a Formula**

Scope of Cell References

References to cells within a sheet are handled as described in this documentation. When a cell is referenced that is beyond the dimensions of the sheet, the cell is still evaluated, but the result is a #REF! error value. For example, if the sheet has less than 20 columns and rows, then the function DDB(B20,1000,10,1) evaluates to DDB(#REF!,1000,10,1), which evaluates to #REF!

Spread.NET does not support Excel's reference operators (for example range, intersection, union) in formulas. However, Spread .NET does support the #NULL! constant in formulas. It does support reading the #NULL! value from Excel files. For more information about what is supported on importing from and exporting to Excel, refer to the Import and Export Reference for the particular Spread product you are using.

Return to **Cell References in a Formula**

Sheet References in a Formula

A formula can have references to cells on the same sheet or to cells on other sheets, as well as ranges of cells on sheets.

In the examples shown below, we use A1 (Letter-Number) notation for the cell reference, but the same would be valid for R1C1 (Number-Number) notation. Simply precede the cell reference, regardless of the style, with the sheet name as described here.

For more information on cell references that do not include sheet names, refer to **Cell References in a Formula**.

Cross-Sheet Referencing

When a reference to a cell includes a reference to a cell on another sheet, this is called cross-sheet referencing.

An example of cross-sheet referencing in a formula that uses the addition operator would be:

```
(FirstRoundData!A2 + SecondRoundData!A2)
```

where the name of one sheet is "FirstRoundData" and the name of another sheet is "SecondRoundData". Sheet names precede the cell reference with the name of the sheet followed by an exclamation point (!). This formula could be on any sheet in the Spread since it explicitly names the sheets of each of the cells as operands. This example adds the values in the cell A2 on two different sheets. By making the sheet name explicit there is no confusion as to which cell A2 is meant. If you do not include the sheet name, the current sheet (in which the formula exists) is used. If the formula in the above example was on the SecondRoundData page, then the formula could be written as:

```
(FirstRoundData!A2 + A2)
```

It might be less confusing to put the cell on the current page first, as in:

```
(A2 + FirstRoundData!A2)
```

3-D Referencing

When a reference to a cell includes the same cell or a cell range on multiple sheets, this is called three-dimensional referencing.

An example of 3-D referencing in a SUM formula that uses the cell range on multiple sheets is described below.

```
SUM(Sheet1:Sheet10!A1:A2)
```

where the name of one sheet is "Sheet1" and the name of another sheet is "Sheet10". This formula uses SUM function to create a 3-D reference that executes the sum operation across multiple sheets via adding up the values in cell range A1:A2 in all the sheets that lie between Sheet1 to Sheet10.

3-D Referencing can be extensively used to quickly calculate data across multiple spreadsheets that possess identical pattern and identical data type.

Sheet Naming

As long as the sheet name conforms to normal variable name rules (with the first character being a letter or an underscore and the remaining characters being letters, digits, or underscores) then the formula can use just the sheet name followed by the exclamation point. Otherwise, the sheet name needs to be enclosed in single quotes. If the sheet name itself contains a single quote, then use two single quotes in the formula. For example, if the name of the sheet includes a single quote (or apostrophe) as in these names for sales of a given month, then a reference to the sheet would look like this in a formula:

```
('November's Sales'!A2 + 'December's Sales'!A2)
```

with two single quotes (or apostrophes) before the s. If the sheet name has a space, use single quotes around the sheet name. In the following example the sheet name is East Coast Sales.

```
('East Coast Sales'!A2 + 'West Coast Sales'!A1)
```

If you have a quote in the name of the sheet, you need to add the delimiter that is required for that language. For instance, in C#, if the sheet name is "Zippy" Sales, where the quotes are part of the sheet name, a formula that includes a reference to this sheet might look like this:

```
(/"Zippy/" Sales!A2 + 'West Coast Sales'!A1)
```

where a single quotes surrounds the entire sheet name and the backslash (/) delimiter precedes the quotes. For Visual Basic, you would use two double quote characters as in:

```
(""Zippy"" Sales!A2 + 'West Coast Sales'!A1)
```

Using Ranges in Sheet References

For cross-sheet referencing of a range of cells in another page, precede the range with the sheet name. For example:

```
SUM(SecondRoundData!A2:A10)
```

This adds the values in cells A2 to A10 of the sheet named SecondRoundData. There is no reason to include the sheet name in the second half of the range reference since the cells are on the same sheet. You cannot specify two different sheets in a range; a range of cells is only on a particular sheet, not between sheets.

Return to the **Formula Overview**.

Operators in a Formula

The following table lists the available operators. For each operator, an example is given of the syntax of using a literal value as well as a cell reference. The type of value returned is given for each type of operator.

Type of Operator	Example Syntax	Result		
Operator	Description	Literal & Literal	Cell Ref & Literal	Type Returned
Binary Operators				
+	Add	5 + 3	A1 + 3	double
-	Subtract	5 - 3	A1 - 3	double
*	Multiply	5 * 3	A1 * 3	double
/	Divide	5 / 3	A1 / 3	double
^	Exponent	5 ^ 3	A1 ^ 3	double
&	Concatenate	"F" & "p"	A1 & "p"	string
=	Equal		A1 <> 3	boolean
<>	Not Equal		A1 = 3	boolean
<	Less Than		A1 < 3	boolean
>	Greater Than		A1 > 3	boolean
<=	Less Than Or Equal		A1 <= 3	boolean
>=	Greater Than Or Equal		A1 >= 3	boolean
Unary Operators				
-	Negate	-(5/3)	-(A1/3)	double
+	Plus	+(5/3)	+(A1/3)	double
%	Percent	(5/3)%	(A1/3)%	double

Operators specify the type of calculation that you want to perform on the elements of a formula. Most of the operators return double-precision floating point values for mathematical operations and boolean (or logical) values for comparison operators.

In Spread, all arithmetic operators (including the unary +) check their arguments and return a #VALUE error if any of the arguments are strings that can not be converted to a number. This is mathematically correct behavior and can not be overridden. For example, the three formulas +B5 and 0+B5 and --B5 should all produce the same result and, in Spread, they do.

Because more than one operator may be used in a formula, so be sure you understand the **Order of Precedence**.

The mathematical operators and unary operators may also be used with date-time and time-span values, as summarized in **Using Operators with Dates and Times**.

Return to the **Formula Overview**.

Order of Precedence

When there are several operators in a formula, the formula performs the operations in a specific order. The formula is parsed from left to right, according to a specific order for each operator or function in the formula. You can prioritize the order of operations by using parentheses in the formula.

If you combine several operators in a single formula, the operations are performed in the order shown in the following table. Unary operations precede binary operations. If a formula contains operators with the same precedence, the operators are evaluated from left to right. To change the order of evaluation, enclose the part of the formula to be calculated first in parentheses; this has the highest precedence. Where the order of precedence is the same for two operators, the formula is evaluated from left to right.

Order of Precedence from Highest to Lowest

Operator	Description
left to right	Direction
()	Parentheses (for grouping)
-	Negate
+	Plus
%	Percent
^	Exponent
* and /	Multiply and Divide
+ and -	Add and Subtract
&	Concatenate
=, <, >, <=, >=, <>	Compare

Return to **Operators in a Formula**.

Return to the **Formula Overview**.

Using Operators with Dates and Times

You can use several of the operators with dates and times as summarized here:

Operator	Type of Operation	Result
Plus	+ TimeSpan	TimeSpan
Negate	- TimeSpan	TimeSpan
Add	DateTime + TimeSpan	DateTime
Add	TimeSpan + DateTime	DateTime
Add	TimeSpan + TimeSpan	TimeSpan
Subtract	DateTime - DateTime	TimeSpan
Subtract	DateTime - TimeSpan	DateTime
Subtract	TimeSpan - TimeSpan	DateTime

The same order of precedence applies, including use of parentheses, as described in **Order of Precedence**. For more information about functions that use and return DateTime and TimeSpan objects, refer to **Date and Time Functions**.

If a DateTime or TimeSpan calculation results in an exception (for example, an OverflowException), the operator returns the #NUM! error.

Return to **Operators in a Formula**.

Return to the **Formula Overview**.

Functions in a Formula

Functions are code segments that perform calculations by using specific values, called arguments, in a particular order that can be used in formulas. For example, the SUM function adds values or ranges of cells and the PMT function calculates the loan payments based on an interest rate, the length of the loan, and the principal amount of the loan. Functions may be either built-in functions that come with Spread or user-defined functions that you create.

Arguments can be numbers, text, logical values, arrays, cell ranges, cell references, or error values. The value you use for an argument must be valid for the given function. Arguments can also be constants, formulas, or other functions. Using a function as an argument for another function is known as nesting a function. Some arguments are optional; this reference displays "[Optional]" before the description of the argument for those arguments that are not required. These are described in **Optional Arguments**.

The structure of a function begins with the function name, followed by an opening parenthesis, the arguments for the function separated by commas, and a closing parenthesis. If you are entering the function into a cell directly, type an equal sign (=) before the function name. The following topics describe the formula functions available. Each includes an example. Examples that provide results give decimal values for 10 decimal places.

Other topics that are relevant include:

- **Categories of Functions**
- **Optional Arguments**
- **Missing Arguments**
- **Volatile Functions**

Return to the **Formula Overview**.

Categories of Functions

These functions are categorized into one of these function types:

- **Database Functions**
- **Date and Time Functions**
- **Engineering Functions**
- **Financial Functions**
- **Information Functions**
- **Logical Functions**
- **Lookup Functions**
- **Math and Trigonometry Functions**
- **Statistical Functions**
- **Text Functions**
- **Volatile Functions**
- **Web Functions**

For a complete list of functions, listed alphabetically by name, refer to **Formula Functions**.

Return to **Functions in a Formula**.

Return to the **Formula Overview**.

Database Functions

The functions that relate to database and list management are:

DAVERAGE	DCOUNT	DCOUNTA	DGET
DMAX	DMIN	DPRODUCT	DSTDEV
DSTDEVP	DSUM	DVAR	DVARP

These functions apply a mathematical or statistical operation to a subset of values in a range of cells treated as a database. The database table can be thought of as a two-dimensional array organized into rows and columns. Or it can be thought of as a one-dimensional array of records where each record is a structure that has one or more fields. In the context of database tables, the terms "row" and "record" mean the same thing and the terms "column" and "field" mean the same thing. Database refers to a range of cells where the first row in the range represents field labels. The remaining rows in the range represent records. The columns in the range represent fields.

Return to the list of **Categories of Functions**.

Date and Time Functions

The functions that relate to date-time values and time-span values are:

DATE	DATEDIF	DATEVALUE	DAY
DAYS	DAYS360	EDATE	EOMONTH
HOUR	ISO.CEILING	MINUTE	MONTH
NETWORKDAYS	NETWORKDAYS.INTL	NOW	SECOND
TIME	TIMEVALUE	TODAY	WEEKDAY
WEEKNUM	WORKDAY	WORKDAY.INTL	YEAR
YEARFRAC			

For most of these functions you can specify the date argument as a `DateTime` object, as in the result of a function such as `DATE(2003,7,4)`, or a `TimeSpan` object, as in the result of a function such as `TIME(12,0,0)`. For compatibility with Excel, it also allows dates to be specified as a number (as in `37806.5`) or as a string (as in `"7/4/2003 12:00"`). The numbers and strings are converted to instances of the `DateTime` class.

Dates as numeric values are in the form `x.y`, where `x` is the "number of days since December 30, 1899" and `y` is the fraction of day. Numbers to the left represent the date. Times as numeric values are decimal fractions ranging from 0 to 0.99999999, representing the times from 0:00:00 (12:00:00 A.M.) to 23:59:59 (11:59:59 P.M.).

The following three formulas produce the same result:

```
YEAR(DATE(2004,8,9))
```

```
YEAR(38208)
```

```
YEAR("8/9/2004")
```

In Excel, dates can range from 01/01/1900 to 12/31/9999, and in the .NET framework, instances of the `DateTime` class can range from 01/01/0001 to 12/31/9999. In Spread, we generally support the larger range found in the .NET framework. For Excel compatibility there are a few cases where the function allows only the smaller range (for example, the `DATE` function can only be used to enter dates since 01/01/1900).

You may see some differences in values if exporting to or importing from Excel. Both Excel and OLE automation use doubles to represent dates and times, with the integer portion of the double representing the number of days from a base date. In Excel, the base date that is used is 01/01/1900 and the year 1900 is treated as a leap year. In OLE automation, Microsoft corrected this by using the base date of 12/31/1899. As OLE automation does, our spreadsheets treat 1900 as a non-leap year and thus use the base date of 12/31/1899.

Return to the list of **Categories of Functions**.

Engineering Functions

The functions that relate to engineering calculations are:

BESSELI	BESSELJ	BESSELK	BESSELY
BIN2DEC	BIN2HEX	BIN2OCT	BITAND
BITLSHIFT	BITOR	BITRSHIFT	BITXOR
COMPLEX	CONVERT	DEC2BIN	DEC2HEX
DEC2OCT	DELTA	ERF	ERFC
ERFC.PRECISE	ERF.PRECISE	GESTEP	HEX2BIN
HEX2DEC	HEX2OCT	IMABS	IMAGINARY
IMARGUMENT	IMCONJUGATE	IMCOS	IMCOSH
IMCOT	IMCSC	IMCSCH	IMDIV
IMEXP	IMLN	IMLOG10	IMLOG2
IMPOWER	IMPRODUCT	IMREAL	IMSEC
IMSECH	IMSIN	IMSINH	IMSQRT
IMSUB	IMSUM	IMTAN	OCT2BIN
OCT2DEC	OCT2HEX		

For more information on the engineering functions that involve complex numbers, refer to **Complex Numbers in Engineering Functions**.

Return to the list of **Categories of Functions**.

Complex Numbers in Engineering Functions

Many of the engineering functions involve complex numbers. A complex number consists of two parts, a real part and an imaginary part. You can think of a complex number as being a point (x,y) in a plane. You can think of a real number as being a point (x,0) on the x-axis of the plane. Note that real numbers are a subset of complex numbers with zero for the coefficient of the imaginary part.

There is not a complex number data type. Instead, complex numbers are represented using strings of the form "x+yi" where x and y are real numbers and x is the real part and yi is the imaginary part. For example:

"2+3i"

"1.23E4+5.67E8i"

Note that if either the real part or the imaginary part is zero then the zero part can be optionally omitted from the text representation. For example:

"3" is equivalent to "3+0i"

"4i" is equivalent to "0+4i"

Since real numbers are a subset of complex numbers, a real number can be used in place of a string of the form "x+yi". For example:

3 is equivalent to "3+0i"

The functions that return a complex number return a string of the form "x+yi". For example:

COMPLEX(3,5) returns "3+5i"

The functions that accept a complex number can accept either a number or a string of the form "x+yi". For example:

IMSUM("1+2i", "3+4i") returns "4+6i"

IMSUM(1, 3) returns "4"

When a string cannot be converted to a number Spread returns a #VALUE error. For example:

COS("abc") returns #VALUE!

IMCOS("abc") returns #VALUE!

Spread allows either suffix "j" or the suffix "i" to denote the imaginary part. For example:

"3+4j" is equivalent to "3+4i"

Spread allows mixed suffixes in the a given formula and always returns the "i" suffix. For example:

IMSUM("1+2i", "3+4i") returns "4+6i"

IMSUM("1+2j", "3+4j") returns "4+6i"

IMSUM("1+2i", "3+4j") returns "4+6i"

Spread does not allow spaces before the real part or before the imaginary part. For example:

IMABS("3+4i") returns 5

IMABS(" 3+4i") returns #VALUE!

IMABS("3 +4i") returns #VALUE!

IMABS("3+4i ") returns #VALUE!

Return to **Engineering Functions**.

Return to the list of **Categories of Functions**.

Financial Functions

The functions that relate to financial calculations such as interest calculations are:

ACCRINT	ACCRINTM	AMORDEGRC	AMORLINC
COUPDAYS	COUPDAYBS	COUPDAYSNC	COUPNCD
COUPNUM	COUPPCD	CUMIPMT (on-line documentation)	CUMPRINC (on-line documentation)
DB	DDB	DISC	DOLLAR
DOLLARDE	DOLLARFR	DURATION	EFFECT
EUROCONVERT	FV	FVSCHEDULE	INTRATE
IPMT	IRR	ISPMT	MDURATION
MIRR	NOMINAL	NPER	NPV
ODDFPRICE	ODDFYIELD	ODDLPRICE	ODDLYIELD
PDURATION	PMT	PPMT	PRICE
PRICEDISC	PRICEMAT	PV	RATE
RECEIVED	RRI	SLN	SYD
TBILLEQ	TBILLPRICE	TBILLYIELD	VDB
XIRR	XNPV	YIELD	YIELDDISC
YIELDMAT			

For the financial functions that use it, refer to **Day Count Basis**.

Return to the list of **Categories of Functions**.

For the arguments of some of these functions and for the results of some of these functions, money paid out is represented by negative numbers and money you receive is represented by positive numbers. How the currency values are displayed depends upon how you set up the cell type and the format settings.

Day Count Basis

For many of the financial functions, the day count basis is used:

Basis Number	Day Count Basis
0 (or omitted)	United States of America (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

[NASD is the National Association of Securities Dealers.]

Return to **Financial Functions**.

Return to the list of **Categories of Functions**.

Information Functions

The functions that relate to information about a cell or the value in a cell are:

CELL	COUNTBLANK	ERRORTYPE	ERROR.TYPE
INFO	ISBLANK	ISERR	ISERROR
ISEVEN	ISFORMULA	ISLOGICAL	ISNA
ISNONTEXT	ISNUMBER	ISODD	ISREF
ISTEXT	N	NA	SHEET
SHEETS	TYPE		

Return to the list of **Categories of Functions**.

Logical Functions

The functions that relate to logical operations are:

- | | | | |
|---------------|--------------|------------|----------------|
| AND | FALSE | IF | IFERROR |
| IFNA | IFS | NOT | OR |
| SWITCH | TRUE | XOR | |

Return to the list of **Categories of Functions**.

Lookup Functions

The functions that relate to referencing and finding other parts of the spreadsheet are:

ADDRESS	AREAS	CHOOSE	COLUMN
COLUMNS	FORMULATEXT	HLOOKUP	HYPERLINK
INDEX	INDIRECT	LOOKUP	MATCH
OFFSET	ROW	ROWS	RTD
TRANSPOSE	VLOOKUP		

Return to the list of **Categories of Functions**.

Math and Trigonometry Functions

The functions that relate to mathematical calculations are:

ABS	ACOS	ACOSH	ACOT
ACOTH	AGGREGATE	ARABIC	ASIN
ASINH	ATAN	ATAN2	ATANH
BASE	CEILING	CEILING.MATH	CEILING.PRECISE
COMBIN	COMBINA	COS	COSH
COT	COTH	CSC	CSCH
DECIMAL	DEGREES	EVEN	EXP
FACT	FACTDOUBLE	FLOOR	FLOOR.MATH
FLOOR.PRECISE	GCD	INT	ISO.CEILING
LCM	LN	LOG	LOG10
MDETERM	MINVERSE	MMULT	MOD
MROUND	MULTINOMIAL	MUNIT	ODD
PI	POWER	PRODUCT	QUOTIENT
RADIANS	RAND	RANDBETWEEN	ROMAN
ROUND	ROUNDDOWN	ROUNDUP	SEC
SECH	SERIESSUM	SIGN	SIN
SINH	SQRT	SQRTPI	SUBTOTAL
SUM	SUMIF	SUMIFS	SUMPRODUCT
SUMSQ	SUMX2MY2	SUMX2PY2	SUMXMY2
TAN	TANH	TRUNC	

Return to the list of **Categories of Functions**.

Statistical Functions

The functions that relate to statistical operations are:

AVEDEV	AVERAGE	AVERAGEA	AVERAGEIF
AVERAGEIFS	BETADIST	BETA.DIST	BETA.INV
BETA.INV	BINOMDIST	BINOM.DIST	BINOM.DIST.RANGE
BINOM.INV	CHIDIST	CHIINV	CHISQ.DIST
CHISQ.DIST.RT	CHISQ.INV	CHISQ.INV.RT	CHISQ.TEST
CHITEST	CONFIDENCE	CONFIDENCE.NORM	CONFIDENCE.T
CORREL	COUNT	COUNTIF	COUNTIFS
COUNTA	COVAR	COVARIANCE.P	COVARIANCE.S
CRITBINOM	DEVSQ	EXPONDIST	EXPON.DIST
FDIST	F.DIST	F.DIST.RT	FINV
F.INV	F.INV.RT	FISHER	FISHERINV
FORECAST	FREQUENCY	FTEST	F.TEST
GAMMA	GAMMADIST	GAMMA.DIST	GAMMAINV
GAMMA.INV	GAMMALN	GAMMALN.PRECISE	GAUSS
GEOMEAN	GROWTH	HARMEAN	HYPGEOMDIST
HYPGEOM.DIST	INTERCEPT	KURT	LARGE
LINEST	LOGEST	LOGINV	LOGNORMDIST
LOGNORM.DIST	LOGNORM.INV	MAX	MAXA
MAXIFS	MEDIAN	MIN	MINA
MINIFS	MODE	MODE.MULT	MODE.SNGL
NEGBINOMDIST	NEGBINOM.DIST	NORMDIST	NORM.DIST
NORMINV	NORM.INV	NORMSDIST	NORM.S.DIST
NORMSINV	NORM.S.INV	PEARSON	PERCENTILE
PERCENTILE.EXC	PERCENTILE.INC	PERCENTRANK	PERCENTRANK.EXC
PERCENTRANK.INC	PERMUT	PERMUTATIONA	PHI
POISSON	POISSON.DIST	PROB	QUARTILE
QUARTILE.EXC	QUARTILE.INC	RANK	RANK.AVG
RANK.EQ	RSQ	SKEW	SKEW.P
SLOPE	SMALL	STANDARDIZE	STDEV
STDEVA	STDEVP	STDEV.P	STDEVPA
STDEV.S	STEYX	TDIST	T.DIST
T.DIST.2T	T.DIST.RT	TINV	T.INV
T.INV.2T	TREND	TRIMMEAN	TTEST
T.TEST	VAR	VARA	VARP

VAR.P

VARPA

VAR.S

WEIBULL

WEIBULL.DIST

ZTEST

Z.TEST

Return to the list of **Categories of Functions**.

Text Functions

The functions that relate to handling text are:

ASC	BAHTTEXT	CHAR	CLEAN
CODE	CONCAT	CONCATENATE	DBCS
DOLLAR	EXACT	FIND	FINDB
FIXED	JIS	LEFT	LEFTB
LEN	LENB	LOWER	MID
MIDB	NUMBERVALUE	PROPER	REPLACE
REPLACEB	REPT	RIGHT	RIGHTB
SEARCH	SEARCHB	SUBSTITUTE	T
TEXT	TEXTJOIN	TRIM	UPPER
UNICHAR	UNICODE	USDOLLAR	VALUE

Return to the list of **Categories of Functions**.

Web Functions

The functions that relate to web-based operations are:

ENCODEURL

FILTERXML

WEBSERVICE

Return to the list of **Categories of Functions**.

Optional Arguments

Some functions have a variable number of arguments with some (typically the last) arguments being optional. These are displayed in this reference with the word **Optional** in brackets "[Optional]" before the argument in the table of arguments. For example, consider the payment function (PMT) which has five arguments with the last two being optional. In Spread, you can make any of the following calls:

```
PMT(rate,nper,pv,fv,type)
```

```
PMT(rate,nper,pv,fv)
```

```
PMT(rate,nper,pv,fv,)
```

```
PMT(rate,nper,pv,,type)
```

```
PMT(rate,nper,pv,,)
```

```
PMT(rate,nper,pv)
```

The optional arguments may be omitted. Any missing optional argument is handled with the default value being passed. For example

```
FIXED(1234.5678,,FALSE)
```

evaluates the same as

```
FIXED(1234.5678,2,FALSE)
```

since the default value for the number of decimal places is 2.

See more about **Missing Arguments**.

Return to **Functions in a Formula**.

Return to the **Formula Overview**.

Missing Arguments

Missing arguments are intended to be used with functions that have optional arguments (as discussed in **Optional Arguments**). If a missing argument is passed in for a required argument then the function will evaluate to the #N/A error.

See more about **Resultant Error Values**.

Return to **Functions in a Formula**.

Return to the **Formula Overview**.

Volatile Functions

Formulas that contain volatile functions are recalculated every time any other function is recalculated. If the `EnableCrossSheetReferences` property for the control is false, then the functions are recalculated only on the sheet where the changes occur.

Array Formulas

You can use array formulas with the control.

An array formula can perform multiple calculations on one or more items in an array. Array formulas can return multiple results or a single result.

To create an array formula, select the cells you wish to put the formula in, type the formula, and then use Ctrl + Shift + Enter. This puts braces around the formula and places an instance of the formula in each cell of the selected range.

In order to remove or change an array formula, you must select the original formula range first.

Arrays in a Formula

Formulas may include functions that operate on arrays. Spread supports array constants in formulas. Use curly brackets { } to enclose the array elements. Use a comma to separate elements within a row. Use a semicolon to separate rows within the array. Individual elements can be number values, text values, logical values, or error values. Some examples of arrays are:

```
CORREL({5,10,15,20,25},{4,8,16,32,64})
```

```
CORREL({73000,45000,40360},{42,70,40})
```

```
ROWS({1,2,3;4,5,6})
```

Return to the **Formula Overview**.

Dynamic Array Formulas

Dynamic array formulas are useful especially when users want to implement effective utilization of data cache in the spreadsheets. This is possible because these formulas allow random access with low memory footprints.

When a cell contains a dynamic array formula, multiple values are returned because the elements of the array spill into the adjacent empty cells. Unlike generic arrays, dynamic arrays automatically resize when the data is inserted or removed from the source range.

Spilled Array Formulas

Dynamic array formulas that return more than one result and spill successfully to the nearby cells with values spanning to a cell range containing multiple rows and columns are known as spilled array formulas.

 **Note:** Spilled array formulas are not supported in Tables. However, while working with dynamic array formulas that spill to a number of rows and columns, the cell ranges used in the spreadsheets can be formatted explicitly to appear like tables.

For more information on dynamic array formulas and its examples, refer to the following topics:

1. **UNIQUE**
2. **SORT**
3. **SORTBY**
4. **RANDARRAY**
5. **SEQUENCE**
6. **SINGLE**
7. **FILTER**

Data Types Using Formulas

When you assign cell data using the Text property, the spreadsheet uses the cell type to parse the assigned string into the needed data type. For example, a NumberCellType parses a string into a double data type. When you assign the cell data using the Value property, the spreadsheet accepts the assigned object as is and no parsing occurs, so if you set it with a string, it remains a string. Some numeric functions (for example, SUM) ignore non-numeric values in a cell range. For example, if the cell range A1:A3 contains the values {1, "2", 3}, then the formula SUM(A1:A3) evaluates to 4 because the SUM function ignores the string "2". Be sure that you set the value correctly for any cells used in the calculation of a formula and that you set them with the correct data type.

Return to the **Formula Overview**.

Custom Functions in Formulas

Formulas may include custom, user-defined functions. If you have functions that you use on a regular basis that are not in the built-in functions or you wish to combine some of the built-in functions into a single function, you can do so by defining your own custom functions. They can be called as you would call any of the built-in functions. Custom functions can have up to 255 arguments.

Duplicate Function Names

A custom function can have the same name as a built-in function. The custom function takes priority over the built-in function. Custom functions are dynamically linked at evaluation time. Thus, the application can redefine an existing custom function.

Excel Function Names

In Spread, the HYPERLINK function is treated as a custom function since we do not have that as a built-in function. A custom function in Spread gets exported as a custom function in Excel. However, there is no way to export the implementation of the custom function. Thus, Excel sees the exported custom function as an unimplemented custom function which evaluates to the #NAME? error. When you enter or leave edit mode via the formula bar, Excel reparses the formula and recognizes the function as the built-in HYPERLINK function. Unfortunately, there is no way to tell the Spread control that a given custom function in Spread should be exported as a built-in function in Excel.

Suppose the application needs an Excel function that Spread does not support. The application would have to supply a custom function to mimic the Excel function. Spread's implementation of a custom function cannot be exported to an Excel file, so the custom function gets exported as an undefined custom function. In Excel, an undefined custom function will evaluate to the #VALUE! error. When you enter edit mode and then leave edit mode in Excel, Excel will reparse the formula (even if you made no changes to the formula). During reparsing, Excel will treat the function as the built-in function (instead of a custom function). The formula will then evaluate to the expected value (instead of the #VALUE! error). Your example of a problem formula does not appear to fall into the above described scenario because the formula only uses the MAX and SUM functions. However, it is still possible that the formula could be referencing a cell that uses a custom function which would get you back into the above described scenario. Editing the referenced cell would get rid of the #VALUE! error in the referenced cell. The recalculations would cascade back the cell in question.

See Also

Refer to the product Developer's Guide for more details on how to create a custom function.

Refer to the product Assembly Reference for more details on the methods that add or get custom functions.

Return to the **Formula Overview**.

Custom Names in Formulas

Formulas may include custom, user-defined names. Custom names are identifiers to represent information in the spreadsheet. A custom name can refer to a cell, a range of cells, a computed value, or a formula. (Methods that deal with custom names provide the same functionality as the Name in Excel.) A custom name can contain up to 255 characters and can include letters, numbers, or underscores. The first character must be a letter or an underscore.

The name's value can be assigned or retrieved as either a string object or as an expression object. Refer to the Assembly Reference for more details on the methods that add or get custom names.

From the example in C#:

```
dataModel.AddCustomName("Total", new FarPoint.CalcEngine.CellExpression(3, 2));
```

a name called Total is created that represents the cell at absolute location 3,2. Assuming A1 notation (ReferenceStyle = A1), then this would be equivalent to:

```
dataModel.AddCustomName("Total", "$D$3", 0, 0);
```

In Excel, this would be equivalent to:

Name: Total Refers To: =\$D\$3

Once the name is defined, the name can be used in formulas. When the formula is evaluated, the name's value is referenced and evaluated. Given the above definition, the following two formula assignments would produce the same result:

```
spread.ActiveSheet.SetFormula(0, 0, "Total"); spread.ActiveSheet.SetFormula(0, 0, "$D$3");
```

Note that the string versions of the AddCustomName and GetCustomName methods take the base row or base column arguments. These arguments are used to parse or unparse relative addresses in A1 notation. These arguments are ignored when using absolute addresses or when using R1C1 notation. A1 notation requires a base location from which the relative offset is computed.

For example:

```
dataModel.AddCustomName("Beta", "D3", 0, 0); // same as "R[2]C[3]" dataModel.AddCustomName("Gamma", "D3", 4, 4); // same as "R[-2]C[-1]"
```

In other words, cell D3 is +3/+2 from cell A1 but -1/-2 from cell E5. In Excel, the Insert > Name > Define dialog uses the active cell as the base location.

Refer to the product Developer's Guide for more details on how to create a custom name.

Refer to the Assembly Reference for more details on the methods that add or get custom names.

Return to the **Formula Overview**.

Resultant Error Values

The values that can be displayed in a cell as a result of an invalid entry or invalid formula are as follows:

Value	Description
#DIV/0!	This displays when a formula includes a division by zero or when a formula uses, in the divisor, a cell reference to a blank cell or to a cell that contains zero.
#N/A	This displays when a value is not available to a function or formula or when an argument in an array formula is not the same size as the range that contains the array formula.
#NAME?	This displays when text in a formula is not recognized or when the name of a function is misspelled, or when including text without using double quotation marks. This can also happen when you omit a colon (:) in a cell range reference.
#NULL!	This displays when you specify an intersection of two areas that do not intersect. Possible causes include a mistyped reference operator or a mistyped cell reference.
#NUM!	This displays when a number in a formula or function can not be calculated, when a formula produces a number that is too large or too small to represent, or when using an unacceptable argument in a function that requires a number. If you are using a function that iterates, such as IRR or RATE, and the function cannot find a result, this value is displayed.
#REF!	This displays when a cell reference is not valid or when you deleted cells referred to by a formula.
#VALUE!	This displays when the wrong type of argument or operand is used, such as using text when the formula requires a number or a logical value, or using a range instead of a single value.

Return to the **Formula Overview**.

Formula Functions

Spread.NET provides these built-in functions, listed alphabetically.

For a list of functions by type, refer to **Categories of Functions**.

Functions A to C

ABS	ACCRINT	ACCRINTM	ACOS
ACOSH	ACOT	ACOTH	ADDRESS
AGGREGATE	AMORDEGRC	AMORLINC	AND
ARABIC	AREAS	ASC	ASIN
ASINH	ATAN	ATAN2	ATANH
AVEDEV	AVERAGE	AVERAGEA	AVERAGEIF
AVERAGEIFS	BAHTTEXT	BASE	BESSELI
BESSELJ	BESSELK	BESSELY	BETA.DIST
BETA.INV	BETADIST	BETAINV	BIN2DEC
BIN2HEX	BIN2OCT	BINOM.DIST	BINOM.DIST.RANGE
BINOM.INV	BINOMDIST	BITAND	BITLSHIFT
BITOR	BITRSHIFT	BITXOR	CALL
CEILING	CEILING.MATH	CEILING.PRECISE	CELL
CHAR	CHIDIST	CHIINV	CHISQ.DIST
CHISQ.DIST.RT	CHISQ.INV	CHISQ.INV.RT	CHISQ.TEST
CHITEST	CHOOSE	CLEAN	CODE
COLUMN	COLUMNS	COMBIN	COMBINA
COMPLEX	CONCAT	CONCATENATE	CONFIDENCE
CONFIDENCE.NORM	CONFIDENCE.T	CONVERT	CORREL
COS	COSH	COT	COTH
COUNT	COUNTA	COUNTBLANK	COUNTIF
COUNTIFS	COUPDAYBS	COUPDAYS	COUPDAYSNC
COUPNCD	COUPNUM	COUPPCD	COVAR
COVARIANCE.P	COVARIANCE.S	CRITBINOM	CSC
CSCH	CUMIPMT (on-line documentation)	CUMPRINC (on-line documentation)	

Functions D to G

DATE	DATEDIF	DATEVALUE	DAVERAGE
DAY	DAYS	DAYS360	DB
DBCS	DCOUNT	DCOUNTA	DDB
DEC2BIN	DEC2HEX	DEC2OCT	DECIMAL
DEGREES	DELTA	DEVSQ	DGET

DISC	DMAX	DMIN	DOLLAR
DOLLARDE	DOLLARFR	DPRODUCT	DSTDEV
DSTDEVP	DSUM	DURATION	DVAR
DVARP	EDATE	EFFECT	ENCODEURL
EOMONTH	ERF	ERF.PRECISE	ERFC
ERFC.PRECISE	ERROR.TYPE	ERRORTYPE	EUROCONVERT
EVEN	EXACT	EXP	EXPON.DIST
EXPONDIST	F.DIST	F.DIST.RT	F.INV
F.INV.RT	F.TEST	FACT	FACTDOUBLE
FALSE	FDIST	FILTER	FILTERXML
FIND	FINDB	FINV	FISHER
FISHERINV	FIXED	FLOOR	FLOOR.MATH
FLOOR.PRECISE	FORECAST	FORECAST.LINEAR	FORMULATEXT
FREQUENCY	FTEST	FV	FVSCHEDULE
GAMMA	GAMMA.DIST	GAMMA.INV	GAMMADIST
GAMMAINV	GAMMALN	GAMMALN.PRECISE	GAUSS
GCD	GEOMEAN	GESTEP	GROWTH

Functions H to L

HARMEAN	HEX2BIN	HEX2DEC	HEX2OCT
HLOOKUP	HOUR	HYPERLINK	HYPGEOM.DIST
HYPGEOMDIST	IF	IFERROR	IFNA
IFS	IMABS	IMAGINARY	IMARGUMENT
IMCONJUGATE	IMCOS	IMCOSH	IMCOT
IMCSC	IMCSCH	IMDIV	IMEXP
IMLN	IMLOG10	IMLOG2	IMPOWER
IMPRODUCT	IMREAL	IMSEC	IMSECH
IMSIN	IMSINH	IMSQRT	IMSUB
IMSUM	IMTAN	INDEX	INDIRECT
INFO	INT	INTERCEPT	INTRATE
IPMT	IRR	ISBLANK	ISERR
ISERROR	ISEVEN	ISFORMULA	ISLOGICAL
ISNA	ISNONTEXT	ISNUMBER	ISO.CEILING
ISODD	ISOWEEKNUM	ISPMT	ISREF
ISTEXT	JIS	KURT	LARGE
LCM	LEFT	LEFTB	LEN
LENB	LINEST	LN	LOG
LOG10	LOGEST	LOGINV	LOGNORM.DIST

LOGNORM.INV

LOGNORMDIST

LOOKUP

LOWER

Functions M to Q

MATCH

MAX

MAXA

MAXIFS

MDETERM

MDURATION

MEDIAN

MID

MIDB

MIN

MINA

MINIFS

MINUTE

MINVERSE

MIRR

MMULT

MOD

MODE

MODE.MULT

MODE.SNGL

MONTH

MROUND

MULTINOMIAL

MUNIT

N

NA

NEGBINOM.DIST

NEGBINOMDIST

NETWORKDAYS

NETWORKDAYS.INTL

NOMINAL

NORM.DIST

NORM.INV

NORM.S.DIST

NORM.S.INV

NORMDIST

NORMINV

NORMSDIST

NORMSINV

NOT

NOW

NPER

NPV

NUMBERVALUE

OCT2BIN

OCT2DEC

OCT2HEX

ODD

ODDFPRICE

ODDFYIELD

ODDLPRICE

ODDLYIELD

OFFSET

OR

PDURATION

PEARSON

PERCENTILE

PERCENTILE.EXC

PERCENTILE.INC

PERCENTRANK

PERCENTRANK.EXC

PERCENTRANK.INC

PERMUT

PERMUTATIONA

PHI

PHONETIC

PI

PMT

POISSON

POISSON.DIST

POWER

PPMT

PRICE

PRICEDISC

PRICEMAT

PROB

PRODUCT

PROPER

PV

QUARTILE

QUARTILE.EXC

QUARTILE.INC

QUOTIENT

Functions R to S

RADIANS

RAND

RANDARRAY

RANDBETWEEN

RANK

RANK.AVG

RANK.EQ

RATE

RECEIVED

REPLACE

REPLACEB

REPT

RIGHT

RIGHTB

ROMAN

ROUND

ROUNDDOWN

ROUNDUP

ROW

ROWS

RRI

RSQ

RTD

SEARCH

SEARCHB

SEC

SECH

SECOND

SERIESSUM

SEQUENCE

SHEET

SHEETS

SIGN

SIN

SINH

SINGLE

SKEW

SKEW.P

SLN

SLOPE

SMALL

SORT

SORTBY

SQRT

SQRTPI

STANDARDIZE

STDEV

STDEV.P

STDEV.S

STDEVA

STDEVP

STDEVPA

STEYX	SUBSTITUTE	SUBTOTAL	SUM
SUMIF	SUMIFS	SUMPRODUCT	SUMSQ
SUMX2MY2	SUMX2PY2	SUMXMY2	SWITCH
SYD			

Functions T to Z

T	T.DIST	T.DIST.2T	T.DIST.RT
T.INV	T.INV.2T	T.TEST	TAN
TANH	TBILLEQ	TBILLPRICE	TBILLYIELD
TDIST	TEXT	TEXTJOIN	TIME
TIMEVALUE	TINV	TODAY	TRANSPOSE
TREND	TRIM	TRIMMEAN	TRUE
TRUNC	TTEST	TYPE	UNICHAR
UNICODE	UNIQUE	UPPER	USDOLLAR
VALUE	VAR	VAR.P	VAR.S
VARA	VARP	VARPA	VDB
VLOOKUP	WEBSERVICE	WEEKDAY	WEEKNUM
WEIBULL	WEIBULL.DIST	WORKDAY	WORKDAY.INTL
XIRR	XNPV	XOR	YEAR
YEARFRAC	YIELD	YIELDDISC	YIELDMAT
Z.TEST	ZTEST		

For more information on how to use these functions, refer to the **Formula Overview**, and to the chapter on Managing Formulas in the product Developers Guide.

Return to the **Formula Reference**.

Functions A to C

Functions A to C

ABS	ACCRINT	ACCRINTM	ACOS
ACOSH	ACOT	ACOTH	ADDRESS
AGGREGATE	AMORDEGRC	AMORLINC	AND
ARABIC	AREAS	ASC	ASIN
ASINH	ATAN	ATAN2	ATANH
AVEDEV	AVERAGE	AVERAGEA	AVERAGEIF
AVERAGEIFS	BAHTTEXT	BASE	BESSELI
BESSELJ	BESSELK	BESSELY	BETA.DIST
BETA.INV	BETADIST	BETAINV	BIN2DEC
BIN2HEX	BIN2OCT	BINOM.DIST	BINOM.DIST.RANGE
BINOM.INV	BINOMDIST	BITAND	BITLSHIFT
BITOR	BITRSHIFT	BITXOR	CALL
CEILING	CEILING.MATH	CEILING.PRECISE	CELL
CHAR	CHIDIST	CHIINV	CHISQ.DIST
CHISQ.DIST.RT	CHISQ.INV	CHISQ.INV.RT	CHISQ.TEST
CHITEST	CHOOSE	CLEAN	CODE
COLUMN	COLUMNS	COMBIN	COMBINA
COMPLEX	CONCAT	CONCATENATE	CONFIDENCE
CONFIDENCE.NORM	CONFIDENCE.T	CONVERT	CORREL
COS	COSH	COT	COTH
COUNT	COUNTA	COUNTBLANK	COUNTIF
COUNTIFS	COUPDAYBS	COUPDAYS	COUPDAYSNC
COUPNCD	COUPNUM	COUPPCD	COVAR
COVARIANCE.P	COVARIANCE.S	CRITBINOM	CSC
CSCH	CUMIPMT (on-line documentation)	CUMPRINC (on-line documentation)	

ABS

This function calculates the absolute value of the specified value.

Syntax

`ABS(value)`

`ABS(expression)`

Arguments

This function can take either a value or an expression as an argument.

Remarks

This function turns negative values into positive values.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ABS(R3C2)`

`ABS(B3)`

`ABS(-4)` gives the result 4

`ABS(14-24)` gives the result 10

`ABS(4)` gives the result 4

Version Available

This function is available in product version 1.0 or later.

See Also

SIGN | Math and Trigonometry Functions

ACCRINT

This function calculates the accrued interest for a security that pays periodic interest.

Syntax

`ACCRINT(issue,first,settle,rate,par,frequency,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>issue</i>	Date that the security is issued
<i>first</i>	First date for calculating the interest for the security
<i>settle</i>	Settlement date for the security
<i>rate</i>	Annual interest rate for the security
<i>par</i>	[Optional] Par value for the security; if omitted, the calculation uses a value of \$1,000
<i>frequency</i>	Frequency of payment, number of payments per year
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function requires that the issue is less than the settlement (otherwise a #NUM! error is returned). If the rate or par is less than or equal to 0, then a #NUM! error is returned. If the frequency is a number other than 1, 2, or 4, then a #NUM! error is returned. If the basis is less than 0 or greater than 4, a #NUM! error is returned.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`ACCRINT(A1,A2,A3,B4,D9,E9,0)`

`ACCRINT(DATE(2003,1,1), DATE(2003,1,7), DATE(2005,1,7), 0.5, 1000, 2)` gives the result 1008.33333

Version Available

This function is available in product version 1.0 or later.

See Also

ACCRINTM | **INTRATE** | **Financial Functions**

ACCRINTM

This function calculates the accrued interest at maturity for a security that pays periodic interest.

Syntax

`ACCRINTM(issue,maturity,rate,par,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>issue</i>	Date that the security is issued
<i>maturity</i>	Maturity date for security
<i>rate</i>	Annual interest rate for the security
<i>par</i>	[Optional] Par value for the security; if omitted, the calculation uses a value of \$1,000
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function requires that issue is a valid date (otherwise a #Value! error is returned). If the rate or par is less than or equal to 0, then a #NUM! error is returned. If the basis is less than 0 or greater than 4, a #NUM! error is returned. If the issue is less than the settlement, then a #NUM! error is returned.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`ACCRINTM(A2,A3,B4,D9,3)`

`ACCRINTM(R1C1,R2C3,R4C2,R9C4,3)`

Version Available

This function is available in product version 1.0 or later.

See Also

ACCRINT | **INTRATE** | **Financial Functions**

ACOS

This function calculates the arccosine, that is, the angle whose cosine is the specified value.

Syntax

`ACOS(value)`

Arguments

For the argument, you can specify the cosine of the angle you want to return, which must be a value between -1 and 1.

Remarks

The result angle is in radians between 0 (zero) and PI (pi). If you want to convert the result to degrees, multiply the result by 180/PI.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ACOS(B3)`

`ACOS(R3C2)`

`ACOS(0.5)` gives the result 1.0471975512

Version Available

This function is available in product version 1.0 or later.

See Also

ACOSH | ASIN | COS | COSH | Math and Trigonometry Functions

ACOSH

This function calculates the inverse hyperbolic cosine of the specified value.

Syntax

`ACOSH(value)`

Arguments

For the argument, you can specify any real number greater than or equal to 1.

Remarks

This function is the inverse of the hyperbolic cosine, so `ACOSH(COSH(n))` gives the result *n*.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ACOSH(B3)`

`ACOSH(R3C2)`

`ACOSH(1)` gives the result 0

`ACOSH(10)` gives the result 2.9932228461

`ACOS(R3C2)`

Version Available

This function is available in product version 1.0 or later.

See Also

[ACOS](#) | [ASINH](#) | [Math and Trigonometry Functions](#)

ACOT

This function returns the arccotangent (inverse of the cotangent) of the specified numeric value.

Syntax

`ACOT(value)`

Arguments

For the argument, you can specify the cotangent of the angle you want to return. The argument should be a real number.

Remarks

The resultant angle is in radians and its value lies between 0 (zero) and PI (pi). If you want to convert the result from radians to degrees, multiply the resultant value by 180/PI.

Data Types

Accepts numeric data. Returns numeric data.

Examples

ACOT(5) gives the result 0.1973

ACOT(10) gives the result 0.0996

ACOT(15) gives the result 0.0665

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

ACOTH

This function calculates the inverse hyperbolic cotangent of the specified numeric value.

Syntax

$ACOTH(value)$

Arguments

For the argument, you can specify any number whose absolute value is greater than 1.

Remarks

This function is the inverse of the hyperbolic cotangent. If the specified number is less than 1, this function returns a #NUM! error value. If the specified number is not a numeric value, it returns a #VALUE! error.

Data Types

Accepts numeric data. Returns numeric data.

Examples

$ACOTH(8)$ gives the result 0.1256

$ACOTH(12)$ gives the result 0.0835

$ACOTH(4)$ gives the result 0.2554

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

ADDRESS

This function uses the row and column numbers to create a cell address in text.

Syntax

`ADDRESS(row,column,absnum,a1style,sheettext)`

Arguments

This function has these arguments:

Argument	Description
<i>row</i>	Row number in the cell reference
<i>column</i>	Column number in the cell reference
<i>absnum</i>	[Optional] Type of reference to return; can be any of: Value - Type of Cell Reference Returned 1 or omitted - Absolute 2 - Absolute row, relative column 3 - Relative row, absolute column 4 - Relative
<i>a1style</i>	[Optional] Logical value that indicates whether the reference style is A1; if TRUE or omitted, the style is A1; if FALSE, then the style is R1C1
<i>sheettext</i>	[Optional] Name of the sheet to use as an external reference; if omitted, no sheet name is used

Data Types

Accepts numeric and string data. Returns string data.

Examples

`ADDRESS (2, 4, 2, FALSE)`

Version Available

This function is available in product version 2.0 or later.

See Also

COLUMNS | **ROWS** | **INDEX** | **Lookup Functions**

AGGREGATE

This function calculates an aggregate value in a list or database.

Syntax

AGGREGATE(*functionnum*, *options*, *reference1*, *reference2*, ...)

or

AGGREGATE(*functionnum*, *options*, *array*, *k*)

Arguments

This function has the following arguments:

Argument Description

<i>functionnum</i>	Refers to the number code that specifies the function to use (see table below).
<i>options</i>	Refers to the numerical value that determines which values must be ignored in the evaluation range for the function (see table below)
<i>reference1</i>	Refers to the argument for which you want the aggregate value.
<i>reference2</i>	[Optional] Refers to the additional arguments for which you want the aggregate value.
<i>array</i>	Refers to an array, array formula, or a reference to a range of cells for which you want the aggregate value.
<i>k</i>	Refers to the additional arguments for which you want the aggregate value.

The *functionnum* argument is the number that represents the built-in function to use, as given in this table.

Built-In Function	Function Code
AVERAGE	1
COUNT	2
COUNTA	3
MAX	4
MIN	5
PRODUCT	6
STDEV.S	7
STDEV.P	8
SUM	9
VAR.S	10
VAR.P	11
MEDIAN	12
MODE.SNGL	13
LARGE	14
SMALL	15
PERCENTILE.INC	16
QUARTILE.INC	17
PERCENTILE.EXC	18
QUARTILE.EXC	19

The *options* argument has the following options.

Option	Description
0 or omitted	Ignore nested SUBTOTAL and AGGREGATE functions
1	Ignore hidden rows and nested SUBTOTAL and AGGREGATE functions
2	Ignore error values and nested SUBTOTAL and AGGREGATE functions
3	Ignore hidden rows, error values, and nested SUBTOTAL and AGGREGATE functions
4	Ignore nothing
5	Ignore hidden rows
6	Ignore error values
7	Ignore hidden rows and error values

Remarks

This function is designed for columns of data, or vertical cell range. This function has an option to ignore hidden rows and error values.

If a second *reference* argument is required but it is not provided by the user, a #VALUE! error is returned. Also, if one or more of the references are 3-D references, a #VALUE! error value is returned.

The *reference2* argument is optional; however, the following functions are required in this argument.

- LARGE
- SMALL
- PERCENTILE.INC
- QUARTILE.INC
- PERCENTILE.EXC
- QUARTILE.EXC

Data Types

Accepts numeric values. Returns numeric data.

Examples

`AGGREGATE(14,6,B1:B7,3)` gives the result 5, where B1:B7 is the cell range containing a numeric list.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

AMORDEGRC

This function returns the depreciation for an accounting period, taking into consideration prorated depreciation, and applies a depreciation coefficient in the calculation based on the life of the assets.

Syntax

`AMORDEGRC(cost,datepurchased,firstperiod,salvage,period,drate,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>cost</i>	Cost of the asset
<i>datepurchased</i>	Purchase date of the asset
<i>firstperiod</i>	End date of the first period
<i>salvage</i>	Salvage value at the end of the life of the asset
<i>period</i>	Accounting period
<i>drate</i>	Rate of depreciation
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis)

Remarks

This function returns the depreciation until the last period of the asset life or until the total value of depreciation is greater than the cost of the assets minus the salvage value. The depreciation coefficients are:

Life of assets	Depreciation Coefficient
Between 3 and 4 years	1.5
Between 5 and 6 years	2
More than 6 years	2.5

The depreciation rate will grow to 50 percent for the period proceeding the last period and will grow to 100 percent for the last period. If the life of assets is between 0 (zero) and 1, 1 and 2, 2 and 3, or 4 and 5, the #NUM! error value is returned.

This function differs from **AMORLINC**, which does not apply a depreciation coefficient in the calculation depending on the life of the assets.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`AMORDEGRC(B1,B2,B3,B4,B5,B6,B7)`

`AMORDEGRC(2800,DATE(2003,9,4),DATE(2006,12,31),200,1,0.02,1)` gives the result 117

Version Available

This function is available in product version 2.0 or later.

See Also

AMORLINC | **Financial Functions**

AMORLINC

This function calculates the depreciation for an accounting period, taking into account prorated depreciation.

Syntax

`AMORLINC(cost,datepurchased,firstperiod,salvage,period,drate,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>cost</i>	Cost of the asset
<i>datepurchased</i>	Purchase date of the asset
<i>firstperiod</i>	End date of the first period
<i>salvage</i>	Salvage value at the end of the life of the asset
<i>period</i>	Accounting period
<i>drate</i>	Rate of depreciation
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis)

Remarks

This function differs from **AMORDEGRC**, which applies a depreciation coefficient in the calculation depending on the life of the assets.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`AMORLINC(B1,B2,B3,B4,B5,B6,B7)`

Version Available

This function is available in product version 2.0 or later.

See Also

AMORDEGRC | Financial Functions

AND

This function calculates logical AND.

Syntax

`AND(bool1,bool2,...)`

`AND(array)`

`AND(array1,array2,...)`

`AND(expression)`

`AND(expression1,expression2,...)`

Arguments

For the arguments of this function, provide numeric (0 or 1) or logical values (TRUE or FALSE) for up to 255 arguments. You can also specify a single array instead of listing the values separately, or up to 255 arrays. You can also specify the logical argument as an expression.

Remarks

This function returns TRUE if all its arguments are true; otherwise, returns FALSE if at least one argument is false.

Data Types

Accepts logical data (Boolean values of TRUE or FALSE) or numerical values (0 or 1). Returns logical data (Boolean values of TRUE or FALSE).

Examples

`AND(D12,E12)`

`AND(R12C42,R12C5,R12C1)`

`AND(D2:D12)`

`AND(R12C1:R12C9)`

`AND(true,true,true)` gives the result TRUE

`AND(TRUE(),FALSE())` gives the result FALSE

`AND(5+3=8,5+1=6)` gives the result TRUE

Version Available

This function is available in product version 1.0 or later.

See Also

NOT | OR | Logical Functions

ARABIC

This function converts the specified Roman value to an Arabic value.

Syntax

ARABIC(*text*)

Arguments

For the argument, a string enclosed in quotation marks or a reference to the cell (possessing text value only) can be specified.

Remarks

In this function, specifying the text in lower case or upper case doesn't matter because the case of the argument (supplied as a *text value*) is ignored.

Data Types

Accepts string data. Returns numeric data.

Examples

ARABIC("XIXI") gives the result 20

ARABIC("LIV") gives the result 54

ARABIC(C2) gives the result 61 where C2 is a cell reference containing the text value LXI.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

AREAS

This function returns the number of areas in the specified reference.

Syntax

`AREAS(reference)`

Arguments

Specify the cell reference for the argument.

Remarks

In this function, single argument can refer to multiple areas. To do so, extra set of parentheses is used.

Data Types

Accepts cell reference for argument. Returns numeric data.

Examples

`AREAS(B7:D9)` gives the result 1

`AREAS((B2:D4,E5,F6:I9))` gives the result 3

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

ASC

This function transforms full-width (double-byte) characters to half-width (single-byte) characters.

Syntax

`ASC(text)`

Arguments

For the arguments, you need to give either a text value or a reference to a cell containing the text to be changed.

Remarks

If the text does not contain full-width letters, then the text is not changed.

Data Types

Accepts string data. Returns string data.

Examples

`ASC("SPREAD")` gives the result "SPREAD"

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

ASIN

This function calculates the arcsine, that is, the angle whose sine is the specified value.

Syntax

`ASIN(value)`

Arguments

For the argument, specify the sine of the angle you want to return, which must be a value between -1 and 1 .

Remarks

The result angle is in radians between $-\pi/2$ and $\pi/2$. If you want to convert the result to degrees, multiply the result by $180/\pi$.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ASIN(B3)`

`ASIN(R3C2)`

`ASIN(0.5)` gives the result 0.5235987756

Version Available

This function is available in product version 1.0 or later.

See Also

ACOS | SIN | SINH | Math and Trigonometry Functions

ASINH

This function calculates the inverse hyperbolic sine of a number.

Syntax

`ASINH(value)`

Arguments

For the argument, you can specify any real number.

Remarks

This function is the inverse of the hyperbolic sine, so `ASINH(SINH(n))` gives the result *n*.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ASINH(E4)`

`ASINH(R4C5)`

`ASINH(-5.5)` gives the result -2.40606

`ASINH(100)` gives the result 5.2983423656

Version Available

This function is available in product version 1.0 or later.

See Also

ACOSH | ASIN | SIN | Math and Trigonometry Functions

ATAN

This function calculates the arctangent, that is, the angle whose tangent is the specified value.

Syntax

`ATAN(value)`

Arguments

For the argument, specify the tangent of the angle you want to return, which must be a value between -1 and 1 .

Remarks

The result angle is in radians between $-\pi/2$ and $\pi/2$. If you want to convert the result to degrees, multiply the result by $180/\pi$.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ATAN(B3)`

`ATAN(R3C2)`

`ATAN(1)` gives the result 0.7853981634

Version Available

This function is available in product version 1.0 or later.

See Also

[ACOS](#) | [ASIN](#) | [TAN](#) | [Math and Trigonometry Functions](#)

ATAN2

This function calculates the arctangent of the specified x- and y-coordinates.

Syntax

`ATAN2(x,y)`

Arguments

This function can take real numbers as arguments.

Remarks

The arctangent is the angle from the x-axis to a line containing the origin (0, 0) and a point with coordinates (x, y).

The result is given in radians between $-\pi$ and π , excluding $-\pi$. If you want to convert the result to degrees, multiply the result by $180/\pi$.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ATAN2 (A1, E3)`

`ATAN2 (R1C1, R3C5)`

`ATAN2(1,1)` gives the result 0.7853981634

Version Available

This function is available in product version 1.0 or later.

See Also

ACOS | ASIN | ATAN | TAN | Math and Trigonometry Functions

ATANH

This function calculates the inverse hyperbolic tangent of a number.

Syntax

`ATANH(value)`

Arguments

For the argument, you can specify any real number between 1 and -1 , excluding -1 and 1 .

Remarks

This function is the inverse of the hyperbolic tangent, so `ATANH(TANH(n))` gives the result *n*.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ATANH(B5)`

`ATANH(R5C2)`

`ATANH(0.55)` gives the result `0.6183813136`

`ATANH(-0.2)` gives the result `-0.2027325541`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOSH | ASINH | ATAN | TAN | Math and Trigonometry Functions

AVEDEV

This function calculates the average of the absolute deviations of the specified values from their mean.

Syntax

`AVEDEV(value1,value2,...)`

`AVEDEV(array)`

`AVEDEV(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This is a measure of the variability in a data set.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`AVEDEV(B5,L32,N25,D17)`

`AVEDEV(B1:B5)`

`AVEDEV(B1:B17,L1:L17,N2:N8)`

`AVEDEV(R5C2,R32C12,R25C15) AVEDEV(R1C2:R1C7)`

`AVEDEV(98,79,85)` gives the result 7.1111111111

Version Available

This function is available in product version 1.0 or later.

See Also

AVERAGE | **DEVSQ** | **Statistical Functions**

AVERAGE

This function calculates the average of the specified numeric values.

Syntax

`AVERAGE(value1,value2,...)`

`AVERAGE(array)`

`AVERAGE(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This is a measure of the variability in a data set.

This function differs from **AVERAGEA**, which accepts text or logical values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`AVERAGE(A1,B3,D5,E9,L8,L9)`

`AVERAGE(R1C1,R3C2)`

`AVERAGE(A1:A9)`

`AVERAGE(A1:A9,B1:B9,D5:D8)`

`AVERAGE(98,72,85)` gives the result 85

Version Available

This function is available in product version 1.0 or later.

See Also

AVEDEV | **AVERAGEA** | **CONFIDENCE** | **DEVSQ** | **MEDIAN** | **VAR** | **Statistical Functions**

AVERAGEA

This function calculates the average of the specified values, including text or logical values as well as numeric values.

Syntax

`AVERAGEA(value1,value2,...)`

`AVERAGEA(array)`

`AVERAGEA(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range). Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This is a measure of the variability in a data set.

This function differs from **AVERAGE** because it allows text or logical values as well as numeric values.

Data Types

Accepts numeric, logical, or text data for all arguments. Returns numeric data.

Examples

`AVERAGEA(A1,B3,D5,E9,L8,L9)`

`AVERAGEA(R1C1,R3C2)`

`AVERAGEA(A1:A9)`

`AVERAGEA(A1:A9,B1:B9,D5:D8)`

`AVERAGEA(98,72,85)` gives the result 85

Version Available

This function is available in product version 2.0 or later.

See Also

AVEDEV | **DEVSQ** | **MEDIAN** | **VAR** | **AVERAGE** | **Statistical Functions**

AVERAGEIF

This function calculates the average of the specified numeric values provided that they meet the specified criteria.

Syntax

`AVERAGEIF(value1,value2,...,condition)`

`AVERAGEIF(array,condition)`

`AVERAGEIF(array1,array2,...,condition)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range). Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This is a measure of the variability in a data set.

Data Types

Accepts numeric data. The condition accepts text, numeric, or expression data. Returns numeric data.

Examples

`AVERAGEIF(A1,B3,D5,E9,L8,L9,"<5000")`

`AVERAGEIF(R1C1,R3C2,"<>0")`

Version Available

This function is available in product version 5.0 or later.

See Also

AVEDEV | DEVSQ | MEDIAN | VAR | AVERAGE | Statistical Functions

AVERAGEIFS

This function calculates the average of all cells that meet multiple specified criteria.

Syntax

`AVERAGEIFS(value1,condition1,value2,...,condition2...)`

`AVERAGEIFS(array,condition)`

`AVERAGEIFS(array1,array2,...,condition)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range). Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well. You can have up to 127 arguments for the conditions.

Remarks

This is a measure of the variability in a data set.

Data Types

Accepts numeric data. The condition accepts text, numeric, or expression data. Returns numeric data.

Examples

`AVERAGEIFS(B2:B5,B2:B5,">90",B2:B5,"<100")`

`AVERAGEIFS(R1C1,R3C2,"<>0")`

Version Available

This function is available in product version 5.0 or later.

See Also

AVEDEV | DEVSQ | MEDIAN | VAR | AVERAGE | Statistical Functions

BAHTTEXT

This function converts the specified value to Thai text and adds a suffix "Baht" to it.

Syntax

BAHTTEXT(*value*)

Arguments

The argument can accept a numeric value, or a cell reference that contains a numeric value.

Data Types

Accepts numeric data. Returns string data.

Examples

BAHTTEXT(1234) gives the result หนึ่งพันสองร้อยสามสิบสี่บาทถ้วน

BAHTTEXT(C3) gives the result หนึ่งพันสองร้อยสามสิบสี่บาทถ้วน, where C3 is the cell reference that contains the value 1234

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

BASE

This function converts a number into a text representation according to specified base.

Syntax

`BASE(value,base,min_len)`

Arguments

This function has the following arguments:

Argument	Description
----------	-------------

<i>value</i>	Refers to the value that you want to convert. This value must be greater than or equal to 0 and less than 2^{53}
<i>base</i>	Refers to the base that you want to convert the <i>value</i> into. This value must be greater than or equal to 2 and less than or equal to 36
<i>min_len</i>	[Optional] Refers to the minimum length of the resultant string. This value must be greater than or equal to 0 and less than or equal to 255

Remarks

This functions returns an error when the specified value in the arguments doesn't meet the minimum and maximum constraints as mentioned in the above table.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`BASE(9,3)` gives the result 100.

`BASE(8,2,1)` gives the result 1000.

`BASE(6,3,2)` gives the result 20.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

BESSELI

This function calculates the modified Bessel function of the first kind evaluated for purely imaginary arguments.

Syntax

BESSELI(*value,order*)

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Value at which to evaluate the function
<i>order</i>	Number representing the order of the function; if it is not an integer, it is truncated

Remarks

If *value* or *order* is nonnumeric then a #Value! error is returned. If *order* is less than 0 then the #NUM! error is returned.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

BESSELI (A4, D5)

BESSELI (R4C1, R5C4)

BESSELI (1.8, 2) gives the result 0.5260402117

Version Available

This function is available in product version 1.0 or later.

See Also

BESSELJ | **BESSELY** | **Engineering Functions**

BESSELJ

This function calculates the Bessel function of the first kind.

Syntax

`BESSELJ(value,order)`

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Value at which to evaluate the function
<i>order</i>	Number representing the order of the function; if it is not an integer, it is truncated

Remarks

If value or order is nonnumeric then a #Value! error is returned. If order is less than 0 then the #NUM! error is returned.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`BESSELJ(A4,D5)`

`BESSELJ(R4C1,R5C4)`

`BESSELJ(1.85,2)` gives the result 0.31812827879

Version Available

This function is available in product version 1.0 or later.

See Also

BESSELI | BESSELK | Engineering Functions

BESSELK

This function calculates the modified Bessel function of the second kind evaluated for purely imaginary arguments.

Syntax

`BESSELK(value,order)`

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Value at which to evaluate the function
<i>order</i>	Number representing the order of the function; if it is not an integer, it is truncated

Remarks

This function is also called the Neumann function. If value or order is nonnumeric then a #Value! error is returned. If order is less than 0 then the #NUM! error is returned.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`BESSELK(A4,D5)`

`BESSELK(R4C1,R5C4)`

`BESSELK(1.85,2)` gives the result 0.32165379

Version Available

This function is available in product version 1.0 or later.

See Also

BESSELJ | **BESSELY** | **Engineering Functions**

BESSELY

This function calculates the Bessel function of the second kind.

Syntax

`BESSELY(value,order)`

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Value at which to evaluate the function
<i>order</i>	Number representing the order of the function; if it is not an integer, it is truncated

Remarks

If value or order is nonnumeric then a #Value! error is returned. If order is less than 0 then the #NUM! error is returned.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`BESSELY(A4,D5)`

`BESSELY(R4C1,R5C4)`

`BESSELY(2.85,1)` gives the result 0.2801918953

Version Available

This function is available in product version 1.0 or later.

See Also

BESSELJ | **BESSELK** | **Engineering Functions**

BETA.DIST

This function calculates the cumulative beta distribution function.

Syntax

`BETA.DIST(x,alpha,beta,cumulative,lower,upper)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>x</i>	Value at which to evaluate the function, between the values of lower and upper
<i>alpha</i>	Alpha parameter of the distribution
<i>beta</i>	Beta parameter of the distribution
<i>cumulative</i>	A logical value that determines the function form. The function returns the cumulative distribution function if this argument is true. It returns the probability density function, if the argument is false
<i>lower</i>	[Optional] Lower bound of the interval for x; 0 if omitted
<i>upper</i>	[Optional] Upper bound of the interval for x; 1 if omitted

Remarks

If you omit values for *upper* and *lower*, the calculation uses the standard cumulative beta distribution, so that *lower* is zero and *upper* is one.

Data Types

Accepts numeric data for all arguments except *cumulative*. Accepts TRUE or FALSE for *cumulative*. Returns numeric data.

Examples

`BETA.DIST(A1,A3,B4,TRUE,1,3)`

`BETA.DIST(2,8,10,TRUE,1,3)` gives the result 0.6854705810546875

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

BETADIST

BETA.INV

This function calculates the inverse of the cumulative beta density function.

Syntax

`BETA.INV(prob,alpha,beta,lower,upper)`

Arguments

This function has these arguments:

Argument	Description
<i>prob</i>	Probability of the distribution
<i>alpha</i>	Alpha parameter of the distribution
<i>beta</i>	Beta parameter of the distribution
<i>lower</i>	[Optional] Lower bound of the interval for x; 0 if omitted
<i>upper</i>	[Optional] Upper bound of the interval for x; 1 if omitted

Remarks

If you omit values for *upper* and *lower*, the calculation uses the standard cumulative beta distribution, so that *lower* is zero and *upper* is one. This function returns the #VALUE! error value if any argument is nonnumeric.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`BETA.INV(0.75,B3,C3,2,4)`

`BETA.INV(0.75,R3C2,R3C3,2,4)`

`BETA.INV(0.75,9,12,2,4)` gives the result 3.0011968805340232

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

BETAINV

BETADIST

This function calculates the cumulative beta distribution function.

Syntax

`BETADIST(x,alpha,beta,lower,upper)`

Arguments

This function has these arguments:

Argument	Description
<i>x</i>	Value at which to evaluate the function, between the values of lower and upper
<i>alpha</i>	Alpha parameter of the distribution
<i>beta</i>	Beta parameter of the distribution
<i>lower</i>	[Optional] Lower bound of the interval for x; 0 if omitted
<i>upper</i>	[Optional] Upper bound of the interval for x; 1 if omitted

Remarks

If you omit values for *upper* and *lower*, the calculation uses the standard cumulative beta distribution, so that *lower* is zero and *upper* is one.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

```
BETADIST(3,B3,C3,2,4)
```

```
BETADIST(3,R3C2,R3C3,2,4)
```

```
BETADIST(3,6,9,2,4) gives the result 0.7880249023
```

Version Available

This function is available in product version 1.0 or later.

See Also

BETAINV | **Statistical Functions**

BETAINV

This function calculates the inverse of the cumulative beta distribution function.

Syntax

`BETAINV(prob, alpha, beta, lower, upper)`

Arguments

This function has these arguments:

Argument	Description
<i>prob</i>	Probability of the distribution
<i>alpha</i>	Alpha parameter of the distribution
<i>beta</i>	Beta parameter of the distribution
<i>lower</i>	[Optional] Lower bound of the interval for x; 0 if omitted
<i>upper</i>	[Optional] Upper bound of the interval for x; 1 if omitted

Remarks

If you omit values for *upper* and *lower*, the calculation uses the standard cumulative beta distribution, so that *lower* is zero and *upper* is one.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`BETAINV(0.75, B3, C3, 2, 4)`

`BETAINV(0.75, R3C2, R3C3, 2, 4)`

`BETAINV(0.75, 9, 12, 2, 4)` gives the result 3.0011968805

Version Available

This function is available in product version 1.0 or later.

See Also

BETADIST | Statistical Functions

BIN2DEC

This function converts a binary number to a decimal number.

Syntax

`BIN2DEC(number)`

Arguments

For the argument of this function, specify the binary numeric value to convert.

Remarks

An error value is returned if the number contains more than 10 digits or is invalid.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`BIN2DEC(11111111)`

Version Available

This function is available in product version 2.0 or later.

See Also

[BIN2HEX](#) | [BIN2OCT](#) | [DEC2BIN](#) | [OCT2DEC](#) | [Engineering Functions](#)

BIN2HEX

This function converts a binary number to a hexadecimal number.

Syntax

`BIN2HEX(number,places)`

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Binary numeric value to convert
<i>places</i>	[Optional] Number of characters to return; if not an integer, the number is truncated

Remarks

An error value is returned if the *number* contains more than 10 digits or is invalid, or if the value of *places* is non-numeric or negative. If *places* is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Data Types

Accepts numeric data. Returns numeric data in hexadecimal format.

Examples

`BIN2HEX(11110)`

Version Available

This function is available in product version 2.0 or later.

See Also

[BIN2DEC](#) | [BIN2OCT](#) | [DEC2HEX](#) | [OCT2HEX](#) | [Engineering Functions](#)

BIN2OCT

This function converts a binary number to an octal number.

Syntax

`BIN2OCT(number,places)`

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Binary numeric value to convert
<i>places</i>	[Optional] Number of characters to return; if not an integer, the number is truncated

Remarks

An error value is returned if the *number* contains more than 10 digits or is invalid, or if the value of *places* is non-numeric or negative. If *places* is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`BIN2OCT(1001,2)`

Version Available

This function is available in product version 2.0 or later.

See Also

[BIN2DEC](#) | [BIN2HEX](#) | [OCT2BIN](#) | [DEC2OCT](#) | [Engineering Functions](#)

BINOM.DIST

This function calculates the individual term binomial distribution probability.

Syntax

`BINOM.DIST(x,n,p,cumulative)`

Arguments

This function has these arguments:

Argument	Description
<i>x</i>	Number representing the number of successes in trials; if not an integer, the number is truncated
<i>n</i>	Number representing the number of independent trials; if not an integer, the number is truncated
<i>p</i>	Probability of success on each trial; number between 0 and 1
<i>cumulative</i>	Logical value that determines the form of the function; if TRUE, then this function returns the cumulative distribution function, which is the probability that there are at most x successes; if FALSE, it returns the probability mass function, which is the probability that there are x successes

Remarks

Use this function in problems with a fixed number of tests or trials, when there are two mutually exclusive possible outcomes (a "success" and a "failure"), when trials are independent, and when the probability of one outcome is constant throughout the experiment. This function can, for example, calculate the probability that two of the next three babies born are male.

The binomial probability mass function is calculated as follows:

$$BINOMDIST(x, n, p, FALSE) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}$$

where *x* is the number of successes, *n* is the number of trials, and *p* is the probability of success on any one trial. The cumulative binomial distribution is calculated as follows:

$$BINOMDIST(x, n, p, TRUE) = \sum_{y=x}^n BINOMDIST(y, n, p, FALSE)$$

where *n* is the number of trials, *x* is the number of successes, and *p* is the possibility of success on any one trial.

Data Types

Accepts numeric data for all arguments, except cumulative, which accepts logical data. Returns numeric data.

Example

A baby can be either male or female; for this example, assume the odds are 50/50 that a baby is either male or female. If female equals TRUE, we can use the following to determine the probability of the next 5 babies in 10 born being female. The probability of the first baby being female is 0.5, and the probability of exactly 5 of 10 babies born being female is:

`BINOM.DIST(5,10,0.5,FALSE)` gives the result 0.24609375

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

BINOMDIST

BINOM.DIST.RANGE

This function uses binomial distribution to calculate the probability of a trial result.

Syntax

`BINOM.DIST.RANGE(N,p,s,s2)`

Arguments

This function has the following arguments:

Argument	Description
----------	-------------

<i>N</i>	Refers to the number representing the number of independent trials. If this value is not an integer, the number is truncated.
<i>p</i>	Refers to the probability of success on each trial. The value of this number must lie between 0 and 1
<i>s</i>	Refers to the number representing the number of successes in trials. If this value is not an integer, the number is truncated.
<i>s2</i>	[Optional] Refers to the probability that number of successful trials will fall between <i>s</i> and <i>s2</i> . This value must be greater than or equal to <i>s</i> and less than or equal to <i>N</i> .

Remarks

All the numeric arguments in this function will be truncated to integers.

If the arguments passed are not numeric values or the values do not lie within the constraints, this function will return an error.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`BINOM.DIST.RANGE(50,0.35,28)` gives the result 0.00116

`BINOM.DIST.RANGE(30,0.25,15,25)` gives the result 0.00274

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

BINOM.INV

This function returns the criterion binomial, the smallest value for which the cumulative binomial distribution is greater than or equal to a criterion value.

Syntax

`BINOM.INV(n,p,alpha)`

Arguments

This function has these arguments:

Argument	Description
<i>n</i>	Number of trials; if not an integer, the number is truncated
<i>p</i>	Probability of success on each trial; number between 0 and 1
<i>alpha</i>	Alpha, value for the criterion

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`BINOM.INV(B5,0.75,0.92)`

`BINOM.INV(R5C2,R8C14,0.75)`

`BINOM.INV(14,0.75,0.85)` gives the result 14

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

BINOMDIST

BINOMDIST

This function calculates the individual term binomial distribution probability.

Syntax

$BINOMDIST(x,n,p,cumulative)$

Arguments

This function has these arguments:

Argument	Description
x	Number representing the number of successes in trials; if not an integer, the number is truncated
n	Number representing the number of independent trials; if not an integer, the number is truncated
p	Probability of success on each trial; number between 0 and 1
$cumulative$	Logical value that determines the form of the function; if TRUE, then this function returns the cumulative distribution function, which is the probability that there are at most x successes; if FALSE, it returns the probability mass function, which is the probability that there are x successes

Remarks

Use this function in problems with a fixed number of tests or trials, when there are two mutually exclusive possible outcomes (a "success" and a "failure"), when trials are independent, and when the probability of one outcome is constant throughout the experiment. This function can, for example, calculate the probability that two of the next three babies born are male.

The binomial probability mass function is calculated as follows:

$$BINOMDIST(x, n, p, FALSE) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}$$

where x is the number of successes, n is the number of trials, and p is the probability of success on any one trial. The cumulative binomial distribution is calculated as follows:

$$BINOMDIST(x, n, p, TRUE) = \sum_{y=x}^n BINOMDIST(y, n, p, FALSE)$$

where n is the number of trials, x is the number of successes, and p is the possibility of success on any one trial.

Data Types

Accepts numeric data for all arguments, except cumulative, which accepts logical data. Returns numeric data.

Example

A baby can be either male or female; for the sake of this example, assume the odds are 50/50 that a baby is either male or female. If female equals TRUE, we can use the following to determine the probability of the next 5 babies in 10 born being female. The probability of the first baby being female is 0.5, and the probability of exactly 5 of 10 babies born being female is:

`BINOMDIST(5,10,0.5,FALSE)` gives the result 0.2460937500

Version Available

This function is available in product version 1.0 or later.

See Also

BETADIST | CRITBINOM | EXPONDIST | GAMMADIST | NEGBINOMDIST | WEIBULL | Statistical Functions

BITAND

This function calculates bitwise AND of the two specified values.

Syntax

BITAND(*value1*, *value2*)

Arguments

For the arguments of this function, both *values* must be in decimal form and greater than or equal to 0.

Remarks

In this function, if the values of both parameter's bits at same position is 1 only then the value of each bit position is counted.

Data Types

Accepts data in decimal form. Returns data in decimal form.

Examples

BITAND(3,6) gives the result 2.

BITAND(4,5) gives the result 4.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

BITLSHIFT

This function calculates the value shifted left by the specified number of bits.

Syntax

BITLSHIFT(*value*, *shift_amt*)

Arguments

This function has the following arguments:

Argument	Description
<i>value</i>	Refers to a value that you want to shift.
<i>shift_amt</i>	Refers to a number of bits to be shifted by.

Remarks

Shifting a number left is similar to adding zeros (0) at right end of the binary representation of that number. If *shift_amt* is a negative number, then the *value* will be shifted to right.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

BITLSHIFT(3,5) gives the result 96.

BITLSHIFT(1,4) gives the result 16.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

BITOR

This function calculates the bitwise OR of two specified numbers.

Syntax

`BITOR(value1, value2)`

Arguments

For the arguments of this function, both *values* must be in decimal form and greater than or equal to 0.

Remarks

In this function, if value of any parameter's bits at same position is 1 only then the value of each bit position is counted.

Data Types

Accepts decimal form of data. Returns decimal form of data.

Examples

`BITOR(22,4)` gives the result 22.

`BITOR(6,3)` gives the result 7.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

BITRSHIFT

This function calculates the value shifted right by specified number of bits.

Syntax

`BITRSHIFT(value, shift_amt)`

Arguments

This function has the following arguments:

Argument	Description
<i>value</i>	Refers to the value that you want to shift.
<i>shift_amt</i>	Refers to the number of bits to be shifted by.

Remarks

Shifting a number right is similar to removing digits from right end of the binary representation of the number. If *shift_amt* is a negative number, then *value* will be shifted to left.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`BITRSHIFT(5,2)` gives the result 1.

`BITRSHIFT(15,2)` gives the result 3.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

BITXOR

This function returns bitwise XOR of the two specified numbers.

Syntax

`BITXOR(value1, value2)`

Arguments

For the arguments of this function, both *values* must be in decimal form and greater than or equal to 0.

Remarks

In this function, if value of any parameter's bits at same position is not equal only then the value of each bit position is counted.

Data Types

Accepts decimal form of data. Returns decimal form of data.

Examples

`BITXOR(6,2)` gives the result 4.

`BITXOR(8,3)` gives the result 11.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

CALL

This function calls a procedure/function defined in a DLL to the worksheet cell.

Syntax

CALL(*register_id*, [*argument1*],...)

CALL(*module_text*, *procedure*, *type_text*, [*argument1*],...)

Arguments

This function has the following arguments:

Argument	Description
<i>register_id</i>	Refers to the data specifying the returned value after executing REGISTER or REGISTER.ID function.
<i>module_text</i>	Refers to the string data specifying the name of the DLL that contains the function.
<i>procedure</i>	Refers to the string data specifying the name of the function to be called.
<i>type_text</i>	Refers to the data specifying the data types of all arguments and of the return value.
<i>argument1</i> ,...	[Optional] Refers to the data specifying different arguments to be passed to <i>procedure</i> .

Remarks

If this function is used incorrectly, it may cause the computer to restart.

Data Types

Accepts string data.

Examples

CALL("D:\sum2.dll","Sum2","J")

CALL(B10) where B10 is the cell reference containing the Register function.

Version Available

This function is available in product version 11.0 or later.

CEILING

This function rounds a number up to the nearest multiple of a specified value.

Syntax

`CEILING(value,signif)`

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Number to round
<i>signif</i>	Number representing the rounding factor

Remarks

Use either both positive or both negative numbers for the arguments. Regardless of the sign of the numbers, the value is rounded away from zero.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`CEILING(C4,B2)`

`CEILING(B3,0.05)`

`CEILING(R4C3,1)`

`CEILING(4.65,2)` gives the result 6

`CEILING(-2.78,-1)` gives the result -3

Version Available

This function is available in product version 1.0 or later.

See Also

FLOOR | EVEN | ODD | TRUNC | Math and Trigonometry Functions

CEILING.MATH

This function rounds a number up to the nearest multiple of a specified value or the nearest integer.

Syntax

CEILING.PRECISE(*value*,*signif*)

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Number to round
<i>signif</i>	Number representing the rounding factor

Use either both positive or both negative numbers for the arguments. Regardless of the sign of the numbers, the value is rounded away from zero.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

CEILING.PRECISE(C4,B2)

CEILING.PRECISE(B3,0.05)

CEILING.PRECISE(R4C3,1)

CEILING.PRECISE(4.65,2) gives the result 6

CEILING.PRECISE(-2.78,-1) gives the result -3

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

CEILING.PRECISE

This function rounds a number up to the nearest multiple of a specified value or the nearest integer.

Syntax

`CEILING.PRECISE(value,signif)`

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Number to round
<i>signif</i>	Number representing the rounding factor

Remarks

Use either both positive or both negative numbers for the arguments. Regardless of the sign of the numbers, the value is rounded away from zero.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`CEILING.PRECISE(C4,B2)`

`CEILING.PRECISE(B3,0.05)`

`CEILING.PRECISE(R4C3,1)`

`CEILING.PRECISE(4.65,2)` gives the result 6

`CEILING.PRECISE(-2.78,-1)` gives the result -3

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

CELL

This function evaluates information about the formatting, location, or contents of a cell.

Syntax

CELL(*info*, *reference*)

Arguments

This function has the following arguments:

Argument Description

- info* Refers to a text value that specifies what type of cell information to return. This argument is required.
- reference* Refers to the cell to get information about. This argument is optional.

The *info* argument has the following options.

Option Returns

- address* The first cell reference, as text.
- col* The column number of the cell.
- color* The value 1 if the cell is formatted in color for negative values; otherwise, returns 0 (zero).
- contents* The value of the upper-left cell in the reference.
- filename* A filename (including full path) of the file that contains the reference.
- format* A text value that corresponds to the number format of the cell (as shown in the following table).
- parentheses* The value 1 if the cell is formatted with parentheses for positive or all values; otherwise, returns 0.
- prefix* A text value that corresponds to the label prefix for the cell. If the cell contains left-aligned text, a single quotation mark (') is returned. If the cell contains right-aligned text, a double quotation mark (") is returned. If the text is centered, a caret (^) is returned. A backslash (\) is returned if the text is fill aligned and empty text is returned if the cell contains anything else.
- protect* The value 0 is returned if the cell is not locked and 1 is returned if the cell is locked.
- row* The row number of the cell.
- type* A text value that corresponds to the type of data in the cell ("b" if empty, "i" if the cell contains a text constant, and "v" if the cell contains anything else).
- width* The column width of the cell.

The following items are returned when using the *format* option.

Format

- General
- 0
- #,##0
- 0.00
- #,##0.00
- #,##0_);(,\$#,##0)

Returns

- "G"
- "Fo"
- ",,0"
- "F2"
- ",,2"
- "Co"

\$#,##0_);[Red](\$#,##0)	"C0"
\$#,##0.00_);(\$#,##0.00)	"C2"
\$#,##0.00_);[Red](\$#,##0.00)	"C2-"
o%	"P0"
o.00%	"P2"
o.00E+00	"S2"
# ??/? or # ??/??	"G"
m/d/yy or m/d/yy h:mm or mm/dd/yy	"D4"
d-mmm-yy or dd-mmm-yy	"D1"
d-mmm or dd-mmm	"D2"
mmm-yy	"D3"
mm/dd	"D5"
h:mm AM/PM	"D7"
h:mm:ss AM/PM	"D6"
h:mm	"D9"
h:mm:ss	"D8"

Remarks

If the *reference* argument is a range of cells, this function only returns information for the upper left cell of the range.

Data Types

Accepts string or numeric data. Returns numeric or string data.

Examples

CELL("row", A10) gives the result 10 (10 is the number of cell A10).

CELL("type", B5) returns the data type of the information contained in the cell B5.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

CHAR

This function returns the character specified by a number.

Syntax

`CHAR(value)`

Arguments

For the argument, use a number between 1 and 255 specifying which character you want from the Windows character set (ANSI).

Data Types

Accepts numeric data. Returns string data.

Examples

`CHAR(B2)`

`CHAR(R2C2)`

`CHAR(66)` gives the result B

`CHAR(218)` gives the result Ú

Version Available

This function is available in product version 1.0 or later.

See Also

CODE | **CONCATENATE** | **LOWER** | **PROPER** | **UPPER** | **Text Functions**

CHIDIST

This function calculates the one-tailed probability of the chi-squared distribution.

Syntax

`CHIDIST(value,deg)`

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Value at which to evaluate the function
<i>deg</i>	Number of degrees of freedom; if not an integer, the number is truncated

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`CHIDIST(B5,D7)`

`CHIDIST(R5C2,R7C4)`

`CHIDIST(6.7,4)` gives the result 0.1526169403

Version Available

This function is available in product version 1.0 or later.

See Also

CHIINV | CHITEST | Statistical Functions

CHIINV

This function calculates the inverse of the one-tailed probability of the chi-squared distribution.

Syntax

`CHIINV(prob,deg)`

Arguments

This function has these arguments:

Argument	Description
<i>prob</i>	Probability of the chi-squared distribution
<i>deg</i>	Number of degrees of freedom; if not an integer, the number is truncated

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`CHIINV(B5,D7)`

`CHIINV(R5C2,R7C4)`

`CHIINV(0.75,4)` gives the result 1.9225575262

Version Available

This function is available in product version 1.0 or later.

See Also

CHIDIST | **CHITEST** | **Statistical Functions**

CHISQ.DIST

This function calculates the chi-squared distribution.

Syntax

CHISQ.DIST(*value,deg,cumulative*)

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>value</i>	Value at which to evaluate the function
--------------	---

<i>deg</i>	Number of degrees of freedom; if not an integer, the number is truncated
------------	--

<i>cumulative</i>	A logical value that determines the form of the function. If <i>cumulative</i> is TRUE, this function returns the cumulative distribution function; if FALSE, it returns the probability density function
-------------------	---

Data Types

Accepts numeric data for *value* and *deg* arguments. Returns numeric data.

Examples

CHISQ.DIST(B5,D7,TRUE)

CHISQ.DIST(R5C2,R7C4,TRUE)

CHISQ.DIST(6.7,4,TRUE) gives the result 0.8473830596613241

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

CHISQ.TEST

CHISQ.DIST.RT

This function calculates the right-tailed probability of the chi-squared distribution.

Syntax

CHISQ.DIST.RT(*value,deg*)

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Value at which to evaluate the function
<i>deg</i>	Number of degrees of freedom; if not an integer, the number is truncated

Remarks

The #VALUE! error value is returned if either argument is nonnumeric.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

CHISQ.DIST.RT(B5,D7)

CHISQ.DIST.RT(R5C2,R7C4)

CHISQ.DIST.RT(6.7,4) gives the result 0.15261694033867584

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

CHISQ.DIST

CHISQ.INV

This function calculates the inverse of the left-tailed probability of the chi-squared distribution.

Syntax

`CHISQ.INV(prob,deg)`

Arguments

This function has these arguments:

Argument	Description
<i>prob</i>	Probability of the chi-squared distribution
<i>deg</i>	Number of degrees of freedom; if not an integer, the number is truncated

Remarks

If the argument is nonnumeric, the function returns the #VALUE! error value.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`CHISQ.INV(B5,D7)`

`CHISQ.INV(R5C2,R7C4)`

`CHISQ.INV(0.75,4)` gives the result 5.385269057779394

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

CHISQ.DIST

CHISQ.INV.RT

This function calculates the inverse of the right-tailed probability of the chi-squared distribution.

Syntax

`CHISQ.INV.RT(prob,deg)`

Arguments

This function has these arguments:

Argument	Description
<i>prob</i>	Probability of the chi-squared distribution
<i>deg</i>	Number of degrees of freedom; if not an integer, the number is truncated

Remarks

The #VALUE! error value is returned if either argument is nonnumeric.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`CHISQ.INV.RT(B5,D7)`

`CHISQ.INV.RT(R5C2,R7C4)`

`CHISQ.INV.RT(0.75,4)` gives the result 1.9225575262293264

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

CHISQ.DIST

CHISQ.TEST

This function calculates the test for independence from the chi-squared distribution.

Syntax

`CHISQ.TEST(obs_array,exp_array)`

Arguments

This function has these arguments:

Argument	Description
<i>obs_array</i>	Array of observed values to test against expected values
<i>exp_array</i>	Array of expected values against which to test observed values

Remarks

The arrays in the arguments must be of the same size.

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

`CHISQ.TEST(B1:C8,B12:C19)`

`CHISQ.TEST(R1C2:R8C3,R12C2:R19C3)`

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

CHISQ.DIST

CHITEST

This function calculates the test for independence from the chi-squared distribution.

Syntax

`CHITEST(obs_array,exp_array)`

Arguments

This function has these arguments:

Argument	Description
<i>obs_array</i>	Array of observed values to test against expected values
<i>exp_array</i>	Array of expected values against which to test observed values

The arrays in the arguments must be of the same size.

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

`CHITEST(B1:C8,B12:C19)`

`CHITEST(R1C2:R8C3,R12C2:R19C3)`

Version Available

This function is available in product version 1.0 or later.

See Also

CHIDIST | **CHIINV** | **AVERAGE** | **Statistical Functions**

CHOOSE

This function returns a value from a list of values.

Syntax

`CHOOSE(index,value1,value2,...)`

Arguments

This function has these arguments:

Argument	Description
<i>index</i>	Index of the specified values to return; an integer value between 1 and 255
<i>value1</i> , etc.	Values from which to choose; can have up to 255 values; can be numbers, cell references, cell ranges, defined names, formulas, functions, or text

The value arguments can be range references as well as single values. For example, the formula:

```
SUM(CHOOSE(2,A1:A25,B1:B10,C1:C5))
```

evaluates to:

```
SUM(B1:B10)
```

which then returns a value based on the values in the range B1:B10.

Remarks

This function is evaluated first, returning the reference B1:B10. The **SUM** function is then evaluated using B1:B10.

Data Types

The index argument accepts numeric data. The value arguments accept any data. Returns the type of data of the specified value.

Examples

```
CHOOSE(3,A1,B1,C1,D1,E1) gives the result C1
```

```
CHOOSE(3,R1C1,R1C2,R1C3,R1C4,R1C5) gives the result R1C3
```

```
CHOOSE(2,"dogs","birds","fish","cats","mice") gives the result birds
```

Version Available

This function is available in product version 1.0 or later.

See Also

INDEX | **SUM** | **Lookup Functions**

CLEAN

This function removes all non-printable characters from text.

Syntax

CLEAN(*text*)

Arguments

The text argument is any data from which you want to remove non-printable characters.

Remarks

Use this function to remove text that contains characters that might not print with your operating system. For example, you can use this function to remove some low-level computer code, which is frequently at the beginning and end of data files and cannot be printed

Data Types

Accepts string data. Returns string data.

Example

In this example, CHR(7) returns a non-printable character
CLEAN(CHAR(7)&"text"&CHAR(7)) gives the result text

Version Available

This function is available in product version 1.0 or later.

See Also

TRIM | **SUBSTITUTE** | **Text Functions**

CODE

This function returns a numeric code to represent the first character in a text string. The returned code corresponds to the Windows character set (ANSI).

Syntax

`CODE(text)`

Arguments

The argument is the text from which you want to determine the code of the first character.

Data Types

Accepts string data. Returns string data.

Examples

`CODE(H6)`

`CODE(R6C8)`

`CODE("B")` gives the result 66

`CODE("Buffalo")` gives the result 66

Version Available

This function is available in product version 1.0 or later.

See Also

CHAR | Text Functions

COLUMN

This function returns the column number of a reference.

Syntax

`COLUMN(reference)`

Arguments

The argument is a cell or a single area.

Remarks

If the reference is omitted, the reference of the cell that the function is in is used.

Data Types

Accepts cell references. Returns numeric data.

Examples

`COLUMN(A9)` gives the result 1

`COLUMN(A1:A5)` gives the result 1

Version Available

This function is available in product version 3.0 or later.

See Also

ROWS | **INDEX** | **Lookup Functions**

COLUMNS

This function returns the number of columns in an array.

Syntax

`COLUMNS(array)`

Arguments

The argument is an array, an array formula, or a range of cells.

Data Types

Accepts cell references or array. Returns numeric data.

Examples

`COLUMNS(B6:D12)` gives the result 3

`COLUMNS(R6C2:R12C4)` gives the result 3

`COLUMNS(B8:H8)` gives the result 7

`COLUMNS(R[2]C[1]:R[3]C[8])` gives the result 8

Version Available

This function is available in product version 1.0 or later.

See Also

ROWS | **INDEX** | **Lookup Functions**

COMBIN

This function calculates the number of possible combinations for a specified number of items.

Syntax

`COMBIN(k,n)`

Arguments

This function has these arguments:

Argument Description

<i>k</i>	Number representing the number of items; if not an integer, the number is truncated; must be positive and greater than or equal to <i>n</i>
<i>n</i>	Number of items in each possible permutation; if not an integer, the number is truncated; must be positive

Remarks

A combination is any set or subset of items, regardless of the internal order of the items. Contrast with permutation (the **PERMUT** function).

The number of combinations is calculated as follows:

$$COMBIN(k, n) = \binom{n}{k} = \frac{PERMUT(k, n)}{k!} = \frac{n!}{k!(n-k)!}$$

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`COMBIN(C4, B2)`

`COMBIN(B3, 5)`

`COMBIN(R1C2, 2)`

`COMBIN(8, 2)` gives the result 28

`COMBIN(100, 3)` gives the result 161700

Version Available

This function is available in product version 1.0 or later.

See Also

PERMUT | **Math and Trigonometry Functions**

COMBINA

This function calculates the number of possible combinations (along with repetitions) for the specified number of items.

Syntax

COMBINA(*Value*, *Value_chosen*)

Arguments

This function has the following arguments:

Argument	Description
<i>Value</i>	Refers to the number representing the number of items. This value must be positive and greater than or equal to the second argument i.e. <i>Value_chosen</i> .
<i>Value_chosen</i>	Refers to the number of items in each possible combination. This value must be positive.

Remarks

The following equation is used to calculate the COMBINA function:

$$(Value + Value_chosen - 1) / (Value - 1)$$

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

COMBINA(5,4) gives the result 70.

COMBINA(6,5) gives the result 252.

COMBINA(11,3) gives the result 286.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

COMPLEX

This function converts real and imaginary coefficients into a complex number.

Syntax

`COMPLEX(realcoeff,imagcoeff,suffix)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>realcoeff</i>	Coefficient of the real part of the complex number
------------------	--

<i>imagcoeff</i>	Coefficient of the imaginary part of the complex number
------------------	---

<i>suffix</i>	(Optional) Suffix of the imaginary part of the complex number, may be either "i" or "j". If omitted, "i" is used.
---------------	---

Remarks

For the suffix, use lowercase for "i" and "j" to prevent errors.

An error is returned if the real or imaginary coefficients are non-numeric.

For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

`COMPLEX(3, 5)`

`COMPLEX(3, 5, "j")`

Version Available

This function is available in product version 2.0 or later.

See Also

IMAGINARY | **IMREAL** | **Engineering Functions** | **Complex Numbers in Engineering Functions**

CONCAT

This function joins the text fetched from multiple strings.

Syntax

CONCAT(*textvalue1*, *textvalue2*,...)

Arguments

This function has the following arguments:

- | | |
|-------------------|--|
| <i>textvalue1</i> | Refers to a text string, or array of strings, to be joined. |
| <i>textvalue2</i> | [Optional] Refers to the additional text strings to be joined. |

Remarks

There can be a maximum of 253 text items in arguments (including *textvalue1*).

In the result, this function never includes the delimiter (spaces, ampersands etc.) between each text value and never eliminates the empty arguments.

Data Types

Accepts string data. Returns string data.

Examples

CONCAT(Riot,"",Pauler) gives the result Riot Pauler.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

CONCATENATE

This function combines multiple text strings or numbers into one text string.

Syntax

```
CONCATENATE(text1,text2,...)
```

Arguments

The arguments can be strings, formulas that return a string, or references to cells containing a string. Up to 255 arguments may be included.

Data Types

Accepts string data for both arguments. Returns string data.

Examples

```
CONCATENATE (B4, D5)
```

```
CONCATENATE (R4C2, R5C4)
```

```
CONCATENATE ("Gold ", "Medal") gives the result Gold Medal
```

Version Available

This function is available in product version 1.0 or later.

See Also

CHAR | **EXACT** | **Text Functions**

CONFIDENCE

This function returns the confidence interval for a population mean.

Syntax

`CONFIDENCE(alpha,stdev,size)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>alpha</i>	Alpha, significance level used in calculating confidence level, where confidence level is 100 times (1-alpha)%
--------------	--

<i>stdev</i>	Population standard deviation for the range
--------------	---

<i>size</i>	Number representing the size of the sample; if not an integer, the number is truncated
-------------	--

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`CONFIDENCE(0.5,B4,D5)`

`CONFIDENCE(0.5,R4C2,R5C4)`

`CONFIDENCE(0.05,3.5,150)` gives the result 0.560106363

Version Available

This function is available in product version 1.0 or later.

See Also

[AVERAGE](#) | [CHITEST](#) | [Statistical Functions](#)

CONFIDENCE.NORM

This function returns the confidence interval for a population mean.

Syntax

`CONFIDENCE.NORM(alpha,stdev,size)`

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>alpha</i>	Alpha, significance level used in calculating confidence level, where confidence level is 100 times (1- <i>alpha</i>)%
<i>stdev</i>	Population standard deviation for the range
<i>size</i>	Number representing the size of the sample; if not an integer, the number is truncated

Remarks

The #VALUE! error value is returned if any argument is nonnumeric. The #NUM! error value is returned if size < 1.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`CONFIDENCE.NORM(0.5,B4,D5)`

`CONFIDENCE.NORM(0.5,R4C2,R5C4)`

`CONFIDENCE.NORM(0.05,3.5,150)` gives the result 0.5601063629983405

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

CONFIDENCE

CONFIDENCE.T

This function returns the confidence interval for a population mean.

Syntax

`CONFIDENCE.T(alpha,stdev,size)`

Arguments

This function has these arguments:

Argument	Description
<i>alpha</i>	Alpha, significance level used in calculating confidence level, where confidence level is 100 times (1- <i>alpha</i>)%
<i>stdev</i>	Population standard deviation for the range
<i>size</i>	Number representing the size of the sample; if not an integer, the number is truncated

Remarks

The function uses a Student's t distribution. If *size* = 1, the function returns a #DIV/O! error value. If any argument is nonnumeric, the function returns the #VALUE! error value.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`CONFIDENCE.T(0.5,B4,D5)`

`CONFIDENCE.T(0.5,R4C2,R5C4)`

`CONFIDENCE.T(0.05,3.5,150)` gives the result 0.5646928012079743

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

CONFIDENCE

CONVERT

This function converts a number from one measurement system to its equivalent in another measurement system.

Syntax

CONVERT(*number,from-unit,to-unit*)

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Numeric value to convert
<i>from-unit</i>	Convertible units (see table below) of numeric value to convert
<i>to-unit</i>	Convertible units (see table below) of desired result

Remarks

In this context a measurement system is a set of units for different types of measurements. This function converts a number with one set of units to a number in different set of units.

An error value is returned if the convertible units (*from-unit* and *to-unit*) are invalid or are from different categories of unit types (different tables below).

The following tables list the convertible units by their unit type:

Weight and Mass Unit Type

Gram
 Slug
 Pound Mass
 U
 Ounce Mass

Convertible Units

"g"
 "sg"
 "lbm"
 "u"
 "ozm"

Distance Unit Type

Meter
 Statute mile
 Nautical mile
 Inch
 Foot
 Yard
 Angstrom
 Pica (1/72 in.)

Convertible Units

"m"
 "mi"
 "Nmi"
 "in"
 "ft"
 "yd"
 "ang"
 "Pica"

Time Unit Type

Year

Convertible Units

"yr"

<i>Day</i>	"day"
<i>Hour</i>	"hr"
Minute	"mn"
Second	"sec"

Pressure Unit Type

Pascal	"Pa"
<i>Atmosphere</i>	"atm"
<i>mm of Mercury</i>	"mmHg"

Force Unit Type

Newton	"N"
<i>Dyne</i>	"dyn"
Pound force	"lbf"

Energy Unit Type

Joule	"J"
<i>Erg</i>	"e"
Thermodynamic calorie	"c"
IT calorie	"cal"
Electron volt	"eV"
Horsepower-hour	"Hph"
Watt-hour	"Wh"
Foot-pound	"flb"
BTU	"BTU"

Power Unit Type

Horsepower	"HP"
<i>Watt</i>	"W"

Magnetism Unit Type

Tesla	"T"
<i>Gauss</i>	"ga"

Temperature Unit Type

Degree Celsius	"C"
<i>Degree Fahrenheit</i>	"F"
Degree Kelvin	"K"

Liquid Measure Unit Type

Teaspoon	"tsp"
----------	-------

Convertible Units

"Pa"
"atm"
"mmHg"

Convertible Units

"N"
"dyn"
"lbf"

Convertible Units

"J"
"e"
"c"
"cal"
"eV"
"Hph"
"Wh"
"flb"
"BTU"

Convertible Units

"HP"
"W"

Convertible Units

"T"
"ga"

Convertible Units

"C"
"F"
"K"

Convertible Units

"tsp"

<i>Tablespoon</i>	"tbs"
Fluid ounce	"oz"
Cup	"cup"
U.S. pint	"pt"
U.K. pint	"uk_pt"
Quart	"qt"
Gallon	"gal"
Liter	"l"

Data Types

Accepts numeric and string data. Returns numeric data.

Examples

```
CONVERT (68, "F", "C")
```

Version Available

This function is available in product version 2.0 or later.

See Also

OCT2BIN | **HEX2DEC** | **DEC2OCT** | **Engineering Functions**

CORREL

This function returns the correlation coefficient of the two sets of data.

Syntax

`CORREL(array1,array2)`

Arguments

The two arrays of data in the arguments of this function should meet these criteria:

- The data should contain numbers, names, ranges, or references that are numeric. If some cells do not contain numeric data, they are ignored.
- The arrays should be the same size, with the same number of data points.
- The arrays should not be empty, nor should the standard deviation of their values equal zero.

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

`CORREL(C1:C10,D1:D10)`

`CORREL(R1C3:R10C3,R1C4:R10C4)`

`CORREL({5,10,15,20,25},{4,8,16,32,64})` gives the result 0.9332565253

`CORREL({73000,45000,40360},{42,70,40})` gives the result -0.3261046660

Version Available

This function is available in product version 1.0 or later.

See Also

COVAR | **Statistical Functions**

COS

This function returns the cosine of the specified angle.

Syntax

`COS(angle)`

Arguments

This function can take any real number as an argument. The *angle* argument is the angle in radians for which you want the cosine.

Remarks

If the angle is in degrees, multiply it by $\text{PI}/180$ to convert it to radians.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`COS(B2)`

`COS(R1C3)`

`COS(45*PI()/180)` gives the result 0.7071067812

`COS(RADIANS(30))`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOS | ACOSH | COSH | Math and Trigonometry Functions

COSH

This function returns the hyperbolic cosine of the specified value.

Syntax

`COSH(value)`

Arguments

This function can take any real number as an argument.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`COSH(B3)`

`COSH(R1C2)`

`COSH(4)` gives the result 27.3082328360

Version Available

This function is available in product version 1.0 or later.

See Also

[ACOSH](#) | [COS](#) | [Math and Trigonometry Functions](#)

COT

This function returns the cotangent of the specified angle.

Syntax

$COT(\textit{angle})$

Arguments

This function can take any real number as an argument. The *angle* argument is the angle in radians for which you want the cotangent.

Remarks

If the *angle* is in degrees, multiply it by $\text{PI}/180$ to convert it to radians.

Data Types

Accepts numeric data. Returns numeric data.

Examples

$COT(60)$ gives the result 3.1246

$COT(35)$ gives the result 2.1105

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

COTH

This function returns the hyperbolic cotangent of the specified angle.

Syntax

$\text{COTH}(\text{angle})$

Arguments

This function can take any real number as an argument.

Remarks

The absolute value of *angle* must be less than 2^{27} .

Data Types

Accepts numeric data. Returns numeric data.

Examples

$\text{COTH}(45)$ gives the result 1.

$\text{COTH}(90)$ gives the result 1.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

COUNT

This function returns the number of cells that contain numbers.

Syntax

`COUNT(value1,value2,...)`

`COUNT(array)`

Arguments

The arguments may be separate values or an array of values. Up to 255 arguments of individual cells may be included.

Remarks

This function counts the number of cells that contain numbers in the specified cell range.

This function differs from **COUNTA** which also includes text or logical values as well as numbers.

Data Types

Accepts cell references. Returns numeric data.

Examples

`COUNT(B2, B5, B8, D5, D8)`

`COUNT(A1:G5)`

`COUNT(R6C3:R9C4, 2)`

Version Available

This function is available in product version 2.0 or later.

See Also

COUNTA | **Statistical Functions**

COUNTA

This function returns the number of number of cells that contain numbers, text, or logical values.

Syntax

`COUNTA(value1,value2,...)`

`COUNTA(array)`

Arguments

The arguments may be separate values or an array of values. Up to 255 arguments of individual cells may be included.

Remarks

This function counts the number of non-empty cells in the specified cell range.

This function differs from **COUNT** because it includes text or logical values as well as numbers.

Data Types

Accepts cell references. Returns numeric data.

Examples

`COUNTA(B2, D2, E4, E5, E6)`

`COUNTA(A1:G5)`

`COUNTA(R6C3:R9C4)`

Version Available

This function is available in product version 2.0 or later.

See Also

COUNT | **Statistical Functions**

COUNTBLANK

This function returns the number of empty (or blank) cells in a range of cells on a sheet.

Syntax

COUNTBLANK(*cellrange*)

Arguments

This function takes a cell range reference or array as an argument.

Remarks

This function counts the number of empty or blank cells in the specified cell range on one sheet. This function does not count cells containing an empty string "". A cell is empty if the cell's Value is null (Nothing in VB). Note that there is a difference being a cell's Value being null and a cell's Value being the empty string "". For example, consider the following Spread code in C#:

```
spread.Sheets[0].Cells[0,0].Value = null; // empty
spread.Sheets[0].Cells[1,0].Value = ""; // string
spread.Sheets[0].Cells[2,0].Value = "abc"; // string
spread.Sheets[0].Cells[3,0].Value = 123.0; // number
spread.Sheets[0].Cells[4,0].Formula = "COUNTBLANK(A1:A4)";
```

The formula in cell A5 evaluates to 1 because cell A1 is the only cell in the range A1:A4 that is empty.

Note: Spread's implementation of functions generally tries to follow the behavior found in popular spreadsheet applications. However, not all these applications agree whether the empty string "" should be treated the same as an empty cell. In Spread, both the COUNTBLANK and **ISBLANK** functions consistently treat the empty string "" differently than an empty cell.

Data Types

Accepts cell range reference. Returns numeric data.

Examples

```
COUNTBLANK (A1 : G5)
COUNTBLANK (R6C3 : R9C4)
```

Version Available

This function is available in product version 1.0 or later.

See Also

COUNTIF | ISBLANK | TYPE | **Information Functions**

COUNTIF

This function returns the number of cells that meet a certain condition.

Syntax

COUNTIF(*cellrange*,*condition*)

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>cellrange</i>	Range of cells to count; cell range reference
------------------	---

<i>condition</i>	Condition that determines which cells are counted, as a text, number, or expression (where expressions use the relational operators detailed in Operators in a Formula)
------------------	---

Data Types

Accepts cell range reference. Returns numeric data.

Examples

COUNTIF(A1:G5,"test")

COUNTIF(R6C3:R9C4,"<2")

Version Available

This function is available in product version 2.0 or later.

See Also

COUNT | COUNTA | COUNTBLANK | SUMIF | **Statistical Functions**

COUNTIFS

This function returns the number of cells that meet multiple conditions.

Syntax

COUNTIFS(*cellrange*,*condition*)

Arguments

This function has these arguments:

Argument	Description
<i>cellrange</i>	Range of cells to count; cell range reference
<i>condition</i>	Condition that determines which cells are counted, as a text, number, or expression (where expressions use the relational operators detailed in Operators in a Formula)

Data Types

Accepts cell range reference. Returns numeric data.

Examples

COUNTIFS (A1:G5, "test", B3:D3, "=Yes")

COUNTIFS (R6C3:R9C4, "<2")

Version Available

This function is available in product version 5.0 or later.

See Also

COUNT | COUNTA | COUNTBLANK | SUMIF | **Statistical Functions**

COUPDAYBS

This function calculates the number of days from the beginning of the coupon period to the settlement date.

Syntax

`COUPDAYBS(settlement,maturity,frequency,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>settlement</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>frequency</i>	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns an error if *settlement* or *maturity* is invalid (**#VALUE!**), or if *frequency* is a number other than 1, 2, or 4 (**#NUM!**). All arguments are truncated to integers. If *basis* is greater than 4 or less than 0, a **#NUM!** error is returned. If *settlement* is greater than or equal to *maturity*, a **#NUM!** error is returned.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`COUPDAYBS(A1,A2,A3,A4)`

Version Available

This function is available in product version 2.0 or later.

See Also

COUPDAYS | **Financial Functions**

COUPDAYS

This function returns the number of days in the coupon period that contains the settlement date.

Syntax

`COUPDAYS(settlement,maturity,frequency,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>settlement</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>frequency</i>	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns an error if *settlement* or *maturity* is invalid (#VALUE!), or if *frequency* is a number other than 1, 2, or 4 (#NUM!). All arguments are truncated to integers. If *basis* is greater than 4 or less than 0, a #NUM! error is returned. If *settlement* is greater than or equal to *maturity*, a #NUM! error is returned.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`COUPDAYS(A1,A2,A3,A4)`

Version Available

This function is available in product version 2.0 or later.

See Also

COUPDAYBS | **DURATION** | **Financial Functions**

COUPDAYSNC

This function calculates the number of days from the settlement date to the next coupon date.

Syntax

`COUPDAYSNC(settlement,maturity,frequency,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>settlement</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>frequency</i>	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns an error if *settlement* or *maturity* is invalid (#VALUE!), or if *frequency* is a number other than 1, 2, or 4 (#NUM!). All arguments are truncated to integers. If *basis* is greater than 4 or less than 0, a #NUM! error is returned. If settlement is greater than or equal to maturity, a #NUM! error is returned.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`COUPDAYSNC(A1,A2,A3,A4)`

Version Available

This function is available in product version 2.0 or later.

See Also

COUPDAYS | **COUPDAYBS** | **Financial Functions**

COUPNCD

This function returns a date number of the next coupon date after the settlement date.

Syntax

COUPNCD(*settlement*,*maturity*,*frequency*,*basis*)

Arguments

This function has these arguments:

Argument	Description
<i>settlement</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>frequency</i>	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns an error if *settlement* or *maturity* is invalid (#VALUE!), or if *frequency* is a number other than 1, 2, or 4 (#NUM!). All arguments are truncated to integers. If *basis* is greater than 4 or less than 0, a #NUM! error is returned. If *settlement* is greater than or equal to *maturity*, a #NUM! error is returned.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

COUPNCD (A1, A2, A3, A4)

COUPNCD (A1, A2, A3, A4)

Version Available

This function is available in product version 2.0 or later.

See Also

COUPPCD | **Financial Functions**

COUPNUM

This function returns the number of coupons due between the settlement date and maturity date.

Syntax

`COUPNUM(settlement,maturity,frequency,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>settlement</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>frequency</i>	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns an error if *settlement* or *maturity* is invalid (**#VALUE!**), or if *frequency* is a number other than 1, 2, or 4 (**#NUM!**). All arguments are truncated to integers. If *basis* is greater than 4 or less than 0, a **#NUM!** error is returned. If *settlement* is greater than or equal to *maturity*, a **#NUM!** error is returned.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`COUPNUM(A1,A2,A3,A4)`

`COUPNUM(R6C3:R9C4)`

Version Available

This function is available in product version 2.0 or later.

See Also

COUPDAYS | **Financial Functions**

COUPPCD

This function returns a date number of the previous coupon date before the settlement date.

Syntax

COUPPCD(*settlement,maturity,frequency,basis*)

Arguments

This function has these arguments:

Argument	Description
<i>settlement</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>frequency</i>	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns an error if *settlement* or *maturity* is invalid (#VALUE!), or if *frequency* is a number other than 1, 2, or 4 (#NUM!). All arguments are truncated to integers. If *basis* is greater than 4 or less than 0, a #NUM! error is returned. If *settlement* is greater than or equal to *maturity*, a #NUM! error is returned.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

COUPPCD(B1,B2,B3,B4)

COUPPCD(R6C3,R9C4,R1C1,R2C2)

Version Available

This function is available in product version 2.0 or later.

See Also

COUPNCD | **Financial Functions**

COVAR

This function returns the covariance, which is the average of the products of deviations for each data point pair in two sets of numbers.

Syntax

`COVAR(array1,array2)`

Arguments

The two arrays of data in the arguments of this function should meet these criteria:

- The data should contain numbers, names, arrays, or references that are numeric. If some cells do not contain numeric data, they are ignored.
- The data sets should be the same size, with the same number of data points.
- The data sets should not be empty, nor should the standard deviation of their values equal zero.

Remarks

Use this covariance function to determine the relationship between two sets of data. For example, you can examine whether greater income accompanies greater levels of education in a population.

The covariance is calculated as follows, where n is the size of the arrays and μ is the mean.

$$COVAR(X, Y) = \frac{\left(\sum_1^n (x - \mu_x)(y - \mu_y) \right)}{(n - 1)}$$

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

`COVAR(J2:J5,L2:L5)`

`COVAR(R2C12:R15C12,R2C14:R15C14)`

`COVAR({7,5,6},{7,4,4})` gives the result 1

`COVAR({5,10,15,20,25},{4,8,16,32,64})` gives the result 144

Version Available

This function is available in product version 1.0 or later.

See Also

CORREL | **VAR** | **Statistical Functions**

COVARIANCE.P

Summary

This function returns the population covariance, which is the average of the products of deviations for each data point pair in two sets of numbers.

Syntax

COVARIANCE.P(*array1,array2*)

Arguments

The two arrays of data in the arguments of this function should meet these criteria:

- The data should contain numbers, names, arrays, or references that are numeric. If some cells do not contain numeric data, they are ignored.
- The data sets should be the same size, with the same number of data points.
- The data sets should not be empty, nor should the standard deviation of their values equal zero.

Remarks

Use this covariance function to determine the relationship between two sets of data. For example, you can determine whether greater income accompanies greater levels of education in a population.

The covariance is calculated as follows, where \bar{x} and \bar{y} are the sample means, AVERAGE(array1) and AVERAGE(array2), and n is the sample size.

$$COV(X, Y) = \frac{\sum(x - \bar{x})(y - \bar{y})}{n}$$

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

COVARIANCE.P(J2:J5,L2:L5)

COVARIANCE.P(R2C12:R15C12,R2C14:R15C14)

COVARIANCE.P({7,5,6},{7,4,4}) gives the result 1

COVARIANCE.P({5,10,15,20,25},{4,8,16,32,64}) gives the result 144

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

COVARIANCE.S

Summary

This function returns the sample covariance, which is the average of the products of deviations for each data point pair in two sets of numbers.

Syntax

`COVARIANCE.S(array1,array2)`

Arguments

The two arrays of data in the arguments of this function should meet these criteria:

- The data should contain numbers, names, arrays, or references that are numeric. If some cells do not contain numeric data, they are ignored.
- The data sets should be the same size, with the same number of data points.
- The data sets should not be empty, nor should the standard deviation of their values equal zero.

Remarks

Use this covariance function to determine the relationship between two sets of data. For example, you can determine whether greater income accompanies greater levels of education.

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

`COVARIANCE.S(J2:J5,L2:L5)`

`COVARIANCE.S(R2C12:R15C12,R2C14:R15C14)`

`COVARIANCE.S({7,5,6},{7,4,4})` gives the result 1.5

`COVARIANCE.S({5,10,15,20,25},{4,8,16,32,64})` gives the result 180

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

CRITBINOM

This function returns the criterion binomial, the smallest value for which the cumulative binomial distribution is greater than or equal to a criterion value.

Syntax

`CRITBINOM(n,p,alpha)`

Arguments

This function has these arguments:

Argument	Description
<i>n</i>	Number of trials; if not an integer, the number is truncated
<i>p</i>	Probability of success on each trial; number between 0 and 1
<i>alpha</i>	Alpha, value for the criterion

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`CRITBINOM(B5,0.75,0.92)`

`CRITBINOM(R5C2,R8C14,0.75)`

`CRITBINOM(14,0.75,0.85)` gives the result 12

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | Statistical Functions

CSC

This function returns the cosecant of the specified angle.

Syntax

$CSC(\textit{angle})$

Arguments

This function can take any real number as an argument. The *angle* argument is the angle in radians for which you want the cosecant.

Remarks

If the *angle* is in degrees, multiply it by $\text{PI}/180$ to convert it to radians.

Data Types

Accepts numeric data. Returns numeric data.

Examples

$CSC(40)$ gives the result 1.555

$CSC(15)$ gives the result 3.863

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

CSCH

CSCH

This function returns the hyperbolic cosecant of the specified angle.

Syntax

$\text{CSCH}(\text{angle})$

Arguments

This function can take any real number as an argument.

Remarks

The absolute value of *angle* must be less than 2^{27} .

Data Types

Accepts numeric data. Returns numeric data.

Examples

$\text{CSCH}(45)$ gives the result 1.414

$\text{CSCH}(2.5)$ gives the result 0.165

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

Functions D to G

Functions D to G

DATE	DATEDIF	DATEVALUE	DAVERAGE
DAY	DAYS	DAYS360	DB
DBCS	DCOUNT	DCOUNTA	DDB
DEC2BIN	DEC2HEX	DEC2OCT	DECIMAL
DEGREES	DELTA	DEVSQ	DGET
DISC	DMAX	DMIN	DOLLAR
DOLLARDE	DOLLARFR	DPRODUCT	DSTDEV
DSTDEVP	DSUM	DURATION	DVAR
DVARP	EDATE	EFFECT	ENCODEURL
EOMONTH	ERF	ERF.PRECISE	ERFC
ERFC.PRECISE	ERROR.TYPE	ERRORTYPE	EUROCONVERT
EVEN	EXACT	EXP	EXPON.DIST
EXPONDIST	F.DIST	F.DIST.RT	F.INV
F.INV.RT	F.TEST	FACT	FACTDOUBLE
FALSE	FDIST	FILTER	FILTERXML
FIND	FINDB	FINV	FISHER
FISHERINV	FIXED	FLOOR	FLOOR.MATH
FLOOR.PRECISE	FORECAST	FORECAST.LINEAR	FORMULATEXT
FREQUENCY	FTEST	FV	FVSCHEDULE
GAMMA	GAMMA.DIST	GAMMA.INV	GAMMADIST
GAMMAINV	GAMMALN	GAMMALN.PRECISE	GAUSS
GCD	GEOMEAN	GESTEP	GROWTH

DATE

This function returns the serial date value for a particular date, specified by the year, month, and day.

Syntax

`DATE(year,month,day)`

Arguments

This function has these arguments:

Argument	Description
<i>year</i>	Number representing the year, from 1 to 9999, using four digits; if not integer, number is truncated
<i>month</i>	Number representing the month of the year; if not integer, number is truncated
<i>day</i>	Number representing the day of the month; if not integer, number is truncated

If month is greater than 12, then month increments by the number of months over 12 and the year advances, if needed. For example, `DATE(2003,16,2)` returns the serial number 38079 representing April 2, 2004.

If day is greater than the number of days in the specified month, then day increments the number of days from the first day of the next month. For example, `DATE(2004,1,35)` returns the serial number 38021 representing February 4, 2004.

If values for the arguments are not integers, any decimal places are truncated. Negative values for months are taken from the year into previous years. Negative values for days are taken from the month into previous months.

Data Types

Accepts numeric data. Returns the serial number of the date.

Examples

`DATE(A1,B1,C1)`

`DATE(R1C1,R1C2,R1C3)`

`DATE(2019,1,1)` gives the result 43466 which can be formatted as 01-01-2019

`DATE(2019,2,13)` gives the result 43509 which can be formatted as 13-02-2019

Version Available

This function is available in product version 1.0 or later.



Note: If a user uses `LegacyBehaviors.CalculationEngine`, DATE function will return the `DateTime` object instead of the serial numeric value. For more details, please refer to [breaking changes for legacy behaviors](#).

See Also

[DATEVALUE](#) | [TIME](#) | [Date and Time Functions](#)

DATEDIF

This function returns the number of days, months, or years between two dates.

Syntax

`DATEDIF(date1,date2,outputcode)`

Arguments

The first two arguments are any dates, as strings, numeric values, or DateTime objects.

The output codes are:

Code	Description
-------------	--------------------

"D"	The number of days between date1 and date2
-----	--

"M"	The number of complete months between date1 and date2
-----	---

"Y"	The number of complete years between date1 and date2
-----	--

"YD"	The number of days between date1 and date2 as if they were in the same year
------	---

"YM"	The number of months between date1 and date2 as if they were in the same year
------	---

"MD"	The number of days between date1 and date2 as if they were in the same month and year
------	---

Data Types

Accepts strings, numeric values, and DateTime objects. Strings and numbers are converted to DateTime objects.

Examples

```
DATEDIF(A1,B1,C1)
```

```
DATEDIF(R1C1,R1C2,R1C3)
```

```
DATEDIF("2001/1/1","2003/1/1","Y")
```

Version Available

This function is available in product version 2.0 or later.

See Also

[DATEVALUE](#) | [TIME](#) | [Date and Time Functions](#)

DATEVALUE

This function returns the serial number of a date entered as text.

Syntax

DATEVALUE(*date_string*)

Arguments

The argument for this function is a date as a string.

Remarks

Use this function to convert a date represented by a text string to an Excel serial number that can be used to perform calculations.

Data Types

Accepts string data. Returns an excel serial number as date.

Examples

DATEVALUE (B18)

DATEVALUE (R18C2)

DATEVALUE ("2019/02/13") gives the result 43509.

Version Available

This function is available in product version 1.0 or later.



Note: If a user uses LegacyBehaviors.CalculationEngine, DATEVALUE function will return the DateTime object instead of the serial numeric value. For more details, please refer to [breaking changes for legacy behaviors](#).

See Also

DATE | TIMEVALUE | **Date and Time Functions**

DAVERAGE

This function calculates the average of values in a column of a list or database that match the specified conditions.

Syntax

DAVERAGE(*database*, *field*, *criteria*)

Arguments

This function has these arguments:

Argument	Description
<i>database</i>	Range of cells that make up the database; cell range reference or array
<i>field</i>	Column in the database, referred to by label or index
<i>criteria</i>	Range of cells that specify which rows in the database are used; cell range reference or array

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to **Database Functions**.

Data Types

Accepts cell ranges or arrays for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DAVERAGE(A4:E10, 3, A4:E10)
```

```
DAVERAGE(A1:A9, "Income", D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DVAR | DVARP | AVERAGE | VAR | VARP | Database Functions

DAY

This function returns the day number of the month (integer 1 to 31) that corresponds to the specified date.

Syntax

`DAY(date)`

Arguments

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in `DATE(2003,7,4)`. For more details on the date inputs, refer to the discussion in **Date and Time Functions**.

Data Types

Accepts numeric, string, or DateTime object data. Returns numeric data.

Examples

`DAY(A2)`

`DAY(R2C1)`

`DAY(366778)` gives the result 14

`DAY(33239)` gives the result 1 (because 33239 is the value for January 1, 1991)

`DAY("7/4/2003 12:00")`

`DAY(DATE(2003,7,4))`

Version Available

This function is available in product version 1.0 or later.

See Also

DATE | DATEVALUE | WEEKDAY | MONTH | Date and Time Functions

DAYS

This function calculates the number of days between two dates.

Syntax

`DAYS(end_date, start_date)`

Arguments

Specify the *end_date* and *start_date* argument as a number (as in 37806), a string or reference to cells containing the information.

Data Types

Accepts numeric or string data for both arguments. Returns numeric data.

Examples

`DAYS("5/15/11", "4/1/11")` gives the result 44.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

DAYS360

This function returns the number of days between two dates based on a 360-day year.

Syntax

DAYS360(*startdate*,*enddate*,*method*)

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>startdate</i>	Date from which to calculate days
------------------	-----------------------------------

<i>enddate</i>	Date to which to calculate days
----------------	---------------------------------

<i>method</i>	[Optional] Method for calculating days; if FALSE or omitted, uses U.S. (NASD) method; if TRUE, uses European method.
---------------	--

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in DATE(2003,7,4). For more details on the date inputs, refer to the discussion in **Date and Time Functions**

The methods for calculating the number of days can vary. The U.S. or NASD method works as follows:

- If the starting date is the 31st of a month, it becomes equal to the 30th of the same month.
- If the ending date is the 31st of a month and the starting date is earlier than the 30th of a month, the ending date becomes equal to the 1st of the next month.
- If the ending date is the 31st of a month and the starting date is the 30th or 31st of a month, the ending date becomes equal to the 30th of the ending date month.

The European method considers starting dates or ending dates that occur on the 31st of a month to be equal to the 30th of the same month.

Remarks

Use this function to help compute payments if your accounting system is based on a 360-day year (twelve 30-day months).

Data Types

Accepts numeric, string, or DateTime object data for the two date arguments and boolean for the method argument. Returns numeric data.

Examples

DAYS360 (B8, C8)

DAYS360 (R8C2, R8C3)

DAYS360 ("7/15/2004", "12/25/2004") gives the result 160

Version Available

This function is available in product version 1.0 or later.

See Also

DAY | DATEVALUE | Date and Time Functions

DB

This function calculates the depreciation of an asset for a specified period using the fixed-declining balance method.

Syntax

`DB(cost,salvage,life,period,month)`

Arguments

This functions has these arguments:

Argument	Description
<i>cost</i>	Initial cost of the asset
<i>salvage</i>	Value at the end of the depreciation period
<i>life</i>	Number of periods over which the asset is being depreciated
<i>period</i>	Period for which you want to calculate the depreciation; use the same units as the life argument
<i>month</i>	[Optional] Number of months in the first year; if omitted, the calculation uses 12 months

Remarks

The fixed-declining balance method computes depreciation at a fixed rate. This function uses the following equation to calculate depreciation for a period:

$(\text{cost} - \text{total depreciation from prior periods}) \times \text{rate}$

where:

$\text{rate} = 1 - ((\text{salvage}/\text{cost})^{(1/\text{life})})$, rounded to three decimal places

Depreciation for the first and last periods is a special case. For the first period, the function uses this equation:

$\text{dep} = \text{cost} \times \text{rate} \times \text{month}/12$

For the last period, the function uses this equation:

$\text{dep} = ((\text{cost} - \text{total dep. from prior periods}) \times \text{rate} \times (12 - \text{month}))/12$.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`DB(B1,1000,10,1)`

`DB(R1C2,10000,10,1)`

`DB(500000,5000,5,1,10)` gives the result \$25,0833.3333333333

Version Available

This function is available in product version 1.0 or later.

See Also

DDB | SLN | SYD | Financial Functions

DBCS

This function transforms half-width (single-byte) characters to full-width (double-byte) characters.

Syntax

DBCS(*text_value*)

Arguments

For the argument, you need to provide a text value or a reference to a cell that contains the text value to be changed.

Remarks

If the text value does not contain half-width letters, then the text is not modified.

Data Types

Accepts string data. Returns string data.

Examples

DBCS("SPREAD") gives the result "SPREAD"

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

DCOUNT

This function counts the cells that contain numbers in a column of a list or database that match the specified conditions.

Syntax

DCOUNT(*database*, *field*, *criteria*)

Arguments

This function has these arguments:

Argument	Description
<i>database</i>	Range of cells that make up the database; cell range reference or array
<i>field</i>	[Optional] Column in the database, referred to by label or index
<i>criteria</i>	Range of cells that specify which rows in the database are used; cell range reference or array

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index). The field argument is optional. If omitted the function counts all the records that meet the criteria.

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to **Database Functions**.

Data Types

Accepts cell ranges or arrays for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DCOUNT(A4:E10, "Type", A4:E10)
```

```
DCOUNT(A1:A9, 3, D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DCOUNTA | **COUNT** | **COUNTA** | **Database Functions**

DCOUNTA

This function counts the non-blank cells in a column of a list or database that match the specified conditions.

Syntax

DCOUNTA(*database*, *field*, *criteria*)

Arguments

This function has these arguments:

Argument	Description
<i>database</i>	Range of cells that make up the database; cell range reference or array
<i>field</i>	[Optional] Column in the database, referred to by label or index
<i>criteria</i>	Range of cells that specify which rows in the database are used; cell range reference or array

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index). The field argument is optional. If omitted the function counts all the records that meet the criteria.

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to **Database Functions**.

Data Types

Accepts cell ranges or arrays for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DCOUNTA(A4:E10, "Type", A4:E10)
```

```
DCOUNTA(A1:A9, 3, D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DCOUNT | **COUNT** | **COUNTA** | **DAVERAGE** | **Database Functions**

DDB

This function calculates the depreciation of an asset for a specified period using the double-declining balance method or another method you specify.

Syntax

`DDB(cost,salvage,life,period,factor)`

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>cost</i>	Initial cost of the asset
<i>salvage</i>	Value at the end of depreciation
<i>life</i>	Number of periods over which the asset is being depreciated
<i>period</i>	Period for which you want to calculate the depreciation in the same units as the <i>life</i> argument
<i>factor</i>	[Optional] Rate at which the value declines; if omitted, the calculation uses 2 (double-declining method)

All arguments must be positive numbers.

Remarks

This function uses the following calculation for depreciation for a period:

$cost - salvage(\text{total depreciation from prior periods}) \times \text{factor}/\text{life}$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`DDB(B1,1000,10,1)`

`DDB(R1C2,10000,10,1)`

`DDB(500000,5000,5,1,4)` gives the result \$40,0000

Version Available

This function is available in product version 1.0 or later.

See Also

DB | **SYD** | **Financial Functions**

DEC2BIN

This function converts a decimal number to a binary number.

Syntax

DEC2BIN(*number*,*places*)

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Decimal numeric value to convert in the range of -512 to 511
<i>places</i>	[Optional] Number of characters to return; if not an integer, the number is truncated

If *places* argument is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Remarks

An error value is returned if the *number* is non-numeric or outside the range, or if the *places* value is non-numeric, negative, or too small.

Data Types

Accepts numeric data. Returns numeric data.

Examples

DEC2BIN(3,3)

Version Available

This function is available in product version 2.0 or later.

See Also

DEC2HEX | DEC2OCT | BIN2DEC | OCT2BIN | Engineering Functions

DEC2HEX

This function converts a decimal number to a hexadecimal number.

Syntax

DEC2HEX(*number,places*)

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Decimal numeric value to convert in the range of -549,755,813,888 to 549,755,813,887
<i>places</i>	[Optional] Number of characters to return; if not an integer, the number is truncated

If *places* argument is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Remarks

An error value is returned if the *number* is non-numeric or outside the range, or if the *places* value is non-numeric, negative, or too small.

Data Types

Accepts numeric data. Returns numeric data.

Examples

DEC2HEX(103,4)

Version Available

This function is available in product version 2.0 or later.

See Also

DEC2BIN | DEC2OCT | BIN2HEX | OCT2HEX | **Engineering Functions**

DEC2OCT

This function converts a decimal number to an octal number.

Syntax

DEC2OCT(*number*,*places*)

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Decimal numeric value to convert in the range of -536,870,912 and 536,870,911
<i>places</i>	[Optional] Number of characters to return; if not an integer, the number is truncated

If *places* argument is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Remarks

An error value is returned if the *number* is non-numeric or outside the range, or if the *places* value is non-numeric, negative, or too small.

Data Types

Accepts numeric data. Returns numeric data.

Examples

DEC2OCT(-99)

Version Available

This function is available in product version 2.0 or later.

See Also

DEC2BIN | DEC2HEX | BIN2OCT | OCT2BIN | **Engineering Functions**

DECIMAL

This function converts the text representation (of a number in specified base) into a decimal number.

Syntax

DECIMAL(*text*,*base*)

Arguments

This function has the following arguments:

Argument	Description
----------	-------------

<i>text</i>	Refers to any combination of valid alpha-numeric characters according to the base. This value is not case sensitive.
<i>base</i>	This value must be an integer and it should be greater than or equal to 2 (binary) and less than or equal to 36

Remarks

The length of argument *text* must be less than or equal to 255 characters.

Data Types

Accepts numeric or string data for argument *text*. Accepts only numeric data for argument *base*. Returns numeric or string data.

Examples

DECIMAL("FF",21) gives the result 330.

DECIMAL(11,2) gives the result 3

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

DEGREES

This function converts the specified value from radians to degrees.

Syntax

`DEGREES(angle)`

Arguments

This function takes any real number angle value as the argument.

Remarks

This function converts angle in radians to angle in degrees.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`DEGREES (B3)`

`DEGREES (R1C2)`

`DEGREES(PI())` gives the result 180

Version Available

This function is available in product version 1.0 or later.

See Also

RADIANS | PI | Math and Trigonometry Functions

DELTA

This function identifies whether two values are equal. Returns 1 if they are equal; returns 0 otherwise.

Syntax

`DELTA(value1,value2)`

Arguments

This function takes two values as arguments.

Remarks

Also called the Kronecker Delta function. This is a discrete version of the Dirac delta function.

Data Types

Accepts numeric data. Returns numeric data (0 or 1).

Examples

`DELTA(A1,5)`

`DELTA(R1C4,R2C5)`

`DELTA(3,3)` gives the result 1

`DELTA(3,2)` gives the result 0

`DELTA(3,2.99999)` gives the result 0

`DELTA(3,QUOTIENT(6,2))` gives the result 1

Version Available

This function is available in product version 1.0 or later.

See Also

GESTEP | Engineering Functions

DEVSQ

This function calculates the sum of the squares of deviations of data points (or of an array of data points) from their sample mean.

Syntax

`DEVSQ(value1,value2, ...)`

`DEVSQ(array)`

`DEVSQ(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This is a measure of the variability in a data set.

The sum of squared deviations is calculated as follows, where n is the number of values.

$$DEVSQ(x_1, x_2, \dots, x_n) = \sum_{1}^n (x - \bar{x})^2$$

If an array or cell reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

Data Types

Accepts numeric data for all arguments or array of numeric data. Returns numeric data.

Examples

`DEVSQ(B3, B5, B9, B10)`

`DEVSQ(B3:B14)`

`DEVSQ(R3C2, R5C2, R9C2)`

`DEVSQ(R3C2:R3C12)`

`DEVSQ(35,31,47,51,37,31,58,39)` gives the result 680.875

Version Available

This function is available in product version 1.0 or later.

See Also

[AVEDEV](#) | [AVERAGE](#) | [Statistical Functions](#)

DGET

This function extracts a single value from a column of a list or database that matches the specified conditions.

Syntax

`DGET(database, field, criteria)`

Arguments

Argument	Description
<i>database</i>	Range of cells that make up the database; cell range reference or array
<i>field</i>	Column in the database, referred to by label or index
<i>criteria</i>	Range of cells that specify which rows in the database are used; cell range reference or array

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

If no value matches the criteria argument, a #VALUE! error is returned. A #NUM! error is returned if more than one match is found.

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to **Database Functions**.

Data Types

Accepts cell ranges or arrays for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

`DGET(A4:E10, "Type", A4:E10)`

`DGET(A1:A9, 3, D5:D8)`

Version Available

This function is available in product version 2.5 or later.

See Also

DAVERAGE | **DCOUNT** | **Database Functions**

DISC

This function calculates the discount rate for a security.

Syntax

`DISC(settle,mature,pricep,redeem,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security
<i>mature</i>	Maturity date for the security
<i>pricep</i>	Amount invested in the security
<i>redeem</i>	Amount to be received at maturity
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

Settle, mature, and basis are truncated to integers. If settle or mature is not a valid serial date number, a #VALUE! error is returned. If pricep or redeem is less than or equal to 0, a #NUM! error is returned. If basis is less than 0 or greater than 4, a #NUM! error is returned. If settle is greater than or equal to mature, a #NUM! error is returned.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`DISC(A1,B1,C4,100,2)`

`DISC("3/15/2003","5/15/2003",R3C4,R5C5,4)`

`DISC("5/15/2004","9/1/2004",98.2,100,3)` gives the result 0.0602752294

Version Available

This function is available in product version 1.0 or later.

See Also

RATE | INTRATE | PRICEDISC | Financial Functions

DMAX

This function returns the largest number in a column of a list or database that matches the specified conditions.

Syntax

`DMAX(database, field, criteria)`

Arguments

Argument	Description
<i>database</i>	Range of cells that make up the database; cell range reference or array
<i>field</i>	Column in the database, referred to by label or index
<i>criteria</i>	Range of cells that specify which rows in the database are used; cell range reference or array

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to **Database Functions**.

Data Types

Accepts cell ranges or arrays for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

`DMAX(A4:E10, "Type", A4:E10)`

`DMAX(A1:A9, 3, D5:D8)`

Version Available

This function is available in product version 2.5 or later.

See Also

DAVERAGE | **DCOUNT** | **DMIN** | **MAX** | **MIN** | **Database Functions**

DMIN

This function returns the smallest number in a column of a list or database that matches the specified conditions.

Syntax

`DMIN(database, field, criteria)`

Arguments

Argument	Description
----------	-------------

<i>database</i>	Range of cells that make up the database; cell range reference or array
-----------------	---

<i>field</i>	Column in the database, referred to by label or index
--------------	---

<i>criteria</i>	Range of cells that specify which rows in the database are used; cell range reference or array
-----------------	--

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to **Database Functions**.

Data Types

Accepts cell ranges or arrays for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DMIN(A4:E10, "Type", A4:E10)
```

```
DMIN(A1:A9, 3, D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DAVERAGE | **DCOUNT** | **DMAX** | **MAX** | **MIN** | **Database Functions**

DOLLAR

This function converts a number to text using currency format, with the decimals rounded to the specified place.

Syntax

`DOLLAR(value,digits)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>value</i>	Numeric value to convert to text using the currency format
--------------	--

<i>digits</i>	[Optional] Number of decimal places to maintain; if negative, the value is rounded to the left of the decimal point; if omitted, the function rounds to two decimal places
---------------	--

Remarks

This function uses the current regional Windows settings to determine the format of the returned string.

Data Types

Accepts numeric data for both arguments. Returns string data.

Examples

`DOLLAR(B5,D2)`

`DOLLAR(R5C2,R2C4)`

`DOLLAR(1234.5678,3)` gives the result \$1,234.568

`DOLLAR(123.45,1)` gives the result \$123.5

Version Available

This function is available in product version 1.0 or later.

See Also

DOLLARDE | DOLLARFR | FIXED | Text Functions

DOLLARDE

This function converts a fraction dollar price to a decimal dollar price.

Syntax

DOLLARDE(*fractionaldollar*,*fraction*)

Arguments

This function has these arguments:

Argument	Description
<i>fractionaldollar</i>	Numeric value expressed as a fraction
<i>fraction</i>	Denominator of the fraction; if not an integer, the number is truncated

Remarks

If fraction is not an integer, it is truncated. If fraction is less than 0, a #NUM! error is returned. If fraction is 0, a #DIV/0! error is returned.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

DOLLARDE(1.10,17)

DOLLARDE(R5C2,R2C4)

Version Available

This function is available in product version 2.0 or later.

See Also

DOLLAR | **DOLLARFR** | **Financial Functions**

DOLLARFR

This function converts a decimal number dollar price to a fraction dollar price.

Syntax

DOLLARFR(*decimaldollar*,*fraction*)

Arguments

This function has these arguments:

Argument	Description
<i>decimaldollar</i>	Decimal number
<i>fraction</i>	Denominator of the fraction; if not an integer, the number is truncated

Remarks

If fraction is not an integer, it is truncated. If fraction is less than 0, a #NUM! error is returned. If fraction is 0, a #DIV/0! error is returned.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

DOLLARFR(B5, D2)

DOLLARFR(R5C2, R2C4)

DOLLARFR(1.125,16) gives the result 1.02

Version Available

This function is available in product version 2.0 or later.

See Also

DOLLAR | **DOLLARDE** | **Financial Functions**

DPRODUCT

This function multiplies the values in a column of a list or database that match the specified conditions.

Syntax

DPRODUCT(*database*, *field*, *criteria*)

Arguments

Argument	Description
----------	-------------

<i>database</i>	Range of cells that make up the database; cell range reference or array
-----------------	---

<i>field</i>	Column in the database, referred to by label or index
--------------	---

<i>criteria</i>	Range of cells that specify which rows in the database are used; cell range reference or array
-----------------	--

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to **Database Functions**.

Data Types

Accepts cell ranges or arrays for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DPRODUCT(A4:E10, "Type", A4:E10)
```

```
DPRODUCT(A1:A9, 3, D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DSUM | **DCOUNT** | **PRODUCT** | **SUM** | **Database Functions**

DSTDEV

This function estimates the standard deviation of a population based on a sample by using the numbers in a column of a list or database that match the specified conditions.

Syntax

`DSTDEV(database, field, criteria)`

Arguments

Argument	Description
----------	-------------

<i>database</i>	Range of cells that make up the database; cell range reference or array
-----------------	---

<i>field</i>	Column in the database, referred to by label or index
--------------	---

<i>criteria</i>	Range of cells that specify which rows in the database are used; cell range reference or array
-----------------	--

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to **Database Functions**.

Data Types

Accepts cell ranges or arrays for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DSTDEV(A4:E10, "Type", A4:E10)
```

```
DSTDEV(A1:A9, 3, D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DSTDEVP | **DAVERAGE** | **STDEV** | **Database Functions**

DSTDEVP

This function calculates the standard deviation of a population based on the entire population using the numbers in a column of a list or database that match the specified conditions.

Syntax

`DSTDEVP(database, field, criteria)`

Arguments

Argument	Description
----------	-------------

<i>database</i>	Range of cells that make up the database; cell range reference or array
-----------------	---

<i>field</i>	Column in the database, referred to by label or index
--------------	---

<i>criteria</i>	Range of cells that specify which rows in the database are used; cell range reference or array
-----------------	--

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to **Database Functions**.

Data Types

Accepts cell ranges or arrays for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DSTDEVP(A4:E10, "Type", A4:E10)
```

```
DSTDEVP(A1:A9, 3, D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DSTDEV | **DAVERAGE** | **STDEV** | **Database Functions**

DSUM

This function adds the numbers in a column of a list or database that match the specified conditions.

Syntax

`DSUM(database, field, criteria)`

Arguments

Argument	Description
----------	-------------

<i>database</i>	Range of cells that make up the database; cell range reference or array
-----------------	---

<i>field</i>	Column in the database, referred to by label or index
--------------	---

<i>criteria</i>	Range of cells that specify which rows in the database are used; cell range reference or array
-----------------	--

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to **Database Functions**.

Data Types

Accepts cell ranges or arrays for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DSUM(A4:E10, "Type", A4:E10)
```

```
DSUM(A1:A9, 3, D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DPRODUCT | **DCOUNT** | **SUM** | **PRODUCT** | **Database Functions**

DURATION

This function returns the Macauley duration for an assumed par value of \$100.

Syntax

DURATION(*settlement,maturity,coupon,yield,frequency,basis*)

Arguments

This function has these arguments:

Argument	Description
<i>settlement</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>coupon</i>	Annual coupon rate
<i>yield</i>	Annual yield for the security
<i>frequency</i>	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when *settlement* or *maturity* is invalid or a #NUM! error when *frequency* is a number other than 1, 2, or 4. Settlement, maturity, frequency, and basis are truncated to integers. If coupon is less than 0 or yield is less than 0, a #NUM! error is returned. If basis is less than 0 or greater than 4, a #NUM! error is returned. If settlement is greater than or equal to maturity, a #NUM! error is returned.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

DURATION (C1, C2, C3, C4, C5, C6)

DURATION (R5C2, R2C4, R3C1, R4C1, R5C1)

Version Available

This function is available in product version 2.0 or later.

See Also

COUPDAYS | **MDURATION** | **Financial Functions**

DVAR

This function estimates the variance of a population based on a sample by using the numbers in a column of a list or database that match the specified conditions.

Syntax

`DVAR(database, field, criteria)`

Arguments

Argument	Description
----------	-------------

<i>database</i>	Range of cells that make up the database; cell range reference or array
-----------------	---

<i>field</i>	Column in the database, referred to by label or index
--------------	---

<i>criteria</i>	Range of cells that specify which rows in the database are used; cell range reference or array
-----------------	--

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to **Database Functions**.

Data Types

Accepts cell ranges or arrays for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DVAR(A4:E10, "Type", A4:E10)
```

```
DVAR(A1:A9, 3, D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DSTDEV | DSTDEVP | DVARP | DAVERAGE | DMIN | DMAX | Database Functions

DVARP

This function calculates the variance of a population based on the entire population by using the numbers in a column of a list or database that match the specified conditions.

Syntax

`DVARP(database, field, criteria)`

Arguments

Argument	Description
----------	-------------

<i>database</i>	Range of cells that make up the database; cell range reference or array
-----------------	---

<i>field</i>	Column in the database, referred to by label or index
--------------	---

<i>criteria</i>	Range of cells that specify which rows in the database are used; cell range reference or array
-----------------	--

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to **Database Functions**.

Data Types

Accepts cell ranges or arrays for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DVARP(A4:E10, "Type", A4:E10)
```

```
DVARP(A1:A9, 3, D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DSTDEV | DSTDEVP | DVAR | DAVERAGE | DMIN | DMAX | Database Functions

EDATE

This function calculates the date that is the indicated number of months before or after a specified date.

Syntax

`EDATE(startdate,months)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>startdate</i>	Starting date
------------------	---------------

<i>months</i>	Number of months before (negative) or after (positive) the starting date; if not an integer, the number is truncated.
---------------	---

Remarks

Use this function to calculate maturity dates or due dates that fall on the same day of the month as the date of issue.

Data Types

Accepts numeric, string, or DateTime object data for the startdate argument and numeric data for the months argument. Returns a numeric value (an excel serial number as new date).

Examples

`EDATE(A1,-6)`

`EDATE(R1C1,4)`

`EDATE("2004/01/09",2)` gives the result 38055

Version Available

This function is available in product version 1.0 or later.



Note: If a user uses LegacyBehaviors.CalculationEngine, EDATE function will return the DateTime object instead of the serial numeric value. For more details, please refer to [breaking changes for legacy behaviors](#).

See Also

[DATE](#) | [EOMONTH](#) | [Date and Time Functions](#)

EFFECT

This function calculates the effective annual interest rate for a given nominal annual interest rate and the number of compounding periods per year.

Syntax

`EFFECT(nomrate,comper)`

Arguments

This function has these arguments:

Argument	Description
<i>nomrate</i>	Nominal interest rate
<i>comper</i>	Number of compounding periods; if not an integer, the number is truncated

Remarks

The #VALUE! error is returned if either argument is nonnumeric. The #NUM error is returned if nomrate is less than or equal to zero or if comper is less than one. Comper is truncated to an integer.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`EFFECT(J12,B3)`

`EFFECT(R12C10,R3C2)`

`EFFECT(6.5%,8)` gives the result 0.66878782

Version Available

This function is available in product version 1.0 or later.

See Also

[INTRATE](#) | [NOMINAL](#) | [Financial Functions](#)

ENCODEURL

This function returns a URL encoded string.

Syntax

ENCODEURL(*text*)

Arguments

Specify the text to be encoded for the argument.

Remarks

This function is used to replace the special characters, such as "/", or "#, or "," and so on, which either is not a valid character for URL or either has its own meaning.

Data Types

Accepts string data. Returns string data.

Examples

ENCODEURL(D1) gives the result TOP, where D1 is the cell reference with text TOP

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

EOMONTH

This function calculates the last day of the month (end of month) that is the indicated number of months before or after the starting date. Typically, this function is used to evaluate expiry dates, due dates and other dates that are supposed to land at the end of a month.

Syntax

`EOMONTH(startdate,months)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>startdate</i>	Starting date
------------------	---------------

<i>months</i>	Number of months before (negative) or after (positive) the starting date; if not an integer, the number is truncated
---------------	--

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in DATE(2003,7,4).

Data Types

Accepts numeric, string, or DateTime object data for the startdate argument and numeric data for the months argument. Returns the serial number for the last day of the month.

Examples

`EOMONTH(A3,6)`

`EOMONTH(R3C1,-4)`

`EOMONTH("2019/02/13",2)` gives the result 43585

Version Available

This function is available in product version 1.0 or later.



Note: If a user uses LegacyBehaviors.CalculationEngine, EOMONTH function will return the DateTime object instead of the serial numeric value. For more details, please refer to [breaking changes for legacy behaviors](#).

See Also

[EDATE](#) | [MONTH](#) | [Date and Time Functions](#)

ERF

This function calculates the error function integrated between a lower and an upper limit.

Syntax

`ERF(limit,upperlimit)`

Arguments

This function has these arguments:

Argument	Description
<i>limit</i>	Either this is the lower limit, if the upper limit is supplied, or it is the upper limit (with 0 as the lower limit) if the second argument is not supplied
<i>upperlimit</i>	[Optional] Upper limit for integrating the function

Remarks

If *upperlimit* is supplied, the function is integrated from *limit* to *upperlimit*. If not supplied, the function is integrated from 0 to *limit*.

If there *upperlimit* is not supplied, the function calculates:

$$ERF(x) = \frac{2}{\pi} \int_0^x (e^{-t^2}) dt$$

where x is the *limit* argument.

If there *upperlimit* is supplied, the function calculates:

$$ERF(lo, hi) = \frac{2}{\pi} \int_{lo}^{hi} (e^{-t^2}) dt$$

where lo is the *limit* argument and hi is the *upperlimit* argument.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ERF(K16)`

`ERF(R16C11,R16,C12)`

`ERF(0.49)` gives the result 0.51166826

`ERF(0.25,0.85)` gives the result 0.494341544

Version Available

This function is available in product version 1.0 or later.

See Also

ERFC | STEYX | Engineering Functions

ERF.PRECISE

This function calculates the error function.

Syntax

ERF.PRECISE(*limit*)

Arguments

This function has the following argument:

Argument	Description
<i>limit</i>	If <i>limit</i> is nonnumeric, the function returns the #VALUE! error value

Data Types

Accepts numeric data. Returns numeric data.

Examples

ERF.PRECISE(K16)

ERF.PRECISE(R16C11)

ERF.PRECISE(0.49) gives the result 0.5116682610468377

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

ERF

ERFC

This function calculates the complementary error function integrated between a lower limit and infinity.

Syntax

`ERFC(lowerlimit)`

Arguments

The argument is the lower limit from which to integrate to infinity when calculating this function.

Remarks

This function calculates the complementary error function as follows:

$$ERFC(x) = \frac{2}{\pi} \int_x^{\infty} (e^{-t^2}) dt$$

where x is the lower limit specified in the argument.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ERFC(K16)`

`ERFC(R16C11)`

`ERFC(0.49)` gives the result 0.48833174

Version Available

This function is available in product version 1.0 or later.

See Also

ERF | STEYX | Engineering Functions

ERFC.PRECISE

This function calculates the complementary ERF function integrated between a lower limit and infinity.

Syntax

ERFC.PRECISE(*lowerlimit*)

Arguments

The argument is the lower limit from which to integrate to infinity when calculating this function.

Remarks

If *lowerlimit* is nonnumeric, this function returns the #VALUE! error value.

Data Types

Accepts numeric data. Returns numeric data.

Examples

ERFC.PRECISE(K16)

ERFC.PRECISE(R16C11)

ERFC.PRECISE(0.49) gives the result 0.4883317389531623

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

ERFC

ERROR.TYPE

This function returns a number corresponding to one of the error values.

Syntax

ERROR.TYPE(*errorvalue*)

Arguments

The valid error values that can be used in the arguments and their corresponding returned values are summarized here:

Error Value	Function Returns
#NULL!	1
#DIV/0!	2
#VALUE!	3
#REF!	4
#NAME?	5
#NUM!	6
#N/A	7

Remarks

You can use this function in an IF-THEN structure to test for the error value and return a text string, such as a message, instead of the error value.

Data Types

Accepts error value as data. Returns numeric data.

Examples

ERROR.TYPE(B13)

ERROR.TYPE(R13C2)

ERROR.TYPE(#REF!) gives the result 4

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

ERRORTYPE

ERRORTYPE

This function returns a number corresponding to one of the error values.

Syntax

`ERRORTYPE(errorvalue)`

Arguments

The valid error values that can be used in the arguments and their corresponding returned values are summarized here:

Error Value	Function Returns
#NULL!	1
#DIV/0!	2
#VALUE!	3
#REF!	4
#NAME?	5
#NUM!	6
#N/A	7

Remarks

You can use this function in an IF-THEN structure to test for the error value and return a text string, such as a message, instead of the error value.

Data Types

Accepts error value as data. Returns numeric data.

Examples

`ERRORTYPE(B13)`

`ERRORTYPE(R13C2)`

`ERRORTYPE(#REF!)` gives the result 4

Version Available

This function is available in product version 1.0 or later.

See Also

ISERROR | **Information Functions**

EUROCONVERT

This function converts currency from a Euro member currency (including Euros) to another Euro member currency (including Euros).

Syntax

EUROCONVERT(*currency,source,target,fullprecision,triangulation*)

Arguments

This function has these arguments:

Argument	Description
<i>currency</i>	Number to convert
<i>source</i>	ISO currency code for the number to convert (see table below)
<i>target</i>	ISO currency code for the result of the conversion (see table below)
<i>fullprecision</i>	[Optional] Logical value representing whether to display the value in full precision or not; if omitted, the value is not displayed in full precision
<i>triangulation</i>	[Optional] Integer greater than or equal to 3 that specifies the number of significant digits to be used for the intermediate Euro value when converting between two Euro member currencies

If *triangulation* is omitted, the calculation does not round the intermediate Euro value. If it is included when converting from a Euro member currency to the Euro, the calculation finds the intermediate Euro value that could then be converted to a Euro member currency.

Remarks

This function does not convert all currencies; only those Euro member currencies listed in this table.

Country/Region	ISO Currency Code
Belgium	BEF
Luxembourg	LUF
Germany	DEM
Spain	ESP
France	FRF
Ireland	IEP
Italy	ITL
Netherlands	NLG
Austria	ATS
Portugal	PTE
Finland	FIM
Euro member state	EUR

ISO Currency Codes are from ISO 4217, the international standard describing three-letter codes to define the names of currencies. ISO is the nickname for the International Organization for Standardization. The first two letters of the code

are the two-letter country codes (ISO 3166) and the third is usually the initial of the currency itself. So BEF is Belgium Franc.

Data Types

Accepts numeric and string data for most arguments; the *fullprecision* argument is a logical value. Returns numeric data.

Examples

```
EUROCONVERT (B5, "DEM", "EUR")
```

```
EUROCONVERT (R5C2, "DEM", "EUR", TRUE, 3)
```

Version Available

This function is available in product version 2.0 or later.

See Also

ROUND | **Financial Functions**

EVEN

This function rounds the specified value up to the nearest even integer.

Syntax

`EVEN(value)`

Arguments

The argument can be any numeric value.

Remarks

Regardless of the sign of the number specified by the argument, the number is rounded away from zero.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`EVEN(A3)`

`EVEN(R1C2)`

`EVEN(5)` gives the result 6

`EVEN(-2.5)` gives the result -4

Version Available

This function is available in product version 1.0 or later.

See Also

CEILING | FLOOR | ODD | ISEVEN | Math and Trigonometry Functions

EXACT

This function returns true if two strings are the same; otherwise, false.

Syntax

```
EXACT(text1,text2)
```

Arguments

The arguments are text strings.

Remarks

This function compares the string in the first argument to the string in the second argument. Although this function is case-sensitive, it ignores formatting differences.

Data Types

Accepts string data for both arguments. Returns boolean data (true or false).

Examples

```
EXACT(A3,A5)
```

```
EXACT(R3C1,R5C1)
```

```
EXACT("SPREAD","spread") gives the result FALSE
```

Version Available

This function is available in product version 1.0 or later.

See Also

CONCATENATE | **Text Functions**

EXP

This function returns e raised to the power of the specified value.

Syntax

`EXP(value)`

Arguments

The argument for this function is any numeric value.

Remarks

Mathematically, this function is (e^x).

This function is the inverse of **LN**, so `EXP (LN(x))` results in x.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`EXP (B3)`

`EXP (R1C2)`

`EXP(1)` gives the result 2.7182818285

Version Available

This function is available in product version 1.0 or later.

See Also

LN | **LOG** | **POWER** | **Math and Trigonometry Functions**

EXPON.DIST

This function returns the exponential distribution or the probability density.

Syntax

EXPON.DIST(*value*,*lambda*,*cumulative*)

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Value of the function; must be positive or zero
<i>lambda</i>	Parameter value; must be greater than zero
<i>cumulative</i>	Logical value indicating whether to return the cumulative distribution; set to TRUE to return the cumulative distribution; set to FALSE to return the probability density

Remarks

Use this function to model the time between events, such as how long an automated bank teller takes to deliver cash. For example, you can use this function to determine the probability that the process takes at most one minute.

The cumulative distribution is calculated as follows:

$$EXPONDIST(x, \lambda, TRUE) = 1 - e^{(-\lambda x)}$$

where x is the *value* argument, lambda is the *lambda* argument.

The probability density is calculated as follows:

$$EXPONDIST(x, \lambda, FALSE) = \lambda e^{(-\lambda x)}$$

where x is the *value* argument, lambda is the *lambda* argument.

Data Types

Accepts numeric data, except the third argument, which accepts logical data. Returns numeric data.

Examples

EXPON.DIST(C12,10,TRUE)

EXPON.DIST(R12C3,8,FALSE)

EXPON.DIST(0.2,10,TRUE) gives the result 0.8646647167633873

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

EXPONDIST

EXPONDIST

This function returns the exponential distribution or the probability density.

Syntax

`EXPONDIST(value,lambda,cumulative)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>value</i>	Value of the function; must be positive or zero
--------------	---

<i>lambda</i>	Parameter value; must be greater than zero
---------------	--

<i>cumulative</i>	Logical value indicating whether to return the cumulative distribution; set to TRUE to return the cumulative distribution; set to FALSE to return the probability density
-------------------	---

Remarks

Use this function to model the time between events, such as how long an automated bank teller takes to deliver cash. For example, you can use this function to determine the probability that the process takes at most one minute.

The cumulative distribution is calculated as follows:

$$EXPONDIST(x, \lambda, FALSE) = \lambda e^{-\lambda x}$$

where x is the *value* argument, lambda is the *lambda* argument.

The probability density is calculated as follows:

$$EXPONDIST(x, \lambda, TRUE) = 1 - e^{-\lambda x}$$

where x is the *value* argument, lambda is the *lambda* argument.

Data Types

Accepts numeric data, except the third argument, which accepts logical data. Returns numeric data.

Examples

`EXPONDIST(C12,10,TRUE)`

`EXPONDIST(R12C3,8,FALSE)`

`EXPONDIST(0.2,10,TRUE)` gives the result 0.8646647168

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | Statistical Functions

F.DIST

This function calculates the F probability distribution, to see degrees of diversity between two sets of data.

Syntax

F.DIST(*value,degnum,degden,cumulative*)

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>value</i>	Value at which to evaluate the function
<i>degnum</i>	Number of degrees of freedom for the numerator; if not an integer, the number is truncated
<i>degden</i>	Number of degrees of freedom for the denominator; if not an integer, the number is truncated
<i>cumulative</i>	A logical value that determines the form of the function. If <i>cumulative</i> is TRUE, this function returns the cumulative distribution function; if FALSE, it returns the probability density function

Data Types

Accepts numeric data for all arguments except *cumulative*. Returns numeric data.

Examples

F.DIST(A1,2,2,TRUE)

F.DIST(R1C1,2,1,TRUE)

F.DIST(16.83975,5,3,TRUE) gives the result 0.9789999175380504

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

FDIST

F.DIST.RT

This function calculates the F probability distribution, to see degrees of diversity between two sets of data.

Syntax

F.DIST.RT(*value, degnum, degden*)

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Value at which to evaluate the function
<i>degnum</i>	Number of degrees of freedom for the numerator; if not an integer, the number is truncated
<i>degden</i>	Number of degrees of freedom for the denominator; if not an integer, the number is truncated

Remarks

The #VALUE! error value is returned if any argument is nonnumeric.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

F.DIST.RT(A1,2,2)

F.DIST.RT(R1C1,2,1)

F.DIST.RT(16.83975,5,3) gives the result 0.021000082461949843

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

FDIST

F.INV

This function returns the inverse of the F probability distribution.

Syntax

$F.INV(p, degnum, degden)$

Arguments

This function has these arguments:

Argument	Description
p	Probability associated with the F cumulative distribution
$degnum$	Number of degrees of freedom for the numerator; if not an integer, the number is truncated
$degden$	Number of degrees of freedom for the denominator; if not an integer, the number is truncated

Remarks

This function calculates the inverse of the F probability distribution, so if $p = F.DIST(x, \dots)$, then $F.INV(p, \dots) = x$.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

$F.INV(A1, 2, 2)$

$F.INV(R1C1, 2, 1)$

$F.INV(0.021, 5, 3)$ gives the result 0.11813305544967191

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

FINV

F.INV.RT

This function returns the inverse of the F probability distribution.

Syntax

F.INV.RT(*p*,*degnum*,*degden*)

Arguments

This function has these arguments:

Argument	Description
<i>p</i>	Probability associated with the F cumulative distribution
<i>degnum</i>	Number of degrees of freedom for the numerator; if not an integer, the number is truncated
<i>degden</i>	Number of degrees of freedom for the denominator; if not an integer, the number is truncated

Remarks

This function calculates the inverse of the F probability distribution, so if $p = \text{FDIST.RT}(x, \dots)$, then $\text{FINV.RT}(p, \dots) = x$. The #VALUE! error value is returned if any argument is nonnumeric.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

F.INV.RT(A1,2,2)

F.INV.RT(R1C1,2,1)

F.INV.RT(0.021,5,3) gives the result 16.83979663538795

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

FINV

F.TEST

This function returns the result of an F-test, which returns the two-tailed probability that the variances in two arrays are not significantly different.

Syntax

`F.TEST(array1,array2)`

Arguments

The arguments may be arrays of values.

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

`F.TEST(A1:D34,A35:D68)`

`F.TEST(R1C1:R34C4,R35C1:R68C4)`

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

FTEST

FACT

This function calculates the factorial of the specified number.

Syntax

`FACT(number)`

Arguments

The argument can be any numeric value.

Remarks

The factorial is the product of the positive integers less than or equal to a number and is calculated as $1 \times 2 \times 3 \times \dots \times \textit{number}$, and is typically written as $n!$ for n being the number. For example, $4!$ is $1 \times 2 \times 3 \times 4$, which is 24. The argument must be a non-negative number. If you provide a number that is not an integer for the argument, the decimal portion of the number is ignored.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`FACT(B3)`

`FACT(R1C2)`

`FACT(10)` gives the result 3628800

Version Available

This function is available in product version 1.0 or later.

See Also

FACTDOUBLE | **PRODUCT** | **Math and Trigonometry Functions**

FACTDOUBLE

This function calculates the double factorial of the specified number.

Syntax

FACTDOUBLE(*number*)

Arguments

The argument can be any non-negative numeric value.

Remarks

The *number* argument must be a non-negative number. If you provide a number that is not an integer for the *number* argument, the decimal portion of the number is ignored. The double factorial is calculated as follows for even numbers:

$$n!! = n(n-2)(n-4) \dots (4)(2)$$

The double factorial is calculated as follows for odd numbers:

$$n!! = n(n-2)(n-4) \dots (3)(1)$$

Data Types

Accepts numeric data. Returns numeric data.

Examples

FACTDOUBLE(E3)

FACTDOUBLE(R3C5)

FACTDOUBLE(6) gives the result 48

Version Available

This function is available in product version 1.0 or later.

See Also

FACT | **PRODUCT** | **Math and Trigonometry Functions**

FALSE

This function returns the value for logical FALSE.

Syntax

FALSE()

Remarks

This function does not accept arguments.

Data Types

Does not accept data. Returns numeric (boolean) data.

Example

FALSE() gives the result 0 (FALSE)

Version Available

This function is available in product version 1.0 or later.

See Also

IF | **TRUE** | **Logical Functions**

FDIST

This function calculates the F probability distribution, to see degrees of diversity between two sets of data.

Syntax

`FDIST(value,degnum,degden)`

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Value at which to evaluate the function
<i>degnum</i>	Number of degrees of freedom for the numerator; if not an integer, the number is truncated
<i>degden</i>	Number of degrees of freedom for the denominator; if not an integer, the number is truncated

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`FDIST(A1,2,2)`

`FDIST(R1C1,2,1)`

`FDIST(16.83975,5,3)` gives the result 0.021

Version Available

This function is available in product version 1.0 or later.

See Also

FINV | Statistical Functions

FILTER

This function allows users to filter a cell range on the basis of the defined criteria. The Filter operation can be performed based on a single criterion or multiple criteria.

In order to combine two or more filter conditions, users can use the "*" operator and the "+" operator. The * operator will multiply two sets of conditions in order to join the filter criteria with AND logic [when both the filter conditions have to be TRUE]. The + operator will simply join the two sets of conditions with OR logic [when one filter condition can be TRUE and the other can be FALSE].

Syntax

`FILTER(array,include,[if_empty])`

Arguments

FILTER function has the following arguments:

Argument	Description
----------	-------------

<i>array</i>	[required] Specifies the range or array that you want to filter.
<i>include</i>	[required] Specifies the filter condition expressed using an intersecting sub-range and conditional expressions.
<i>if_empty</i>	[optional] Specifies the optional value that users want to return when the filter result is empty. If value is not specified for this parameter, then #CALC! error is thrown.

Data Types

Accepts a cell range or an array of data that you want to filter. Returns a filtered array.

Examples

For instance - The cell F5 in the following image contains the formula "`=FILTER(A5:D17, C5:C17=F1)`". This formula filters the cell range A5 to D17 based on one filter criteria (when the cell range C5 to C17 matches the Product value in cell F1 i.e. Apple). As a result, all the values in the cell range A5 to D17 containing product as "Apple" will be displayed.

In another example, the cell F14 in the following image contains the formula "`=FILTER(A5:D17, (C5:C17=F1)*(A5:A17=F2))`". This formula filters the cell range A5 to D17 based on two filter conditions that are specified by the multiplication (*) operator. The first condition is the cell range C5 to C17 should match the Product value in cell F1 i.e. Apple and the second condition is the cell range A5 to A17 should match the region "East". As a result, all the values in the cell range A5 to D17 containing Product as "Apple" and Region as "East" will be displayed.

	A	B	C	D	E	F	G	H	I	J
1					Product:	Apple				
2					Region:	East				
3						Filtering performed on one Criteria				
4	Region	Sales Rep	Product	Units		Region	Sales Rep	Product	Units	
5	East	Tom	Apple	6380		East	Tom	Apple	6380	
6	North	Fred	Grape	2344		East	Hector	Apple	2341	
7	Wast	Amy	Pear	3434		North	Fred	Apple	2334	
8	Sast	Sal	Banana	5461		South	Sravan	Apple	7682	
9	East	Hector	Apple	2341						
10	East	Xi	Banana	3234						
11	West	Amy	Banana	6532						
12	South	Sal	Pear	7323		Filtering performed on two Criteria				
13	North	Fred	Apple	2334		Region	Sales Rep	Product	Units	
14	North	Tom	Grape	8734		East	Tom	Apple	6380	
15	East	Hector	Grape	1932		East	Hector	Apple	2341	
16	South	Sravan	Apple	7682						
17	West	Xi	Grape	3293						

Version Available

This function is available in Spread for Windows Forms 12.1 or later.

FILTERXML

This function returns specific data from the XML content using the specified XPath.

Syntax

`FILTERXML(xml,xpath)`

Arguments

This function has the following arguments:

Argument	Description
<i>xml</i>	Refers to a valid XML formatted string
<i>xpath</i>	Refers to a standard XPath formatted string

Remarks

If the XML string is invalid or if it contains a namespace with a prefix which is not valid, this function returns a #VALUE! error value.

Data Types

Accepts string data. Returns string data.

Examples

`FILTERXML(A3,"//cd/@title")`

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

FIND

This function finds one text value within another and returns the text value's position in the text you searched.

Syntax

`FIND(findtext,intext,start)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>findtext</i>	Text you are trying to find; if empty (" "), the function matches the first character in the search string (that is, the character numbered <i>start</i> or 1); cannot contain wildcard characters
<i>intext</i>	Text through which you are searching
<i>start</i>	[Optional] Number representing character at which to start the search; the first character of <i>intext</i> is 1; if omitted, the calculation starts at 1; if not an integer, the number is truncated

Remarks

This function performs a case-specific search (for example, to specify a capital letter and not lower case letters).

Data Types

Accepts string data for the *findtext* argument, string data for the *intext* argument, and numeric data for the *start* argument. Returns numeric data.

Examples

```
FIND("G",A2,1)
```

```
FIND("G",R2C1,1)
```

```
FIND("P","FarPoint Technologies") gives the result 4
```

```
FIND("n","FarPoint Technologies",8) gives the result 4
```

Version Available

This function is available in product version 1.0 or later.

See Also

REPLACE | **SUBSTITUTE** | **Text Functions**

FINDB

This function finds the specified text string(1) within another text string(2) and returns the number of the starting position of the specified text string(1) from the first character of the another text string(2).

Syntax

FINDB(*findtext*,*intext*,*start*)

Arguments

This function has the following arguments:

Argument	Description
----------	-------------

<i>findtext</i>	Refers to the text you are trying to find. If the value is empty (" "), the function matches the first character in the search string (i.e. the character numbered start or 1). This value cannot contain wildcard characters.
<i>intext</i>	Refers to the text through which you are searching.
<i>start</i>	[Optional] Refers to the number representing character at which to start the search. The first character of <i>intext</i> is 1; if omitted, the calculation starts at 1. If this value is not an integer, the number is truncated.

Remarks

The FINDB function counts 2 bytes per character, but this happens only when a DBCS language is set as the default language.

This function performs a case-specific search (for example, to specify a capital letter and not lower case letters).

Data Types

Accepts string data for the *findtext* argument, string data for the *intext* argument, and numeric data for the *start* argument. Returns numeric data.

Examples

FINDB("ea","rheabuto") gives the result 3.

FINDB("to","rheabuto") gives the result 7.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

FINV

This function returns the inverse of the F probability distribution.

Syntax

`FINV(p,degnum,degden)`

Arguments

This function has these arguments:

Argument	Description
<i>p</i>	Probability associated with the F cumulative distribution
<i>degnum</i>	Number of degrees of freedom for the numerator; if not an integer, the number is truncated
<i>degden</i>	Number of degrees of freedom for the denominator; if not an integer, the number is truncated

If either *degnum* or *degden* is not an integer, it is truncated.

Remarks

This function calculates the inverse of the F probability distribution, so if $p = \text{FDIST}(x, \dots)$, then $\text{FINV}(p, \dots) = x$.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`FINV(A1, 2, 2)`

`FINV(R1C1, 2, 1)`

`FINV(0.021, 5, 3)` gives the result 16.83975

Version Available

This function is available in product version 1.0 or later.

See Also

FDIST | **Statistical Functions**

FISHER

This function returns the Fisher transformation for a specified value.

Syntax

`FISHER(value)`

Arguments

Provide a numeric value that is less than 1 and greater than -1 for which you want the transformation.

Remarks

This transformation produces an approximately normal distribution. Use this function to perform hypothesis testing on the correlation coefficient.

The Fisher transformation is calculated as follows:

$$FISHER(x) = \frac{1}{2} \ln \frac{(1+x)}{(1-x)}$$

where *x* is the *value* argument.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`FISHER(A43)`

`FISHER(R4C12)`

`FISHER(-0.65)` gives the result -0.7752987062

Version Available

This function is available in product version 1.0 or later.

See Also

FISHERINV | Statistical Functions

FISHERINV

This function returns the inverse of the Fisher transformation for a specified value.

Syntax

`FISHERINV(value)`

Arguments

The argument is the specified numeric value.

Remarks

Use this transformation when analyzing correlations between ranges or arrays of data. This function calculates the inverse of the Fisher transformation, so if $y = \mathbf{FISHER}(x)$, then $\mathbf{FISHERINV}(y) = x$.

The inverse Fisher transformation is calculated as follows:

$$\mathbf{FISHERINV}(y) = \frac{e^{2y} - 1}{e^{2y} + 1}$$

where y is the *value* argument.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`FISHERINV(A43)`

`FISHERINV(R4C12)`

`FISHERINV(0.56)` gives the result 0.5079774329

Version Available

This function is available in product version 1.0 or later.

See Also

FISHER | **Statistical Functions**

FIXED

This function rounds a number to the specified number of decimal places, formats the number in decimal format using a period and commas (if so specified), and returns the result as text.

Syntax

`FIXED(num,digits,notcomma)`

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>num</i>	Number to round and convert to text
------------	-------------------------------------

<i>digits</i>	[Optional] Number of decimal places; if omitted, uses two places
---------------	--

<i>notcomma</i>	[Optional] Logical value whether not to use commas; if omitted or FALSE, returns with commas
-----------------	--

Data Types

Accepts numeric data for first two arguments; accepts logical value for the third argument. Returns string (text) data.

Examples

`FIXED(B3)`

`FIXED(R3C2,2,FALSE)`

`FIXED(4.2365,3)`

Version Available

This function is available in product version 1.0 or later.

See Also

DOLLAR | **Text Functions**

FLOOR

This function rounds a number down to the nearest multiple of a specified value.

Syntax

`FLOOR(value,signif)`

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Number to round
<i>signif</i>	Number representing the rounding factor

Use either both positive or both negative numbers for the arguments. Regardless of the sign of the numbers, the value is rounded toward zero.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

```
FLOOR(C4,B2)
```

```
FLOOR(B3,0.05)
```

```
FLOOR(R1C2,1)
```

```
FLOOR(4.65,2) gives the result 4
```

```
FLOOR(-2.78,-1) gives the result -2
```

Version Available

This function is available in product version 1.0 or later.

See Also

CEILING | EVEN | ODD | TRUNC | Math and Trigonometry Functions

FLOOR.MATH

This function rounds a number down to the nearest multiple of the specified value.

Syntax

FLOOR.MATH(*value,signif,mode*)

Arguments

This function has the following arguments:

Argument	Description
<i>value</i>	Number to round
<i>signif</i>	[Optional] Number representing the rounding factor
<i>mode</i>	[Optional] Represents the direction (towards or away from 0) to round negative value

Remarks

Positive numbers with decimal parts are rounded down to the nearest integer.

Negative numbers with decimal parts are rounded away from 0 to the nearest integer.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

FLOOR.MATH(-3.1,3,4) gives the result -3

FLOOR.MATH(-6.3,8) gives the result -8

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

FLOOR.PRECISE

Summary

This function rounds a number down to the nearest multiple of a specified value or to the nearest integer.

Syntax

FLOOR.PRECISE(*value*,*signif*)

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Number to round
<i>signif</i>	[Optional] Number representing the rounding factor

Use either both positive or both negative numbers for the arguments. Regardless of the sign of the numbers, the value is rounded toward zero.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

FLOOR.PRECISE(C4,B2)

FLOOR.PRECISE(B3,0.05)

FLOOR.PRECISE(R1C2,1)

FLOOR.PRECISE(4.65,2) gives the result 4

FLOOR.PRECISE(-2.78,-1) gives the result -3

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

FORECAST

This function calculates a future value using existing values.

Syntax

`FORECAST(value,Yarray,Xarray)`

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Value for which to predict the future dependent value
<i>Yarray</i>	An array of known dependent values (y's)
<i>Xarray</i>	An array of known independent values (x's)

Remarks

The predicted value is a y value for a given x value. The known values are existing x values and y values, and the new value is predicted by using linear regression. You can use this function to predict future sales, inventory requirements, or consumer trends.

This function is calculated as follows:

$$FORECAST(v, Y, X) = \bar{Y} - \left[\frac{n \sum xy - \sum x \sum y}{n \sum x - (\sum x)^2} \right] \bar{X} + \left[\frac{n \sum xy - \sum x \sum y}{n \sum x - (\sum x)^2} \right] v$$

where v is the *value* argument, Y is the *Yarray* argument, X is the *Xarray* argument, and n is the size of the arrays.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`FORECAST(30,G1:G9,F1:F9)`

`FORECAST(30,R1C7:R9C7,R1C6:R9C6)`

`FORECAST(45,{53000,57000,58000,69000,74500,55620,80000,68700},{35,31,47,51,37,31,58,39})` gives the result 67060.8665320360

Version Available

This function is available in product version 1.0 or later.

See Also

INTERCEPT | Statistical Functions

FORECAST.LINEAR

This function calculates future value by using existing values.

Syntax

FORECAST.LINEAR(*x*, *known_y*, *known_x*)

Arguments

This function has the following arguments:

Argument	Description
<i>x</i>	Refers to the numeric data specifying data point to predict a value for.
<i>known_y</i>	Refers to the numeric data (array) specifying known y-values.
<i>known_x</i>	Refers to the numeric data (array) specifying known x-values.

Remarks

This function predicts a new value on a linear basis. Hence, if the data follows seasonal variances, this function is not useful.

Data Types

Accepts only numeric data. Returns numeric data.

Examples

FORECAST.LINEAR(7, B4:B8, C4:C8)

Version Available

This function is available in product version 11.0 or later.

FORMULATEXT

This function returns a formula as a string.

Syntax

FORMULATEXT(*reference*)

Arguments

Specify the reference to single cell or range of cells for the argument.

Remarks

In this function, text displayed in the formula bar of the specified cell is returned. Reference argument can refer to cell (or cells) of another worksheet or workbook too. Value of upper leftmost cell or row is returned, if an entire row or column is referred.

Data Types

Accepts cell reference for argument. Returns string data.

Examples

FORMULATEXT(B7) gives the value #N/A

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

FREQUENCY

This function calculates how often values occur within a range of values. This function returns a vertical array of numbers.

Syntax

`FREQUENCY(dataarray,binarray)`

Arguments

This function has these arguments:

Argument	Description
<i>dataarray</i>	Array of values or a reference to a set of values for which to count frequencies
<i>binarray</i>	Array of intervals or a reference to intervals into which to group the values of <i>dataarray</i>

Remarks

The number of elements in the returned array is one greater than the number of elements in *binarray*. The extra element in the returned array is the count of values in *dataarray* that is above the highest value in *binarray*.

Use the **INDEX** function to get individual elements from the returned arrays.

Data Types

Accepts an array. Returns an array.

Examples

`FREQUENCY(A1:A7,C2:C5)`

Version Available

This function is available in product version 2.0 or later.

See Also

AVEDEV | **AVERAGEA** | **CONFIDENCE** | **DEVSQ** | **MEDIAN** | **VAR** | **Statistical Functions**

FTEST

This function returns the result of an F-test, which returns the one-tailed probability that the variances in two arrays are not significantly different.

Syntax

`FTEST(array1,array2)`

Arguments

The arguments may be arrays of values.

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

`FTEST(A1:D34,A35:D68)`

`FTEST(R1C1:R34C4,R35C1:R68C4)`

Version Available

This function is available in product version 1.0 or later.

See Also

ZTEST | **TTEST** | **Statistical Functions**

FV

This function returns the future value of an investment based on a present value, periodic payments, and a specified interest rate.

Syntax

`FV(rate,numper,paymt,pval,type)`

Arguments

This function has these arguments:

Argument	Description
<i>rate</i>	Interest rate expressed as percentage (per period)
<i>numper</i>	Total number of payment periods
<i>paymt</i>	Payment made each period
<i>pval</i>	[Optional] Present value; if omitted, uses zero and the calculation is based on the <i>paymt</i> argument.
<i>type</i>	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

Remarks

Use consistent units for specifying the rate and number of periods arguments. If you make monthly payments on a five-year loan at 8 percent annual interest, use 0.08/12 for the rate argument and 5*12 for the number of periods argument. If you make annual payments on the same loan, use 0.08 for rate and 5 for number of periods.

For the arguments, money paid out (such as deposits in an investment) is represented by negative numbers; money you receive (such as dividend checks) is represented by positive numbers.

See the **PV** function for the equations for calculating financial values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`FV(A1/12, 48, B1, 1000, 0)`

`FV(R1C1/12, 48, R1C2, 1000, 0)`

`FV(0.005, 60, -100, 100, 1)` gives the result \$6877.00

Version Available

This function is available in product version 1.0 or later.

See Also

FVSCCHEDULE | **NPER** | **PMT** | **PV** | **Financial Functions**

FVSCCHEDULE

This function returns the future value of an initial principal after applying a series of compound interest rates. Calculate future value of an investment with a variable or adjustable rate.

Syntax

`FVSCCHEDULE(principal,schedule)`

Arguments

This function has these arguments:

Argument	Description
<i>principal</i>	Present value of the principal
<i>schedule</i>	Schedule, array of interest rates to apply

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`FVSCCHEDULE(4,A1:C1)`

`FVSCCHEDULE(45,R1C1:R7C1)`

`FVSCCHEDULE(1000,{0.8,0.6,0.7})` gives the result 4896

Version Available

This function is available in product version 1.0 or later.

See Also

FV | **Financial Functions**

GAMMA

This function calculates the gamma function value.

Syntax

`GAMMA(value)`

Arguments

For the argument, you can specify any real number whose value is either greater than 1 or equal to 1.

Remarks

The value passed in the arguments should not be a negative integer or 0. If the specified number is a 0 or a negative integer, this function returns the #NUM! error.

If the specified number possesses invalid characters, this function returns the #VALUE! error.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`GAMMA(7.5)` gives the result 1871.254

`GAMMA(1)` gives the result 1

`GAMMA(-1.23)` gives the result #NUM!

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

GAMMA.DIST

This function returns the gamma distribution.

Syntax

`GAMMA.DIST(x,alpha,beta,cumulative)`

Arguments

This function has these arguments:

Argument Description

<i>x</i>	Value at which to evaluate the distribution
<i>alpha</i>	Alpha parameter of the distribution
<i>beta</i>	Beta parameter of the distribution
<i>cumulative</i>	Logical value that determines the form of the function. If <code>TRUE</code> , then this function returns the cumulative distribution function; if <code>FALSE</code> , it returns the probability density function.

Remarks

The equation for this function is:

$$GAMMADIST(x, \alpha, \beta, TRUE) = \frac{1}{\beta^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{-x/\beta}$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`GAMMA.DIST(A5,1,3,FALSE)`

`GAMMA.DIST(R5C1,2,1,TRUE)`

`GAMMA.DIST(4,3,2,TRUE)` gives the result 0.3233235838169362

`GAMMA.DIST(4,3,2,FALSE)` gives the result 0.1353352832366127

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

GAMMA

GAMMA.INV

This function returns the inverse of the gamma cumulative distribution.

Syntax

`GAMMA.INV(p,alpha,beta)`

Arguments

This function has these arguments:

Argument	Description
<i>p</i>	Probability
<i>alpha</i>	Alpha parameter of the distribution
<i>beta</i>	Beta parameter of the distribution

Remarks

This function calculates the inverse of the F probability distribution, so if $p = \text{GAMMA.DIST}(x, \dots)$, then $\text{GAMMA.INV}(p, \dots) = x$. The standard gamma distribution is returned if $beta = 1$.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`GAMMA.INV(A3,3,4)`

`GAMMA.INV(0.8902,R3C8,R3C9)`

`GAMMA.INV(0.75,2,3)` gives the result 8.077903586669088

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

GAMMA

GAMMADIST

This function returns the gamma distribution.

Syntax

`GAMMADIST(x,alpha,beta,cumulative)`

Arguments

This function has these arguments:

Argument	Description
<i>x</i>	Value at which to evaluate the distribution
<i>alpha</i>	Alpha parameter of the distribution
<i>beta</i>	Beta parameter of the distribution
<i>cumulative</i>	Logical value that determines the form of the function. If cumulative is TRUE, then this function returns the cumulative distribution function; if FALSE, it returns the probability mass function.

Remarks

The equation for this function is:

$$GAMMADIST(x, \alpha, \beta, TRUE) = \frac{1}{\beta^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{-x/\beta}$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`GAMMADIST(A5,1,3,FALSE)`

`GAMMADIST(R5C1,2,1,TRUE)`

`GAMMADIST(4,3,2,TRUE)` gives the result 0.3233235838

`GAMMADIST(4,3,2,FALSE)` gives the result 0.1353352832

Version Available

This function is available in product version 1.0 or later.

See Also

BETADIST | GAMMAINV | GAMMALN | KURT | POISSON | Statistical Functions

GAMMAINV

This function returns the inverse of the gamma cumulative distribution.

Syntax

`GAMMAINV(p,alpha,beta)`

Arguments

This function has these arguments:

Argument	Description
<i>p</i>	Probability
<i>alpha</i>	Alpha parameter of the distribution
<i>beta</i>	Beta parameter of the distribution

Remarks

This function calculates the inverse of the F probability distribution, so if $p = \text{GAMMADIST}(x, \dots)$, then $\text{GAMMAINV}(p, \dots) = x$.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`GAMMAINV(A3, 3, 4)`

`GAMMAINV(0.8902, R3C8, R3C9)`

`GAMMAINV(0.75, 2, 3)` gives the result 8.0779035867

Version Available

This function is available in product version 1.0 or later.

See Also

GAMMADIST | GAMMALN | Statistical Functions

GAMMALN

This function returns the natural logarithm of the Gamma function, G(x).

Syntax

`GAMMALN(value)`

Arguments

The argument is any numeric value.

Remarks

This function is calculated as the natural logarithm (LN) of the Gamma function.

The equation for this function is:

$$GAMMALN(x) = LN\left(\int_0^{\infty} e^{-u} u^{x-1} du\right)$$

where x is the *value* argument.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`GAMMALN(A4)`

`GAMMALN(R4C1)`

`GAMMALN(12)` gives the result 17.5023078459

Version Available

This function is available in product version 1.0 or later.

See Also

[GAMMADIST](#) | [GAMMAINV](#) | [LN](#) | [Statistical Functions](#)

GAMMALN.PRECISE

This function returns the natural logarithm of the Gamma function, $G(x)$.

Syntax

`GAMMALN.PRECISE(value)`

Arguments

The argument is any numeric value.

Remarks

This function is calculated as the natural logarithm (LN) of the Gamma function. If *value* is nonnumeric, the function returns the #VALUE! error value. If $x \leq 0$, this function returns the #NUM! error value.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`GAMMALN.PRECISE(A4)`

`GAMMALN.PRECISE(R4C1)`

`GAMMALN.PRECISE(12)` gives the result 17.502307845873887

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

GAMMA

GAUSS

This function returns the probability (in the form of a numeric value) specifying that a member of a standard normal population will fall between the mean and specified standard deviations from the mean.

Syntax

GAUSS(*z*)

Arguments

For the argument, you can specify any real number.

Remarks

This function returns an error value if *z* is not a valid number or a valid data type.

Data Types

Accepts numeric data. Returns numeric data.

Examples

GAUSS(10) gives the result 0.5

GAUSS(-5) gives the result -0.499

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

GCD

This function returns the greatest common divisor of two numbers.

Syntax

`GCD(number1,number2)`

Arguments

The arguments are two numeric values or arrays. If the arguments are not integers, they are truncated to integers. This function can have up to 255 arguments.

Remarks

The greatest common divisor is the largest integer that divides both numbers without a remainder.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`GCD(B5,G7)`

`GCD(R5C2,R7C7)`

`GCD(3348,972)` gives the result 108 `GCD(12.8,16.3)` gives the result 4

Version Available

This function is available in product version 1.0 or later.

See Also

LCM | Math and Trigonometry Functions

GEOMEAN

This function returns the geometric mean of a set of positive data.

Syntax

`GEOMEAN(value1,value2,...)`

`GEOMEAN(array)`

`GEOMEAN(array1,array2,...)`

Arguments

You can specify a set of numeric values. You can also use a single array or a reference to an array instead of arguments separated by commas. If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero. This function can have up to 255 arguments.

Data should be provided so that the value arguments should be greater than zero.

Remarks

You can use this function to calculate average growth rate given compound interest with variable rates.

The equation for this function is:

$$GEOMEAN(x_1, x_2, \dots, x_n) = \sqrt[n]{x_1 x_2 \dots x_n}$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`GEOMEAN(F1:F9)`

`GEOMEAN(R1C6:R9C6)`

`GEOMEAN(35,31,47,51,37,31,58,39)` gives the result 40.1461796637

Version Available

This function is available in product version 1.0 or later.

See Also

HARMEAN | **Statistical Functions**

GESTEP

This function, greater than or equal to step, returns an indication of whether a number is equal to a threshold.

Syntax

`GESTEP(number,step)`

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Value to test against the step (which is either step or zero)
<i>step</i>	[Optional] Value of the threshold against which to test; if omitted, uses zero

Remarks

If the *number* is greater than or equal to the *step*, this function returns one. Otherwise it returns zero.

Data Types

Accepts numeric data for all arguments. Returns numeric (0 or 1) data.

Examples

`GESTEP(B5,7)`

`GESTEP(43)` gives the result 1

Version Available

This function is available in product version 1.0 or later.

See Also

DELTA | Engineering Functions

GROWTH

This function calculates predicted exponential growth. This function returns the y values for a series of new x values that are specified by using existing x and y values.

Syntax

`GROWTH(y,x,newx,constant)`

Remarks

This function has these arguments:

Argument	Description
<i>y</i>	Set of y values that are known in the relationship $y=b*m^x$
<i>x</i>	(Optional) X is an optional set of x values that may be known in the relationship $y=b*m^x$
<i>newx</i>	New x values for which this functions returns the corresponding y values
<i>constant</i>	Logical value that specifies whether to force the constant b to equal 1

If constant is true or omitted then b is calculated normally. If constant is false then b is equal to 0 and the m values are adjusted so that $y=m^x$.

If x is omitted then x defaults to the array {1,2,3...}, that has the same dimensions as y.

If newx is omitted then it defaults to x.

Remarks

Use the **INDEX** function to get individual elements from the returned array.

Data Types

Accepts an array. Returns an array.

Examples

`GROWTH(A2:A7,C2:C7,A9:A10)`

Version Available

This function is available in product version 2.0 or later.

See Also

AVEDEV | AVERAGEA | FREQUENCY | DEVSQ | MEDIAN | TREND | VAR | Statistical Functions

Functions H to L

Functions H to L

HARMEAN	HEX2BIN	HEX2DEC	HEX2OCT
HLOOKUP	HOUR	HYPERLINK	HYPGEOM.DIST
HYPGEOMDIST	IF	IFERROR	IFNA
IFS	IMABS	IMAGINARY	IMARGUMENT
IMCONJUGATE	IMCOS	IMCOSH	IMCOT
IMCSC	IMCSCH	IMDIV	IMEXP
IMLN	IMLOG10	IMLOG2	IMPOWER
IMPRODUCT	IMREAL	IMSEC	IMSECH
IMSIN	IMSINH	IMSQRT	IMSUB
IMSUM	IMTAN	INDEX	INDIRECT
INFO	INT	INTERCEPT	INTRATE
IPMT	IRR	ISBLANK	ISERR
ISERROR	ISEVEN	ISFORMULA	ISLOGICAL
ISNA	ISNONTEXT	ISNUMBER	ISO.CEILING
ISODD	ISOWEEKNUM	ISPMT	ISREF
ISTEXT	JIS	KURT	LARGE
LCM	LEFT	LEFTB	LEN
LENB	LINEST	LN	LOG
LOG10	LOGEST	LOGINV	LOGNORM.DIST
LOGNORM.INV	LOGNORMDIST	LOOKUP	LOWER

HARMEAN

This function returns the harmonic mean of a data set.

Syntax

`HARMEAN(value1,value2,...)`

`HARMEAN(array)`

`HARMEAN(array1,array2,...)`

Arguments

You can specify a set of numeric values. You can also use a single array or a reference to an array instead of arguments separated by commas. If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero. This function can have up to 255 arguments.

Data should be provided so that the value arguments should be greater than zero.

Remarks

The harmonic mean is always less than the geometric mean, which is always less than the arithmetic mean

The equation for this function is:

$$HARMEAN(x_n) = \frac{1}{\frac{1}{n} \sum \frac{1}{x}}$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`HARMEAN(F1:F9)`

`HARMEAN(R1C6:R9C6)`

`HARMEAN(35,31,47,51,37,31,58,39)` gives the result 39.2384929823

Version Available

This function is available in product version 1.0 or later.

See Also

GEOMEAN | Statistical Functions

HEX2BIN

This function converts a hexadecimal number to a binary number.

Syntax

HEX2BIN(*number*, *places*)

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Hexadecimal numeric value to convert, must be between FFFFFFFE00 and 1FF
<i>places</i>	[Optional] Number of characters to return; if not an integer, the number is truncated

Remarks

This functions returns an error when the *number* is not a valid hexadecimal value or if the value for *places* is non-numeric or negative. If *places* is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Data Types

Accepts numeric data. Returns numeric data.

Examples

HEX2BIN("F", 5)

Version Available

This function is available in product version 2.0 or later.

See Also

HEX2DEC | HEX2OCT | BIN2HEX | OCT2HEX | Engineering Functions

HEX2DEC

This function converts a hexadecimal number to a decimal number.

Syntax

HEX2DEC(*number*)

Arguments

Specify the number to convert, which is limited to a maximum of 10 characters.

Remarks

An error value is returned if the *number* is invalid or more than 10 characters.

Data Types

Accepts numeric data. Returns numeric data.

Examples

HEX2DEC ("FF")

Version Available

This function is available in product version 2.0 or later.

See Also

HEX2BIN | **HEX2OCT** | **BIN2DEC** | **OCT2DEC** | **Engineering Functions**

HEX2OCT

This function converts a hexadecimal number to an octal number.

Syntax

HEX2OCT(*number*, *places*)

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Hexadecimal numeric value to convert, must be between FFE000000 and 1FFFFFFF
<i>places</i>	[Optional] Number of characters to return; if not an integer, the number is truncated

Remarks

This functions returns an error when the *number* is not a valid hexadecimal number or if the value for *places* is non-numeric or negative. If *places* is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Data Types

Accepts numeric data. Returns numeric data.

Examples

HEX2OCT ("2B")

Version Available

This function is available in product version 2.0 or later.

See Also

HEX2BIN | HEX2DEC | BIN2OCT | DEC2OCT | Engineering Functions

HLOOKUP

This function searches for a value in the top row and then returns a value in the same column from a specified row.

Syntax

`HLOOKUP(value,array,row,approx)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>value</i>	Value to be found in the first row
--------------	------------------------------------

<i>array</i>	Array or range that contains the data to search
--------------	---

<i>row</i>	Row number in the array from which the matching value will be returned
------------	--

<i>approx</i>	[Optional] Logical value indicating whether to find an approximate match; if omitted, uses TRUE and finds an approximate match
---------------	--

Remarks

If *approx* is FALSE, it finds an exact match, not an approximate match. If it cannot find one, it returns an #N/A error value.

If *approx* is TRUE or omitted, and the *value* cannot be found, then the largest value that is less than the *value* is used.

This function is similar to **VLOOKUP** except that it searches by row (horizontally), instead of vertically (by column).

Data Types

Accepts numeric or string data. Returns numeric data.

Examples

```
HLOOKUP("Test",A1:D5,3,TRUE)
```

Version Available

This function is available in product version 2.0 or later.

See Also

VLOOKUP | **LOOKUP** | **Lookup Functions**

HOUR

This function returns the hour that corresponds to a specified time.

Syntax

`HOUR(time)`

Arguments

Specify the *time* argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), a DateTime object, as in DATE(2003,7,4), or a TimeSpan object, as in TIME(12,0,0). For more details on the date and time inputs, refer to the discussion in **Date and Time Functions**

Dates as numeric values are in the form x.y, where x is the "number of days since December 30, 1899" and y is the fraction of day. Numbers to the left represent the date. Times as numeric values are decimal fractions ranging from 0 to 0.99999999, representing the times from 0:00:00 (12:00:00 A.M.) to 23:59:59 (11:59:59 P.M.).

Remarks

The hour is returned as an integer, ranging from 0 (12:00 A.M.) to 23 (11:00 P.M.).

Data Types

Accepts numeric, string, DateTime object, or TimeSpan object data. Returns numeric data.

Examples

`HOUR(A2)`

`HOUR(R2C1)`

`HOUR(0.25)` gives the result 6

`HOUR(347.25)` gives the result 6

`HOUR("2:22 PM")` gives the result 14

`HOUR("2:22 AM")` gives the result 2

`HOUR(TIME(12,0,0))`

Version Available

This function is available in product version 1.0 or later.

See Also

[MINUTE](#) | [SECOND](#) | [Date and Time Functions](#)

HYPERLINK

This function creates shortcut that opens document stored on the Internet.

Syntax

HYPERLINK(*link*,[*display_name*])

Arguments

This function has the following arguments:

Argument	Description
----------	-------------

<i>link</i>	Refers to the URL of the document
-------------	-----------------------------------

<i>display_name</i>	[Optional] Refers to the text that appears in the cell; if this is excluded, whole <i>link</i> appears in the cell
---------------------	--

Remarks

Click the cell and hold the mouse button until pointer becomes a cross, to select the cell and not jump to the hyperlink destination.

Data Types

Accepts string data for both arguments. Returns string data.

Examples

HYPERLINK("<http://grapecity.com>", "Click for demo") gives the result Click for Demo

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

HYPGEOM.DIST

This function returns the hypergeometric distribution.

Syntax

HYPGEOM.DIST(*x,n,M,N,cumulative*)

Arguments

The arguments are as follows, and are truncated if not integers:

Argument	Description
<i>x</i>	An integer representing the number of successes in the sample
<i>n</i>	An integer representing the size of the sample
<i>M</i>	An integer representing the number of successes in the population
<i>N</i>	An integer representing the size of the population
<i>cumulative</i>	A logical value that determines the form of the function. If <i>cumulative</i> is TRUE, then this function returns the cumulative distribution function; if this argument is FALSE, it returns the probability mass function

Remarks

If any argument is nonnumeric, the function returns the #VALUE! error value.

Data Types

Accepts numeric data for all arguments except *cumulative*. Returns numeric data.

Examples

HYPGEOM.DIST(A22,B23,62,1000,C10)

HYPGEOM.DIST(R22C11,R22C12,R34C14,R35C15,R10C5)

HYPGEOM.DIST(2,37,6,100,200) gives the result 0.6079646750428083

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

HYPGEOMDIST

HYPGEOMDIST

This function returns the hypergeometric distribution.

Syntax

`HYPGEOMDIST(x,n,M,N)`

Arguments

The arguments are as follows, and are truncated if not integers:

Argument	Description
<i>x</i>	An integer representing the number of successes in the sample
<i>n</i>	An integer representing the size of the sample
<i>M</i>	An integer representing the number of successes in the population
<i>N</i>	An integer representing the size of the population

Remarks

The equation for this function is:

$$HYPGEOMDIST(x, n, M, N) = \frac{\binom{M}{x} \binom{N-M}{n-x}}{\binom{N}{n}}$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`HYPGEOMDIST(A22,B23,62,1000)`

`HYPGEOMDIST(R22C11,R22C12,R34C14,R35C15)`

`HYPGEOMDIST(2,37,6,100)` gives the result 0.3327981975

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | **GAMMADIST** | **Statistical Functions**

IF

This function performs a comparison and returns one of two provided values based on that comparison.

Syntax

`IF(valueTest,valueTrue,valueFalse)`

Arguments

This function has these arguments:

Argument	Description
<i>valueTest</i>	Value or expression to evaluate
<i>valueTrue</i>	Value to return if the test evaluates to true
<i>valueFalse</i>	Value to return if the test evaluates to false

Remarks

The value of *valueTest* is evaluated. If it is non-zero (or TRUE), then *valueTrue* is returned. If it is zero (or FALSE), then *valueFalse* is returned. The value of *valueTest* must be or evaluate to numeric data, where non-zero values indicate TRUE, and a value of zero indicates FALSE. It may contain one of the relational operators: greater than (>), less than (<), equal to (=), or not equal to (<>).

Data Types

Accepts numeric (boolean) data. Returns any data type.

Example

`IF(A3<>2000,1900,2000)`

`IF(R1C2>65,1000,2000)`

`IF(C4,B2,B4)`

`IF(1>2,5,10)` gives the result 10

`IF(1<2,""dogs"" ,""cats"")` gives the result dogs

Version Available

This function is available in product version 1.0 or later.

See Also

AND | **FALSE** | **Logical Functions**

IFERROR

This function evaluates a formula and returns a value you provide if there is an error or the formula result.

Syntax

IFERROR(*value*,*error*)

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Value or expression to evaluate
<i>error</i>	Value to return if the formula returns an error

Remarks

The following error types are evaluated, #VALUE!, #REF!, #NUM!, #NAME?, #DIV/O, #N/A, or #NULL

Data Types

Accepts any type of formula for the value. Returns any data type.

Example

IFERROR(A3/A5,"dogs")

Version Available

This function is available in product version 5.0 or later.

See Also

AND | **FALSE** | **Logical Functions**

IFNA

This function calculates the specified value if the formula returns #N/A error, otherwise it returns the actual result of the formula.

Syntax

IFNA(*value*, *value_if_na*)

Arguments

This function has the following arguments:

Argument	Description
<i>value</i>	Refers to a number, or expression that needs to be tested
<i>value_if_na</i>	Refers to an alternate number, or expression that is returned if <i>value</i> returns #N/A error

Remarks

If *value* or *value_if_na* is empty, IFNA treats it as an empty string value ("").

Data Types

Accepts numeric, string (or any expression) data for both arguments.

Examples

IFNA(VLOOKUP("Sweden",\$A\$6:\$B\$8,0),"Not found") gives the result Not found.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

IFS

This function performs a comparison and returns the specified associated value if comparison is TRUE.

Syntax

`IFS(condition,value1,[condition2,value2],...)`

Arguments

This function has the following arguments:

Argument	Description
<i>condition1</i>	Refers to a value or expression to evaluate.
<i>value1</i>	Refers to a value to return if the test evaluates to true.
<i>[condition2,value2],...</i>	[Optional] Refers to more conditions to be compared (nested IF statements).

Remarks

This function allows users to test upto 127 different conditions.

Data Types

Accepts numeric (boolean) data for both arguments. Returns any data type.

Examples

`IFS(A2 > 89,"A",A2 > 79,"B")` gives the result B.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

IMABS

This function returns the absolute value or modulus of a complex number.

Syntax

`IMABS(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the absolute value.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns number data.

Examples

```
IMABS("3+5j")
```

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | Engineering Functions | Complex Numbers in Engineering Functions

IMAGINARY

This function returns the imaginary coefficient of a complex number.

Syntax

`IMAGINARY(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the imaginary coefficient.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns number data.

Examples

```
IMAGINARY("3+5j")
```

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | **IMREAL** | **Engineering Functions** | **Complex Numbers in Engineering Functions**

IMARGUMENT

This function returns the argument theta, which is an angle expressed in radians.

Syntax

`IMARGUMENT(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the argument theta.

Remarks

The *complexnum* argument is a complex number for which to return the argument theta.

An error is returned if number is not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns number data.

Examples

`IMARGUMENT ("3+5j ")`

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | IMCOS | IMSIN | Engineering Functions | Complex Numbers in Engineering Functions

IMCONJUGATE

This function returns the complex conjugate of a complex number.

Syntax

`IMCONJUGATE(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the conjugate.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

```
IMCONJUGATE("3+5j")
```

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | IMABS | Engineering Functions | Complex Numbers in Engineering Functions

IMCOS

This function returns the cosine of a complex number.

Syntax

`IMCOS(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the cosine.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

```
IMCOS ("3+5j")
```

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | **IMSIN** | **IMARGUMENT** | **Engineering Functions** | **Complex Numbers in Engineering Functions**

IMCOSH

This function calculates the hyperbolic cosine of specified complex number.

Syntax

`IMCOSH(complex_num)`

Arguments

The *complex_num* argument is a complex number to return the hyperbolic cosine for.

Remarks

An error is returned if the *complex_num* argument is not in the form "x+yi" or "x+yj".

Data Types

Accepts numeric and string data. Returns string data.

Examples

`IMCOSH("8-3i")` gives the result -1475.563

`IMCOSH("7-2i")` gives the result -228.180

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

IMCOT

This function calculates the cotangent of specified complex number.

Syntax

`IMCOT(complex_num)`

Arguments

The *complex_num* argument is a complex number to return the cotangent for.

Remarks

An error is returned if the *complexn_um* argument is not in the form "x+yi" or "x+yj".

Data Types

Accepts numeric and string data. Returns string data.

Examples

`IMCOT("2-3i")` gives the result -0.0037

`IMCOT("1-3i")` gives the result 0.0044

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

IMCSC

This function calculates the cosecant of specified complex number.

Syntax

`IMCSC(complex_num)`

Arguments

The *complex_num* argument is a complex number to return the cosecant for.

Remarks

An error is returned if the *complex_num* argument is not in the form "x+yi" or "x+yj".

Data Types

Accepts numeric and string data. Returns string data.

Examples

`IMCSC("13+3i")` gives the result 0.0420

`IMCSC("7+2i")` gives the result 0.1819

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

IMCSCH

This function calculates the hyperbolic cosecant of specified complex number.

Syntax

`IMCSCH(complex_num)`

Arguments

The *complex_num* argument is a complex number to return the hyperbolic cosecant for.

Remarks

An error is returned if the *complex_num* argument is not in the form "x+yi" or "x+yj".

Data Types

Accepts numeric and string data. Returns string data.

Examples

`IMCSCH("15+i")` gives the result 3.3055

`IMCSCH("8+4i")` gives the result -0.0004

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

IMDIV

This function returns the quotient of two complex numbers.

Syntax

`IMDIV(complexnum,complexdenom)`

Arguments

This function has these arguments:

Argument	Description
<i>complexnum</i>	Complex numerator or dividend
<i>complexdenom</i>	Complex denominator or divisor

Remarks

An error is returned if the arguments are not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMDIV("3+5j","10+20i")`

Version Available

This function is available in product version 2.0 or later.

See Also

[IMPRODUCT](#) | [IMSQRT](#) | [Engineering Functions](#) | [Complex Numbers in Engineering Functions](#)

IMEXP

This function returns the exponential of a complex number.

Syntax

`IMEXP(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the exponential.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMEXP("2+5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

IMLN | **IMLOG10** | **IMLOG2** | **IMPOWER** | **Engineering Functions** | **Complex Numbers in Engineering Functions**

IMLN

This function returns the natural logarithm of a complex number.

Syntax

`IMLN(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the natural logarithm.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

```
IMLN("2+5j")
```

Version Available

This function is available in product version 2.0 or later.

See Also

IMEXP | IMLOG10 | IMLOG2 | Engineering Functions | Complex Numbers in Engineering Functions

IMLOG10

This function returns the common logarithm of a complex number.

Syntax

`IMLOG10(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the common logarithm.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

```
IMLOG10("2+5j")
```

Version Available

This function is available in product version 2.0 or later.

See Also

[IMEXP](#) | [IMLN](#) | [IMLOG2](#) | [Engineering Functions](#) | [Complex Numbers in Engineering Functions](#)

IMLOG2

This function returns the base-2 logarithm of a complex number.

Syntax

`IMLOG2(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the base-2 logarithm.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMLOG2 ("2+5j ")`

Version Available

This function is available in product version 2.0 or later.

See Also

IMEXP | IMLN | IMLOG10 | Engineering Functions | Complex Numbers in Engineering Functions

IMPOWER

This function returns a complex number raised to a power.

Syntax

`IMPOWER(complexnum,powernum)`

Arguments

This function has these arguments:

Argument	Description
<i>complexnum</i>	Complex number to raise to a power
<i>powernum</i>	Power to which to raise the complex number

The power (*powernum* argument) can be an integer, negative, or fractional.

Remarks

An error is returned if *complexnum* is not in the form "x+yi" or "x+yj" or if *powernum* is non-numeric. For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMPOWER("2+5j", 4)`

Version Available

This function is available in product version 2.0 or later.

See Also

[IMEXP](#) | [IMPRODUCT](#) | [Engineering Functions](#) | [Complex Numbers in Engineering Functions](#)

IMPRODUCT

This function returns the product of up to 29 complex numbers in the "x+yi" or "x+yj" text format.

Syntax

`IMPRODUCT(complexnum1,complexnum2, ...)`

Arguments

The arguments are the complex numbers to multiply. There can be up to 29 of them.

Arrays in the x+yi format or range references are allowed.

Remarks

An error is returned if the arguments are not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

```
IMPRODUCT("2+5j", 4)
```

```
IMPRODUCT({"1+2i", "3+4i"})
```

Version Available

This function is available in product version 2.0 or later.

See Also

IMDIV | IMPOWER | Engineering Functions | Complex Numbers in Engineering Functions

IMREAL

This function returns the real coefficient of a complex number in the $x+yi$ or $x+yj$ text format.

Syntax

`IMREAL(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the real coefficient.

Remarks

An error is returned if the *complexnum* argument is not in the form " $x+yi$ " or " $x+yj$ ". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns number data.

Examples

```
IMREAL("2-5j")
```

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | **IMAGINARY** | **Engineering Functions** | **Complex Numbers in Engineering Functions**

IMSEC

This function calculates the secant of specified complex number.

Syntax

`IMSEC(complex_num)`

Arguments

The *complex_num* argument is a complex number to return the secant for.

Remarks

An error is returned if the *complex_num* argument is not in the form "x+yi" or "x+yj".

Data Types

Accepts numeric and string data. Returns string data.

Examples

`IMSEC("11-2i")` gives the result 0.00126

`IMSEC("3-6i")` gives the result -0.00490

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

IMSECH

This function calculates the hyperbolic secant of specified complex number.

Syntax

`IMSECH(complex_num)`

Arguments

The *complex_num* argument is a complex number to return the hyperbolic secant for.

Remarks

An error is returned if the *complex_num* argument is not in the form "x+yi" or "x+yj".

Data Types

Accepts numeric and string data. Returns string data.

Examples

`IMSECH("3+6i")` gives the result 0.09544

`IMSECH("7+2i")` gives the result -0.00075

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

IMSIN

This function returns the sine of a complex number in the $x+yi$ or $x+yj$ text format.

Syntax

`IMSIN(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the sine.

Remarks

An error is returned if the *complexnum* argument is not in the form " $x+yi$ " or " $x+yj$ ". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

```
IMSIN("2-5j")
```

Version Available

This function is available in product version 2.0 or later.

See Also

IMCOS | **IMARGUMENT** | **Engineering Functions** | **Complex Numbers in Engineering Functions**

IMSINH

This function calculates the hyperbolic sine of specified complex number.

Syntax

`IMSINH(complex_num)`

Arguments

The *complex_num* argument is a complex number to return the hyperbolic sine for.

Remarks

An error is returned if the *complex_num* argument is not in the form "x+yi" or "x+yj".

Data Types

Accepts numeric and string data. Returns string data.

Examples

`IMSINH("6+4i")` gives the result -131.848

`IMSINH("4+4i")` gives the result -17.83

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

IMSQRT

This function returns the square root of a complex number in the $x+yi$ or $x+yj$ text format.

Syntax

`IMSQRT(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the square root.

Remarks

An error is returned if the *complexnum* argument is not in the form " $x+yi$ " or " $x+yj$ ". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

```
IMSQRT("2-5j")
```

Version Available

This function is available in product version 2.0 or later.

See Also

IMDIV | IMPRODUCT | Engineering Functions | Complex Numbers in Engineering Functions

IMSUB

This function returns the difference of two complex numbers in the $x+yi$ or $x+yj$ text format.

Syntax

`IMSUB(complexnum1,complexnum2)`

Arguments

The *complexnum1* is a complex number from which to subtract the other complex number *complexnum2*.

Remarks

An error is returned if the arguments are not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

```
IMSUB("2+5j", "5+3i")
```

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | IMSUM | Engineering Functions | Complex Numbers in Engineering Functions

IMSUM

This function returns the sum of two or more complex numbers in the $x+yi$ or $x+yj$ text format.

Syntax

`IMSUM(complexnum1,complexnum2, ...)`

Arguments

The arguments are the complex numbers to multiply. There can be up to 29 of them.

Arrays in the "x+yi" or "x+yj" format or range references are allowed.

Remarks

An error is returned if the arguments are not in the form "x+yi" or "x+yj". For more information, refer to **Complex Numbers in Engineering Functions**.

Data Types

Accepts number and string data. Returns string data.

Examples

```
IMSUM("2+5j", "5+3i")
```

```
IMSUM(A1:B5)
```

```
IMSUM({"2+5j", "5+3i"})
```

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | **IMSUB** | **Engineering Functions** | **Complex Numbers in Engineering Functions**

IMTAN

This function calculates the tangent of specified complex number.

Syntax

`IMTAN(complex_num)`

Arguments

The *complex_num* argument is a complex number to return the tangent for.

Remarks

An error is returned if the *complex_num* argument is not in the form "x+yi" or "x+yj".

Data Types

Accepts numeric and string data. Returns string data.

Examples

`IMTAN("8+2i")` gives the result -0.0109

`IMTAN("9+3i")` gives the result -0.0037

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

INDEX

This function returns a value or the reference to a value from within an array or range.

Syntax

`INDEX(return, row, col, area)`

Arguments

The arguments are as follows, and are truncated if not integers:

Argument	Description
<i>return</i>	Returns a value or a reference of a cell or range of cells
<i>row</i>	Row number in the range
<i>col</i>	Column number in the range
<i>area</i>	[If <i>return</i> is a cell range reference] Area of the range

Data Types

Accepts numeric data. Returns numeric data.

Examples

`INDEX (A2 : C3, 2, 2)`

`INDEX (R2C1 : R3C3, 5, 3)`

Version Available

This function is available in product version 1.0 or later.

See Also

CHOOSE | Lookup Functions

INDIRECT

This function returns the reference specified by a text string.

Syntax

INDIRECT(*Reftext*,*A1*)

Arguments

This function has these arguments:

Argument	Description
<i>Reftext</i>	A reference to a cell that contains an A1 reference, an R1C1 reference, a name defined as a reference, or a text string reference to a cell. This argument is required.
<i>A1</i>	A logical value that specifies what type of reference is contained in the cell <i>Reftext</i> . This argument is optional.

Remarks

Use INDIRECT when you want to change the reference to a cell within a formula without changing the formula itself. *Reftext* is interpreted as an A1 reference if *A1* is TRUE or omitted. *Reftext* is interpreted as an R1C1 reference if *A1* is FALSE. If *Reftext* is not a valid cell reference, INDIRECT returns the #REF! error value. Changing a cell value causes the INDIRECT function and all dependent cells to recalculate.

Data Types

Accepts any data. Returns any data type.

Example

INDIRECT("A1")

INDIRECT(A1)

INDIRECT("R[-&B1&]C[-&B2&]", false)

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

LOOKUP

INFO

This function returns information about the operating system.

Syntax

INFO(*text*)

Arguments

The *text* argument specifies the type of information to be returned.

The following types of text are available:

Text Type	Returns
directory	Current directory path or folder
numfile	Number of active worksheets
origin	Returns the absolute cell reference of the top and leftmost cell visible in the window, based on the current scrolling position
osversion	Current operating system version (text)
recalc	Current recalculation mode
release	Microsoft Excel version (text)
system	Operating environment

Data Types

Accepts any data. Returns any data type.

Example

INFO("recalc") gives the result Automatic.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

INT

This function rounds a specified number down to the nearest integer.

Syntax

`INT(value)`

Arguments

Use any numeric value for the argument.

Remarks

You can use this function to return the decimal portion of the value in a cell by subtracting the value of this function for the cell from the value in the cell, as illustrated in the first example.

The **TRUNC** and INT functions are similar in that both return integers. Use the TRUNC function to remove the decimal portion of the number; the TRUNC function does not round up or down. Use the INT function to round numbers down to the nearest integer-based decimal portion of the number. These functions differ also when using negative numbers: TRUNC(-4.2) returns -4, but INT(-4.2) returns -5 because -5 is the lower number.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`INT(A3)`

`R1C2-INT(R1C2)`

`INT(2.85)` gives the result 2

`INT(-2.85)` gives the result -3

Version Available

This function is available in product version 1.0 or later.

See Also

CEILING | EVEN | FLOOR | TRUNC | Math and Trigonometry Functions

INTERCEPT

This function returns the coordinates of a point at which a line intersects the y-axis, by using existing x values and y values.

Syntax

`INTERCEPT(dependent, independent)`

Arguments

This function has these arguments:

Argument	Description
<i>dependent</i>	An array of known dependent values (y's)
<i>independent</i>	An array of known independent values (x's)

You can use numbers, arrays, or references for the arguments.

Remarks

The intercept point is based on a best-fit regression line plotted through the known x-values and known y-values. Use the intercept when you want to determine the value of the dependent variable when the independent variable is 0 (zero). For example, you can use this function to predict a metal's electrical resistance at 0°C when your data points were taken at room temperature and higher.

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

The number of dependent data points must be equal to the number of independent data points.

The equation for this function is:

$$INTERCEPT(Y, X) = \bar{Y} - \left[\frac{n \sum xy - \sum x \sum y}{n \sum x - (\sum x)^2} \right] \bar{X}$$

where Y is the array of dependent variables, X is the array of independent variables, and n is the size of the arrays.

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

`INTERCEPT (G1:G9, F1:F9)`

`INTERCEPT (R1C7:R9C7, R1C6:R9C6)`

`INTERCEPT ({53000,57000,58000,69000,74500,55620,80000, 68700},{35,31,47,51,37,31,58,39})` gives the result 37060.4809987149

Version Available

This function is available in product version 1.0 or later.

See Also

FORECAST | Statistical Functions

INTRATE

This function calculates the interest rate for a fully invested security.

Syntax

`INTRATE(settle,mature,invest,redeem,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security.
<i>mature</i>	Maturity date for the security.
<i>invest</i>	Amount invested in the security.
<i>redeem</i>	Amount to be received at maturity.
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when *settle* or *mature* is invalid. Settle, mature, and basis are truncated to integers. If invest or redeem is less than or equal to 0, a #NUM! error is returned. If basis is less than 0 or greater than 4, a #NUM! error is returned. If settle is greater than or equal to mature, a #NUM! error is returned.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`INTRATE(A1,B3,70000,72000,3)`

`INTRATE(R1C1,R4C4,82000,86500,2)`

`INTRATE("3/1/2003","5/31/2003",65000,70000,2)` gives the result 0.304311074

Version Available

This function is available in product version 1.0 or later.

See Also

ACCRINT | **EFFECT** | **RATE** | **RECEIVED** | **Financial Functions**

IPMT

This function calculates the payment of interest on a loan.

Syntax

`IPMT(rate,per,nper,pval,fval,type)`

Arguments

This function has these arguments:

Argument	Description
<i>rate</i>	Value of interest rate per period.
<i>per</i>	Number of the period for which to find the interest, between 1 and <i>nper</i>
<i>nper</i>	Total number of payment periods in an annuity.
<i>pval</i>	Present value, worth now
<i>fval</i>	[Optional] Future value, cash value after the last payment; if omitted, the calculation uses zero
<i>type</i>	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

Remarks

The result is represented by a negative number because it is money paid out by you.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`IPMT(0.65,A1,B3,C42)`
`IPMT(R1C1,R12C12,R13C13,R32C1)`
`IPMT(0.45, 2, 30, 6000)` gives the result `-$2,699.98`

Version Available

This function is available in product version 1.0 or later.

See Also

PMT | PPMT | RATE | Financial Functions

IRR

This function returns the internal rate of return for a series of cash flows represented by the numbers in an array.

Syntax

`IRR(arrayvals,estimate)`

Remarks

This function has these arguments:

Argument	Description
<i>arrayvals</i>	An array of numbers for which you want to estimate the internal rate of return representing payments and income occurring at regular intervals (and use positive for income and negative for payment)
<i>estimate</i>	[Optional] An estimate of the internal rate of return; if omitted, the calculation uses 0.1 (10 percent)

Values must contain at least one positive value (some income) and one negative value (a payment) to calculate the internal rate of return.

Remarks

This function uses the order of values to interpret the order of payments and income. Be sure to enter your payment and income values in the sequence you want with correct signs. The payments and income must occur at regular time intervals, such as monthly or annually.

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

The function is calculated using an iterative technique. Starting with the estimate, this function cycles through the calculation until the result is accurate within 0.00001 (0.001 percent). If this function cannot find a result that works after 50 iterations, it returns an error.

If the function returns an error or if the result is not close to what you expected, try again with a different value for the estimate.

This function is closely related to NPV, the net present value function. The rate of return calculated by IRR is the interest rate corresponding to a 0 (zero) net present value.

For a schedule of cash flows that is non-periodic, use **XIRR**.

Data Types

Accepts numeric data for both arguments, the first being an array. Returns numeric data.

Examples

`IRR(D1:D6)`

`IRR(R1C4:R6C4, -.02)`

Version Available

This function is available in product version 1.0 or later.

See Also

MIRR | NPV | XIRR | Financial Functions

ISBLANK

This function tests whether a value, an expression, or contents of a referenced cell is empty.

Syntax

`ISBLANK(cellreference)`

`ISBLANK(value)`

`ISBLANK(expression)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

This function returns TRUE if the value refers to an empty cell or to no data.

Note: Spread's implementation of functions generally tries to follow the behavior found in popular spreadsheet applications. However, not all these applications agree whether the empty string "" should be treated the same as an empty cell. In Spread, both the **COUNTBLANK** and **ISBLANK** functions consistently treat the empty string "" differently than an empty cell.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISBLANK(B1)`

`ISBLANK(A4)`

`ISBLANK(A4-52)`

`ISBLANK(4)` gives the result FALSE

Version Available

This function is available in product version 1.0 or later.

See Also

COUNTBLANK | **ISERROR** | **ISREF** | **ISTEXT** | **Information Functions**

ISERR

This function, Is Error Other Than Not Available, tests whether a value, an expression, or contents of a referenced cell has an error other than not available (#N/A).

Syntax

ISERR(*cellreference*)

ISERR(*value*)

ISERR(*expression*)

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

This function returns TRUE if the value refers to an empty cell or to no data.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

ISERR(B12)

ISERR(R12C2)

ISERR(#N/A) gives the result FALSE

ISERR(#REF!) gives the result TRUE

ISERR(C14) gives the result TRUE if C14 contains a #NUM! error.

Version Available

This function is available in product version 1.0 or later.

See Also

ERRORTYPE | ISERROR | ISNA | Information Functions

ISERROR

This function, Is Error of Any Kind, tests whether a value, an expression, or contents of a referenced cell has an error of any kind.

Syntax

`ISERROR(cellreference)`

`ISERROR(value)`

`ISERROR(expression)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

This function returns TRUE if the value refers to an empty cell or to no data.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISERROR(B12)`

`ISERROR(R12C2)`

`ISERROR(#N/A)` gives the result TRUE

`ISERROR(#REF!)` gives the result TRUE

Version Available

This function is available in product version 1.0 or later.

See Also

ERRORTYPE | ISERR | ISNA | Information Functions

ISEVEN

This function, Is Number Even, tests whether a value, an expression, or contents of a referenced cell is even.

Syntax

`ISEVEN(cellreference)`

`ISEVEN(value)`

`ISEVEN(expression)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

If the number specified by the argument is even, the function returns TRUE. If the number specified by the argument is odd, the function returns FALSE. If the number specified by the argument is zero, the function returns TRUE. If the number specified by the argument refers to an empty cell or to no data, the function returns TRUE.

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

Data Types

Accepts numeric data. Returns Boolean (TRUE or FALSE) data.

Examples

`ISEVEN(B3)`

`ISEVEN(R1C2)`

`ISEVEN(574)` gives the result TRUE

`ISEVEN(9)` gives the result FALSE

`ISEVEN(2.4)` gives the result TRUE

`ISEVEN(3.6)` gives the result FALSE

`ISEVEN(ROUND(3.6))` gives the result TRUE

Version Available

This function is available in product version 1.0 or later.

See Also

ISODD | EVEN | Information Functions

ISFORMULA

This function tests whether a value, an expression, or contents of a reference cell is a formula.

Syntax

ISFORMULA(*reference*)

Arguments

Specify the cell reference for the argument.

Remarks

This function returns FALSE if the value refers to an empty cell or to no data.

This function is used to test the contents of a cell.

Data Types

Accepts cell reference for argument. Returns Boolean (TRUE or FALSE) data.

Examples

ISFORMULA(B4) returns false, where B4 is a cell reference with a numeric value 56.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

ISLOGICAL

This function tests whether a value, an expression, or contents of a referenced cell is a logical (Boolean) value.

Syntax

`ISLOGICAL(cellreference)`

`ISLOGICAL(value)`

`ISLOGICAL(expression)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

This function returns FALSE if the value refers to an empty cell or to no data.

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISLOGICAL(B7)`

`ISLOGICAL(R4C8)`

`ISLOGICAL(true)` gives a result TRUE

`ISLOGICAL(OR(B7,B8))` gives a result TRUE

Version Available

This function is available in product version 1.0 or later.

See Also

ISNONTEXT | ISNUMBER | ISTEXT | Information Functions

ISNA

This function, Is Not Available, tests whether a value, an expression, or contents of a referenced cell has the not available (#N/A) error value.

Syntax

`ISNA(cellreference)`

`ISNA(value)`

`ISNA(expression)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

This function returns TRUE if the value is or refers to the Not Available error value, and returns FALSE if the value is or refers to a cell with no data.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISNA(B12)`

`ISNA(R12C2)`

`ISNA(#N/A)` gives the result TRUE

`ISNA(NA())` gives the result TRUE

`ISNA(#REF)` gives the result FALSE

Version Available

This function is available in product version 1.0 or later.

See Also

ERRORTYPE | ISERR | ISERROR | NA | Information Functions

ISNONTEXT

This function tests whether a value, an expression, or contents of a referenced cell has any data type other than text.

Syntax

`ISNONTEXT(cellreference)`

`ISNONTEXT(value)`

`ISNONTEXT(expression)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

This function returns TRUE if the value refers to a blank cell.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISNONTEXT(A3)`

`ISNONTEXT(R3C1)`

`ISNONTEXT(12)` gives the result TRUE

`ISNONTEXT("Total")` gives the result FALSE

Version Available

This function is available in product version 1.0 or later.

See Also

ISLOGICAL | **ISNUMBER** | **ISTEXT** | **Information Functions**

ISNUMBER

This function tests whether a value, an expression, or contents of a referenced cell has numeric data.

Syntax

`ISNUMBER(cellreference)`

`ISNUMBER(value)`

`ISNUMBER(expression)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

This function returns TRUE if the argument is or refers to a number, and returns FALSE if the argument is or refers to a value that is not a number. This function returns FALSE if the value is or refers to a cell with no data.

You might want to use this function to test whether cells contain numeric data before you perform mathematical operations on them, such as averaging the contents of a range of cells.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISNUMBER(B3)`

`ISNUMBER(R1C2)`

`ISNUMBER(12)` gives the result TRUE

Version Available

This function is available in product version 1.0 or later.

See Also

[ISLOGICAL](#) | [ISNONTTEXT](#) | [ISREF](#) | [ISTEXT](#) | [N](#) | **Information Functions**

ISO.CEILING

This function rounds a number up to the nearest integer or multiple of a specified value.

Syntax

ISO.CEILING(*value*,*signif*)

Arguments

This function has these arguments:

Argument	Description
<i>value</i>	Number to round
<i>signif</i>	[Optional] Number representing the rounding factor, default value is 1

If the number or the *signif* is zero, zero is returned. The absolute value of the multiple is used, so this function returns the mathematical ceiling regardless of the number signs and significance.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

ISO.CEILING(4.3) gives the result 5

ISO.CEILING(-2.5, 2) gives the result -2

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

CEILING

ISODD

This function, Is Number Odd, tests whether a value, an expression, or contents of a referenced cell has numeric data.

Syntax

`ISODD(cellreference)`

`ISODD(value)`

`ISODD(expression)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

If the number specified by the argument is odd, the function returns TRUE. If the number specified by the argument is even, the function returns FALSE. If the number specified by the argument is zero, the function returns FALSE. If the number specified by the argument refers to an empty cell or to no data, the function returns TRUE.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISODD(B3)`

`ISODD(R1C2)`

`ISODD(12)` gives the result FALSE

`ISODD(2.5)` gives the result FALSE

`ISODD(3.6)` gives the result TRUE

`ISODD(ROUND(3.6))` gives the result FALSE

Version Available

This function is available in product version 1.0 or later.

See Also

ISEVEN | ODD | Information Functions

ISOWEEKNUM

This function calculates the number ISO week number of the year for specified date.

Syntax

ISOWEEKNUM(*date*)

Arguments

Specify the *date* argument as a number (as in 37806), a string or reference to cell containing the information.

Data Types

Accepts numeric or string data for both arguments. Returns numeric data.

Examples

ISOWEEKNUM("5/9/2015") gives the result 19.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

ISPMT

This function calculates the interest paid during a specific period of an investment.

Syntax

`ISPMT(rate,per,nper,pv)`

Remarks

This function has these arguments:

Argument	Description
<i>rate</i>	Interest rate for the investment.
<i>per</i>	Number of the period for which to find the interest, between 1 and <i>nper</i> .
<i>nper</i>	Total number of payment periods for the investment.
<i>pv</i>	Present value of the investment.

Remarks

Be consistent with the units for *rate* and *nper*.

The cash you pay out is represented by negative numbers and the cash you receive by positive numbers.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`ISPMT (B1, C4, C5, 1)`

`ISPMT (R1C2, R4C3, R6C3, R7C3)`

Version Available

This function is available in product version 2.0 or later.

See Also

IPMT | PMT | PV | Financial Functions

ISREF

This function, Is Reference, tests whether a value, an expression, or contents of a referenced cell is a reference to another cell.

Syntax

ISREF(*cellreference*)

ISREF(*value*)

ISREF(*expression*)

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

If the argument is a reference, this function returns TRUE. If the argument is not a reference, this function returns FALSE.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

ISREF(B3) gives the result TRUE

ISREF(R1C2) gives the result TRUE

ISREF(12) gives the result FALSE

Version Available

This function is available in product version 1.0 or later.

See Also

ISBLANK | **Information Functions**

ISTEXT

This function tests whether a value, an expression, or contents of a referenced cell has text data.

Syntax

`ISTEXT(cellreference)`

`ISTEXT(value)`

`ISTEXT(expression)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

If the data type of the argument is text, this function returns TRUE. If the data type of the argument is not text, this function returns FALSE. If the argument refers to an empty cell, this function returns FALSE.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISTEXT(B3)`

`ISTEXT(R1C2)`

`ISTEXT("Total")` gives the result TRUE

`ISTEXT(12)` gives the result FALSE

Version Available

This function is available in product version 1.0 or later.

See Also

ISLOGICAL | **ISNONTTEXT** | **ISNUMBER** | **T** | **Information Functions**

JIS

This function transforms half-width (single-byte) characters to full-width (double-byte) characters.

Syntax

JIS(text)

Arguments

For the argument, text or a reference to a cell that contains the text to change.

Remarks

If the text does not contain half-width letters, then the text is not changed.

Data Types

Accepts string data. Returns string data.

Examples

JIS("SPREAD") gives the result "SPREAD"

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

KURT

This function returns the kurtosis of a data set.

Syntax

`KURT(value1,value2,value3,value4,...)`

`KURT(array)`

`KURT(array1,array2,...)`

Arguments

For the arguments, you can use numbers, arrays, or references. If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes cells with the value zero in its calculations.

You must provide four or more value arguments. You may provide up to 255 arguments.

Remarks

Kurtosis describes how peaked or flat a distribution is compared with the normal distribution. Positive kurtosis indicates a relatively peaked distribution. Negative kurtosis indicates a relatively flat distribution.

If the standard deviation of the values is zero, this function returns the #DIV/0! error value.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`KURT(F1:F8)`

`KURT(R1C6:R8C6)`

`KURT(F1:F8,G1:G8)`

`KURT(35,31,47,51,37,31,58,39)` gives the result `-0.7496238078`

Version Available

This function is available in product version 1.0 or later.

See Also

GAMMADIST | Statistical Functions

LARGE

This function returns the n th largest value in a data set, where n is specified.

Syntax

`LARGE(array,n)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>array</i>	Array from which to return the n th largest value
--------------	---

<i>n</i>	The position (from the largest value) for which to return the value (for example, 5 to return the fifth largest value). Must be equal to or less than the number of items in the array.
----------	---

Remarks

Use this function to select a value based on its relative standing. For example, you can use it to return the third-place score in a competition.

Data Types

Accepts array and numeric data for all arguments. Returns numeric data.

Examples

`LARGE(F1:F8,2)`

`LARGE(R1C6:R8C6,5)`

`LARGE({35,31,47,51,37,31,58,39},3)` gives the result 47.0000000000

Version Available

This function is available in product version 1.0 or later.

See Also

SMALL | **Statistical Functions**

LCM

This function returns the least common multiple of two numbers.

Syntax

`LCM(number1,number2)`

Arguments

For the arguments, use numeric values or arrays. If the arguments are not integers, they are truncated to integers. This function can have up to 255 arguments.

Remarks

The least common multiple is the smallest positive integer that is a multiple of all integers given.

Use this function to add fractions with different denominators by calculating the least common multiple of both denominators first.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`LCM(B12,C22)`

`LCM(R12C2,R22C3)`

`LCM(300,500)` gives the result 1500

`LCM(12.3,16.99)` gives the result 48

Version Available

This function is available in product version 1.0 or later.

See Also

GCD | Math and Trigonometry Functions

LEFT

This function returns the specified leftmost characters from a text value.

Syntax

`LEFT(mytext,num_chars)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>mytext</i>	Text string that contains the characters you want to extract.
---------------	---

<i>num_chars</i>	[Optional] Number of characters to extract; if omitted, uses one; if not an integer, the number is truncated
------------------	--

The *mytext* argument can be a string, a formula that returns a string, or a reference to a cell containing a string.

The *num_chars* argument has these rules:

- It must be greater than or equal to zero.
- If it is greater than the length of text, this function returns all the text.

Data Types

Accepts string data for the first argument and numeric data the second argument. Returns string data.

Examples

```
LEFT(A2, LEN(A2) - 1)
```

```
LEFT(R2C1, LEN(R2C1) - 1)
```

```
LEFT("TotalPrice") gives the result T
```

```
LEFT("Total Price", 5) gives the result Total
```

Version Available

This function is available in product version 1.0 or later.

See Also

MID | **RIGHT** | **Text Functions**

LEFTB

This function returns the specified leftmost characters from a text value on the basis of the number of bytes in the specified value.

Syntax

LEFTB(*mytext*, *num_bytes*)

Arguments

This function has the following arguments:

Argument	Description
----------	-------------

<i>mytext</i>	Refers to the text string that contains the characters you want to extract.
<i>num_bytes</i>	[Optional] Refers to the number of bytes to extract. If this value is not given, it uses one. If this value is not an integer, the number is truncated.

Remarks

The *mytext* argument can be a string, a formula that returns a string, or a reference to a cell containing a string.

The *num_bytes* argument has the following rules:

- It must be greater than or equal to zero.
- If it is greater than the length of text, this function returns all the text.

Data Types

Accepts string data for the first argument and numeric data the second argument. Returns string data.

Examples

LEFTB("rheabuto",4) gives the result rhea.

LEFTB("rheabuto") gives th result r.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

LEN

This function returns the length of, the number of characters in, a text string.

Syntax

`LEN(value)`

Arguments

The argument is the text whose length you want to find. Spaces count as characters. The argument must be a string or a cell reference to a string value.

Data Types

Accepts string data. Returns numeric data.

Examples

```
LEFT(A2, LEN(A2) - 1)
```

```
LEN("FarPoint Technologies, NC") gives the result 25
```

Version Available

This function is available in product version 1.0 or later.

See Also

CHAR | TRIM | Text Functions

LENB

This function returns the number of bytes used to represent the characters in a text string.

Syntax

LENB(*text_value*)

Arguments

The argument is the text whose length you want to find. Spaces in the specified text value are counted as characters. The argument must be a string or a cell reference to a string value.

Remarks

The LENB function counts 2 bytes per character, but this happens only when a DBCS language is set as the default language.

Data Types

Accepts string data. Returns numeric data.

Examples

LENB("rheabuto") gives the result 8.

LENB("rosy") gives the result 4.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

LINEST

This function calculates the statistics for a line.

Syntax

`LINEST(y,x,constant,stats)`

Arguments

The equation for the line is $y=mx+b$ or $y=m_1x_1+m_2x_2+\dots+b$.

This function has these arguments:

Argument	Description
<i>y</i>	Set of y values that are known in the relationship $y=mx+b$
<i>x</i>	(Optional) X is an optional set of x values that may be known in the relationship $y=mx+b$
<i>constant</i>	Logical value that specifies whether to force the constant b to equal 0. If true or omitted then b is calculated normally; if false then b is equal to 0 and the m values are adjusted so that $y=mx$.
<i>stats</i>	Logical value that specifies whether to return additional regression statistics. If true, then the additional regression statistics are returned if false or omitted then only the m-coefficients and b are returned.

If x is omitted then x defaults to the array {1,2,3...}, that has the same dimensions as y.

Remarks

Use the **INDEX** function to get individual elements from the returned array.

Data Types

Accepts an array. Returns an array.

Examples

`LINEST(A2:A7,C2:C7,,FALSE)`

Version Available

This function is available in product version 2.0 or later.

See Also

GROWTH | TREND | LOGEST | DEVSQ | MEDIAN | VAR | Statistical Functions

LN

This function returns the natural logarithm of the specified number.

Syntax

`LN(value)`

Arguments

For the argument, specify a positive numeric value.

Remarks

This function is the inverse of **EXP**, so LN(EXP(x)) is x.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`LN(B3)`

`LN(R1C2)`

`LN(10)` gives the result 2.3025850930

`LN(exp(1))` gives the result 1

Version Available

This function is available in product version 1.0 or later.

See Also

EXP | **LOG** | **LOGINV** | **Math and Trigonometry Functions**

LOG

This function returns the logarithm base Y of a number X.

Syntax

`LOG(number,base)`

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>number</i>	Number for which to find a logarithm. This must be a positive real number
---------------	---

<i>base</i>	[Optional] Base of the logarithm; if omitted, the calculation uses 10 as the base (See LOG10 .)
-------------	--

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`LOG(B3,C5)`

`LOG(R1C2,R4C4)`

`LOG(255,16)` gives the result 1.9985883592

Version Available

This function is available in product version 1.0 or later.

See Also

LN | **LOG10** | **Math and Trigonometry Functions**

LOG10

This function returns the logarithm base 10 of the number given.

Syntax

`LOG10(value)`

Arguments

The number specified by the argument must be a positive real number.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`LOG10(B3)`

`LOG10(R1C2)`

`LOG10(115)` gives the result 2.0606978404

Version Available

This function is available in product version 1.0 or later.

See Also

LN | LOG | Math and Trigonometry Functions

LOGEST

This function calculates an exponential curve that fits the data and returns an array of values that describes the curve.

Syntax

`LOGEST(y,x,constant,stats)`

Arguments

The equation for the curve is $y=b*m^x$ or $y=(b*(m1^x1)*(m2^x2)*...)$.

This function has these arguments:

Argument Description

<i>y</i>	Set of y values that are known in the relationship $y=b*m^x$
<i>x</i>	(Optional) X is an optional set of x values that may be known in the relationship $y=mx+b$
<i>constant</i>	Logical value that specifies whether to force the constant b to equal 0. If true or omitted then b is calculated normally; if false then b is equal to 0 and the m values are adjusted so that $y=m^x$.
<i>stats</i>	Logical value that specifies whether to return additional regression statistics. If true, then the additional regression statistics are returned if false or omitted then only the m-coefficients and b are returned.

If x is omitted then x defaults to the array {1,2,3...}, that has the same dimensions as y.

Remarks

Use the **INDEX** function to get individual elements from the returned array.

Data Types

Accepts an array. Returns an array.

Examples

`LOGEST(A2:A7,C2:C7,TRUE,FALSE)`

Version Available

This function is available in product version 2.0 or later.

See Also

GROWTH | TREND | LINEST | DEVSQ | MEDIAN | VAR | Statistical Functions

LOGINV

This function returns the inverse of the lognormal cumulative distribution function of x , where $\mathbf{LN}(x)$ is normally distributed with the specified mean and standard deviation.

Syntax

`LOGINV(prob,mean,stdev)`

Arguments

This function has these arguments:

Argument	Description
<i>prob</i>	Value at which to evaluate the function
<i>mean</i>	Value of mean of natural logarithm of x , $\mathbf{LN}(x)$
<i>stdev</i>	Value representing the standard deviation of $\mathbf{LN}(x)$

Remarks

This function calculates the inverse of the lognormal cumulative distribution functions, so if $p = \mathbf{LOGNORMDIST}(x, \dots)$ then $\mathbf{LOGINV}(p, \dots) = x$.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`LOGINV(0.92, B8, G22)`

`LOGINV(0.88, 2, 1.2)` gives the result 30.26479297

Version Available

This function is available in product version 1.0 or later.

See Also

[LN](#) | [LOGNORMDIST](#) | [Statistical Functions](#)

LOGNORM.DIST

This function returns the cumulative natural log normal distribution of x , where $\text{LN}(x)$ is normally distributed with the specified mean and standard deviation. Analyze data that has been logarithmically transformed with this function.

Syntax

`LOGNORM.DIST(x , $mean$, $stdev$, $cumulative$)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

x	Value at which to evaluate the function
$mean$	Value of mean of natural logarithm of x , $\text{LN}(x)$
$stdev$	Value representing the standard deviation of $\text{LN}(x)$
$cumulative$	A logical value that determines the form of the function. If $cumulative$ is <code>TRUE</code> , this function returns the cumulative distribution function; if <code>FALSE</code> , it returns the probability density function

Remarks

If any argument is nonnumeric, this function returns the `#VALUE!` error value.

The equation for the lognormal cumulative distribution function is:

$\text{LOGNORM.DIST}(x,\mu,o) = \text{NORM.S.DIST}(\ln(x)-\mu / o)$.

Data Types

Accepts numeric data for x , $mean$, and $stdev$ arguments. Accepts `TRUE` or `FALSE` for $cumulative$. Returns numeric data.

Examples

`LOGNORM.DIST(0.92,B8,G22,A5)`

`LOGNORM.DIST(42,2,1.2,TRUE)` gives the result 0.9261995869896625

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

LOGNORMDIST

LOGNORM.INV

This function returns the inverse of the lognormal cumulative distribution function of x , where $\text{LN}(x)$ is normally distributed with the specified mean and standard deviation.

Syntax

`LOGNORM.INV(prob,mean,stdev)`

Arguments

This function has these arguments:

Argument	Description
<i>prob</i>	Value at which to evaluate the function
<i>mean</i>	Value of mean of natural logarithm of x , $\text{LN}(x)$
<i>stdev</i>	Value representing the standard deviation of $\text{LN}(x)$

Remarks

This function calculates the inverse of the lognormal cumulative distribution functions, so if $p = \text{LOGNORM.DIST}(x, \dots)$ then $\text{LOGNORM.INV}(p, \dots) = x$. The #VALUE! error value is returned if any argument is nonnumeric.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`LOGNORM.INV(0.92,B8,G22)`

`LOGNORM.INV(0.88,2,1.2)` gives the result 30.264764580330958

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

LOGNORMDIST

LOGNORMDIST

This function returns the cumulative natural log normal distribution of x , where $\text{LN}(x)$ is normally distributed with the specified mean and standard deviation. Analyze data that has been logarithmically transformed with this function.

Syntax

`LOGNORMDIST(x,mean,stdev)`

Arguments

This function has these arguments:

Argument	Description
<i>x</i>	Value at which to evaluate the function
<i>mean</i>	Value of mean of natural logarithm of x , $\text{LN}(x)$
<i>stdev</i>	Value representing the standard deviation of $\text{LN}(x)$

Remarks

If $p = \text{LOGNORMDIST}(x, \dots)$ then $\text{LOGINV}(p, \dots) = x$.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`LOGNORMDIST(0.92, B8, G22)`

`LOGNORMDIST(42, 2, 1.2)` gives the result 0.926199546

Version Available

This function is available in product version 1.0 or later.

See Also

[LN](#) | [LOGINV](#) | [Statistical Functions](#)

LOOKUP

This function searches for a value and returns a value from the same location in a second area.

Syntax

LOOKUP(*lookupvalue*,*lookupvector*,*resultvector*)

LOOKUP(*lookupvalue*,*lookuparray*)

Arguments

Vector Form

The arguments for the vector form are:

Argument	Description
----------	-------------

<i>lookupvalue</i>	Value for which to search; can be number, text, logical value, or name or reference that refers to a value
--------------------	--

<i>lookupvector</i>	Cell range that contains one row or one column; can be text, numbers, or a logical value; values need to be in ascending order
---------------------	--

<i>resultvector</i>	Cell range that contains one row or column; must be the same size as <i>lookupvector</i>
---------------------	--

Array Form

The arguments for the array form are:

Argument	Description
----------	-------------

<i>lookupvalue</i>	Value for which to search; can be number, text, logical value, or name or reference that refers to a value
--------------------	--

<i>lookuparray</i>	Range of cells that contains text, numbers, or logical values; values must be ascending order
--------------------	---

Remarks

Vector Form

The vector form of this function searches for a value from a range with a single row or column and returns a value from the same location in a second one row or one column range.

In the vector form, if *lookupvalue* can not be found, it matches the largest value in *lookupvector* that is less than or equal to *lookupvalue*.

Array Form

The array form of this function searches in the first row or column of an array for the specified value and returns a value from the same location in the last row or column of the array.

In the array form, if *lookuparray* has more columns than rows then the first row is searched. If *lookuparray* has more rows than columns then the first column is searched. The values in *lookuparray* must be in ascending order.

Data Types

Accepts numeric or string data. Returns numeric or string data.

Examples

`LOOKUP(30,A1:A5,B1:B5)`

`LOOKUP("A",{"a","b","c","d";1,2,3,5})`

Version Available

This function is available in product version 2.0 or later.

See Also

HLOOKUP | VLOOKUP | Lookup Functions

LOWER

This function converts text to lower case letters.

Syntax

`LOWER(string)`

Arguments

The argument is the text you want to convert to lower case. This function does not change characters in value that are not letters. The argument may be a string, a reference to a cell containing a string, or a formula that returns a string.

Data Types

Accepts string data. Returns string data.

Examples

`LOWER(A4)`

`LOWER(R4C1)`

`LOWER("Road Race 2")` gives the result road race 2

`LOWER(CONCATENATE(A1,A5))`

Version Available

This function is available in product version 1.0 or later.

See Also

UPPER | T | Text Functions

Functions M to Q

Functions M to Q

MATCH	MAX	MAXA	MAXIFS
MDETERM	MDURATION	MEDIAN	MID
MIDB	MIN	MINA	MINIFS
MINUTE	MINVERSE	MIRR	MMULT
MOD	MODE	MODE.MULT	MODE.SNGL
MONTH	MROUND	MULTINOMIAL	MUNIT
N	NA	NEGBINOM.DIST	NEGBINOMDIST
NETWORKDAYS	NETWORKDAYS.INTL	NOMINAL	NORM.DIST
NORM.INV	NORM.S.DIST	NORM.S.INV	NORMDIST
NORMINV	NORMSDIST	NORMSINV	NOT
NOW	NPER	NPV	NUMBERVALUE
OCT2BIN	OCT2DEC	OCT2HEX	ODD
ODDFPRICE	ODDFYIELD	ODDLPRICE	ODDLYIELD
OFFSET	OR	PDURATION	PEARSON
PERCENTILE	PERCENTILE.EXC	PERCENTILE.INC	PERCENTRANK
PERCENTRANK.EXC	PERCENTRANK.INC	PERMUT	PERMUTATIONA
PHI	PHONETIC	PI	PMT
POISSON	POISSON.DIST	POWER	PPMT
PRICE	PRICEDISC	PRICEMAT	PROB
PRODUCT	PROPER	PV	QUARTILE
QUARTILE.EXC	QUARTILE.INC	QUOTIENT	

MATCH

This function returns the relative position of a specified item in a range.

Syntax

`MATCH(value1,array,type)`

Arguments

You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Argument	Description
<i>value</i>	Value to search for
array	Range to search in
type	[Optional] Value to return if the formula returns an error

Remarks

The value can be a number, text, or logical value or a cell reference to a number, text, or logical value. The array is the range of cells to search.

The type can be 0 (first value that is equal to value), 1 (largest value that is less than or equal to value), or -1 (smallest value that is greater than or equal to value) and is optional.

Data Types

The value can be a number, text, or logical value or a cell reference to a number, text, or logical value. Returns numeric data.

Examples

`MATCH(25,A1:E5)`

Version Available

This function is available in product version 5.0 or later.

See Also

MIN | LOOKUP | Lookup Functions

MAX

This function returns the maximum value, the greatest value, of all the values in the arguments.

Syntax

`MAX(value1,value2,...)`

`MAX(array)`

`MAX(array1,array2,...)`

Arguments

Each argument can be a double-precision floating point value, an integer value, or an array of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

This function differs from **MAXA**, which allows text and logical values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`MAX(A1,B2,C3,D4,E5)`

`MAX(A1:A9)`

`MAX(R1C2:R1C15,R2C2:R2C15)`

`MAX(2,15,12,3,7,19,4)` gives the result 19

Version Available

This function is available in product version 1.0 or later.

See Also

MIN | **MAXA** | **Statistical Functions**

MAXA

This function returns the largest value in a list of arguments, including text and logical values.

Syntax

`MAXA(value1,value2,...)`

`MAXA(array)`

`MAXA(array1,array2,...)`

Arguments

Each argument can be a double-precision floating point value, an integer value, text, or logical values. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This function differs from **MAX** because it allows text and logical values as well as numeric values.

Data Types

Accepts numeric, text, or logical data for all arguments. Returns numeric data.

Examples

`MAXA(A1,B2,C3,D4,E5)`

`MAXA(A1:A9)`

`MAXA(R1C2:R1C15,R2C2:R2C15)`

`MAXA(2,15,12,3,7,19,4)` gives the result 19

Version Available

This function is available in product version 2.0 or later.

See Also

MINA | **MAX** | **Statistical Functions**

MAXIFS

This function returns the maximum value among the values in cells or the cell range provided the specified set of conditions or the criteria meets.

Syntax

MAXIFS(*max_range*, *range1*, *criteria1*, [*range2*,*criteria2*],...)

Arguments

This function has the following arguments:

Argument	Description
<i>max_range</i>	Refers to the range of cells in which maximum will be calculated
<i>range1</i>	Refers to the set of cells to be calculated based on the criteria
<i>criteria1</i>	Refers to the criteria (in terms of number, expression, or text) according to which cells will be calculated
<i>range2</i> , <i>criteria2</i> ...	[Optional] Refers to the additional ranges and their corresponding criteria

Remarks

If the size and shape of *max_range* and *rangeN* (*refers to range1,2,3,4,.....N*) arguments is not same, this function will return the #VALUE! error.

If cells do not match with the specified criteria, this function will return 0.

Data Types

Accepts either numeric data, expression, or text for all arguments. Returns numeric data.

Examples

MAXIFS(C4:C7,B4:B7,"325") gives the result 656

MAXIFS(C4:C7,B4:B7,"55") gives the result 0

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

MDETERM

This function returns the matrix determinant of an array.

Syntax

`MDETERM(array)`

Arguments

The array is a numeric array that has an equal number of columns and rows.

Arrays can be a cell range. If any of the array cells are empty or contain text then an error is returned.

Data Types

Accepts an array. Returns numeric data.

Examples

`MDETERM(A3:E7)`

Version Available

This function is available in product version 2.0 or later.

See Also

MINVERSE | **MMULT** | **Math and Trigonometry Functions**

MDURATION

This function calculates the modified Macauley duration of a security with an assumed par value of \$100.

Syntax

MDURATION(*settlement,maturity,coupon,yield,frequency,basis*)

Arguments

This function has these arguments:

Argument	Description
<i>settlement</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>coupon</i>	Annual coupon rate
<i>yield</i>	Annual yield for the security
<i>frequency</i>	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when *settlement* or *maturity* is invalid or a #NUM! error when *frequency* is a number other than 1, 2, or 4. If coupon is less than 0 or yield is less than 0, a #NUM! error is returned. If basis is less than 0 or greater than 4, a #NUM! error is returned. If settlement is greater than or equal to maturity, a #NUM! error is returned.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

MDURATION(A1, B2, C3, D4, E5, F6)

Version Available

This function is available in product version 2.0 or later.

See Also

DURATION | **Financial Functions**

MEDIAN

This function returns the median, the number in the middle of the provided set of numbers; that is, half the numbers have values that are greater than the median, and half have values that are less than the median.

Syntax

`MEDIAN(value1,value2,...)`

`MEDIAN(array)`

`MEDIAN(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

If there are an even number of arguments, the function calculates the average of the two numbers in the middle.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`MEDIAN(A3,B5,C1,D4,E7)`

`MEDIAN(A1:A9)`

`MEDIAN(R1C2,R3C5,R4C7,R6C7)`

`MEDIAN(89,95,76,88,92)` gives the result 89

Version Available

This function is available in product version 1.0 or later.

See Also

[AVERAGE](#) | [MODE](#) | [Statistical Functions](#)

MID

This function returns the requested number of characters from a text string starting at the position you specify.

Syntax

`MID(text,start_num,num_chars)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>text</i>	Text string containing the characters you want to extract
<i>start_num</i>	Number representing the first character you want to extract in text, with the first character in the text having a value of one (1); if not an integer, the number is truncated
<i>num_chars</i>	Number of characters to return from text; if not an integer, the number is truncated

The *text* argument can be a string, a formula that returns a string, or a reference to a cell containing a string. The *start_num* argument has these rules

- If *start_num* is greater than the length of *text*, this function returns "" (empty text). If *start_num* is less than the length of *text*, but *start_num* plus *num_chars* exceeds the length of *text*, this function returns the characters up to the end of text.

Data Types

Accepts string data for the text argument, numeric data for the *start_num* argument, and numeric data for the *num_chars* argument. Returns string data.

Examples

`MID(B17, 5, 8)`

`MID("wind surfing", 6, 20)` gives the result `surfing`

Version Available

This function is available in product version 1.0 or later.

See Also

[LEFT](#) | [RIGHT](#) | [Text Functions](#)

MIDB

This function returns the requested number of characters from a text string starting at the position you have specified (on the basis of the specified number of bytes).

Syntax

MIDB(*text*,*start_num*,*num_bytes*)

Arguments

This function has the following arguments:

Argument	Description
----------	-------------

<i>text</i>	Refers to the text string containing the characters you want to extract.
<i>start_num</i>	Refers to the number representing the first character you want to extract in text, with the first character in the text having a value of one (1). If this value is not an integer, the number is truncated.
<i>num_bytes</i>	Refers to the number of characters to return from text (in bytes). If this value is not an integer, the number is truncated.

The *text* argument can be a string or a formula that returns a string, or a reference to a cell containing that string.

The *start_num* argument works on the following rule:

- If *start_num* is greater than the length of *text*, this function returns "" (empty text). If *start_num* is less than the length of *text*, but *start_num* plus *num_bytes* exceeds the length of *text*, this function returns the characters up to the end of text.

Remarks

The MIDB function counts 2 bytes per character, but this happens only when a DBCS language is set as the default language.

Data Types

Accepts string data for the text argument, numeric data for the *start_num* argument, and numeric data for the *num_bytes* argument. Returns string data.

Examples

MIDB("rosy garden", 6, 20) gives the result garden.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

MIN

This function returns the minimum value, the least value, of all the values in the arguments.

Syntax

`MIN(value1,value2,...)`

`MIN(array)`

`MIN(array1,array2,...)`

Arguments

Each argument can be a double-precision floating point value, an integer value, or an array of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

This function differs from **MINA**, which includes text and logical values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`MIN(A3,B5,C1,D4,E7)`

`MIN(A1:A9)`

`MIN(R1C2,R3C5,R4C7,R6C7)`

`MIN(2,15,12,3,7,19,4)` gives the result 2

Version Available

This function is available in product version 1.0 or later.

See Also

MAX | **MINA** | **Statistical Functions**

MINA

This function returns the minimum value in a list of arguments, including text and logical values.

Syntax

`MINA(value1,value2,...)`

`MINA(array)`

`MINA(array1,array2,...)`

Arguments

Each argument can be a double-precision floating point value, an integer value, text, logical value, or an array of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This function differs from **MIN** because it includes text and logical values as well as numeric values.

Data Types

Accepts numeric, text, or logical data for all arguments. Returns numeric data.

Examples

`MINA(A3,B5,C1,D4,E7)`

`MINA(A1:A9)`

`MINA(R1C2,R3C5,R4C7,R6C7)`

`MINA(A1,B1)` gives the result 0 if A1 is 10 and B1 is FALSE

Version Available

This function is available in product version 2.0 or later.

See Also

MIN | **MAXA** | **Statistical Functions**

MINIFS

This function returns the minimum value among the values in cells or the cell range provided the specified set of conditions or the criteria meets.

Syntax

`MINIFS(max_range, range1, criteria1, [range2,criteria2],...)`

Arguments

This function has the following arguments:

Argument	Description
<i>min_range</i>	Refers to the range of cells in which minimum will be calculated
<i>range1</i>	Refers to the set of cells to be calculated based on criteria
<i>criteria1</i>	Refers to the criteria (in term of number, expression, or text) according to which cells will be calculated
<i>range2, criteria2...</i>	[Optional] Refers to the additional ranges and thier corresponding criteria

Remarks

If the size and shape of *min_range* and *rangeN* (refers to *range1,2,3,4,.....N*) arguments is not same, this function will return the #VALUE! error.

If cells do not match with the specified criteria, this function will return 0.

Data Types

Accepts either numeric data, expression, or text for all arguments. Returns numeric data.

Examples

`MINIFS(A3:A7,B3:B7,23)` gives the result 0.

`MINIFS(A3:A7,B3:B7,325)` gives the result 325.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

MINUTE

This function returns the minute corresponding to a specified time.

Syntax

`MINUTE(time)`

Arguments

Specify the time argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), a DateTime object, as in `DATE(2003,7,4)`, or a TimeSpan object, as in `TIME(12,0,0)`. For more details on the date and time inputs, refer to the discussion in **Date and Time Functions**

Dates as numeric values are in the form x.y, where x is the "number of days since December 30, 1899" and y is the fraction of day. Numbers to the left represent the date. Times as numeric values are decimal fractions ranging from 0 to 0.99999999, representing the times from 0:00:00 (12:00:00 A.M.) to 23:59:59 (11:59:59 P.M.).

Remarks

The minute is returned as an integer, ranging from 0 to 59.

Data Types

Accepts numeric, string, DateTime object, or TimeSpan object data. Returns numeric data.

Examples

`MINUTE(D1)`

`MINUTE(R1C4)`

`MINUTE(0.7)` gives the result 48

`MINUTE("12:17")` gives the result 17

`MINUTE(TIME(12,0,0))`

Version Available

This function is available in product version 1.0 or later.

See Also

hour | **second** | **Date and Time Functions**

MINVERSE

This function returns the inverse matrix for the matrix stored in an array.

Syntax

`MINVERSE(array)`

Arguments

The array is a numeric array that has an equal number of columns and rows.

Arrays can be a cell range. If any of the array cells are empty or contain text then an error is returned.

Remarks

Use the **INDEX** function to get individual elements from the returned array.

Data Types

Accepts an array. Returns an array.

Examples

`MINVERSE(A3:E7)`

Version Available

This function is available in product version 2.0 or later.

See Also

MDETERM | **MMULT** | **Math and Trigonometry Functions**

MIRR

This function returns the modified internal rate of return for a series of periodic cash flows.

Syntax

`MIRR(arrayvals,payment_int,income_int)`

Arguments

This function has these arguments:

Argument	Description
<i>arrayvals</i>	An array of numbers for which you want to estimate the internal rate of return representing payments and income occurring at regular intervals (and use positive for income and negative for payment)
<i>payment_int</i>	Interest rate on money in cash flows
<i>income_int</i>	Interest rate on money invested from cash flows

Values must contain at least one positive value (some income) and one negative value (a payment) to calculate the internal rate of return. The payments and income must occur at regular time intervals, such as monthly or annually.

Remarks

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

This function uses the order of values to interpret the order of payments and income. Be sure to enter your payment and income values in the sequence you want with correct signs. The payments and income must occur at regular time intervals, such as monthly or annually.

Data Types

Accepts numeric data for all arguments, the first being an array. Returns numeric data.

Examples

`MIRR(D1:D6, D10, D12)`

`MIRR(R1C4:R6C4, R10C4, R12C4)`

`MIRR({7300,-15000,4036,3050},6.5%,8%)` gives the result 0.0564050548577524

Version Available

This function is available in product version 1.0 or later.

See Also

IRR | XIRR | Financial Functions

MMULT

This function returns the matrix product for two arrays.

Syntax

`MMULT(array1,array2)`

Arguments

The arrays are numeric arrays where the columns in array1 match the rows in array2.

Arrays can be a cell range. If any of the array cells are empty or contain text then an error is returned.

Remarks

Use the **INDEX** function to get individual elements from the returned array.

Data Types

Accepts an array for all arguments. Returns an array.

Examples

`MMULT(A2:B3,D5:E6)`

Version Available

This function is available in product version 2.0 or later.

See Also

MDETERM | **MINVERSE** | **Math and Trigonometry Functions**

MOD

This function returns the remainder of a division operation.

Syntax

`MOD(dividend,divisor)`

Arguments

This function has these arguments:

Argument	Description
<i>dividend</i>	Number for which you want to find the remainder by dividing the divisor into it
<i>divisor</i>	Number by which you want to divide the dividend argument

Remarks

The remainder has the same sign as the divisor.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`MOD(B3,10)`

`MOD(C4,B2)`

`MOD(R1C2,12)`

`MOD(255,16)` gives the result 15

`MOD(-3,2)` gives the result 1

Version Available

This function is available in product version 1.0 or later.

See Also

PRODUCT | QUOTIENT | Math and Trigonometry Functions

MODE

This function returns the most frequently occurring value in a set of data.

Syntax

`MODE(value1,value2,...)`

`MODE(array)`

`MODE(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

If no value occurs more than once, the function does not return a value. If more than one value occurs the same number of times, the function returns the first value that repeats that same number of times.

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`MODE(A3,B3,C3,D3)`

`MODE(A1:A9)`

`MODE(R1C2,12,10,R2C3)`

`MODE(A2:A9,B2:B9,B12:35)`

`MODE(89,95,88,97,88,74)` gives the result 88

`MODE(1,2,2,3,4,5,5)` gives the result 2

Version Available

This function is available in product version 1.0 or later.

See Also

AVERAGE | **MEDIAN** | **Statistical Functions**

MODE.MULT

Summary

This function returns a vertical array of the most frequently occurring value in a set of data.

Syntax

MODE.MULT(*value1,value2,...*)

MODE.MULT(*array*)

MODE.MULT(*array1,array2,...*)

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

If no value occurs more than once, the function does not return a value. If more than one value occurs the same number of times, the function returns the first value that repeats that same number of times.

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

MODE.MULT(A3,B3,C3,D3)

MODE.MULT(A1:A9)

MODE.MULT(R1C2,12,10,R2C3)

MODE.MULT(A2:A9,B2:B9,B12:35)

MODE.MULT(89,95,88,97,88,74) gives the result 88

MODE.MULT(1,2,2,3,4,5,5) gives the result 2

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

MODE.SNGL

Summary

This function returns the most frequently occurring value in a set of data.

Syntax

`MODE.SNGL(value1,value2,...)`

`MODE.SNGL(array)`

`MODE.SNGL(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

If no value occurs more than once, the function does not return a value. If more than one value occurs the same number of times, the function returns the first value that repeats that same number of times.

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

This function measures the central tendency which is the center location of a group of numbers in a statistical distribution. Some common measures of tendency are average, median, and mode. Average is the arithmetic mean, and is calculated by adding a group of numbers and then dividing by the number count. Median is the middle number of a group of numbers where half of the numbers have values that are greater than the median and half of the numbers have values that are less than the median.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`MODE.SNGL(A3,B3,C3,D3)`

`MODE.SNGL(A1:A9)`

`MODE.SNGL(R1C2,12,10,R2C3)`

`MODE.SNGL(A2:A9,B2:B9,B12:35)`

`MODE.SNGL(89,95,88,97,88,74)` gives the result 88

`MODE.SNGL(1,2,2,3,4,5,5)` gives the result 2

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

MONTH

This function returns the month corresponding to the specified date value.

Syntax

`MONTH(date)`

Arguments

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in `DATE(2003,7,4)`. For more details on the date inputs, refer to the discussion in **Date and Time Functions**.

Remarks

The month is returned as an integer, ranging from 1 (January) to 12 (December).

Data Types

Accepts numeric, string, or DateTime object data. Returns numeric data.

Examples

`MONTH(L4)`

`MONTH(R4C12)`

`MONTH(366)` gives the result 12

`MONTH("12/17/2004")` gives the result 12

Version Available

This function is available in product version 1.0 or later.

See Also

DAY | **EOMONTH** | **YEAR** | **Date and Time Functions**

MROUND

This function returns a number rounded to the desired multiple.

Syntax

`MROUND(number,multiple)`

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Numeric value to round
<i>multiple</i>	Numeric value representing the rounded result

Remarks

This function rounds to the nearest multiple (either up or down). For even numbers where there may be two choices (one rounding up and one rounding down), the result is the number farther from zero. For example, `MROUND(18,4)` returns 20 even though 16 is as near since 20 is farther from zero. For `MROUND(-18,-4)` returns -20 since that value is farther from zero.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`MROUND(B14,3)`

`MROUND(R14C2,5)`

`MROUND(100,8)` gives the result 104

`MROUND (11,8)` gives the result 8

`MROUND (12,8)` gives the result 16

`MROUND (13,8)` gives the result 16

`MROUND (-12,-8)` gives the result -16

`MROUND (50,8)` gives the result 48

`MROUND (-50,-8)` gives the result -48

Version Available

This function is available in product version 1.0 or later.

See Also

ROUND | Math and Trigonometry Functions

MULTINOMIAL

This function calculates the ratio of the factorial of a sum of values to the product of factorials.

Syntax

`MULTINOMIAL(value1,value2,...)`

`MULTINOMIAL(array)`

`MULTINOMIAL(array1,array2,...)`

Arguments

The arguments are the values to calculate in the multinomial. Each argument can be a double-precision floating point value, an integer value, or an array of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`MULTINOMIAL(D5,D6,D7,D8)`

`MULTINOMIAL(R5C4,R6C4,R7C4,R8C4)`

`MULTINOMIAL(1,2,3)` gives the result 60

Version Available

This function is available in product version 1.0 or later.

See Also

MODE | Math and Trigonometry Functions

MUNIT

This function returns the unit matrix for the specified dimension.

Syntax

MUNIT(*dimension*)

Arguments

For the argument, you need to provide an integer that specifies the dimension of the unit matrix to return. The argument is required.

Remarks

The argument must be greater than 0. The #VALUE! error value is returned if *dimension* is equal to or less than 0.

Data Types

Accepts numeric data. Returns a resultant matrix.

Examples

MUNIT(2) will return a unit matrix 2x2.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

N

This function returns a value converted to a number.

Syntax

`N(value)`

Arguments

Use any value as the argument.

Remarks

It is not always necessary to use this function, because Spread automatically converts values as necessary in many cases.

Data Types

Accepts many types of data. Returns numeric data.

Examples

`N(G12)`

`N(R12C7)`

`N(2.53)` gives the result 2.53

`N(TRUE)` gives the result 1

Version Available

This function is available in product version 1.0 or later.

See Also

ISNUMBER | **Information Functions**

NA

This function returns the error value #N/A that means "not available."

Syntax

NA()

Arguments

This function does not require an argument.

Remarks

It is necessary to include empty parentheses with this function.

Data Types

Returns an error value.

Examples

`NA()`

`NA(R12C7)`

`ISNA(NA())` gives the result `TRUE`

Version Available

This function is available in product version 1.0 or later.

See Also

[ISNA](#) | [ISNUMBER](#) | [Information Functions](#)

NEGBINOM.DIST

This function returns the negative binomial distribution.

Syntax

NEGBINOM.DIST(*x,r,p,cumulative*)

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>x</i>	An integer representing the number of failures in trials
<i>r</i>	An integer representing the threshold number of successes
<i>p</i>	Probability of success on each trial A number between 0 and 1
<i>cumulative</i>	A logical value that determines the form of the function. If <i>cumulative</i> is TRUE, this function returns the cumulative distribution function; if FALSE, it returns the probability density function

Remarks

The number of successes is fixed and the number of trials is variable. If *p* is less than 0 or greater than 1, the function returns the #NUM! error value. If *x* is less than 0 an error is returned. If *r* is less than 1 an error is returned.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

NEGBINOM.DIST(B1,C15,0.335,TRUE)

NEGBINOM.DIST(R1C2,R15C3,0.75,TRUE)

NEGBINOM.DIST(4,13,0.85,TRUE) gives the result 0.9012900017858557

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

NEGBINOMDIST

NEGBINOMDIST

This function returns the negative binomial distribution.

Syntax

`NEGBINOMDIST(x,r,p)`

Arguments

This function has these arguments:

Argument	Description
<i>x</i>	An integer representing the number of failures in trials
<i>r</i>	An integer representing the threshold number of successes
<i>p</i>	Probability of success on each trial A number between 0 and 1.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`NEGBINOMDIST (B1, C15, 0.335)`

`NEGBINOMDIST (R1C2, R15C3, 0.75)`

`NEGBINOMDIST (4, 13, 0.85)` gives the result 0.111399299

Version Available

This function is available in product version 1.0 or later.

See Also

[BINOMDIST](#) | [HYPGEOMDIST](#) | [Statistical Functions](#)

NETWORKDAYS

This function returns the total number of complete working days between the start and end dates.

Syntax

`NETWORKDAYS(startdate,enddate,holidays)`

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>startdate</i>	Date that is the starting date; a number (as in 37806.5), or a DateTime object, as in DATE(2003,7,4)
<i>enddate</i>	Date that is the ending date; a number (as in 37806.5), or a DateTime object, as in DATE(2003,7,4)
<i>holidays</i>	[Optional] Range of dates to exclude from the calculation; if omitted, the calculation assumes no holidays and all weekdays are workdays

Data Types

Accepts numeric, string, or DateTime object data. Returns numeric data.

Examples

`NETWORKDAYS(L4,L5)`

`NETWORKDAYS(R4C12,R1C1,R2C2)`

Version Available

This function is available in product version 2.0 or later.

See Also

WORKDAY | **NOW** | **Date and Time Functions**

NETWORKDAYS.INTL

This function returns the total number of complete working days between the start and end dates.

Syntax

NETWORKDAYS.INTL(*startdate*,*enddate*,*weekend*,*holidays*)

Arguments

This function has these arguments:

Argument	Description
<i>startdate</i>	Date that is the starting date; a number (as in 37806.5), or a DateTime object, as in DATE(2003,7,4)
<i>enddate</i>	Date that is the ending date; a number (as in 37806.5), or a DateTime object, as in DATE(2003,7,4)
<i>weekend</i>	[Optional] A number or string that specifies when weekends occur. Weekend days are days of the week that are not included in the number of whole working days between <i>startdate</i> and <i>enddate</i>
<i>holidays</i>	[Optional] Range of dates to exclude from the calculation. Holidays can be a range of cells that contain the dates, or an array constant of the serial values that represent those dates

The following table lists the *weekend* number values:

Number	Day
1 or omitted	Saturday, Sunday
2	Sunday, Monday
3	Monday, Tuesday
4	Tuesday, Wednesday
5	Wednesday, Thursday
6	Thursday, Friday
7	Friday, Saturday
11	Sunday only
12	Monday only
13	Tuesday only
14	Wednesday only
15	Thursday only
16	Friday only
17	Saturday only

Remarks

Weekend string values are seven characters long and each character in the string represents a day of the week, starting with Monday. A non-workday is 1 and a workday is 0. Only characters 1 and 0 are allowed in the string. The string 1111111 always returns 0.

Weekend days and holidays are not considered to be workdays.

Data Types

Accepts numeric, string, or DateTime object data. Returns numeric data.

Examples

`NETWORKDAYS.INTL(L4,L5)`

`NETWORKDAYS.INTL(R4C12,R1C1,R2C2)`

`NETWORKDAYS.INTL(DATE(2014,9,1),DATE(2014,9,30), "0000111",{ "2014/9/2", "2014/9/3" })` gives the result 16

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

NETWORKDAYS

NOMINAL

This function returns the nominal annual interest rate for a given effective rate and number of compounding periods per year.

Syntax

`NOMINAL(effrate,comper)`

Arguments

This function has these arguments:

Argument	Description
<i>effrate</i>	Value representing the effective interest rate
<i>comper</i>	Number of compounding periods per year; if not an integer, the number is truncated

Remarks

This function returns a #VALUE! error if *effrate* or *comper* is nonnumeric. If *effrate* is less than or equal to 0 or if *comper* is less than 1, a #NUM! error is returned.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`NOMINAL(A4,A5)`

`NOMINAL(R4C1,3)`

`NOMINAL(6.2336%,2)` gives the result 0.061393703

`NOMINAL(6.2336%,6)` gives the result 0.060776004

Version Available

This function is available in product version 1.0 or later.

See Also

EFFECT | INTRATE | Financial Functions

NORM.DIST

This function returns the normal distribution for the specified mean and standard deviation.

Syntax

`NORM.DIST(x,mean,stdev,cumulative)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>x</i>	Value for which to find the distribution
----------	--

<i>mean</i>	Arithmetic mean of the distribution
-------------	-------------------------------------

<i>stdev</i>	Standard deviation of the distribution Must be greater than zero.
--------------	---

<i>cumulative</i>	Set to TRUE to return the cumulative distribution function. Set to FALSE to return the probability mass function.
-------------------	---

Remarks

If *mean* = 0, *stdev* = 1, and *cumulative* = TRUE, this function returns the standard normal distribution, NORMSDIST.

Data Types

The *x*, *mean*, and *stdev* arguments accept numeric data. The *cumulative* argument accepts logical data. Returns numeric data.

Examples

`NORM.DIST(10,A3,B17,FALSE)`

`NORM.DIST(10,R3C1,R17C2,FALSE)`

`NORM.DIST(37,41.125,9.86,TRUE)` gives the result 0.3378430609671818

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

NORMDIST

NORM.INV

This function returns the inverse of the normal cumulative distribution for the given mean and standard deviation.

Syntax

`NORM.INV(prob,mean,stdev)`

Arguments

This function has these arguments:

Argument	Description
<i>prob</i>	Probability of the normal distribution
<i>mean</i>	Arithmetic mean of the distribution
<i>stdestdev</i>	Standard deviation of the distribution; must be greater than zero

Remarks

The #VALUE! error value is returned if any argument is nonnumeric.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`NORM.INV(B3,C12,D14)`

`NORM.INV(R3C2,R12C3,R14C4)`

`NORM.INV(0.978,32,0.252)` gives the result 32.50755088397007

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

NORMINV

NORM.S.DIST

This function returns the standard normal cumulative distribution function.

Syntax

`NORM.S.DIST(value,cumulative)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>value</i>	A numeric value for which you want the distribution
--------------	---

<i>cumulative</i>	A logical value that determines the form of the function. If <i>cumulative</i> is TRUE, the function returns the cumulative distribution function; if FALSE, it returns the probability mass function
-------------------	---

Remarks

The distribution has a mean of zero and a standard deviation of one.

Use this function in place of a table of standard normal curve areas.

If *value* is nonnumeric, the #VALUE! error value is returned.

Data Types

Accepts numeric data for *value*. Accepts TRUE or FALSE for *cumulative*. Returns numeric data.

Examples

`NORM.S.DIST(F1,TRUE)`

`NORM.S.DIST(R1C6,TRUE)`

`NORM.S.DIST(1.288,TRUE)` gives the result 0.901127

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

NORMSDIST

NORM.S.INV

This function returns the inverse of the standard normal cumulative distribution. The distribution has a mean of zero and a standard deviation of one.

Syntax

`NORM.S.INV(prob)`

Arguments

The argument is the probability for the normal distribution.

Remarks

The #VALUE! error value is returned if any argument is nonnumeric.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`NORM.S.INV(A3)`

`NORM.S.INV(R1C2)`

`NORM.S.INV(0.9244)` gives the result 1.435305714537128

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

NORMSDIST

NORMDIST

This function returns the normal cumulative distribution for the specified mean and standard deviation.

Syntax

`NORMDIST(x,mean,stdev,cumulative)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>x</i>	Value for which to find the distribution
<i>mean</i>	Arithmetic mean of the distribution
<i>stdev</i>	Standard deviation of the distribution Must be greater than zero.
<i>cumulative</i>	Set to TRUE to return the cumulative distribution function. Set to FALSE to return the probability mass function.

Remarks

If *mean* = 0 and *stdev* = 1, this function returns the standard normal distribution, NORMSDIST.

Data Types

The *x*, *mean*, and *stdev* arguments accept numeric data. The *cumulative* argument accepts logical data. Returns numeric data.

Examples

```
NORMDIST(10,A3,B17,FALSE)
```

```
NORMDIST(10,R3C1,R17C2,FALSE)
```

```
NORMDIST(37,41.125,9.86,TRUE) gives the result 0.3378810361
```

Version Available

This function is available in product version 1.0 or later.

See Also

[NORMINV](#) | [NORMSDIST](#) | [Statistical Functions](#)

NORMINV

This function returns the inverse of the normal cumulative distribution for the given mean and standard deviation.

Syntax

`NORMINV(prob,mean,stdev)`

Arguments

This function has these arguments:

Argument	Description
<i>prob</i>	Probability of the normal distribution
<i>mean</i>	Arithmetic mean of the distribution
<i>stdevstdev</i>	Standard deviation of the distribution Must be greater than zero.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`NORMINV (B3, C12, D14)`

`NORMINV (R3C2, R12C3, R14C4)`

`NORMINV(0.978,32,0.252)` gives the result 32.50755174

Version Available

This function is available in product version 1.0 or later.

See Also

[NORMDIST](#) | [NORMSINV](#) | [Statistical Functions](#)

NORMSDIST

This function returns the standard normal cumulative distribution function.

Syntax

`NORMSDIST(value)`

Arguments

The argument can be any numeric value.

Remarks

The distribution has a mean of zero and a standard deviation of one.

Use this function in place of a table of standard normal curve areas.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`NORMSDIST(F1)`

`NORMSDIST(R1C6)`

`NORMSDIST(1.288)` gives the result 0.901127

Version Available

This function is available in product version 1.0 or later.

See Also

NORMDIST | NORMSINV | Statistical Functions

NORMSINV

This function returns the inverse of the standard normal cumulative distribution. The distribution has a mean of zero and a standard deviation of one.

Syntax

`NORMSINV(prob)`

Arguments

The argument is the probability for the normal distribution.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`NORMSINV(A3)`

`NORMSINV(R1C2)`

`NORMSINV(0.9244)` gives the result 1.43530571453713

Version Available

This function is available in product version 1.0 or later.

See Also

[NORMINV](#) | [NORMSDIST](#) | [Statistical Functions](#)

NOT

This function reverses the logical value of its argument.

Syntax

`NOT(value)`

Arguments

Provide a numeric or logical value for the argument.

Remarks

If the specified value is zero, then the function returns TRUE. If the specified value is a value other than zero, then the function returns FALSE.

Data Types

Accepts boolean data (TRUE or FALSE). Returns boolean data (TRUE or FALSE).

Examples

`NOT(A3)`

`NOT(R1C2)`

`NOT(D5>100)`

`NOT(0)` gives the result TRUE

`NOT(TRUE)` gives the result FALSE

`NOT(12)` gives the result FALSE

Version Available

This function is available in product version 1.0 or later.

See Also

AND | OR | Logical Functions

NOW

This function is used to determine the current date and time. It returns a serial number that represents the current date and time in Excel. The results are updated each time a worksheet is opened or refreshed.

Syntax

NOW()

Arguments

This function does not accept arguments.

Remarks

This function is updated only when the spreadsheet or cell containing the function is recalculated. This is a volatile function with version 2.5 or later.

Data Types

Does not accept data. Returns a numeric value.

Examples

If it is 04:50:00 P.M. , February 14, 2019, then:

NOW() gives the result 43510.70174

Version Available

This function is available in product version 1.0 or later. This function is a volatile function in version 2.5 or later.



Note: If a user uses LegacyBehaviors.CalculationEngine, the NOW function will return the DateTime object instead of the serial numeric value. For more details, please refer to [breaking changes for legacy behaviors](#).

See Also

DATEVALUE | TIME | **Date and Time Functions**

NPER

This function returns the number of periods for an investment based on a present value, future value, periodic payments, and a specified interest rate.

Syntax

`NPER(rate,paymt,pval,fval,type)`

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>rate</i>	Interest rate expressed as percentage (per period)
<i>paymt</i>	Payment made each period; cannot change over life of the annuity
<i>pval</i>	Present value
<i>fval</i>	[Optional] Future value; if omitted, the calculation uses zero (0)
<i>type</i>	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

For the arguments, money paid out (such as deposits in an investment) is represented by negative numbers; money you receive (such as dividend checks) is represented by positive numbers.

Remarks

Be sure to express the interest rate as per period. For example, if you make monthly payments on a loan at 8 percent interest, use 0.08/12 for the rate argument.

See the **PV** function for the equations for calculating financial values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

```
NPER(A1/12,50,1000,0,1)
```

```
NPER(R1C1/12,50,1000,0,1)
```

```
NPER(0.005,-790,90000,0,1) gives the result 167.7227522114
```

Version Available

This function is available in product version 1.0 or later.

See Also

FV | **PMT** | **PV** | **Financial Functions**

NPV

This function calculates the net present value of an investment by using a discount rate and a series of future payments and income.

Syntax

`NPV(discount,value1,value2,...)`

Arguments

This function has these arguments:

Argument	Description
<i>discount</i>	Rate of discount for one period
<i>value1,...</i>	Values for money paid out (as for a payment) are negative numbers; values for money you receive (as for income) are positive numbers

The function includes in calculations arguments that are numbers, empty cells, logical values, or text representations of numbers; the function ignores arguments that are error values or text that cannot be translated into numbers. If an argument is an array or reference, only numbers in that array or reference are counted. Empty cells, logical values, text, or error values in the array or reference are ignored. This function can have up to 255 arguments.

Remarks

The payments and income must be equally spaced in time and occur at the end of each period. The function uses the order of the values to interpret the order of cash flows. Be sure to enter your payment and income values in the correct sequence.

The investment begins one period before the date of the value1 cash flow and ends with the last cash flow in the list. The calculation is based on future cash flows. If your first cash flow occurs at the beginning of the first period, the first value must be added to the result, not included in the value arguments.

This function is similar to the **PV** function (present value). Use **PV** to work with cash flows that begin at the beginning or the end of the period; this function allows cash flows only at the end of the period. Unlike the variable cash flow values of this function, **PV** cash flows must be constant throughout the investment.

This is also related to the **IRR** function (internal rate of return). **IRR** is equivalent to this function when the rate argument for net present value equals zero: $NPV(IRR(...),...) = 0$.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`NPV(0.065,D12:D19)`

`NPV(R1C1,R12C4:R19C4)`

`NPV(6.5%, -10000, 3000, 3400, 7700)` gives the result \$2,055.38

Version Available

This function is available in product version 1.0 or later.

See Also

IRR | PV | Financial Functions

NUMBERVALUE

This function converts text to a number, in a locale-independent way.

Syntax

NUMBERVALUE(*text*, *decseparator*, *groupseparator*)

Arguments

This function has these arguments:

Argument	Description
<i>text</i>	Text to convert to a number
<i>decseparator</i>	[Optional] Character used to separate the integer and fractional part of the result
<i>groupseparator</i>	[Optional] Character used to separate groupings of numbers

Remarks

If the separator arguments are not specified, separators from the current locale are used. The result is 0 if an empty string is specified as the *text* argument. The #VALUE! error value is returned if the group separator occurs after the decimal separator in the *text* argument.

Data Types

Accepts string data. Returns numeric data.

Examples

NUMBERVALUE("2.4%") gives the result 0.024

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

OCT2BIN

This function converts an octal number to a binary number.

Syntax

OCT2BIN(*number*,*places*)

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>number</i>	Octal numeric value to convert, must be 10 characters or less, and must be between 7777777000 and 777
<i>places</i>	[Optional] Number of characters to return; if not an integer, the number is truncated

Remarks

An error value is returned if the *number* is invalid or if *places* is non-numeric or negative. If *places* is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Data Types

Accepts numeric data. Returns numeric data.

Examples

OCT2BIN(77770000)

Version Available

This function is available in product version 2.0 or later.

See Also

OCT2DEC | OCT2HEX | HEX2BIN | DEC2BIN | Engineering Functions

OCT2DEC

This function converts an octal number to a decimal number.

Syntax

OCT2DEC(*number*)

Arguments

Specify the octal number to convert. The number should not contain more than 10 octal characters. An error value is returned if the number is invalid.

Data Types

Accepts numeric data. Returns numeric data.

Examples

OCT2DEC(7777)

Version Available

This function is available in product version 2.0 or later.

See Also

OCT2BIN | OCT2HEX | HEX2DEC | DEC2OCT | Engineering Functions

OCT2HEX

This function converts an octal number to a hexadecimal number.

Syntax

OCT2HEX(*number*,*places*)

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Octal numeric value to convert, must be 10 characters or less
<i>places</i>	[Optional] Number of characters to return; if not an integer, the number is truncated

Remarks

An error value is returned if the *number* is invalid or if *places* is non-numeric or negative. If *places* is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Data Types

Accepts numeric data. Returns numeric data.

Examples

OCT2HEX(7777)

Version Available

This function is available in product version 2.0 or later.

See Also

OCT2BIN | OCT2DEC | HEX2OCT | DEC2OCT | Engineering Functions

ODD

This function rounds the specified value up to the nearest odd integer.

Syntax

`ODD(value)`

Arguments

The argument can be any numeric value.

Remarks

Regardless of the sign of the number specified by the argument, the number is rounded away from zero.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ODD(A3)`

`ODD(R1C2)`

`ODD(4)` gives the result 5

`ODD(-2.5)` gives the result -3

Version Available

This function is available in product version 1.0 or later.

See Also

CEILING | EVEN | FLOOR | ISODD | Math and Trigonometry Functions

ODDFPRICE

This function calculates the price per \$100 face value of a security with an odd first period.

Syntax

ODDFPRICE(*settle,maturity,issue,first,rate,yield,redeem,freq,basis*)

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>issue</i>	Issue date for the security
<i>first</i>	First coupon date
<i>rate</i>	Annual interest rate
<i>yield</i>	Annual yield for the security
<i>redeem</i>	Redemption value per \$100 face value for the security
<i>freq</i>	Frequency of payment, number of payments per year
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns an error when *settle*, *maturity*, *issue*, or *first* is invalid. *Settle*, *maturity*, *issue*, *first*, and *basis* are truncated to integers. If *rate* or *yield* is less than 0, a #NUM! error is returned. If *basis* is less than 0 or greater than 4, a #NUM! error is returned. *Maturity* should be greater than *first* which should be greater than *settle* which should be greater than *issue*. Otherwise a #NUM! error is returned.

Data Types

Accepts numeric data or dates. Returns numeric data.

Examples

ODDFPRICE (A1, A2, A3, A4, A5, A6, A7, A8, A9)

Version Available

This function is available in product version 2.0 or later.

See Also

ODDLPRICE | PRICE | ODDFYIELD | ODDLYIELD | Financial Functions

ODDFYIELD

This function calculates the yield of a security with an odd first period.

Syntax

ODDFYIELD(*settle,maturity,issue,first,rate,price,redem,freq,basis*)

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>issue</i>	Issue date for the security
<i>first</i>	First coupon date
<i>rate</i>	Interest rate of the security
<i>price</i>	Price of the security
<i>redem</i>	Redemption value per \$100 face value for the security
<i>freq</i>	Frequency of payment, number of payments per year
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when *settle*, *maturity*, *issue*, or *first* is invalid. *Settle*, *maturity*, *issue*, *first*, and *basis* are truncated to integers. If *rate* is less than 0 or *yield* is less than or equal to 0, a #NUM! error is returned. If *basis* is less than 0 or greater than 4, a #NUM! error is returned. *Maturity* should be greater than *first* which should be greater than *settle* which should be greater than *issue*. Otherwise a #NUM! error is returned.

Data Types

Accepts numeric data. Returns numeric data.

Examples

ODDFYIELD (B1, B2, B3, B4, B5, B6, B7, B8, B9)

Version Available

This function is available in product version 2.0 or later.

See Also

PRICE | **ODDLYIELD** | **ODDFPRICE** | **ODDLPRICE** | **Financial Functions**

ODDLPRICE

This function calculates the price per \$100 face value of a security with an odd last coupon period.

Syntax

ODDLPRICE(*settle,maturity,last,rate,yield,redeem,freq,basis*)

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>last</i>	Last coupon date
<i>rate</i>	Annual interest rate
<i>yield</i>	Annual yield for the security
<i>redeem</i>	Redemption value per \$100 face value for the security
<i>freq</i>	Frequency of payment, number of payments per year
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when *settle*, *maturity*, or *last* is invalid. *Settle*, *maturity*, *issue*, *last*, and *basis* are truncated to integers. If *rate* is less than 0 or *yield* is less than 0, a #NUM! error is returned. If *basis* is less than 0 or greater than 4, a #NUM! error is returned. *Maturity* should be greater than *settle* which should be greater than *last*. Otherwise a #NUM! error is returned.

Data Types

Accepts numeric data and dates. Returns numeric data.

Examples

ODDLPRICE (C1, C2, A3, C4, C5, C6, C7, C8)

Version Available

This function is available in product version 2.0 or later.

See Also

PRICE | **ODDFPRICE** | **ODDFYIELD** | **ODDLYIELD** | **Financial Functions**

ODDLYIELD

This function calculates the yield of a security with an odd last period.

Syntax

ODDLYIELD(*settle,maturity,last,rate,price,redeem,freq,basis*)

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>last</i>	Last coupon date
<i>rate</i>	Annual interest rate
<i>price</i>	Price of the security
<i>redeem</i>	Redemption value per \$100 face value for the security
<i>freq</i>	Frequency of payment, number of payments per year
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when *settle*, *maturity*, or *last* is invalid. *Settle*, *maturity*, *last*, and *basis* are truncated to integers. If *rate* is less than 0 or *price* is less than or equal to 0, a #NUM! error is returned. If *basis* is less than 0 or greater than 4, a #NUM! error is returned. *Maturity* should be greater than *settle* which should be greater than *last*. Otherwise a #NUM! error is returned.

Data Types

Accepts numeric data or dates. Returns numeric data.

Examples

ODDLYIELD(G1,G2,G3,G4,G5,G6,G7,G8)

Version Available

This function is available in product version 2.0 or later.

See Also

PRICE | **ODDFPRICE** | **ODDFYIELD** | **ODDLPRICE** | **Financial Functions**

OFFSET

This function returns a reference to a range. The range is a specified number of rows and columns from a cell or range of cells. The function returns a single cell or a range of cells.

Syntax

`OFFSET(reference,rows,cols,height,width)`

Remarks

This function has these arguments:

Argument	Description
<i>reference</i>	The location from which to base the offset
<i>rows</i>	Number of rows to which the upper left cell refers
<i>cols</i>	Number of columns to which the upper left cell refers
<i>height</i>	[Optional] Number of returned rows; if omitted, same as <i>reference</i>
<i>width</i>	[Optional] Number of returned columns; if omitted, same as <i>reference</i>

The *cols* can be positive (right of the reference) or negative (left). If height or width is omitted, it is the same as the reference.

Remarks

This is a volatile function.

Data Types

Accepts a cell range for reference. Accepts numbers for rows, cols, height, and width. Returns a cell range.

Examples

`OFFSET(D3,2,3,1,1)`

`OFFSET(D3:E5,2,3,1,1)`

Version Available

This function is available in product version 2.5 or later.

See Also

HLOOKUP | LOOKUP | Lookup Functions

OR

This function calculates logical OR. It returns TRUE if any of its arguments are true; otherwise, returns FALSE if all arguments are false.

Syntax

`OR(bool1, bool2, ...)`

`OR(array)`

`OR(array1, array2, ...)`

`OR(expression)`

`OR(expression1, expression2, ...)`

Arguments

Provide numeric (1 or 0) or logical values (TRUE or FALSE) for up to 255 arguments. You can also specify a single array instead of listing the values separately, or up to 255 arrays. Similarly, you can specify an expression or up to 255 expressions.

Data Types

Accepts logical data (Boolean values of TRUE or FALSE) or numerical values (0 or 1). Returns logical data (Boolean values of TRUE or FALSE).

Examples

`OR(B3, B6, B9)`

`OR(R1C2, R1C3, R1C4, R1C5)`

`OR(D2:D12)`

`OR(R12C1:R12C9)`

`OR(TRUE, FALSE, FALSE)` gives the result TRUE

`OR(TRUE())` gives the result TRUE

`OR(FALSE(), FALSE())` gives the result FALSE

`OR(1+1=1, 2+2=5)` gives the result FALSE

`OR(5+3=8, 5+4=12)` gives the result TRUE

Version Available

This function is available in product version 1.0 or later.

See Also

AND | NOT | Logical Functions

PDURATION

This function returns the number of periods required by an investment to reach specified value.

Syntax

PDURATION(*rate,pv,fv*)

Arguments

This function has the following arguments:

Argument	Description
<i>rate</i>	Value of interest rate per period
<i>pv</i>	Present value of the investment
<i>fv</i>	Desired future value of the investment

All the arguments of this function has positive values.

Remarks

All the passed arguments should be positive values.

Data Types

Accepts numeric data. Returns numeric data.

Examples

PDURATION(0.025/12,1000,1200) gives the result 87.6

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

PEARSON

This function returns the Pearson product moment correlation coefficient, a dimensionless index between -1.0 to 1.0 inclusive indicative of the linear relationship of two data sets.

Syntax

`PEARSON(array_ind,array_dep)`

Arguments

This function has these arguments:

Argument	Description
<i>array_ind</i>	Array of independent values (x's)
<i>array_dep</i>	Array of dependent values (y's)

The arrays must be the same size.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`PEARSON(B4:G7,B8:G11)`

`PEARSON(R4C2:R7C7,R8C2:R11C7)`

`PEARSON({2,8,4,16,10,12},{8,2,15,14,18,11})` gives the result 0.262017

Version Available

This function is available in product version 1.0 or later.

See Also

RSQ | STEYX | Statistical Functions

PERCENTILE

This function returns the *n*th percentile of values in a range.

Syntax

PERCENTILE(*array*,*n*)

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Array of values representing the data
<i>n</i>	Value representing the percentile value between 0 and 1

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

PERCENTILE (A1:A12,0.95)

PERCENTILE (R1C1:R1C45,0.866)

PERCENTILE ({5,15,25,50,65},0.45) gives the result 23

Version Available

This function is available in product version 1.0 or later.

See Also

PERCENTRANK | QUARTILE | Statistical Functions

PERCENTILE.EXC

Summary

This function returns the *k*th percentile of values in a range where *k* is between 0..1, exclusive.

Syntax

PERCENTILE.EXC(*array*,*k*)

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Array of values representing the data
<i>k</i>	Value representing the percentile value between 0 and 1

Remarks

This function returns the #NUM! error value if the array is empty. If *k* is nonnumeric, #VALUE! is returned. If *k* = 0 or 1, #NUM! is returned. The function interpolates to determine the value at the *k*th percentile if *k* is not a multiple of 1/(n-1). The function interpolates when the value for the specified percentile is between two values in the array. If the function cannot interpolate for the percentile, #NUM! is returned.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

PERCENTILE.EXC(A1:A12,0.95)

PERCENTILE.EXC(R1C1:R1C45,0.866)

PERCENTILE.EXC({5,15,25,50,65},0.45) gives the result 22

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

PERCENTILE.INC

Summary

This function returns the *k*th percentile of values in a range where *k* is between 0..1, inclusive.

Syntax

PERCENTILE.INC(*array*,*k*)

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Array of values representing the data
<i>k</i>	Value representing the percentile value between 0 and 1

Remarks

This function returns the #NUM! error value if the array is empty. If *k* is nonnumeric, #VALUE! is returned. If *k* < 0 or > 1, #NUM! is returned. The function interpolates to determine the value at the *k*th percentile if *k* is not a multiple of 1/(*n*-1).

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

PERCENTILE.INC(A1:A12,0.95)

PERCENTILE.INC(R1C1:R1C45,0.866)

PERCENTILE.INC({5,15,25,50,65},0.45) gives the result 23

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

PERCENTRANK

This function returns the rank of a value in a data set as a percentage of the data set.

Syntax

`PERCENTRANK(array,n,sigdig)`

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Array of data with numeric values that defines the relative ranking
<i>n</i>	Value for which you want to find the rank in percentage
<i>sigdig</i>	[Optional] Number of significant digits for the ranked percentage value; if omitted, the calculation used three significant digits; if not an integer, number is truncated

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PERCENTRANK(A1:A12,0.95)`

`PERCENTRANK(R1C1:R1C45,0.866)`

`PERCENTRANK(A1:A17,23,3)`

`PERCENTRANK(R1C1:R43:C1,255.4,2)`

`PERCENTRANK({10,12,13,14,14,14.5,16,17.5,17.75,20,22},18,4)` gives the result 0.8111

`PERCENTRANK({10,12,13,14,14,14.5,16,17.5,17.75,20,22},18,1)` gives the result 0.8

Version Available

This function is available in product version 1.0 or later.

See Also

PERCENTILE | Statistical Functions

PERCENTRANK.EXC

This function returns the rank of a value in a data set as a percentage (0..1, exclusive) of the data set.

Syntax

PERCENTRANK.EXC(*array*,*n*,*sigdig*)

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Array of data with numeric values that defines the relative ranking
<i>n</i>	Value for which you want to find the rank in percentage
<i>sigdig</i>	[Optional] Number of significant digits for the ranked percentage value; if omitted, the calculation used three significant digits

Remarks

If *array* is empty, this function returns the #NUM! error value. If *sigdig* < 1, this function returns the #NUM! error value.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

PERCENTRANK.EXC(A1:A12,0.95)

PERCENTRANK.EXC(R1C1:R1C45,0.866)

PERCENTRANK.EXC(A1:A17,23,3)

PERCENTRANK.EXC(R1C1:R43:C1,255.4,2)

PERCENTRANK.EXC({10,12,13,14,14,14.5,16,17.5,17.75,20,22},18,1) gives the result 0.8

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

PERCENTRANK

PERCENTRANK.INC

This function returns the rank of a value in a data set as a percentage of the data set.

Syntax

PERCENTRANK.INC(*array,n,sigdig*)

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>array</i>	Array of data with numeric values that defines the relative ranking
<i>n</i>	Value for which you want to find the rank in percentage
<i>sigdig</i>	[Optional] Number of significant digits for the ranked percentage value; if omitted, the calculation uses three significant digits; if not an integer, number is truncated

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

PERCENTRANK.INC(A1:A12,0.95)

PERCENTRANK.INC(R1C1:R1C45,0.866)

PERCENTRANK.INC(A1:A17,23,3)

PERCENTRANK.INC(R1C1:R43:C1,255.4,2)

PERCENTRANK.INC({10,12,13,14,14,14.5,16,17.5,17.75,20,22},18,4) gives the result 0.8111

PERCENTRANK.INC({10,12,13,14,14,14.5,16,17.5,17.75,20,22},18,1) gives the result 0.8

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

PERCENTRANK

PERMUT

This function returns the number of possible permutations for a specified number of items.

Syntax

PERMUT(*k*,*n*)

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>k</i>	Number of items; must be greater than 0; if not an integer, the number is truncated
<i>n</i>	Number of items in each possible permutation; must be positive or 0; if not an integer, the number is truncated

Remarks

A permutation is any set or subset of items where internal order is significant. Contrast with combinations (the **COMBIN** function).

The equation for this function is:

$$PERMUT(k, n) = P_{k,n} = \frac{n!}{(n-k)!}$$

where *k* and *n* are defined in the arguments.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

PERMUT(B3, 5)

PERMUT(C4, B2)

PERMUT(R1C2, 2)

PERMUT(8,2) gives the result 56

PERMUT(100,3) gives the result 970200

Version Available

This function is available in product version 1.0 or later.

See Also

COMBIN | **Math and Trigonometry Functions**

PERMUTATIONA

This function calculates the number of permutations for the specified number of items (along with repetitions) that can be selected from the total items.

Syntax

PERMUTATIONA(*k*,*n*)

Arguments

This function has the following arguments:

Argument	Description
----------	-------------

<i>k</i>	Refers to the total number of items. This value must be greater than 0. If the specified value is not an integer, the number is truncated.
----------	--

<i>n</i>	Refers to the number of items in each possible permutation. This value must be positive or 0. If the specified value is not an integer, the number is truncated.
----------	--

Remarks

If the values in the arguments passed are invalid, this functions returns an error.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

PERMUTATIONA(4,6) gives the result 4096.

PERMUTATIONA(2,5) gives the result 32.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

PHI

This function calculates the value of the density function for the specified standard normal distribution.

Syntax

$\text{PHI}(x)$

Arguments

For the argument, you can specify any real number that you want to calculate in order to determine the density of the standard normal distribution.

Remarks

If the values in the arguments passed are invalid (invalid numeric value or invalid data type), this functions returns an error.

Data Types

Accepts only numeric data. Returns numeric data.

Examples

$\text{PHI}(0.35)$ gives the result 0.3752

$\text{PHI}(0.4)$ gives the result 0.3682

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

PHONETIC

This function extracts phonetic (furigana) characters from the specified string.

Syntax

PHONETIC(*text*)

Arguments

For the argument, you can specify a string, or a cell reference that contains the furigana text.

Remarks

If a range of cells is passed to *text* argument, the furigana text of upper-left corner cell is returned.

Data Types

Accepts string data. Returns string data.

Examples

PHONETIC(pan) gives the result a.

Version Available

This function is available in product version 11.0 or later.

PI

This function returns PI as 3.1415926536.

Syntax

PI()

Arguments

This function does not accept arguments.

Data Types

Does not accept data. Returns numeric data.

Examples

```
PI( )
```

```
DEGREES(PI()) gives the result 180
```

Version Available

This function is available in product version 1.0 or later.

See Also

DEGREES | RADIANS | Math and Trigonometry Functions

PMT

This function returns the payment amount for a loan given the present value, specified interest rate, and number of terms.

Syntax

`PMT(rate,nper,pval,fval,type)`

Arguments

This function has these arguments:

Argument	Description
<i>rate</i>	Value of interest rate per period
<i>nper</i>	Total number of payment periods
<i>pval</i>	Present value, worth now
<i>fval</i>	[Optional] Future value, cash value after the last payment; if omitted, the calculation uses zero
<i>type</i>	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

Remarks

Be sure that the interest rate and the number of payment periods correspond to the same units. If payment periods are monthly, then the interest rate should be calculated per month. If the interest rate is 6 percent annually, you can use 6% or (6/100) or 0.06 for the rate argument if the payment period is a year, but for monthly pay periods, divide the 6% by 12. The payment returned includes principal and interest but, no taxes, reserve payments, or fees.

The result is represented by a negative number because it is money paid out by you.

See the **PV** function for the equation for calculating financial values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PMT(B1,C4,C5,C6,1)`

`PMT(R1C2,8,16,4)`

`PMT(6%/12, 15, 5000)` gives the result `-$346.82`

`PMT(0.005, 15, 5000, 0, 1)` gives the result `-$345.10`

Version Available

This function is available in product version 1.0 or later.

See Also

IPMT | PPMT | PV | Financial Functions

POISSON

This function returns the Poisson distribution.

Syntax

`POISSON(nevents,mean,cumulative)`

Remarks

This function has these arguments:

Argument Description

<i>nevents</i>	Number of events Provide an integer, or the value is truncated. The number must be greater than zero.
<i>mean</i>	Expected numeric value The number must be greater than zero.
<i>cumulative</i>	Set to TRUE to return the cumulative Poisson probability that the number of random events occurring is between zero and <i>nevents</i> inclusive. Set to FALSE to return the Poisson probability mass function that the number of events occurring is exactly <i>nevents</i> .

Remarks

The cumulative Poisson probability is calculated as follows:

$$POISSON(x, \mu, TRUE) = \sum_{j=0}^x \frac{e^{-\lambda} \lambda^j}{j!}$$

The Poisson probability mass function is calculated as follows:

$$POISSON(x, \mu, FALSE) = \frac{e^{-\lambda} \lambda^x}{x!}$$

where x is the number of events (*nevents* argument), mu is the mean (*mean* argument).

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`POISSON(A3,B4,TRUE)`

`POISSON(R1C2,3,FALSE)`

`POISSON(7,4,TRUE)` gives the result 0.948866384

`POISSON(7,4,FALSE)` gives the result 0.059540363

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | GAMMADIST | HYPGEOMDIST | Statistical Functions

POISSON.DIST

This function returns the Poisson distribution.

Syntax

POISSON.DIST(*nevents*,*mean*,*cumulative*)

Remarks

This function has these arguments:

Argument Description

<i>nevents</i>	Number of events; provide an integer, or the value is truncated; the number must be greater than zero
<i>mean</i>	Expected numeric value; the number must be greater than zero
<i>cumulative</i>	Set to TRUE to return the cumulative Poisson probability that the number of random events occurring is between zero and <i>nevents</i> inclusive. Set to FALSE to return the Poisson probability mass function that the number of events occurring is exactly <i>nevents</i> .

Remarks

The cumulative Poisson probability is calculated as follows:

$$POISSON(x, \mu, TRUE) = \sum_{j=0}^x \frac{e^{-\lambda} \lambda^j}{j!}$$

The Poisson probability mass function is calculated as follows:

$$POISSON(x, \mu, FALSE) = \frac{e^{-\lambda} \lambda^x}{x!}$$

where x is the number of events (*nevents* argument), mu is the mean (*mean* argument).

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

POISSON.DIST(A3,B4,TRUE)

POISSON.DIST(R1C2,3,FALSE)

POISSON.DIST(7,4,TRUE) gives the result 0.9488663842071525

POISSON.DIST(7,4,FALSE) gives the result 0.059540362609726345

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

POISSON

POWER

This function raises the specified number to the specified power.

Syntax

`POWER(number,power)`

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Number to raise to the power given in <i>power</i>
<i>power</i>	Power to which to raise the number given in <i>number</i>

Specify the number to raise using the first argument and specify the power to raise it to using the second argument.

Remarks

You can use the exponent operator (^) instead of this function to raise a number to a power; for example, 16^3 .

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`POWER(A3,B4)`

`POWER(R1C2,3)`

`POWER(16,3)` gives the result 4096

Version Available

This function is available in product version 1.0 or later.

See Also

EXP | SQRT | Math and Trigonometry Functions

PPMT

This function returns the amount of payment of principal for a loan given the present value, specified interest rate, and number of terms.

Syntax

`PPMT(rate,per,nper,pval,fval,type)`

Arguments

This function has these arguments:

Argument	Description
<i>rate</i>	Value of interest rate per period.
<i>per</i>	Number of the period for which to find the interest, between 1 and <i>nper</i>
<i>nper</i>	Total number of payment periods in an annuity.
<i>pval</i>	Present value, worth now
<i>fval</i>	[Optional] Future value, cash value after the last payment; if omitted, the calculation uses zero
<i>type</i>	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

Remarks

Be sure to express the interest rate as per annum. For example, if the interest rate is 8 percent, use 8 for the rate argument.

The result is represented by a negative number because it is money paid out by you.

See the **PV** function for the equation for calculating financial values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PPMT(B1,C4,C5,C6,C7,1)`

`PPMT(R1C2,R4C3,R6C3,R7C3,0)`

`PPMT(0.45, 22, 30, 6000, 7000)` gives the result `-$206.47`

Version Available

This function is available in product version 1.0 or later.

See Also

IPMT | **PMT** | **PV** | **Financial Functions**

PRICE

This function calculates the price per \$100 face value of a periodic interest security.

Syntax

PRICE(*settlement,maturity,rate,yield,redem,frequency,basis*)

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>rate</i>	Annual coupon rate
<i>yield</i>	Annual yield for the security
<i>redem</i>	Redemption value per \$100 face value for the security
<i>frequency</i>	Frequency of payment, number of payments per year; must be 1, 2, or 4
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when settle or maturity is invalid. A #NUM! error is returned if frequency is a number other than 1, 2, or 4. Settle, maturity, frequency, and basis are truncated to integers. If yield or rate is less than 0, a #NUM! error is returned. If redem is less than or equal to 0, a #NUM! error is returned. If basis is less than 0 or greater than 4, a #NUM! error is returned. If settle is greater than or equal to maturity, a #NUM! error is returned.

Data Types

Accepts numeric data and dates. Returns numeric data.

Examples

PRICE(A3,A4,A5,A6,A7,A8,A9)

Version Available

This function is available in product version 2.0 or later.

See Also

PRICEMAT | PRICEDISC | ODDFPRICE | ODDLPRICE | Financial Functions

PRICEDISC

This function returns the price per \$100 face value of a discounted security.

Syntax

`PRICEDISC(settle,mature,discount,redeem,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security.
<i>mature</i>	Maturity date for the security.
<i>discount</i>	Amount invested in the security.
<i>redeem</i>	Amount to be received at maturity.
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when settle or mature is invalid. Settle, mature, and basis are truncated to integers. If discount or redeem is less than or equal to 0, a #NUM! error is returned. If basis is less than 0 or greater than 4, a #NUM! error is returned. If settle is greater than or equal to mature, a #NUM! error is returned.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PRICEDISC(A1,A2,A5,A7,1)`

`PRICEDISC(R1C1,R2C1,R5C5,R5C7,2)`

`PRICEDISC("5/15/2004","9/1/2004",0.06,100,3)` gives the result 98.20822

Version Available

This function is available in product version 1.0 or later.

See Also

DISC | PRICEMAT | Financial Functions

PRICEMAT

This function returns the price at maturity per \$100 face value of a security that pays interest.

Syntax

`PRICEMAT(settle,mature,issue,rate,yield,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security
<i>mature</i>	Maturity date for the security
<i>issue</i>	Issue date for the security
<i>rate</i>	Interest rate for the security at the issue date
<i>yield</i>	Annual yield for the security
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when settle, mature, or issue is invalid. Settle, mature, issue, and basis are truncated to integers. If rate or yield is less than 0, a #NUM! error is returned. If basis is less than 0 or greater than 4, a #NUM! error is returned. If settle is greater than or equal to mature, a #NUM! error is returned.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PRICEMAT(A1,A2,A5,A7,A7,1)`

`PRICEMAT(R1C1,R2C1,R5C5,R5C7,R5C9,2)`

`PRICEMAT("5/15/2004","9/1/2004","5/15/2003",0.06,0.07,3)` gives the result 99.5842915904314

Version Available

This function is available in product version 1.0 or later.

See Also

DISC | PRICEDISC | Financial Functions

PROB

This function returns the probability that values in a range are between two limits.

Syntax

`PROB(array,probs,lower,upper)`

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Array of numeric values, which has corresponding probs
<i>probs</i>	Probabilities associated with the numeric values in array
<i>lower</i>	Lower limit on the numeric value for which you want a probability
<i>upper</i>	[Optional] Upper limit on the numeric value for which you want a probability; if omitted, returns the probability of result equal to lower limit

Remarks

If the *upper* argument is not provided, the function uses the value for the *lower* argument only, and returns the probability that the values are equal to the *lower* argument.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PROB({B1:B6},{E1:E6},10,100)`

`PROB({B2,B4,B5,B7},{0.4,0.25,0.1,.025},10,100)`

`PROB({R1C2:R6C2},{R1C5:R6C5},1,50)`

`PROB({0,1,2,3},{0.2,0.3,0.1,0.4},2)` gives the result 0.1

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | CRITBINOM | Statistical Functions

PRODUCT

This function multiplies all the arguments and returns the product.

Syntax

```
PRODUCT(value1,value2,...)
```

```
PRODUCT(array)
```

```
PRODUCT(array1,array2,...)
```

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

```
PRODUCT (B3, B7, 12)
```

```
PRODUCT (C4, B2, B4, C5)
```

```
PRODUCT (A1:A9)
```

```
PRODUCT (R1C2, 2, 10)
```

```
PRODUCT (A1:A8, B1:B8, C2:C18)
```

```
PRODUCT(1,2,3,5,7,11,13) gives the result 30030
```

Version Available

This function is available in product version 1.0 or later.

See Also

FACT | QUOTIENT | SUMPRODUCT | Statistical Functions

PROPER

This function capitalizes the first letter in each word of a text string.

Syntax

`PROPER(text)`

Arguments

The text argument can be a string, a formula that returns a string, or a reference to a cell containing a string.

Remarks

This function capitalizes letters that follow any character other than a letter, for example, a space. This function converts all other letters to lowercase letters.

Data Types

Accepts string data. Returns string data.

Examples

`PROPER(D2)`

`PROPER("INTRO to SPREAD")` gives the result Intro To Spread

`PROPER("Tom's one-time order")` gives the result Tom'S One-Time Order

Version Available

This function is available in product version 1.0 or later.

See Also

[CHAR](#) | [UPPER](#) | [Text Functions](#)

PV

This function returns the present value of an investment based on the interest rate, number and amount of periodic payments, and future value. The present value is the total amount that a series of future payments is worth now.

Syntax

`PV(rate,numper,paymt,fval,type)`

Arguments

This function has these arguments:

Argument	Description
<i>rate</i>	Interest rate expressed as percentage (per period)
<i>numper</i>	Total number of payment periods
<i>paymt</i>	Payment made each period; cannot change over the life of the annuity
<i>fval</i>	[Optional] Future value; if omitted, the calculation is based on the payments
<i>type</i>	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

For the arguments, money paid out (such as deposits in an investment) is represented by negative numbers; money you receive (such as dividend checks) is represented by positive numbers.

Remarks

Use consistent units for specifying the rate and number of periods arguments. If you make monthly payments on a five-year loan at 8 percent annual interest, use 0.08/12 for the rate argument and 5*12 for the number of periods argument. If you make annual payments on the same loan, use 0.08 for rate and 5 for number of periods.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PV(B1/12,N24,-75,0,1)`

`PV(R1C1/12,48,R1C2,0,0)`

`PV(0.005,60,-100,0,1)` gives the result \$5,198.42

Version Available

This function is available in product version 1.0 or later.

See Also

FV | **NPER** | **PMT** | **Financial Functions**

QUARTILE

This function returns which quartile (which quarter or 25 percent) of a data set a value is.

Syntax

`QUARTILE(array,quart)`

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Array or cell range of numeric values for which you want the quartile value
<i>quart</i>	Quartile value for the array (see the table below for returned values)

Remarks

A quarter is 25 percent. So the quartile number is an integer between 0 (the minimum value in the data set) and 4 (the maximum value in the data set) and determines the value to return as listed in the table below.

If the number is...	Then this function returns the...
0	Minimum value
1	First quartile (25th percentile)
2	Median value (50th percentile)
3	Third quartile (75th percentile)
4	Maximum value

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`QUARTILE(A1:A17,2)`

`QUARTILE(R1C1:R17C1,3)`

`QUARTILE({11,21,42,27,18,29,32,52},1)` gives the result 20.25

Version Available

This function is available in product version 1.0 or later.

See Also

PERCENTILE | PERCENTRANK | Statistical Functions

QUARTILE.EXC

Summary

This function returns the quartile (which quarter or 25 percent) of a data set based on percentile values from 0..1, exclusive.

Syntax

QUARTILE.EXC(*array*,*quart*)

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Array or cell range of numeric values for which you want the quartile value
<i>quart</i>	Quartile value for the array (see the table below for returned values)

Remarks

A quarter is 25 percent. So the quartile number is an integer between 0 (the minimum value in the data set) and 4 (the maximum value in the data set) and determines the value to return as listed in the table below.

If the number is...	Then this function returns the...
0	Minimum value
1	First quartile (25th percentile)
2	Median value (50th percentile)
3	Third quartile (75th percentile)
4	Maximum value

The function returns #NUM! if the array is empty. The function returns #NUM! if *quart* = 0 or 4.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

QUARTILE.EXC(A1:A17,2)

QUARTILE.EXC(R1C1:R17C1,3)

QUARTILE.EXC({11,21,42,27,18,29,32,52},1) gives the result 18.75

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

QUARTILE.INC

Summary

This function returns the quartile (which quarter or 25 percent) of a data set based on percentile values from 0..1, inclusive.

Syntax

QUARTILE.INC(*array*,*quart*)

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Array or cell range of numeric values for which you want the quartile value
<i>quart</i>	Quartile value for the array (see the table below for returned values)

Remarks

A quarter is 25 percent. So the quartile number is an integer between 0 (the minimum value in the data set) and 4 (the maximum value in the data set) and determines the value to return as listed in the table below.

If the number is...	Then this function returns the...
0	Minimum value
1	First quartile (25th percentile)
2	Median value (50th percentile)
3	Third quartile (75th percentile)
4	Maximum value

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

QUARTILE.INC(A1:A17,2)

QUARTILE.INC(R1C1:R17C1,3)

QUARTILE.INC({11,21,42,27,18,29,32,52},1) gives the result 20.25

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

QUOTIENT

This function returns the integer portion of a division. Use this to ignore the remainder of a division.

Syntax

`QUOTIENT(numerator,denominator)`

Arguments

This function has these arguments:

Argument	Description
<i>numerator</i>	Numerator or dividend
<i>denominator</i>	Denominator or divisor

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`QUOTIENT(B8,B10)`

`QUOTIENT(R8B2,R10B2)`

`QUOTIENT(14,4)` gives the result 3

Version Available

This function is available in product version 1.0 or later.

See Also

MOD | PRODUCT | Math and Trigonometry Functions

Functions R to S

Functions R to S

RADIANS	RAND	RANDARRAY	RANDBETWEEN
RANK	RANK.AVG	RANK.EQ	RATE
RECEIVED	REPLACE	REPLACEB	REPT
RIGHT	RIGHTB	ROMAN	ROUND
ROUNDDOWN	ROUNDUP	ROW	ROWS
RRI	RSQ	RTD	SEARCH
SEARCHB	SEC	SECH	SECOND
SERIESSUM	SEQUENCE	SHEET	SHEETS
SIGN	SIN	SINH	SINGLE
SKEW	SKEW.P	SLN	SLOPE
SMALL	SORT	SORTBY	SQRT
SQRTPI	STANDARDIZE	STDEV	STDEV.P
STDEV.S	STDEVA	STDEVP	STDEVPA
STEYX	SUBSTITUTE	SUBTOTAL	SUM
SUMIF	SUMIFS	SUMPRODUCT	SUMSQ
SUMX2MY2	SUMX2PY2	SUMXMY2	SWITCH
SYD			

RADIANS

This function converts the specified number from degrees to radians.

Syntax

`RADIANS(value)`

Arguments

This function takes any real number angle value as the argument.

Remarks

Converts angle in degrees to angle in radians.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`RADIANS(B3)`

`RADIANS(R1C2)`

`RADIANS(45)` gives the result 0.7853981634 (which is $\pi/4$)

Version Available

This function is available in product version 1.0 or later.

See Also

DEGREES | PI | Math and Trigonometry Functions

RAND

This function returns an evenly distributed random number between 0 and 1.

Syntax

RAND()

Arguments

This function does not accept arguments.

Remarks

This function returns a new random number.

To generate a random real number between x and y , with y greater than x , use the following expression:

$\text{RAND()}*(y-x)+x$

To generate a random integer between x and y , with y greater than x , use the following expression:

$\text{INT}((y-x+1)*\text{RAND()}+x)$

This is a volatile function with version 2.5 or later. For more information, refer to **Volatile Functions**.

Data Types

Does not accept data. Returns numeric data.

Examples

`RAND()`

`RAND()*100`

`INT(RAND()*100)`

Version Available

This function is available in product version 1.0 or later. This function is a volatile function in version 2.5 or later.

See Also

RANDBETWEEN | **INT** | **Math and Trigonometry Functions**

RANDARRAY

This function returns an array of random numeric values. Users can specify the number of rows and columns, minimum and maximum values and indicate whether to return integers or decimal values.

Syntax

RANDARRAY([rows],[columns],[min],[max],[integer])

Arguments

RANDARRAY function has the following arguments:

Argument	Description
rows	Specifies the number of rows of random numbers to generate (if nothing is specified, the default value 1 is used).
columns	Specifies the number of columns of random numbers to generate (if nothing is specified, the default value 1 is used).
min	Specifies the minimum of values to generate (if nothing is specified, the default value 0 is used).
max	Specifies the maximum of values to generate (if nothing is specified, the default value 1 is used).
integer	Returns an integer only if the specified value is Boolean TRUE. If this value is FALSE, this function returns a decimal value (if nothing is specified, the default value FALSE is used).

Data Types

Accepts the number of rows and columns. Returns an array of random numbers.

 **Note :** RANDARRAY is a volatile function. This means that each time the worksheet calculates, the RANDARRAY function recalculates the new values.

Remarks

By default, if users don't specify the row or column argument, the RANDARRAY function will return a single value between 0 and 1. Also, if minimum and maximum arguments are not specified, then the RANDARRAY function returns a random set of values between 0 and 1. The minimum number argument should always contain a value which is less than the maximum number, else this function will return a #VALUE! error. Further, if the whole-number argument is empty, then this function will choose FALSE as the default value and the decimal value will be returned.

Examples

For instance - The cell A8 in the following image contains the formula "=RANDARRAY(5,3)" and returns a random set of values between 0 and 1.

7	RANDARRAY(5,3) Function				
8	0.301296713654076	0.968978906944218	0.702538246962628		
9	0.167834041222712	0.528294371528925	0.620410664378921		
10	0.884531817060312	0.516241429963227	0.622292071832181		
11	0.65774892066046	0.208537678811478	0.751867999807693		
12	0.387976196850592	0.0638986509434175	0.534441710355955		
13					

RANDARRAY(5,3,1,100) returns a series of random decimal values between 1 and 100.

RANDARRAY(5,3,1,100,TRUE) returns a series of random whole numbers between 1 and 100.

Version Available

This function is available in Spread for Windows Forms 12.1 or later.

RANDBETWEEN

This function returns a random number between the numbers you specify.

Syntax

`RANDBETWEEN(lower,upper)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>lower</i>	Lower number of two numbers between which a random number is chosen; this number must be less than <i>upper</i>
--------------	---

<i>upper</i>	Upper number of two numbers between which a random number is chosen
--------------	---

Remarks

This function returns a new random number every time the sheet is calculated.

This functions returns an integer value. The first argument must be less than the second argument.

This is a volatile function with version 2.5 or later. For more information, refer to **Volatile Functions**.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

```
RANDBETWEEN (A1, B2)
```

```
RANDBETWEEN (10, 20)
```

```
RANDBETWEEN (10, 40) *100
```

```
INT (RANDBETWEEN (1, 10) *100)
```

Version Available

This function is available in product version 1.0 or later. This function is a volatile function in version 2.5 or later.

See Also

RAND | **Math and Trigonometry Functions**

RANK

This function returns the rank of a number in a set of numbers. If you were to sort the set, the rank of the number would be its position in the list.

Syntax

`RANK(number,array,order)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>number</i>	Number whose rank you want to return
---------------	--------------------------------------

<i>array</i>	Reference to the set of numbers
--------------	---------------------------------

<i>order</i>	[Optional] How the number is ranked, either in descending order (0 or omitted) or ascending order (non-zero value)
--------------	--

Remarks

This function gives duplicate numbers the same rank. The presence of duplicate numbers affects the ranks of subsequent numbers. For example, in a list of integers, if the number 12 appears twice and has a rank of 4, then 13 would have a rank of 6 (no number would have a rank of 5).

Data Types

Accepts numeric data for the *number* argument, a reference for the *array* argument, and numeric data for the *order* argument. Returns numeric data.

Examples

`RANK(B3,B1:B8,1)`

`RANK(R3C2,R1C2:R8C2,1)`

`RANK(16,{2,4,8,16,32},1)` gives the result 4

Version Available

This function is available in product version 1.0 or later.

See Also

MEDIAN | **MODE** | **Statistical Functions**

RANK.AVG

Summary

This function returns the rank of a number in a set of numbers.

Syntax

`RANK.AVG(number,array,order)`

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>number</i>	Number whose rank you want to return
---------------	--------------------------------------

<i>array</i>	Reference to the set of numbers
--------------	---------------------------------

<i>order</i>	[Optional] How the number is ranked, either in descending order (0 or omitted) or ascending order (non-zero value)
--------------	--

Remarks

The size of the returned number is relative to other values in the list. The average rank is returned if more than one value has the same rank.

Data Types

Accepts numeric data for the *number* argument, a reference for the *array* argument, and numeric data for the *order* argument. Returns numeric data.

Examples

`RANK.AVG(B3,B1:B8,1)`

`RANK.AVG(R3C2,R1C2:R8C2,1)`

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

RANK.EQ

Summary

This function returns the rank of a number in a set of numbers.

Syntax

RANK.EQ(*number,array,order*)

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>number</i>	Number whose rank you want to return
---------------	--------------------------------------

<i>array</i>	Reference to the set of numbers
--------------	---------------------------------

<i>order</i>	[Optional] How the number is ranked, either in descending order (0 or omitted) or ascending order (non-zero value)
--------------	--

Remarks

The size of the returned number is relative to other values in the list. The top rank of that set of values is returned if more than one value has the same rank. Duplicate numbers are given the same rank. Duplicate numbers affect the ranks of subsequent numbers. For example, if the number of 11 is duplicated with a rank of 6 in a list of ascending numbers, the number 12 would have a rank of 8.

Data Types

Accepts numeric data for the *number* argument, a reference for the *array* argument, and numeric data for the *order* argument. Returns numeric data.

Examples

RANK.EQ(B3,B1:B8,1)

RANK.EQ(R3C2,R1C2:R8C2,1)

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

RATE

This function returns the interest rate per period of an annuity.

Syntax

`RATE(nper,pmt,pval,fval,type,guess)`

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>nper</i>	Total number of payment periods in an annuity
<i>pmt</i>	Value representing the payment made each period
<i>pval</i>	Present value, worth now
<i>fval</i>	Future value, cash value after the last payment
<i>type</i>	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)
<i>guess</i>	Guess for what the rate will be (optional)

Remarks

Guess is assumed to be 10% if omitted.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`RATE(A1, B2, C3, C4, 1)`

`RATE(360, -600, 80000)` gives the result 0.686%

Version Available

This function is available in product version 1.0 or later.

See Also

IPMT | PMT | PPMT | Financial Functions

RECEIVED

This function returns the amount received at maturity for a fully invested security.

Syntax

`RECEIVED(settle,mature,invest,discount,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security
<i>mature</i>	Maturity date for the security
<i>invest</i>	Amount invested in the security
<i>discount</i>	Discount rate for the security
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when settle or mature is invalid. Settle, mature, and basis are truncated to integers. If invest or discount is less than or equal to 0, a #NUM! error is returned. If basis is less than 0 or greater than 4, a #NUM! error is returned. If settle is greater than or equal to mature, a #NUM! error is returned.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`RECEIVED(A1,B2,C3,C4,1)`

`RECEIVED("3/01/2004","6/01/2004",600000,0.03,2)` gives \$604,635.50

Version Available

This function is available in product version 1.0 or later.

See Also

INTRATE | **Financial Functions**

REPLACE

This function replaces part of a text string with a different text string.

Syntax

`REPLACE(old_text,start_char,num_chars,new_text)`

Arguments

This function has these arguments:

Argument	Description
<i>old_text</i>	Original text in which you want to replace characters
<i>start_char</i>	Starting position in the original text to begin the replacement
<i>num_chars</i>	Number of characters in the original text that you want to replace with characters from the new text; if not an integer, the number is truncated
<i>new_text</i>	New text that replaces characters in the original text

Remarks

Use this function to replace a specified number of characters in a specified location with other characters. Use the **SUBSTITUTE** function to replace specific text with other text.

Data Types

Accepts string data for the *old_text* argument, numeric data for the *start_char* argument, numeric data for the *num_chars* argument, and string data for the *new_text* argument. Returns string data.

Examples

This example replaces three characters with one character, starting with the sixth character in the provided text:
`REPLACE("abcdefghijk", 6, 3, "%")` gives the result `abcde%ijk`

Version Available

This function is available in product version 1.0 or later.

See Also

FIND | **SUBSTITUTE** | **Text Functions**

REPLACEB

This function replaces part of a text string with a different text string (based on specified number of bytes).

Syntax

`REPLACEB(old_text,start_char,num_bytes,new_text)`

Arguments

This function has the following arguments:

Argument	Description
----------	-------------

<i>old_text</i>	Refers to the original text in which you want to replace characters.
<i>start_char</i>	Refers to the starting position in the original text to begin the replacement.
<i>num_bytes</i>	Refers to the number of bytes in the original text that you want to replace with characters from the new text. If this function is not an integer, the number is truncated.
<i>new_text</i>	Refers to the new text that replaces characters in the original text.

Remarks

The REPLACEB function counts 2 bytes per character, but this happens only when a DBCS language is set as the default language.

Data Types

Accepts string data for the *old_text* argument, numeric data for the *start_char* argument, numeric data for the *num_bytes* argument, and string data for the *new_text* argument. Returns string data.

Examples

`REPLACEB("lovely",1,3,"lo")` gives the result loely.

`REPLACEB("rosy",1,3,"!")` gives the result !y.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

REPT

This function repeats text a specified number of times.

Syntax

`REPT(text,number)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>text</i>	Text you want to repeat
-------------	-------------------------

<i>number</i>	Number of times you want to repeat the text; if not an integer, the number is truncated; if zero (0), returns empty (" ")
---------------	---

Remarks

The result of this function must be less than or equal to 255 characters.

Data Types

Accepts string data for the *text* argument and numeric data for the *number* argument. Returns string data.

Examples

`REPT(D9, 2)`

`REPT(R9C4, 2)`

`REPT("*4", 3)` gives the result `*4*4*4`

Version Available

This function is available in product version 1.0 or later.

See Also

[CONCATENATE](#) | [Text Functions](#)

RIGHT

This function returns the specified rightmost characters from a text value.

Syntax

`RIGHT(text,num_chars)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>text</i>	Text string from which you want to return characters
-------------	--

<i>num_chars</i>	[Optional] Number of characters to return; if omitted, calculation uses one (1); if not an integer, the number is truncated
------------------	---

The *text* argument can be a string, a formula that returns a string, or a reference to a cell containing a string.

The *num_chars* argument has these rules:

- The *num_chars* argument must be greater than or equal to zero.
- If the *num_chars* argument is greater than the length of text, this function returns all text.

Data Types

Accepts string data for the *text* argument and numeric data for the *num_chars* argument. Returns string data.

Examples

`RIGHT("Total Sales",5)` gives the result Sales

`RIGHT("Collie dog")` gives the result g

Version Available

This function is available in product version 1.0 or later.

See Also

LEFT | MID | Text Functions

RIGHTB

This function returns the specified rightmost characters from a text value on the basis of the number of bytes.

Syntax

`RIGHTB(mytext, num_bytes)`

Arguments

This function has the following arguments:

Argument	Description
<i>mytext</i>	Refers to the text string that contains the characters you want to extract.
<i>num_bytes</i>	[Optional] Refers to the number of bytes to be extracted. If this value is omitted, it uses one. If this value is not an integer, the number is truncated.

Remarks

The *mytext* argument can be a string, a formula that returns a string, or a reference to a cell containing a string.

The *num_bytes* argument works on the following rules:

- It must be greater than or equal to zero.
- If it is greater than the length of text, this function returns all the text.

Data Types

Accepts string data for the first argument and numeric data the second argument. Returns string data.

Examples

`RIGHTB("NOVARO")` gives the result O.

`RIGHTB("MOBILE")` gives the result E.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

ROMAN

This function converts an arabic numeral to a roman numeral text equivalent.

Syntax

ROMAN(*number*,*style*)

Arguments

This function has these arguments:

Argument	Description
<i>number</i>	Arabic number to convert
<i>style</i>	Type of roman numeral

Remarks

The style of roman numeral is set by the numeric value of the style argument:

Style value	Roman numeral style
<i>0 or omitted</i>	Classic
<i>1</i>	More concise
<i>2</i>	More concise
<i>3</i>	More concise
<i>4</i>	Simplified
TRUE	Classic
FALSE	Simplified

An error is returned if the *number* argument is negative.

Data Types

Accepts numeric data. Returns string data.

Examples

ROMAN(100,3)

Version Available

This function is available in product version 2.0 or later.

See Also

ABS | Math and Trigonometry Functions

ROUND

This function rounds the specified value to the nearest number, using the specified number of decimal places.

Syntax

`ROUND(value,places)`

Arguments

Use the *value* argument to specify the number to round. Use the *places* argument to specify the number of decimal places. The *places* argument has these rules:

- Set *places* to a value greater than zero to round to the specified number of decimal places.
- Set *places* to zero to round to the nearest whole number.
- Set *places* to a value less than zero to round the value left of the decimal to the nearest ten, hundred, etc.

Remarks

The result may be rounded up or rounded down.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`ROUND(A3,-2)`

`ROUND(C4,B2)`

`ROUND(R1C2,2)`

`ROUND(PI(),5)` gives the result 3.14159

`ROUND(29.2,-2)` gives the result 0 because 29.2 is closer to 0 than to 100.

`ROUND(-1.963,0)` gives the result -2

Version Available

This function is available in product version 1.0 or later.

See Also

ROUNDDOWN | ROUNDUP | CEILING | FLOOR | MROUND | Math and Trigonometry Functions

ROUNDDOWN

This function rounds the specified number down to the nearest number, using the specified number of decimal places.

Syntax

`ROUNDDOWN(value,places)`

Arguments

Use the *value* argument to specify the number to round. Use the *places* argument to specify the number of decimal places. The *places* argument has these rules:

- Set *places* to a value greater than zero to round to the specified number of decimal places.
- Set *places* to zero to round to the nearest whole number.
- Set *places* to a value less than zero to round the value left of the decimal to the nearest ten, hundred, etc.

Regardless of the sign of the number specified by the *value* argument, the number is rounded away from zero.

Remarks

The result is always rounded down.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`ROUNDDOWN(3.2,0)` gives the result 3

`ROUNDDOWN(D14,3)`

`ROUNDDOWN(R14C4,10)`

`ROUNDDOWN(3.14159,3)` gives the result 3.141

`ROUNDDOWN(-3.14159,1)` gives the result -3.1

`ROUNDDOWN(31415.92654,-2)` gives the result 31400

Version Available

This function is available in product version 1.0 or later.

See Also

ROUND | ROUNDDUP | CEILING | FLOOR | Math and Trigonometry Functions

ROUNDUP

This function rounds the specified number up to the nearest number, using the specified number of decimal places.

Syntax

`ROUNDUP(value,places)`

Arguments

Use the *value* argument to specify the number to round. Use the *places* argument to specify the number of decimal places. The *places* argument has these rules:

- Set *places* to a value greater than zero to round to the specified number of decimal places.
- Set *places* to zero to round to the nearest whole number.
- Set *places* to a value less than zero to round the value left of the decimal to the nearest ten, hundred, etc.

Remarks

Regardless of the sign of the number specified by the *value* argument, the number is rounded away from zero.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`ROUNDUP(A3,-2)`

`ROUNDUP(C4,B2)`

`ROUNDUP(R1C2, 2)`

`ROUNDUP(PI(),5)` gives the result 3.14160

`ROUNDUP(29.2,-2)` gives the result 100

`ROUNDUP(-1.963,0)` gives the result -2

Version Available

This function is available in product version 1.0 or later.

See Also

ROUND | ROUNDDOWN | CEILING | FLOOR | Math and Trigonometry Functions

ROW

This function returns the number of a row from a reference.

Syntax

ROW(reference)

Arguments

The argument is a cell or a single area.

Remarks

If the reference is omitted, the reference of the cell that the function is in is used.

Data Types

Accepts a cell or a single area. Returns numeric data.

Examples

`ROW(B2)` gives the result 2

`ROW(B1:B5)` gives the result 1

Version Available

This function is available in product version 3.0 or later.

See Also

COLUMNS | **INDEX** | **Lookup Functions**

ROWS

This function returns the number of rows in an array.

Syntax

`ROWS(array)`

Arguments

The argument is an array, an array formula, or a range of cells.

Data Types

Accepts array. Returns numeric data.

Examples

`ROWS(B2:B14)` gives the result 13

`ROWS(R2C6:R4C12)` gives the result 3

`ROWS(H2:H8)` gives the result 7

`ROWS(R[2]C[3]:R[8]C[3])` gives the result 7

`ROWS(R3C2:R17C2)` gives the result 15

Version Available

This function is available in product version 1.0 or later.

See Also

COLUMNS | **INDEX** | **Lookup Functions**

RRI

This function returns calculated interest rate for growth of investment.

Syntax

$RRI(nper,pv,fv)$

Arguments

This function has these arguments:

Argument	Description
<i>nper</i>	Number of periods for the investment
<i>pv</i>	Present value of the investment
<i>fv</i>	Future value of the investment

All the arguments of this function has positive values.

Remarks

If the passed arguments are invalid, this function returns an error value.

Data Types

Accepts numeric data. Returns numeric data.

Examples

$RRI(96,10000,11000)$ gives the result 0.0009933

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

RSQ

This function returns the square of the Pearson product moment correlation coefficient (R-squared) through data points in known y's and known x's.

Syntax

`RSQ(array_dep,array_ind)`

Arguments

This function has these arguments:

Argument	Description
<i>array_dep</i>	Array of dependent values (y's)
<i>array_ind</i>	Array of independent values (x's)

The arrays must be the same size.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`RSQ(B2:B14,H2:H14)`

`RSQ(R2C2:R14C2,R2C8:R14C8)`

`RSQ({2,4,6},{10,15,25})` gives the result 0.964286

Version Available

This function is available in product version 1.0 or later.

See Also

PEARSON | **Statistical Functions**

RTD

This function retrieves real-time data from a program that supports COM automation.

Syntax

RTD(*progID*, *server*, *topic1*, *topic2*, ...)

Arguments

This function has the following arguments:

Argument	Description
-----------------	--------------------

<i>progID</i>	Refers to the name of the ProgID of a registered COM automation add-in that has been installed on the local computer.
<i>server</i>	Refers to the name of the server where the add-in is. Leave the argument blank if this is run locally.
<i>topic1</i>	The <i>topic1</i> argument is required. Subsequent topics are optional. There can be 1 to 253 parameters that represent a unique piece of real-time data.

Remarks

The RTD COM automation add-in must be created and registered on a local computer.

Data Types

Accepts string data. Returns string data.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

SEARCH

This function finds one text string in another text string and returns the index of the starting position of the found text.

Syntax

SEARCH(string1,string2)

Arguments

The first argument is a string or cell reference of the text you are searching for and the second argument is a string, cell reference, or cell range of what you want to search.

Data Types

Accepts cell reference or string. Returns numeric data.

Examples

`SEARCH(A2,A4:A9)`

Version Available

This function is available in product version 5.0 or later.

See Also

FIND | **CONCATENATE** | **Text Functions**

SEARCHB

This function searches one text value within another and returns the text value's position in the text you searched.

Syntax

`SEARCHB(findtext,intext,start)`

Arguments

This function has the following arguments:

Argument	Description
----------	-------------

<i>findtext</i>	Refers to the text you are trying to find; if empty (" "), the function matches the first character in the search string (that is, the character numbered start or 1); cannot contain wildcard characters.
<i>intext</i>	Refers to the text through which you are searching.
<i>start</i>	[Optional] Refers to the number representing character at which to start the search. The first character of <i>intext</i> argument is 1. If this value is omitted, the calculation starts at 1. If this value is not an integer, the number is truncated.

Remarks

The SEARCHB function counts 2 bytes per character, but it happens only when a DBCS language is set as the default language.

This function performs a case-specific search (for example, to specify a capital letter and not lower case letters).

Data Types

Accepts string data for the *findtext* argument, string data for the *intext* argument, and numeric data for the *start* argument. Returns numeric data.

Examples

`SEARCHB("MOB","MOBILE")` gives the result 1.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

SEC

This function returns the secant of the specified angle.

Syntax

`SEC(angle)`

Arguments

This function can take any real number as an argument. The *angle* argument is the angle in radians for which you want the secant.

Remarks

If the *angle* is in degrees, multiply it by $\text{PI}/180$ to convert it to radians.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`SEC(60)` gives the result -1.049

`SEC(45)` gives the result 1.903

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

SECH

This function returns the hyperbolic secant of the specified angle.

Syntax

SECH(*angle*)

Arguments

This function can take any real number as an argument. The *angle* argument is the angle in radians for which you want the hyperbolic secant.

Remarks

The absolute value of *angle* must be less than 2^{27} .

Data Types

Accepts numeric data. Returns numeric data.

Examples

SECH(30) gives the result 1.87E-13

SECH(45) gives the result 5.73E-20

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

SECOND

This function returns the seconds (0 to 59) value for a specified time.

Syntax

`SECOND(time)`

Arguments

Specify the time argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), a DateTime object, as in `DATE(2003,7,4)`, or a TimeSpan object, as in `TIME(12,0,0)`. For more details on the date and time inputs, refer to the discussion in **Date and Time Functions**

Dates as numeric values are in the form x.y, where x is the "number of days since December 30, 1899" and y is the fraction of day. Numbers to the left represent the date. Times as numeric values are decimal fractions ranging from 0 to 0.99999999, representing the times from 0:00:00 (12:00:00 A.M.) to 23:59:59 (11:59:59 P.M.).

Remarks

The second is returned as an integer, ranging from 0 to 59

Data Types

Accepts numeric, string, DateTime object, or TimeSpan object data. Returns numeric data.

Examples

`SECOND(A2)`

`SECOND(R2C1)`

`SECOND(0.01)` gives the result 24

`SECOND(TIME(12,0,0))`

Version Available

This function is available in product version 1.0 or later.

See Also

HOURL | MINUTE | Date and Time Functions

SERIESSUM

This function returns the sum of a power series.

Syntax

`SERIESSUM(x,n,m,coeff)`

Arguments

This function has these arguments:

Argument	Description
<i>x</i>	Value to evaluate in the power series
<i>n</i>	Power to which to raise x
<i>m</i>	Step by which to increase n for each term in the series
<i>coeff</i>	Set of coefficients for the series (the values of a1, a2, ... ai)

Remarks

The power series formula is:

$$SERIESSUM(x, n, m, a) \approx a_1x^n + a_2x^{n+m} + a_3x^{n+2m} + \dots + a_ix^{n+(i-1)m}$$

where x, n, and m are the similarly named arguments and a is the *coeff* argument.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SERIESSUM(34, 3, 2, A1:A6)`

`SERIESSUM(12, 3, 1, B2:B24)`

Version Available

This function is available in product version 1.0 or later.

See Also

SUM | Math and Trigonometry Functions

SEQUENCE

SEQUENCE function returns a list of sequential numbers in an array (in ascending order), such as 1, 2, 3, 4 and so on. This function is used with hard-coded arguments in order to allow users to generate a specific sequence of values for the dynamic array formula.

Syntax

SEQUENCE(rows,[columns],[start],[step])

Arguments

SEQUENCE function has the following arguments:

Argument Description

<i>rows</i>	[required] Specifies the number of rows to generate in the sequence. It is mandatory for users to specify this argument. If this argument is not provided, the #CALC! error will be returned.
<i>columns</i>	[optional] Specifies the number of columns to generate in the sequence. If this argument is provided, the function returns an array with the specified number of columns. If nothing is specified, then the default value 1 is used.
<i>start</i>	[optional] Specifies the starting value. If this argument is provided, the function returns values starting with the specified value. In case nothing is specified by the user, then the default value 1 is used.
<i>step</i>	[optional] Specifies the increment value. If this argument is provided, the function returns values incremented with that specified value. In case nothing is specified by the user, then the default value 1 is used.

Data Types

Accepts the number of rows and columns. Returns a sequence of numbers.

Examples

For instance - The cell A2 in the following image contains the formula "=SEQUENCE(4,5)" and returns an array with values spilled to a cell range containing four rows and five columns displaying numbers in the sequence 1, 2, 3, 4 upto 20.

	A	B	C	D	E
1	SEQUENCE(4,5) Function				
2	1	2	3	4	5
3	6	7	8	9	10
4	11	12	13	14	15
5	16	17	18	19	20

Version Available

This function is available in Spread for Windows Forms 12.1 or later.

SHEET

This function returns the sheet number of the reference sheet.

Syntax

SHEET(*value*)

Arguments

For the argument, the name of a sheet or a reference for which you want the sheet number. This argument is optional.

Remarks

The #REF! error value is returned if the *value* argument is an invalid value. The #NA error value is returned if the *value* argument is an invalid sheet name.

Data Types

Accepts string data. Returns numeric data.

Examples

SHEET(Products) gives the result 1, where Products is the name of the first sheet(Sheet1) in the workbook.

SHEET("SheetFour") gives the result 4, where SheetFour is the name of the 4th sheet in the workbook.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

SHEETS

This function returns the number of sheets in a reference.

Syntax

`SHEETS(ref)`

Arguments

For the argument, the reference for which you want to know the number of sheets. This argument is optional.

Remarks

The #REF! error value is returned if the reference is not a valid value. The total number of sheets in the workbook is returned if no parameter is listed.

Data Types

Accepts string data. Returns numeric data.

Examples

`SHEETS(Products)` gives the result 2.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

SIGN

This function returns the sign of a number or expression.

Syntax

`SIGN(cellreference)`

`SIGN(value)`

`SIGN(expression)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Returns 1 if the number is positive, 0 if the number is 0, and -1 if the number is negative.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`SIGN(B3)`

`SIGN(R1C2)`

`SIGN(-5)` gives the result -1

`SIGN(15-8)` gives the result 1

Version Available

This function is available in product version 1.0 or later.

See Also

ABS | Math and Trigonometry Functions

SIN

This function returns the sine of the specified angle.

Syntax

`SIN(angle)`

Arguments

This function can take any real number as an argument. The *angle* argument is the angle in radians for which you want the sine.

Remarks

If the angle is in degrees, multiply it by $\text{PI}/180$ to convert it to radians.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`SIN(B4)`

`SIN(R1C2)`

`SIN(30*PI()/180)` gives the result 0.5

`SIN(RADIANS(45))`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOS | ASIN | COS | SINH | Math and Trigonometry Functions

SINH

This function returns the hyperbolic sine of the specified number.

Syntax

`SINH(value)`

Arguments

You can use any real number for the *value* argument.

Remarks

The equation for calculating the hyperbolic sine is:

$$\text{SINH}(z) = \frac{e^z - e^{-z}}{2}$$

where *z* is the *value* argument.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`SINH(B4)`

`SINH(R1C2)`

`SINH(1)` gives the result 1.1752011936

Version Available

This function is available in product version 1.0 or later.

See Also

[ACOSH](#) | [ASINH](#) | [SIN](#) | [COSH](#) | [Math and Trigonometry Functions](#)

SINGLE

This function returns a single value, a single cell range or an error using the intersection logic. There are two types of intersection logic - Implicit Intersection and Explicit Intersection.

The Implicit intersection logic selects a single value from an array of values while also ensuring that the formula returns only one value that the cell can hold. Implicit intersection can be used when users want to specify a range argument to a function that expects a single value and the formula is not an array formula (a formula entered using Ctrl+Shift+Enter). In this case, the value in the cell of the range which intersects the column or row of the formula cell is used for the function.

When dynamic arrays are enabled, then the "Implicit Intersection" is not supported, and users must use the SINGLE function (or the '@' operator) to specify the "Explicit Intersection" in order to return the single value. This is required because specifying the range argument will pass the range to the function and the results will be spilled as a dynamic array.

Syntax

SINGLE(*value*)

Arguments

value - [required] Specifies the value that you want to evaluate using implicit intersection.

Data Types

Accepts values in the form of a cell range. Returns a single value, a single cell range or an error.

Remarks

If the argument provided by the user contains a range, then the SINGLE function returns the cell at the intersection of the row or column of the formula cell. But, if there is no intersection, or more than one cell falls into the intersection, then this function will return a #VALUE! error. Further, if the argument provided by the user contains an array, the SINGLE function returns the first item (i.e. Row 1, Column 1).

Examples

For instance - The cell A15 in the following image contains the formula "`=SINGLE(A15:E15)`" and returns the result "C" in the cell C16 by evaluating the intersection of the rows and columns in the cell range A15 to E15.

13					
14	SINGLE(A15:E15) Function				
15	A	B	C	D	E
16			C		
17					

Version Available

This function is available in Spread for Windows Forms 12.1 or later.

SKEW

This function returns the skewness of a distribution.

Syntax

`SKEW(number1,number2,...)`

Arguments

The arguments are numeric values. Only the first argument is required. Up to 255 arguments may be included.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SKEW(A1, B2, B3, C1, C4)`

Version Available

This function is available in product version 1.0 or later.

See Also

KURT | Statistical Functions

SKEW.P

This function calculates the skewness of a distribution on the basis of population.

Syntax

`SKEW.P(value1,[value2],...)`

Arguments

The arguments passed are numeric values. Only the first argument is required. Up to 255 arguments may be included.

Remarks

This function doesn't make use of a sample but uses the standard deviation of the whole population.

If the values in the arguments passed are invalid (invalid values or data types), this functions returns an error.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SKEW.P(3,4,5,2,3,4,5,6,4,7)` gives the result 0.303

`SKEW.P(4,7,5,2,3,4,5,1,2,7)` gives the result 0.1619

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

SLN

This function returns the straight-line depreciation of an asset for one period.

Syntax

`SLN(cost,salvage,life)`

Arguments

This function has these arguments:

Argument	Description
<i>cost</i>	Initial cost of the asset
<i>salvage</i>	Value at the end of the depreciation
<i>life</i>	Number of periods over which the asset is being depreciated

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SLN(B1,1000,10)`

`SLN(R1C2,1000,10)`

`SLN(500000,20000,5)` gives the result \$96,000

Version Available

This function is available in product version 1.0 or later.

See Also

DB | DDB | SYD | Financial Functions

SLOPE

This function calculates the slope of a linear regression.

Syntax

`SLOPE(array_dep,array_ind)`

Arguments

This function has these arguments:

Argument	Description
<i>array_dep</i>	Array of dependent values (y's)
<i>array_ind</i>	Array of independent values (x's)

The arrays must be the same size.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`SLOPE (A1 : A4, B1 : B4)`

Version Available

This function is available in product version 1.0 or later.

See Also

SERIESSUM | **Math and Trigonometry Functions**

SMALL

This function returns the n th smallest value in a data set, where n is specified.

Syntax

`SMALL(array,n)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>array</i>	Array from which to return the n th largest value
--------------	---

<i>n</i>	The position (from the largest value) for which to return the value (for example, 5 to return the fifth largest value). Must be equal to or less than the number of items in the array.
----------	---

Remarks

Use this function to select a value based on its relative standing.

Data Types

Accepts array and numeric data for all arguments. Returns numeric data.

Examples

`SMALL(B4:B8,2)`

`SMALL(R4C2:R8C2,2)`

`SMALL({15, 20, 10, 5}, 2)` gives the result 10

Version Available

This function is available in product version 1.0 or later.

See Also

LARGE | **Statistical Functions**

SORT

This function sorts the data in a cell range or an array. The cell contents are extracted from the source array, the data is sorted and the results spill into the resultant range with a dynamic array of values arranged in ascending or descending order. Users can sort the values by one or more columns in the spreadsheet as per custom requirements.

The SORT function requires the sort keys to be included inside the specified array. Further, users can perform sorting on multiple fields by executing the multiple-key sorting operation on the worksheet. For instance - let's say you have a large database that you want to sort in such a way that you can obtain the sales data to analyse which region sells how many product units. Now, in this scenario, you will have to perform sorting on two fields concurrently - 1) Sorting based on the Region column (that specifies the area where the product is being sold) and 2) Sorting based on the Sales column (that contains the figures as to how many products are sold). In multiple-key sorting or multi-level data sorting, the sort index and the sort order can be of the same length which specifies multiple sort keys and their sort key orders respectively.

Syntax

`SORT (array, [sort_index], [sort_order], [by_col])`

Arguments

SORT function has the following arguments:

Argument	Description
<i>array</i>	[required] Specifies the range or array that you want to sort.
<i>sort_index</i>	[optional] Specifies the column index of the row or column to sort by. If nothing is specified, the default value 1 is used.
<i>sort_order</i>	[optional] Specifies the sort order. The value 1 indicates ascending order and the value -1 indicates descending order. The default value is 1 i.e. ascending.
<i>by_col</i>	[optional] If this argument is TRUE, it refers to the "sort by column" operation and if FALSE, it refers to the "sort by row" operation. The default value is Boolean FALSE i.e. the sort by row operation.

If you are implementing multiple-key sorting, the arguments `sort_index` [specifies multiple sort keys] and the `sort_order` [specifies the sort key orders] in the above table can be of the same length.

Data Types

Accepts a range or array that users want to sort. Returns a sorted array.

Examples

For instance - The cell D4 in the following image contains the formula `"=SORT(A4:A15)"` and returns the customer names sorted by age in the increasing order.

	A	B	C	D
1	Dynamic Array Functions			
2				
3	Customer's Name	Age	Unique List	Sort
4	Larry	32	Larry	Gilbert
5	Safeway	23	Safeway	Larry
6	Safeway	23	Raley	Larry
7	Raley	39	Vallarta	Larry
8	Vallarta	18	Gilbert	Larry
9	Safeway	23		Raley
10	Raley	39		Raley
11	Larry	32		Raley
12	Gilbert	19		Safeway
13	Larry	32		Safeway
14	Larry	32		Safeway
15	Raley	39		Vallarta
16				

If you want to sort all the unique values in the range A4 to A15, you can either apply the sort function on the unique list displayed in the column C4 or you can also combine both the functions SORT and UNIQUE into a single formula.

For instance, the cell E4 in the following image contains the formula `"=SORT(C4#)"` where # indicates a list. This formula will sort the list of values in column C (where cell C4 already contains the UNIQUE formula `"=UNIQUE(A4:A15)"`) and displays the results in column E.

Alternatively, you can also combine both the functions SORT and UNIQUE. For instance, the cell F4 in the following image contains the formula `"=SORT(UNIQUE(A4:A15))"` which returns all the unique values in the range A4:A15 sorted alphabetically.

	A	B	C	D	E	F
1	Dynamic Array Functions					
2						
3	Customer's Name	Age	Unique List	Sort	Sort Unique	Sort Unique
4	Larry	32	Larry	Gilbert	Gilbert	Gilbert
5	Safeway	23	Safeway	Larry	Larry	Larry
6	Safeway	23	Raley	Larry	Raley	Raley
7	Raley	39	Vallarta	Larry	Safeway	Safeway
8	Vallarta	18	Gilbert	Larry	Vallarta	Vallarta
9	Safeway	23		Raley		
10	Raley	39		Raley		
11	Larry	32		Raley		
12	Gilbert	19		Safeway		
13	Larry	32		Safeway		
14	Larry	32		Safeway		
15	Raley	39		Vallarta		
16						

Version Available

This function is available in Spread for Windows Forms 12.1 or later.

SORTBY

This function sorts the contents of a cell range or an array on the basis of the values present in a corresponding range or array.

Unlike the SORT function, the SORTBY function doesn't require the sort keys to be included inside the specified array or the sort range. However, it is necessary to allocate appropriate size to the sort keys (with respect to their correct length) while working with this function.

Syntax

`SORTBY(array, by_array1, [order_array1], [by_array2, order_array2], ...)`

Arguments

SORTBY function has the following arguments:

Argument	Description
----------	-------------

array	[required] Specifies the range or array that you want to sort.
by_array1	[required] Specifies the array or range of the first sort key.
order_array1	[optional] Specifies the sort order. The value 1 indicates ascending order and the value -1 indicates descending order. The default value is 1 i.e. ascending.
by_array2	[optional] Specifies the array or range of the second sort key.
order_array2	[optional] Specifies the sort order. The value 1 indicates ascending order and the value -1 indicates descending order. The default value is 1 i.e. ascending. This argument is required only if the by_array2 argument is specified.

This function can accept additional arguments in pairs. Users can specify the next sort key range and order as per custom requirements.

Data Types

Accepts a cell range or an array of data that you want to sort along with another cell range based on which the sort operation will take place. Returns a sorted array.

Examples

For instance - The cell G4 in the following image contains the formula "`=SORTBY(A4:B15,B4:B15)`". This function sorts the cell range A4 to B15 based on another cell range B4 to B15 and returns the customer names displayed along with their ages sorted in the increasing order (the default sort order).

	A	B	C	D	E	F	G	H
1	Dynamic Array Functions							
2								
3	Customer's Name	Age	Unique List	Sort	Sort Unique	Sort Unique	SortBy	
4	Larry	32	Larry	Gilbert	Gilbert	Gilbert	Vallarta	18
5	Safeway	23	Safeway	Larry	Larry	Larry	Gilbert	19
6	Safeway	23	Raley	Larry	Raley	Raley	Safeway	23
7	Raley	39	Vallarta	Larry	Safeway	Safeway	Safeway	23
8	Vallarta	18	Gilbert	Larry	Vallarta	Vallarta	Safeway	23
9	Safeway	23		Raley			Larry	32
10	Raley	39		Raley			Larry	32
11	Larry	32		Raley			Larry	32
12	Gilbert	19		Safeway			Larry	32
13	Larry	32		Safeway			Raley	39
14	Larry	32		Safeway			Raley	39
15	Raley	39		Vallarta			Raley	39
16								

Version Available

This function is available in Spread for Windows Forms 12.1 or later.

SQRT

This function returns the positive square root of the specified number.

Syntax

`SQRT(value)`

Arguments

The argument may be any positive numeric value. You must provide a positive number for the argument.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`SQRT(B4)`

`SQRT(R4C2)`

`SQRT(256)` gives the result 16

Version Available

This function is available in product version 1.0 or later.

See Also

POWER | EXP | Math and Trigonometry Functions

SQRTPI

This function returns the positive square root of a multiple of pi (p).

Syntax

SQRTPI(multiple)

Arguments

Specify the number of multiples of pi (p) of which to calculate the square root.

Remarks

This function calculates the square root of a multiple of pi.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`SQRTPI(A3)`

`SQRTPI(1)` is the same as `SQRT(PI())`

`SQRTPI(5)` gives the result 3.963327

Version Available

This function is available in product version 1.0 or later.

See Also

PI | SQRT | Statistical Functions

STANDARDIZE

This function returns a normalized value from a distribution characterized by mean and standard deviation.

Syntax

`STANDARDIZE(x,mean,stdev)`

Arguments

This function has these arguments:

Argument	Description
<i>x</i>	Value to normalize
<i>mean</i>	Arithmetic mean of the distribution
<i>stdev</i>	Standard deviation of the distribution Must be greater than zero.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`STANDARDIZE(15.6,A4,B2)`

`STANDARDIZE(88,48,1.6)` gives the result 25

Version Available

This function is available in product version 1.0 or later.

See Also

NORMDIST | NORMSDIST | Statistical Functions

STDEV

This function returns the standard deviation for a set of numbers.

Syntax

`STDEV(value1,value2,...)`

Arguments

Each argument can be a cell, a cell range, a float value, or an integer value. This function can have up to 255 arguments.

Remarks

The standard deviation is a measure of how widely values are dispersed from the average value.

The standard deviation is calculated using the "non-biased" or "n-1" method.

The equation for calculating the standard deviation is:

$$STDEV(x_n) = \sqrt{\frac{n \sum x^2 - (\sum x)^2}{n(n-1)}}$$

where x is the value and n is the number of values.

This function assumes that its arguments are a sample of the population. If your data represents the entire population, then compute the standard deviation using the **STDEVP** function.

This function differs from the STDEVA, which allows text or logical values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`STDEV(A1,B2,C3,D4,E5,F6)`

`STDEV(A1:A9)`

`STDEV(R1C2,R3C4,R4C5,R7C2)`

`STDEV(95,89,73,87,85,76,100,96,96)` gives the result 9.3422576382

Version Available

This function is available in product version 1.0 or later.

See Also

AVEDEV | **AVERAGE** | **Statistical Functions**

STDEV.P

Summary

This function returns the standard deviation for an entire specified population (of numeric values).

Syntax

STDEV.P(*value1,value2,...*)

Arguments

Each argument can be a cell, a cell range, a float value, or an integer value. This function can have up to 255 arguments.

Remarks

The standard deviation is a measure of how widely values are dispersed from the average value. If your data represents a sample of the population, then compute the standard deviation using the **STDEV** function.

The standard deviation is calculated using the "biased" or "n" method.

Logical values and text representations of numbers that are typed into the list of arguments are counted. If an argument is an array or reference, only numbers in that array or reference are counted. Empty cells, error values, logical values, or text in the array or reference are ignored.

The equation for calculating the standard deviation for a population is:

$$\sqrt{\frac{\sum (x - \bar{x})^2}{n}}$$

where x is the sample mean, AVERAGE(number1,number2,...), and n is the number of values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

STDEV.P(A1,B2,C3,D4,E5,F6)

STDEV.P(A1:A9)

STDEV.P(R1C2,R3C4,R4C5,R7C2)

STDEV.P(95,89,73,87,85,76,100,96,96) gives the result 8.80796497

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

STDEV.S

Summary

This function returns the standard deviation based on a sample (of numeric values).

Syntax

STDEV.S(*value1,value2,...*)

Arguments

Each argument can be a cell, a cell range, a float value, or an integer value. This function can have up to 255 arguments.

Remarks

The standard deviation is a measure of how widely values are dispersed from the average value. If your data represents the entire population, then compute the standard deviation using the **STDEV.P** function.

The standard deviation is calculated using the "n-1" method.

Logical values and text representations of numbers that are typed into the list of arguments are counted. If an argument is an array or reference, only numbers in that array or reference are counted. Empty cells, error values, logical values, or text in the array or reference are ignored.

The equation for calculating the standard deviation for a population is:

$$\sqrt{\frac{\sum (x - \bar{x})^2}{(n - 1)}}$$

where \bar{x} is the sample mean, AVERAGE(number1,number2,...), and n is the number of values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

STDEV.S(A1,B2,C3,D4,E5,F6)

STDEV.S(A1:A9)

STDEV.S(R1C2,R3C4,R4C5,R7C2)

STDEV.S(95,89,73,87,85,76,100,96,96) gives the result 9.342257638

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

STDEVA

This function returns the standard deviation for a set of numbers, text, or logical values.

Syntax

`STDEVA(value1,value2,...)`

Arguments

Each argument can be a cell, a cell range, a float value, an integer value, text, or a logical value. There can be up to 255 arguments. TRUE evaluates to 1 and FALSE or text evaluates to 0.

Remarks

The standard deviation is a measure of how widely values are dispersed from the average value.

The standard deviation is calculated using the "non-biased" or "n-1" method.

The equation for calculating the standard deviation is the same as for **STDEV**:

$$STDEVA(x_n) = \sqrt{\frac{n \sum x^2 - (\sum x)^2}{n(n-1)}}$$

where x is the value and n is the number of values

This function assumes that its arguments are a sample of the population.

This function differs from **STDEV** because it accepts text or logical values as well as numeric values.

Data Types

Accepts numeric, text, and logical data for all arguments. Returns numeric data.

Examples

`STDEVA(A1,B2,C3,D4,E5,F6)`

`STDEVA(A1:A9)`

`STDEVA(R1C2,R3C4,R4C5,R7C2)`

`STDEVA(95,89,73,87,85,76,100,96,96)` gives the result 9.3422576382

Version Available

This function is available in product version 2.0 or later.

See Also

AVEDEV | **AVERAGE** | **STDEV** | **STDEVPA** | **Statistical Functions**

STDEVP

This function returns the standard deviation for an entire specified population (of numeric values).

Syntax

`STDEVP(value1,value2,...)`

Arguments

Each argument can be a cell, a cell range, a float value, or an integer value. This function can have up to 255 arguments.

Remarks

The standard deviation is a measure of how widely values are dispersed from the average value.

The standard deviation is calculated using the "biased" or "n" method.

The equation for calculating the standard deviation for a population is:

$$STDEVP(x_n) = \sqrt{\frac{n \sum x^2 - (\sum x)^2}{n^2}}$$

where x is the value and n is the number of values.

This function assumes that its arguments are the entire population. If your data represents a sample of the population, then compute the standard deviation using the **STDEV** function.

This function differs from **STDEVPA**, which accepts text or logical values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`STDEVP(A1,B2,C3,D4,E5,F6)`

`STDEVP(A1:A9)`

`STDEVP(R1C2,R3C4,R4C5,R7C2)`

`STDEVP(95,89,73,87,85,76,100,96,96)` gives the result 8.8079649700

Version Available

This function is available in product version 1.0 or later.

See Also

AVERAGE | **STDEV** | **STDEVPA** | **Statistical Functions**

STDEVPA

This function returns the standard deviation for an entire specified population, including text or logical values as well as numeric values.

Syntax

`STDEVPA(value1,value2,...)`

Arguments

Each argument can be a cell, a cell range, a float value, text, a logical value, or an integer value. There can be up to 255 arguments. TRUE evaluates as 1. Text or FALSE evaluates as 0.

Remarks

The standard deviation is a measure of how widely values are dispersed from the average value.

The standard deviation is calculated using the "biased" or "n" method.

The equation for calculating the standard deviation for a population is:

$$STDEVPA(x_n) = \sqrt{\frac{n \sum x^2 - (\sum x)^2}{n^2}}$$

where x is the value and n is the number of values.

This function assumes that its arguments are the entire population. If your data represents a sample of the population, then compute the standard deviation using the **STDEVA** function.

This function differs from **STDEVP** because it accepts text or logical values as well as numeric values.

Data Types

Accepts numeric, text, and logical data for all arguments. Returns numeric data.

Examples

`STDEVPA(A1,B2,C3,D4,E5,F6)`

`STDEVPA(A1:A9)`

`STDEVPA(R1C2,R3C4,R4C5,R7C2)`

`STDEVPA(95,89,73,87,85,76,100,96,96)` gives the result 8.8079649700

Version Available

This function is available in product version 2.0 or later.

See Also

AVERAGE | **STDEVP** | **STDEVA** | **Statistical Functions**

STEYX

This function returns the standard error of the predicted y value for each x. The standard error is a measure of the amount of error in the prediction of y for a value of x.

Syntax

```
STEYX(array_dep,array_ind)
```

Arguments

This function has these arguments:

Argument	Description
<i>array_dep</i>	Array of dependent values (y's)
<i>array_ind</i>	Array of independent values (x's)

The arrays must be the same size.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

```
STEYX(A1:A17,B1:B17)
```

```
STEYX({22,33,49,21,32,37,43},{31,28,29,42,35,37,34}) gives the result 10.14406
```

Version Available

This function is available in product version 1.0 or later.

See Also

ERF | **PEARSON** | **Statistical Functions**

SUBSTITUTE

This function substitutes a new string for specified characters in an existing string.

Syntax

`SUBSTITUTE(text,old_piece,new_piece,instance)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>text</i>	String or reference to a cell containing the string in which you want to replace characters
<i>old_piece</i>	String to be replaced
<i>new_piece</i>	New string to use instead of existing string
<i>instance</i>	[Optional] Which occurrence of the existing string to replace; otherwise every occurrence is replaced

Remarks

Use this function to replace specific text with other text. Use the **REPLACE** function to replace a specific number of characters in a specific location with other characters.

Data Types

Accepts string data for the *text*, *old_piece*, and *new_piece* arguments, and numeric data for the *instance* argument. Returns string data.

Examples

`SUBSTITUTE("Down Trend","Down","Up")` gives the result Up Trend

`SUBSTITUTE("Feb 1, 1991","1","2", 3)` gives the result Feb 1, 1992

Version Available

This function is available in product version 1.0 or later.

See Also

FIND | **REPLACE** | **TRIM** | **Text Functions**

SUBTOTAL

This function calculates a subtotal of a list of numbers using a specified built-in function.

Syntax

`SUBTOTAL(functioncode,value1,value2,...)`

`SUBTOTAL(functioncode,array)`

Arguments

The *functioncode* argument is the number that represents the built-in function to use for the subtotal, as given in this table.

Built-In Function	Function Code (Include Hidden Values)	Function Code (Ignore Hidden Values)
AVERAGE	1	101
COUNT	2	102
COUNTA	3	103
MAX	4	104
MIN	5	105
PRODUCT	6	106
STDEV	7	107
STDEVP	8	108
SUM	9	109
VAR	10	110
VARP	11	111

Each additional argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments can be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

The SUBTOTAL function does not include other SUBTOTAL formula results that are in the same range.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SUBTOTAL(8,A1:B7)`

Version Available

This function is available in product version 2.0 or later.

See Also

SUMPRODUCT | SUM | Math and Trigonometry Functions

SUM

This function returns the sum of cells or range of cells.

Syntax

`SUM(value1,value2,...)`

`SUM(array)`

`SUM(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

Range references with mixed relativity for column or row end points are not supported with the SUM function. R1C[1]:R2C[2] is okay but, R1C1:R2C[2] is not.

The SUM function ignores non-numeric values passed by reference. For example, if A1 contains TRUE, A2 contains "2", and A3 contains 4, then:

`TRUE+"2"+4` evaluates to 7

`A1+A2+A3` evaluates to 7

`SUM(TRUE,"2",4)` evaluates to 7

`SUM(A1,A2,A3)` evaluates to 4

The + operator provides an auto-conversion for non-numeric values passed by constant and for non-numeric values passed by reference. The SUM function provides an auto-conversion for non-numeric values passed by constant but, ignores non-numeric values passed by reference.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SUM(A1,B7,C11)`

`SUM(A1:A9)`

`SUM(A2:A14,B2:B18,D12:D30)`

`SUM(R1C2,R3C5,R6C2,R1C7)`

`SUM(95,89,73,87,85,76,100,96,96)` gives the result 797

Version Available

This function is available in product version 1.0 or later.

See Also

SUMPRODUCT | SERIESSUM | PRODUCT | Math and Trigonometry Functions

SUMIF

This function adds the cells using a given criteria.

Syntax

`SUMIF(array,condition,sumrange)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>array</i>	Range of cells to check; each cell in the array can be a double-precision floating-point value or an integer value
<i>condition</i>	Condition that determines which cells are added, as a text, number, or expression (where expressions use the relational operators detailed in Operators in a Formula)
<i>sumrange</i>	[Optional] Range of cells to add; if omitted, then all the cells in the array are added

Data Types

Accepts numeric data for *array* and *sumrange*. Accepts text, numeric or expression data for *condition*. Returns numeric data.

Examples

```
SUMIF(A1:B7, ">150", C1:C11)
```

```
SUMIF(A1:A9, ">150")
```

Version Available

This function is available in product version 2.0 or later.

See Also

SUMPRODUCT | **SUM** | **COUNTIF** | **Math and Trigonometry Functions**

SUMIFS

This function adds the cells in a range using multiple criteria.

Syntax

`SUMIFS(array,conditionarray,condition,...)`

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Range of cells to check; each cell in the array can be a double-precision floating-point value or an integer value
<i>conditionarray</i>	Range of cells to check; each cell in the array can be a double-precision floating-point value or an integer value
<i>condition</i>	Condition that determines which cells are added, as a text, number, or expression (where expressions use the relational operators detailed in Operators in a Formula)

Data Types

Accepts numeric data for *array*. Accepts text, numeric or expression data for *condition*. Returns numeric data.

Examples

```
SUMIFS(A1:A10, B1:B10, ">0", C1:C10, "<10")
```

Version Available

This function is available in product version 5.0 or later.

See Also

SUMPRODUCT | **SUM** | **COUNTIF** | **Math and Trigonometry Functions**

SUMPRODUCT

This function returns the sum of products of cells. Multiplies corresponding components in the given arrays, and returns the sum of those products.

Syntax

`SUMPRODUCT(array1,array2,...)`

Arguments

There must be at least two arrays (*array1*, *array2*) and optionally up to 255 arrays (*array3*, ...) as arguments. The arrays must have the same dimension.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SUMPRODUCT(A1:A17,B1:B17,C1:C17)`

`SUMPRODUCT({2,3,5,6,4,7},{5,6,4,4,7,2})` gives the result 114

Version Available

This function is available in product version 1.0 or later.

See Also

PRODUCT | **SUM** | **Math and Trigonometry Functions**

SUMSQ

This function returns the sum of the squares of the arguments.

Syntax

`SUMSQ(value1,value2,...)`

`SUMSQ(array)`

`SUMSQ(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SUMSQ(A1,B7,C11)`

`SUMSQ(A1:A9)`

`SUMSQ(R1C2,R3C5,R6C2,R1C7)`

`SUMSQ(95,89,73,87,85,76,100,96,96)` gives the result 71277

Version Available

This function is available in product version 1.0 or later.

See Also

SUMPRODUCT | SUM | Math and Trigonometry Functions

SUMX2MY2

This function returns the sum of the difference of the squares of corresponding values in two arrays.

Syntax

`SUMX2MY2(array_x,array_y)`

Arguments

This function has these arguments:

Argument	Description
<i>array_x</i>	First array of values (x's)
<i>array_y</i>	Second array of values (y's)

The arrays must be the same size.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SUMX2MY2 (A1:A17, B1:B17)`

Version Available

This function is available in product version 1.0 or later.

See Also

SUMX2PY2 | SUMXMY2 | SUM | Math and Trigonometry Functions

SUMX2PY2

This function returns the sum of the sum of squares of corresponding values in two arrays.

Syntax

`SUMX2PY2(array_x,array_y)`

Arguments

This function has these arguments:

Argument	Description
<i>array_x</i>	First array of values (x's)
<i>array_y</i>	Second array of values (y's)

The arrays must be the same size.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SUMX2PY2(A1:A17,B1:B17)`

Version Available

This function is available in product version 1.0 or later.

See Also

SUMX2MY2 | SUMXMY2 | SUM | Math and Trigonometry Functions

SUMXMY2

This function returns the sum of the square of the differences of corresponding values in two arrays.

Syntax

`SUMXMY2(array_x,array_y)`

Arguments

This function has these arguments:

Argument	Description
<i>array_x</i>	First array of values (x's)
<i>array_y</i>	Second array of values (y's)

The arrays must be the same size.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SUMXMY2(A1:A17,B1:B17)`

Version Available

This function is available in product version 1.0 or later.

See Also

SUMX2PY2 | SUMX2MY2 | SUM | Math and Trigonometry Functions

SWITCH

This function compares specified expression against given list of values and returns the result according to the first matching value.

Syntax

SWITCH(*expression,value,result,result_no_match*)

Arguments

This function has the following arguments:

Argument	Description
<i>expression</i>	Value or expression to compare
<i>value</i>	Value compared against expression
<i>result</i>	Value returned if comparison matches
<i>result_no_match</i>	Value returned if comparison do not match

Remarks

In this function, argument *value* and *result* can take upto 126 different entries each.

Data Types

Accepts data of any type. Returns data of any type.

Examples

SWITCH(WEEKDAY(A2),1,"Sunday",2,"Monday",3,"Tuesday","No match") gives the result No match.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

SYD

This function returns the sum-of-years' digits depreciation of an asset for a specified period.

Syntax

`SYD(cost,salvage,life,period)`

Arguments

This function has these arguments:

Argument	Description
cost	Initial cost of the asset
salvage	Value at the end of the depreciation
life	Number of periods over which the asset is being depreciated
period	Period for depreciation; must use the same units as the <i>life</i> argument.

Remarks

This function calculates the digits depreciation as follows:

$$SYD = \frac{(cost - salvage) \times (life - period + 1) \times 2}{(life)(life + 1)}$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SYD(B1,1000,10,1)`

`SYD(R1C2,1000,10,1)`

`SYD(100000,10000,5,2)` gives the result \$2,4000

Version Available

This function is available in product version 1.0 or later.

See Also

DB | DDB | SLN | Financial Functions

Functions T to Z

Functions T to Z

T	T.DIST	T.DIST.2T	T.DIST.RT
T.INV	T.INV.2T	T.TEST	TAN
TANH	TBILLEQ	TBILLPRICE	TBILLYIELD
TDIST	TEXT	TEXTJOIN	TIME
TIMEVALUE	TINV	TODAY	TRANSPOSE
TREND	TRIM	TRIMMEAN	TRUE
TRUNC	TTEST	TYPE	UNICHAR
UNICODE	UNIQUE	UPPER	USDOLLAR
VALUE	VAR	VAR.P	VAR.S
VARA	VARP	VARPA	VDB
VLOOKUP	WEBSERVICE	WEEKDAY	WEEKNUM
WEIBULL	WEIBULL.DIST	WORKDAY	WORKDAY.INTL
XIRR	XNPV	XOR	YEAR
YEARFRAC	YIELD	YIELDDISC	YIELDMAT
Z.TEST	ZTEST		

T

This function returns the text in a specified cell.

Syntax

T(*value*)

Arguments

The argument is any cell reference.

Remarks

If the cell contains text, this function returns text. If the cell contains a number, this function returns an empty string.

Data Types

Accepts cell reference. Returns string data.

Examples

T(B3) If B3 contains "Test" then this function returns "Test".

T(R3C2)

T(A1)

Version Available

This function is available in product version 2.0 or later.

See Also

LEN | ISTEXT | CHAR | UPPER | LOWER | Text Functions

T.DIST

This function returns the probability for the t-distribution.

Syntax

T.DIST(*x,deg,cumulative*)

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>x</i>	Numeric value used to evaluate the distribution
<i>deg</i>	Number of degrees of freedom to characterize the distribution; if not an integer, the number is truncated
<i>cumulative</i>	A logical value that determines the form of the function. If <i>cumulative</i> is TRUE, the function returns the cumulative distribution function; if FALSE, it returns the probability density function

Remarks

The #VALUE! error value is returned if *x* or *deg* are nonnumeric.

Data Types

Accepts numeric data for *x* and *deg* arguments. Returns numeric data.

Examples

T.DIST(A1,B45,TRUE)

T.DIST(0.245,2,TRUE) gives the result 1.4146507236438

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

TDIST

T.DIST.2T

This function returns the t-distribution.

Syntax

T.DIST.2T(*x*,*deg*)

Arguments

This function has these arguments:

Argument	Description
<i>x</i>	Numeric value at which to evaluate the distribution
<i>deg</i>	Number of degrees of freedom to characterize the distribution

Remarks

The #VALUE! error value is returned if any argument is nonnumeric.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

T.DIST.2T(A1,B45,2)

T.DIST.2T(0.245,2,1) gives the result 0.414651

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

TDIST

T.DIST.RT

This function returns the t-distribution.

Syntax

T.DIST.RT(*x*,*deg*)

Arguments

This function has these arguments:

Argument	Description
<i>x</i>	Numeric value at which to evaluate the distribution
<i>deg</i>	Number of degrees of freedom to characterize the distribution

Remarks

The #VALUE! error value is returned if any argument is nonnumeric.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

T.DIST.RT(A1,B45)

T.DIST.RT(0.245,2) gives the result 0.41465072364379996

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

TDIST

T.INV

This function returns the t-value of the student's t-distribution as a function of the probability and the degrees of freedom.

Syntax

T.INV(*prog*,*deg*)

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>prog</i>	Probability of the student's t-distribution
-------------	---

<i>deg</i>	Number of degrees of freedom to characterize the distribution; if not an integer, the number is truncated
------------	---

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

T.INV(A4,2)

T.INV(0.68,4) gives the result 0.5051744394100004

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

TINV

T.INV.2T

This function returns the t-value of the student's t-distribution as a function of the probability and the degrees of freedom.

Syntax

T.INV.2T(*prog*,*deg*)

Arguments

This function has these arguments:

Argument	Description
<i>prog</i>	Probability of the two-tailed student's t-distribution
<i>deg</i>	Number of degrees of freedom to characterize the distribution; if not an integer, the number is truncated

Remarks

The #VALUE! error value is returned if any argument is nonnumeric.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

T.INV.2T(A4,2)

T.INV.2T(0.68,4) gives the result 0.44400612800394834

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

TINV

T.TEST

This function returns the probability associated with a t-test.

Syntax

T.TEST(*array1,array2,tails,type*)

Arguments

This function has these arguments:

Argument	Description
<i>array1</i>	Array of values in first data set
<i>array2</i>	Array of values in second data set
<i>tails</i>	Number of tails
<i>type</i>	Type of t-test to perform (1, 2, or 3)

Remarks

The *tails* and *type* arguments are truncated to integers. The #VALUE! error value is returned if *tails* or *type* is nonnumeric.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

T.TEST(A1:A17,B1:B17,4,3)

T.TEST({2,2,2,3,4},{2,3,3,4,5},1,2) gives the result 0.1260360000000153

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

TTEST

TAN

This function returns the tangent of the specified angle.

Syntax

`TAN(angle)`

Arguments

This function can take any real number as an argument. The *angle* argument is the angle in radians for which you want the tangent.

Remarks

If the angle is in degrees, multiply it by $\text{PI}/180$ to convert it to radians.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`TAN(B3)`

`TAN(R3C2)`

`TAN(45*PI()/180)` gives the result 1

`TAN(RADIANS(20))`

Version Available

This function is available in product version 1.0 or later.

See Also

ATAN | ATAN2 | COS | SIN | Math and Trigonometry Functions

TANH

This function returns the hyperbolic tangent of the specified number.

Syntax

TANH(*value*)

Remarks

You can use any real number for the value argument.

The equation for calculating the hyperbolic sine is:

$$\text{TANH}(z) = \frac{\text{SINH}(z)}{\text{COSH}(z)}$$

Data Types

Accepts numeric data. Returns numeric data.

Examples

TANH(B3)

TANH(R1C2)

TANH(0.5) gives the result 0.4621171573

Version Available

This function is available in product version 1.0 or later.

See Also

ATAN | ATANH | COSH | SINH | TAN | Math and Trigonometry Functions

TBILLEQ

This function returns the equivalent yield for a Treasury bill (or T-bill).

Syntax

TBILLEQ(*settle,mature,discount*)

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the Treasury bill
<i>mature</i>	Maturity date for the Treasury bill
<i>discount</i>	Discount rate for the Treasury bill

Remarks

This function returns a #VALUE! error when settle or mature is invalid. Settle and mature are truncated to integers. If discount is less than or equal to 0, a #NUM! error is returned. If settle is greater than mature or if mature is more than one year after settle, a #NUM! error is returned. This function is calculated as $(365 \times \text{rate}) / (360 - (\text{rate} \times \text{DSM}))$, where DSM is the number of days between settle and mature computed according to the 360 days per year basis.

Data Types

Accepts numeric and DateTime object data for all arguments. Returns numeric data.

Examples

TBILLEQ(A1,B2,C3)

TBILLEQ("3/31/2003","6/1/2003",0.0532) gives the result 0.054437659 (or 5.44%)

Version Available

This function is available in product version 1.0 or later.

See Also

TBILLPRICE | TBILLYIELD | **Financial Functions**

TBILLPRICE

This function returns the price per \$100 face value for a Treasury bill (or T-bill).

Syntax

TBILLPRICE(*settle,mature,discount*)

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the Treasury bill
<i>mature</i>	Maturity date for the Treasury bill
<i>discount</i>	Discount rate for the Treasury bill

Remarks

This function returns a #VALUE! error when settle or mature is invalid. Settle and mature are truncated to integers. If discount is less than or equal to 0, a #NUM! error is returned. If settle is greater than mature or if mature is more than one year after settle, a #NUM! error is returned.

Data Types

Accepts numeric and DateTime object data for all arguments. Returns numeric data.

Examples

TBILLPRICE (A1, B2, C3)

TBILLPRICE ("3/31/2003", "6/1/2003", 0.065) gives the result \$98.88055556

Version Available

This function is available in product version 1.0 or later.

See Also

TBILLEQ | TBILLYIELD | **Financial Functions**

TBILLYIELD

This function returns the yield for a Treasury bill (or T-bill).

Syntax

TBILLYIELD(*settle,mature,priceper*)

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the Treasury bill
<i>mature</i>	Maturity date for the Treasury bill
<i>priceper</i>	Price per \$100 face value for the Treasury bill

Remarks

This function returns a #VALUE! error when *settle* or *mature* is invalid. *Settle* and *mature* are truncated to integers. If *priceper* is less than or equal to 0, a #NUM! error is returned. If *settle* is greater than or equal to *mature* or if *mature* is more than one year after *settle*, a #NUM! error is returned.

Data Types

Accepts numeric and DateTime object data for all arguments. Returns numeric data.

Examples

TBILLYIELD(A1, B2, C3)

TBILLYIELD("3/31/2003", "6/1/2003", 98.65) gives the result 0.0794598041299475 (or 5.80%)

Version Available

This function is available in product version 1.0 or later.

See Also

TBILLEQ | TBILLPRICE | **Financial Functions**

TDIST

This function returns the probability for the t-distribution.

Syntax

`TDIST(x,deg,tails)`

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>x</i>	Probability of the two-tailed student's t-distribution
<i>deg</i>	Number of degrees of freedom to characterize the distribution; if not an integer, the number is truncated
<i>tails</i>	Number of tails to return; if not an integer, the number is truncated; for 1, returns one-tailed distribution; for 2, returns two-tailed distribution

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`TDIST(A1,B45,2)`

`TDIST(0.245,2,1)` gives the result 0.414651

Version Available

This function is available in product version 1.0 or later.

See Also

FDIST | **TINV** | **TTEST** | **Statistical Functions**

TEXT

This function formats a number and converts it to text.

Syntax

`TEXT(value,text)`

Arguments

The text argument requires a string. Value requires numeric data or a reference to a cell that contains numeric data.

Data Types

Returns string data.

Examples

`TEXT(A1,"$0.00")` gives the result \$10.00if A1 contains 10

Version Available

This function is available in product version 5.0 or later.

See Also

[CHAR](#) | [EXACT](#) | [Text Functions](#)

TEXTJOIN

This function combines the text from multiple strings, and includes the specified delimiter between each text value.

Syntax

`TEXTJOIN(delimiter, ignore_empty, value1, value2,...)`

Arguments

This function has the following arguments:

Argument	Description
<i>delimiter</i>	Refers to a text string (either empty, or one or more characters inside double quotes) or a cell reference containing text value. If you pass a number in this argument, it will be recognized as text.
<i>ignore_empty</i>	Accepts a boolean TRUE or FALSE. If this value is TRUE, it ignores empty cells.
<i>value1</i>	Refers to a text string, or an array of strings to be joined.
<i>value2</i>	[Optional] Refers to the additional text strings to be joined.

Remarks

The resultant string can hold a maximum of 32767 characters. If the resultant string exceeds this limit, this function will return an error.

Data Types

Returns string data.

Examples

`TEXTJOIN(" ",TRUE,"You","may","get","late","for","party.")` gives the result You may get late for party.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

TIME

This function returns the decimal number for the specified time.

Syntax

`TIME(hour,minutes,seconds)`

Arguments

This function has these arguments:

Argument	Description
<i>hour</i>	Hour as a number from 0 to 23.
<i>minutes</i>	Minutes as a number from 0 to 59.
<i>seconds</i>	Seconds as a number from 0 to 59.

Data Types

Accepts numeric data for all arguments. Returns a decimal value.

Examples

`TIME(A1,B1,C1)`

`TIME(R1C1,R1C2,R1C3)`

`TIME(12,0,0)` gives the result 0.5

`TIME(16,48,10)` gives the result 0.7

Version Available

This function is available in product version 1.0 or later.

 **Note:** If a user uses LegacyBehaviors.CalculationEngine, TIME function will return the DateTime object instead of the serial numeric value. For more details, please refer to [breaking changes for legacy behaviors](#).

See Also

HOUR | **MINUTE** | **DAY** | **NOW** | **TODAY** | **Date and Time Functions**

TIMEVALUE

This function returns the decimal number representing a particular time in Excel.

Syntax

`TIMEVALUE(time_string)`

Arguments

Specify the time as a text string.

Remarks

Use this function to convert the time represented by a text string to a decimal number.

Data Types

Accepts string data. Returns the decimal number that represents the time in Excel.

Examples

`TIMEVALUE (B18)`

`TIMEVALUE (R18C2)`

`TIMEVALUE ("5:29")` gives the result 0.228472

`TIMEVALUE ("5:29 PM")` gives the result 0.728472

`TIMEVALUE ("17:29")` gives the result 0.728472

Version Available

This function is available in product version 1.0 or later.



Note: If a user uses LegacyBehaviors.CalculationEngine, TIMEVALUE function will return the DateTime object instead of the serial numeric value. For more details, please refer to [breaking changes for legacy behaviors](#).

See Also

TIME | DATEVALUE | Date and Time Functions

TINV

This function returns the t-value of the student's t-distribution as a function of the probability and the degrees of freedom.

Syntax

`TINV(prog,deg)`

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>prog</i>	Probability of the two-tailed student's t-distribution
-------------	--

<i>deg</i>	Number of degrees of freedom to characterize the distribution; if not an integer, the number is truncated
------------	---

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`TINV(A4,2)`

`TINV(0.68,4)` gives the result 0.444006

Version Available

This function is available in product version 1.0 or later.

See Also

TDIST | TTEST | Statistical Functions

TODAY

This function returns a serial number representing the current date in Excel. The results are updated as soon as the worksheet is opened or refreshed.

Syntax

TODAY()

Arguments

This function does not accept arguments.

Remarks

If you use this function in a date-time cell (`DateTimeCellType`), the cell formats the value using the date format settings.

This function is updated only when the spreadsheet or cell containing the function is recalculated. This is a volatile function with version 2.5 or later.

Data Types

Does not accept data. Returns a numeric value (a serial number) that represents the current date in Excel.

Examples

If today is the 13th of February in the year 2019, then
TODAY() gives the result 43509.

Version Available

This function is available in product version 1.0 or later. This function is a volatile function in version 2.5 or later.

 **Note:** If a user uses `LegacyBehaviors.CalculationEngine`, TODAY function will return the `DateTime` object instead of the serial numeric value. For more details, please refer to [breaking changes for legacy behaviors](#).

See Also

DATE | **DAY** | **NOW** | **TIME** | **Date and Time Functions**

TRANSPOSE

This function returns a vertical range of cells as a horizontal range or a horizontal range of cells as a vertical range.

Syntax

TRANSPOSE(array)

Arguments

The *array* argument is a range of cells or an array that you want to switch.

Remarks

This function uses the first row of the array as the first column of the new array and so on.

Use the **INDEX** function to get individual elements from the returned array.

Data Types

Accepts an array. Returns an array.

Examples

`TRANSPOSE(A2:A5)`

Version Available

This function is available in product version 2.0 or later.

See Also

HLOOKUP | **INDEX** | **LOOKUP** | **VLOOKUP** | **Lookup Functions**

TREND

This function returns values along a linear trend. This function fits a straight line to the arrays known x and y values. Trend returns the y values along that line for the array of specified new x values.

Syntax

`TREND(y,x,newx,constant)`

Arguments

This function has these arguments:

Argument	Description
<i>y</i>	Set of y values that are known in the relationship $y=mx+b$
<i>x</i>	(Optional) X is an optional set of x values that may be known in the relationship $y=mx+b$
<i>newx</i>	New x values for which this functions returns the corresponding y values
<i>constant</i>	Logical value that specifies whether to force the constant b to equal 0

Remarks

If constant is true or omitted then b is calculated normally. If constant is false then b is equal to 0 and the m values are adjusted so that $y=mx$.

If x is omitted then x defaults to the array {1,2,3...}, that has the same dimensions as y.

If newx is omitted then it defaults to x.

Use the **INDEX** function to get individual elements from the returned array.

Data Types

Accepts an array. Returns an array.

Examples

`TREND(A2:A7,C2:C7,A9:A10)`

Version Available

This function is available in product version 2.0 or later.

See Also

AVEDEV | AVERAGEA | FREQUENCY | DEVSQ | GROWTH | INDEX | MEDIAN | VAR | Statistical Functions

TRIM

This function removes extra spaces from a string and leaves single spaces between words.

Syntax

`TRIM(text)`

Arguments

The argument specifies the string containing the spaces you want to remove.

Data Types

Accepts string data. Returns string data.

Examples

`TRIM(" First Quarter")` gives the result `First Quarter`

Version Available

This function is available in product version 1.0 or later.

See Also

CLEAN | **SUBSTITUTE** | **Text Functions**

TRIMMEAN

This function returns the mean of a subset of data excluding the top and bottom data.

Syntax

`TRIMMEAN(array,percent)`

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Array of values to trim and find the mean
<i>percent</i>	Fractional amount of data in array to trim (to exclude from calculation)

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`TRIMMEAN(A1:A17,0.25)`

Version Available

This function is available in product version 1.0 or later.

See Also

GEOMEAN | **HARMEAN** | **Statistical Functions**

TRUE

This function returns the value for logical TRUE.

Syntax

TRUE()

Arguments

This function does not accept arguments.

Data Types

Does not accept data. Returns numeric (boolean) data.

Example

TRUE() gives the result 1 (TRUE)

Version Available

This function is available in product version 1.0 or later.

See Also

FALSE | **IF** | **Logical Functions**

TRUNC

This function removes the specified fractional part of the specified number.

Syntax

`TRUNC(value,precision)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>value</i>	Number to truncate
--------------	--------------------

<i>precision</i>	Integer representing the precision; if greater than zero, truncates to the specified number of decimal places; if zero (or not specified), truncate to the nearest whole number; if less than zero, rounds the value left of the decimal to the nearest order of tens
------------------	---

Remarks

The TRUNC and INT functions are similar in that both can return integers. Use the TRUNC function to remove the decimal portion of the number; the TRUNC function does not round up or down. Use the INT function to round numbers down to the nearest integer based decimal portion of the number.

These functions differ also when using negative numbers: TRUNC(-4.2, 0) returns -4, but INT(-4.2) returns -5 because -5 is the lower number.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`TRUNC(B16)`

`TRUNC(R16C2)`

`TRUNC(5.745)` gives the result 5

`TRUNC(-5.745)` gives the result -5

`TRUNC(5.745,2)` gives the result 5.74

`TRUNC(PI())` gives the result 3

Version Available

This function is available in product version 1.0 or later.

See Also

CEILING | EVEN | FLOOR | INT | Math and Trigonometry Functions

TTEST

This function returns the probability associated with a t-test.

Syntax

`TTEST(array1,array2,tails,type)`

Arguments

This function has these arguments:

Argument	Description
<i>array1</i>	Array of values in first data set
<i>array2</i>	Array of values in second data set
<i>tails</i>	Number of tails
<i>type</i>	Type of t-test to perform (1, 2, or 3)

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`TTEST(A1:A17,B1:B17,4,3)`

`TTEST({2,2,2,3,4},{2,3,3,4,5},1,2)` gives the result 0.126036

Version Available

This function is available in product version 1.0 or later.

See Also

FTEST | TDIST | TINV | ZTEST | Statistical Functions

TYPE

This function returns the type of value.

Syntax

TYPE(value)

Arguments

The argument is any value as summarized here:

Type of Value	Returned Number
Number	1
DateTime object	1
TimeSpan object	1
Text	2
Logical value	4
Error value	16
Array	64

Data Types

Accepts many types of data. Returns numeric data.

Examples

```
TYPE(G15)
```

```
TYPE(R15C7)
```

```
TYPE(154) gives the result 1
```

```
TYPE("String") gives the result 2
```

```
TYPE(TRUE) gives the result 4
```

Version Available

This function is available in product version 1.0 or later.

See Also

ERRORTYPE | ISERROR | ISLOGICAL | ISNUMBER | ISTEXT | Information Functions

UNICHAR

This function returns the Unicode character specified by a number.

Syntax

UNICHAR(*value*)

Arguments

For the argument, specify the Unicode number representing a character.

Remarks

The resultant Unicode character can be a string in UTF-8 or UTF-16 code.

If 0 is passed in the argument or the passed numerals are out of range, this function returns an error.

Data Types

Accepts numeric data. Returns string data.

Examples

UNICHAR(86) gives the result V.

UNICHAR(79) gives the result O.

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

UNICODE

This function returns the number (code) corresponding to first character of specified text.

Syntax

UNICODE(*value*)

Arguments

For the argument, specify a text value to obtain the Unicode value.

Remarks

If the passed arguments contain an invalid data type, this function will return an error.

Data Types

Accepts string data. Returns numeric data.

Examples

UNICODE("D") gives the result 68

UNICODE("tor") gives the result 116

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

UNIQUE

This function returns a list of all the unique values in a cell range.

Syntax

UNIQUE(array,[by_col],[occurs_once])

Arguments

UNIQUE function has the following arguments:

Argument Description

- array* [required] Specifies the range or array from which you want to return unique values.
- by_col* [optional] Specifies the logical value that indicates how to compare. If this argument is TRUE, it refers to "by column" and if FALSE, it refers to the "by row".
- occurs_once* [optional] Specifies a logical value. If this argument is TRUE, it will return unique values that occur only once. In case the value is FALSE, all the unique values will be included in the result.

Data Types

Accepts a range or array. Returns a list of unique values.

Examples

For instance - The cell C4 in the following image contains the formula "=UNIQUE(A4:A15)" and returns only the unique customer names from the values in cell range A4 to A15. Based on the number of unique values, the dynamic array formula spills to the cell range C5 to C8 automatically.

	A	B	C
1	Dynamic Array Functions		
2			
3	Customer's Name	Age	Unique List
4	Larry	32	Larry
5	Safeway	23	Safeway
6	Safeway	23	Raley
7	Raley	39	Vallarta
8	Vallarta	18	Gilbert
9	Safeway	23	
10	Raley	39	
11	Larry	32	
12	Gilbert	19	
13	Larry	32	
14	Larry	32	
15	Raley	39	
16			

Version Available

This function is available in Spread for Windows Forms 12.1 or later.

UPPER

This function converts text to uppercase letters.

Syntax

`UPPER(string)`

Arguments

The argument is the text you want to convert to uppercase. The argument may be a string, a reference to a cell containing a string, or a formula that returns a string.

Remarks

This function does not change characters in value that are not letters.

Data Types

Accepts string data. Returns string data.

Examples

`UPPER(G15)`

`UPPER(R15C7)`

`UPPER("Report")` gives the result `REPORT`

`UPPER("summary")` gives the result `"SUMMARY"`

Version Available

This function is available in product version 1.0 or later.

See Also

PROPER | LOWER | T | Text Functions

USDOLLAR

This function converts a number to text using currency format, with the decimals rounded to the specified place.

Syntax

`DOLLAR(number,digits)`

Arguments

This function has the following arguments:

Argument	Description
<i>number</i>	Refers to the numeric value to convert to text using the currency format
<i>digits</i>	[Optional] Refers to the number of decimal places to maintain; if negative, the value is rounded to the left of the decimal point; if omitted, the system locale setting is used to determine the number of decimal places

Remarks

This function always shows U.S. currency.

Data Types

Accepts numeric data for both arguments. Returns string data.

Examples

`USDOLLAR (A5, B2)`

`USDOLLAR (R1B2, R3D4)`

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

VALUE

This function converts a text string that is a number to a numeric value.

Syntax

`VALUE(text)`

Arguments

This function has these arguments:

Argument	Description
<i>text</i>	Number in quotation marks or a reference to a cell with the text.

Remarks

The text can be in number, date, or time format. If the text is not in the correct format, a #VALUE! error is returned.

Data Types

Accepts string data. Returns numeric data.

Examples

`VALUE("$9,000")` gives the result 9000

Version Available

This function is available in product version 3.0 or later.

See Also

DOLLAR | DOLLARFR | FIXED | Text Functions

VAR

This function returns the variance based on a sample of a population, which uses only numeric values.

Syntax

`VAR(value1,value2,...)`

`VAR(array)`

`VAR(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

The variance returns how spread out a set of data is.

This function uses the following equation to calculate the variance, where n is the number of values.

$$VAR(x_n) = \frac{n \sum x^2 - (\sum x)^2}{n(n-1)}$$

where x is the value and n is the number of values.

This function assumes that its arguments are a sample of the population. If your data represents the entire population, then compute the variance using the **VARP** function.

This function differs from **VARA**, which accepts text and logical values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`VAR(B3,C4,B2,D10,E5)`

`VAR(A1:A9)`

`VAR(R1C2,100,R2C5,102)`

`VAR(R1C1:R9C1)`

`VAR(R1C1:R1C9)`

`VAR(98,85,76,87,92,89,90)` gives the result 45.8095238095

Version Available

This function is available in product version 1.0 or later.

See Also

AVERAGE | COVAR | VARP | VARA | Statistical Functions

VAR.P

Summary

This function returns variance based on the entire population, which uses only numeric values.

Syntax

VAR.P(*value1,value2,...*)

VAR.P(*array*)

VAR.P(*array1,array2,...*)

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

The variance returns how spread out a set of data is.

Logical values and text representations of numbers that are typed into the list of arguments are counted. If an argument is an array or reference, only numbers in that array or reference are counted. Empty cells, error values, logical values, or text in the array or reference are ignored.

This function uses the following equation to calculate the variance,

$$\frac{\sum (x - \bar{x})^2}{n}$$

where \bar{x} is the sample mean AVERAGE(number1,number2,...) and n is the number of values.

This function assumes that its arguments are the entire population. If your data represents only a sample of the population, then compute the variance using the **VAR.S** function.

This function differs from **VARPA**, which accepts logical or text values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

VAR.P(B3,C4,B2,D10,E5)

VAR.P(A1:A9)

VAR.P(R1C2,100,R2C5,102)

VAR.P(98,85,76,87,92,89,90) gives the result 39.26530612

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

VAR.S

Summary

This function returns variance based on a sample, which uses only numeric values.

Syntax

`VAR.S(value1,value2,...)`

`VAR.S(array)`

`VAR.S(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

The variance returns how spread out a set of data is.

Logical values and text representations of numbers that are typed into the list of arguments are counted. If an argument is an array or reference, only numbers in that array or reference are counted. Empty cells, error values, logical values, or text in the array or reference are ignored.

This function uses the following equation to calculate the variance,

$$\frac{\sum(x - \bar{x})^2}{(n - 1)}$$

where x is the sample mean `AVERAGE(number1,number2,...)` and n is the number of values.

This function differs from **VARA**, which accepts logical or text values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`VAR.S(B3,C4,B2,D10,E5)`

`VAR.S(A1:A9)`

`VAR.S(R1C2,100,R2C5,102)`

`VAR.S(98,85,76,87,92,89,90)` gives the result 45.80952381

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

VARA

This function returns the variance based on a sample of a population, which includes numeric, logical, or text values.

Syntax

`VARA(value1,value2,...)`

`VARA(array)`

`VARA(array1,array2,...)`

Remarks

Each argument can be a double-precision floating-point value, an integer value, text, a logical value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

The variance returns how spread out a set of data is.

This function uses the following equation to calculate the variance, where n is the number of values.

$$VARA(x_n) = \frac{n \sum x^2 - (\sum x)^2}{n(n-1)}$$

where x is the value and n is the number of values.

This function assumes that its arguments are a sample of the population. If your data represents the entire population, then compute the variance using the **VARPA** function.

This function differs from **VAR** because it accepts text and logical values as well as numeric values.

Data Types

Accepts numeric, logical, and text data for all arguments. Returns numeric data.

Examples

`VARA(B3,C4,B2,D10,E5)`

`VARA(A1:A9)`

`VARA(R1C2,100,R2C5,102)`

`VARA(R1C1:R9C1)`

`VARA(R1C1:R1C9)`

`VARA(98,85,76,87,92,89,90)` gives the result 45.8095238095

Version Available

This function is available in product version 2.0 or later.

See Also

AVERAGEA | VAR | VARP | Statistical Functions

VARP

This function returns variance based on the entire population, which uses only numeric values.

Syntax

`VARP(value1,value2,...)`

`VARP(array)`

`VARP(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

The variance returns how spread out a set of data is.

This function uses the following equation to calculate the variance, where n is the number of values.

$$VARP(x_n) = \frac{n \sum x^2 - (\sum x)^2}{n^2}$$

where x is the value and n is the number of values.

This function assumes that its arguments are the entire population. If your data represents only a sample of the population, then compute the variance using the **VAR** function.

This function differs from **VARPA**, which accepts logical or text values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`VARP(B3,C4,B2,D10,E5)`

`VARP(A1:A9) VARP(R1C2,100,R2C5,102)`

`VARP(98,85,76,87,92,89,90)` gives the result 39.2653061224

Version Available

This function is available in product version 1.0 or later.

See Also

AVERAGE | **VAR** | **VARPA** | **Statistical Functions**

VARPA

This function returns variance based on the entire population, which includes numeric, logical, or text values.

Syntax

`VARPA(value1,value2,...)`

`VARPA(array)`

`VARPA(array1,array2,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, text, a logical value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

The variance returns how spread out a set of data is.

Each argument can be a double-precision floating-point value, an integer value, text, a logical value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

This function uses the following equation to calculate the variance, where n is the number of values.

$$VARPA(x_n) = \frac{n \sum x^2 - (\sum x)^2}{n^2}$$

where x is the value and n is the number of values.

This function assumes that its arguments are the entire population. If your data represents only a sample of the population, then compute the variance using the **VARA** function.

This function differs from **VARP** because it accepts logical and text values as well as numeric values.

Data Types

Accepts numeric, logical, and text data for all arguments. Returns numeric data.

Examples

`VARPA(B3,C4,B2,D10,E5)`

`VARPA(A1:A9) VARPA(R1C2,100,R2C5,102)`

`VARPA(98,85,76,87,92,89,90)` gives the result 39.2653061224

Version Available

This function is available in product version 2.0 or later.

See Also

AVERAGEA | VARA | VARP | Statistical Functions

VDB

This function returns the depreciation of an asset for any period you specify using the variable declining balance method.

Syntax

`VDB(cost,salvage,life,start,end,factor,switchnot)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>cost</i>	Initial cost of the asset
<i>salvage</i>	Value at the end of the depreciation period
<i>life</i>	Number of periods over which the asset is being depreciated
<i>start</i>	Number representing the starting period for which to calculate the depreciation in the same units as <i>life</i> ; if not an integer, the number is truncated
<i>end</i>	Number representing the ending period for which to calculate the depreciation in the same units as <i>life</i> ; if not an integer, the number is truncated
<i>factor</i>	[Optional] Rate at which the balance declines; if omitted, uses two (2)
<i>switchnot</i>	[Optional] Logical value specifying whether to switch to straight-line depreciation when depreciation is greater than the declining balance calculation; if omitted uses FALSE

Remarks

If *factor* is omitted, the calculation uses two, which represents the double-declining balance method. For other methods, use a different value. For more information about the double-declining balance method, see **DDB**.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`VDB(B1,1000,10,1,8)`

`VDB(50000,500,1200,100,1000,1)` gives the result \$37,122.94

Version Available

This function is available in product version 1.0 or later.

See Also

DB | **DDB** | **SLN** | **SYD** | **Financial Functions**

VLOOKUP

This function searches for a value in the leftmost column and returns a value in the same row from a column you specify.

Syntax

VLOOKUP(*value,array,colindex,approx*)

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>value</i>	Value for which to search
--------------	---------------------------

<i>array</i>	Array or cell range that contains the data to search
--------------	--

<i>colindex</i>	Column number in the array from which the matching value is returned
-----------------	--

<i>approx</i>	[Optional] Logical value indicating whether to find an approximate match; if omitted, uses TRUE and finds an approximate match
---------------	--

Remarks

If *approx* is FALSE, it finds an exact match, not an approximate match. If it cannot find one, it returns an #N/A error value.

If *approx* is TRUE or omitted, and the *value* cannot be found, then the largest value that is less than the *value* is used.

This function is similar to **HLOOKUP** except that it searches vertically (by column), instead of by row (horizontally).

Data Types

Accepts numeric or string data. Returns numeric data.

Examples

VLOOKUP(2,A1:D10,3)

Version Available

This function is available in product version 2.0 or later.

See Also

HLOOKUP | **LOOKUP** | **Lookup Functions**

WEBSERVICE

This function returns data from a web service on the Internet or Intranet.

Syntax

`WEBSERVICE(url)`

Arguments

For the argument, the web service URL.

Remarks

The #VALUE! error value is returned if the argument cannot return data or the URL contains more than 2048 characters. The #VALUE! error value is returned if the argument string is not valid or contains more than 32767 characters. The #VALUE! error value is also returned if the string contains unsupported protocols.

Data Types

Accepts string data. Returns string data.

Examples

`WEBSERVICE(A3)`

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

WEEKDAY

This function returns the number corresponding to the day of the week for a specified date.

Syntax

`WEEKDAY(date,type)`

Arguments

This function has these arguments:

Argument	Description
<i>date</i>	Date for which you want to determine the day of the week provided
<i>type</i>	[Optional] Number that represents the numbering scheme for the returned weekday value; can be any of: Value 1 or omitted 2 3
	Number returned Numbers 1 (Sunday) through 7 (Saturday) Numbers 1 (Monday) through 7 (Sunday) Numbers 0 (Monday) through 6 (Sunday)

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in DATE(2003,7,4). For more details on the date inputs, refer to the discussion in **Date and Time Functions**.

Remarks

The returned day of the week is given as an integer, ranging from 0 to 6 or 1 to 7, depending on the setting of the *type* argument.

Data Types

Accepts numeric, string, or DateTime object for both arguments. Returns numeric data.

Examples

`WEEKDAY(A2)`

`WEEKDAY(R2C1)`

`WEEKDAY(36828)` gives the result 1 equivalent to Sunday

`WEEKDAY(46,2)` gives the result 3

Version Available

This function is available in product version 1.0 or later.

See Also

DATE | DAY | MONTH | WEEKNUM | WORKDAY | Date and Time Functions

WEEKNUM

This function returns a number that indicates the week of the year numerically.

Syntax

WEEKNUM(*date*,*weektype*)

Arguments

This function has these arguments:

Argument	Description	
<i>date</i>	Date for which you want to determine the number of week	
<i>weektype</i>	Type of week determined by on which day the week starts	
	Value	Number returned
	1 (assumed if omitted)	Week starts on a Sunday
	2	Week starts on a Monday

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in DATE(2003,7,4). For more details on the date inputs, refer to the discussion in **Date and Time Functions**.

Data Types

Accepts numeric, string, DateTime object, or TimeSpan object data. Returns numeric data.

Examples

WEEKNUM(A2)

WEEKNUM(R2C1,2)

WEEKNUM(23,1) gives the result 4

Version Available

This function is available in product version 1.0 or later.

See Also

MONTH | WEEKDAY | **Date and Time Functions**

WEIBULL

This function returns the two-parameter Weibull distribution, often used in reliability analysis.

Syntax

`WEIBULL(x,alpha,beta,cumulative)`

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>x</i>	Value at which to evaluate the distribution
<i>alpha</i>	Scale parameter of the distribution, represented by alpha
<i>beta</i>	Shape parameter of the distribution, represented by beta
<i>cumulative</i>	Logical value that determines the form of the function. If <code>cumulative</code> is <code>TRUE</code> , then this function returns the cumulative distribution function; if <code>FALSE</code> , it returns the probability mass function.

Data Types

Accepts numeric data for all arguments except `cumulative`, which is logical (boolean). Returns numeric data.

Examples

`WEIBULL(3, D4, D5, FALSE)`

`WEIBULL(50, 10, 20, TRUE)`

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | **Statistical Functions**

WEIBULL.DIST

This function returns the two-parameter Weibull distribution, often used in reliability analysis.

Syntax

`WEIBULL.DIST(x,alpha,beta,cumulative)`

Arguments

This function has these arguments:

Argument	Description
-----------------	--------------------

<i>x</i>	Value at which to evaluate the distribution
<i>alpha</i>	Scale parameter of the distribution, represented by alpha
<i>beta</i>	Shape parameter of the distribution, represented by beta
<i>cumulative</i>	Logical value that determines the form of the function. If cumulative is TRUE, then this function returns the cumulative distribution function; if FALSE, it returns the probability mass function

Data Types

Accepts numeric data for all arguments except cumulative, which is logical (boolean). Returns numeric data.

Examples

`WEIBULL.DIST(3,D4,D5,FALSE)`

`WEIBULL.DIST(50,10,20,TRUE)`

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

WEIBULL

WORKDAY

This function returns the number of working days before or after the starting date.

Syntax

`WORKDAY(startdate,numdays,holidays)`

Arguments

This function has these arguments:

Argument	Description
<i>startdate</i>	Date that is the starting date; a number (as in 37806.5), or a DateTime object, as in DATE(2003,7,4)
<i>numdays</i>	Number of non-weekend or non-holiday days before or after the starting date; days in the future are positive and days in the past are negative; if not an integer, the number is truncated
<i>holidays</i>	[Optional] Range of dates to exclude from the calculation; if omitted, the calculation assumes no holidays and all weekdays are workdays

Data Types

Accepts numeric, string, or DateTime object data. Returns numeric data.

Examples

`WORKDAY(A2,A4)`

`WORKDAY(R2C1,R5C5)`

`WORKDAY(A1,A2,A5:A7)`

Version Available

This function is available in product version 2.0 or later.

See Also

DATE | NETWORKDAYS | MONTH | Date and Time Functions

WORKDAY.INTL

This function returns the serial number of the date before or after a specified number of workdays with custom weekend parameters.

Syntax

WORKDAY.INTL(*startdate*,*numdays*,*weekend*,*holidays*)

Arguments

This function has these arguments:

Argument	Description
<i>startdate</i>	Date that is the starting date; a number (as in 37806.5), or a DateTime object, as in DATE(2003,7,4)
<i>numdays</i>	Number of workdays before or after the starting date; days in the future are positive and days in the past are negative; if not an integer, the number is truncated
<i>weekend</i>	[Optional] A number or string that specifies when weekends occur. Weekend days are days of the week that are not counted as working days
<i>holidays</i>	[Optional] Range of dates to exclude from the calculation; if omitted, the calculation assumes no holidays and all weekdays are workdays

The following table lists the *weekend* number values:

Number	Day
1 or omitted	Saturday, Sunday
2	Sunday, Monday
3	Monday, Tuesday
4	Tuesday, Wednesday
5	Wednesday, Thursday
6	Thursday, Friday
7	Friday, Saturday
11	Sunday only
12	Monday only
13	Tuesday only
14	Wednesday only
15	Thursday only
16	Friday only
17	Saturday only

Remarks

Weekend string values are seven characters long and each character in the string represents a day of the week, starting with Monday. A non-workday is 1 and a workday is 0. Only characters 1 and 0 are allowed in the string. The string 1111111 always returns 0.

Weekend days and holidays are not considered to be workdays.

Data Types

Accepts numeric, string, or DateTime object data. Returns numeric data.

Examples

```
WORKDAY.INTL(A2,A4)
```

```
WORKDAY.INTL(R2C1,R5C5)
```

```
WORKDAY.INTL(A1,A2,A5:A7)
```

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

WORKDAY

XIRR

This function calculates the internal rate of return for a schedule of cash flows that may not be periodic.

Syntax

`XIRR(values,dates,guess)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>values</i>	Series of cash flows that correspond to a schedule of payments in dates. The first payment is optional and corresponds to a cost or payment that occurs at the beginning of the investment
<i>dates</i>	Schedule of payment dates that corresponds to the cash flow payments in <i>values</i>
<i>guess</i>	[Optional] Estimate of the internal rate of return that you guess is close to the result of this function; if omitted, the calculation uses 0.1 (10 percent)

Remarks

For a schedule of cash flows that is periodic, use **IRR**. Numbers in dates are truncated to integers. Both a positive and negative cash flow are required to prevent a #NUM! error. A #VALUE! error is returned if dates is invalid. If a number in dates precedes the starting date, a #NUM! error is returned. If values and dates contain a different number of values, a #NUM! error is returned. If the function can not find a result that works after 100 tries, a #NUM! error is returned.

Data Types

Accepts numeric data for *values* and *guess*, DateTime object data for *dates*. Returns numeric data.

Examples

`XIRR(B2:B6,C2:C6,0.2)`

Version Available

This function is available in product version 2.0 or later.

See Also

IRR | **XNPV** | **MIRR** | **Financial Functions**

XNPV

This function calculates the net present value for a schedule of cash flows that may not be periodic.

Syntax

`XNPV(rate,values,dates)`

Arguments

This function has these arguments:

Argument	Description
----------	-------------

<i>rate</i>	Discount rate to apply to the cash flows
<i>values</i>	Series of cash flows that correspond to a schedule of payments in dates. The first payment is optional and corresponds to a cost or payment that occurs at the beginning of the investment
<i>dates</i>	Schedule of payment dates that corresponds to the cash flow payments in <i>values</i>

Remarks

Numbers in dates are truncated to integers. A #VALUE! error is returned if any argument is nonnumeric or if any date is invalid. If a number in dates precedes the starting date, a #NUM! error is returned. If values and dates have a different number of values, a #NUM! error is returned.

Data Types

Accepts numeric data for *rate* and *values*, and DateTime object data for *dates*. Returns numeric data.

Examples

`XNPV(0.09,B2:B6,C2:C6)`

Version Available

This function is available in product version 2.0 or later.

See Also

IRR | **NPV** | **MIRR** | **XIRR** | **Financial Functions**

XOR

This function returns logical exclusive OR of specified numbers.

Syntax

`XOR(value1,value2,..)`

Arguments

For the arguments of this function, provide numeric (0 or 1) or logical values (TRUE or FALSE) up to 255 arguments.

Remarks

This function returns TRUE (or 1) when number of true inputs is odd; otherwise, returns FALSE (or 0) when number of true inputs is even.

Data Types

Accepts logical data (Boolean values of TRUE or FALSE) or numerical values (0 or 1). Returns logical data (Boolean values of TRUE or FALSE).

Examples

`XOR(3>0,2<9)` gives the result TRUE

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

YEAR

This function returns the year as an integer for a specified date.

Syntax

`YEAR(date)`

Arguments

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in DATE(2003,7,4). For more details on the date inputs, refer to the discussion in **Date and Time Functions**.

Remarks

The Spread control correctly treats the year 1900 as a non-leap year and uses a base date of 12/31/1899.

Data Types

Accepts numeric, string, DateTime object, or TimeSpan object data. Returns numeric data.

Examples

`YEAR(A2)`

`YEAR(R2C1)`

`YEAR(0.007)` gives the result (which may be different from Excel) 1899

`YEAR(DATE(2004,8,9))` gives the result 2004

`YEAR(38208)` gives the result 2004

`YEAR("8/9/2004")` gives the result 2004

Version Available

This function is available in product version 1.0 or later.

See Also

DATE | MONTH | TODAY | YEARFRAC | Date and Time Functions

YEARFRAC

This function returns the fraction of the year represented by the number of whole days between the start and end dates.

Syntax

`YEARFRAC(startdate,enddate,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>startdate</i>	Starting date (DateTime object)
<i>enddate</i>	Ending date (DateTime object)
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This functions returns an error when start, end, or basis is invalid.

Data Types

Accepts numeric, string, DateTime object data for the date arguments and numeric data for the optional argument. Returns numeric data.

Examples

`YEARFRAC(A1, A2, A3)`

Version Available

This function is available in product version 2.0 or later.

See Also

DATE | **MONTH** | **TODAY** | **YEAR** | **Date and Time Functions**

YIELD

This function calculates the yield on a security that pays periodic interest.

Syntax

`YIELD(settle,maturity,rate,price,redeem,frequency,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>rate</i>	Annual coupon rate
<i>price</i>	Price per \$100 face value for the security
<i>redeem</i>	Redemption value per \$100 face value
<i>frequency</i>	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when settle or maturity is invalid. A #NUM! error is returned if frequency is a number other than 1, 2, or 4. If rate is less than 0, a #NUM! error is returned. If price or redeem is less than or equal to 0, a #NUM! error is returned. If basis is less than 0 or greater than 4, a #NUM! error is returned. If settle is greater than or equal to maturity, a #NUM! error is returned. Settle, maturity, frequency, and basis are truncated to integers.

Data Types

Accepts numeric data and dates. Returns numeric data.

Examples

`YIELD(A1, A2, A3, A4, A5, A6, A7)`

Version Available

This function is available in product version 2.0 or later.

See Also

YIELDDISC | YIELDMAT | ODDFYIELD | Financial Functions

YIELDDISC

This function calculates the annual yield for a discounted security.

Syntax

YIELDDISC(*settle*,*maturity*,*price*,*redeem*,*basis*)

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>price</i>	Price per \$100 face value for the security
<i>redeem</i>	Redemption value per \$100 face value
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when *settle* or *maturity* is invalid. If *price* or *redeem* is less than or equal to 0, a #NUM! error is returned. If *basis* is less than 0 or greater than 4, a #NUM! error is returned. If *settle* is greater than or equal to *maturity*, a #NUM! error is returned. *Settle*, *maturity*, and *basis* are truncated to integers.

Data Types

Accepts numeric data and dates. Returns numeric data.

Examples

YIELDDISC (B1, B2, B3, B4, B5)

Version Available

This function is available in product version 2.0 or later.

See Also

YIELD | **YIELDMAT** | **ODDLYIELD** | **Financial Functions**

YIELDMAT

This function calculates the annual yield of a security that pays interest at maturity.

Syntax

`YIELDMAT(settle,maturity,issue,issrate,price,basis)`

Arguments

This function has these arguments:

Argument	Description
<i>settle</i>	Settlement date for the security
<i>maturity</i>	Maturity date for the security
<i>issue</i>	Issue date for the security
<i>issrate</i>	Interest rate for the security at the date of issue
<i>price</i>	Price per \$100 face value for the security
<i>basis</i>	[Optional] Integer representing the basis for day count (Refer to Day Count Basis .)

Remarks

This function returns a #VALUE! error when settle, maturity, or issue is invalid. If issrate is less than 0 or price is less than or equal to 0, a #NUM! error is returned. If basis is less than 0 or greater than 4, a #NUM! error is returned. If settle is greater than or equal to maturity, a #NUM! error is returned. Settle, maturity, issue, and basis are truncated to integers.

Data Types

Accepts numeric and date data. Returns numeric data.

Examples

`YIELDMAT(C1,C2,C3,C4,C5,C6)`

Version Available

This function is available in product version 2.0 or later.

See Also

YIELD | **YIELDDISC** | **PRICEMAT** | **Financial Functions**

Z.TEST

This function returns the significance value of a z-test. The z-test generates a standard score for x with respect to the set of data and returns the two-tailed probability for the normal distribution.

Syntax

Z.TEST(*array,x,sigma*)

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Array of data to test
<i>x</i>	Value at which to test
<i>sigma</i>	[Optional] Known standard deviation for the population; if omitted, the calculation uses the sample standard deviation

Remarks

If sigma is not specified, the calculated standard deviation of the data in array is used.

The equation for calculating the z-test is as follows, where *n* is the number of data points.

$$ZTEST(array, x, \sigma) = 1 - NORMSDIST\left(\frac{\mu - x}{\sigma \div n}\right)$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

Z.TEST(A2:D12,40,0.877)

Z.TEST(R2C1:R12C4,2)

Z.TEST({5,10,15,12,11,8,16,7},10) gives the result 0.355512703503418

Z.TEST({5,10,15,12,11,8,16,7},10,3) gives the result 0.31867594409823696

Version Available

This function is available in Spread for Windows Forms 11.0 or later.

See Also

ZTEST | TTEST | Statistical Functions

ZTEST

This function returns the significance value of a z-test. The z-test generates a standard score for x with respect to the set of data and returns the two-tailed probability for the normal distribution.

Syntax

`ZTEST(array,x,sigma)`

Arguments

This function has these arguments:

Argument	Description
<i>array</i>	Array of data to test
<i>x</i>	Value at which to test
<i>sigma</i>	[Optional] Known standard deviation for the population; if omitted, the calculation uses the sample standard deviation

Remarks

If sigma is not specified, the calculated standard deviation of the data in array is used.

The equation for calculating the z-test is as follows, where *n* is the number of data points.

$$ZTEST(array, x, \sigma) = 1 - NORMSDIST\left(\frac{\mu - x}{\sigma \div n}\right)$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`ZTEST(A2:D12,40,0.877)`

`ZTEST(R2C1:R12C4,2)`

`ZTEST({5,10,15,12,11,8,16,7},10)` gives the result 0.355512703503418

`ZTEST({5,10,15,12,11,8,16,7},10,3)` gives the result 0.318675944098237

Version Available

This function is available in product version 1.0 or later.

See Also

FTEST | **TTEST** | **Statistical Functions**

2 Index

A1 (Letter-Number) Notation, 26

ABS, 66

ACCRINT, 67

ACCRINTM, 68

ACOS, 69

ACOSH, 70

ACOT, 71

ACOTH, 72

adding values, 501

ADDRESS, 73

AGGREGATE, 74-75

AMORDEGRC, 76-77

AMORLINC, 78

AND, 79

ARABIC, 80

AREAS, 81

arguments, 51

Array Formulas, 54

Arrays in a Formula, 55

ASC, 82

ASIN, 83

ASINH, 84

ATAN, 85

ATAN2, 86

ATANH, 87

AVEDEV, 88

AVERAGE, 89

AVERAGEA, 90

AVERAGEIF, 91

AVERAGEIF function, 91

AVERAGEIFS, 92

AVERAGEIFS function, 92

BAHTTEXT, 93

BASE, 94

BESSELI, 95

BESSELJ, 96

BESSELK, 97

BESSELY, 98

BETA.DIST, 99

BETA.INV, 100

BETADIST, 101

- BETAINV, 102
- BIN2DEC, 103
- BIN2HEX, 104
- BIN2OCT, 105
- BINOM.DIST, 106-107
- BINOM.DIST.RANGE, 108
- BINOM.INV, 109
- BINOMDIST, 110-111
- BITAND, 112
- BITLSHIFT, 113
- BITOR, 114
- BITRSHIFT, 115
- BITXOR, 116
- CALL, 117
- Categories of Functions, 36
- CEILING, 118
- CEILING.MATH, 119
- CEILING.PRECISE, 120
- CELL, 121-122
- Cell References in a Formula, 25
- CHAR, 123
- CHIDIST, 124
- CHIINV, 125
- CHISQ.DIST, 126
- CHISQ.DIST.RT, 127
- CHISQ.INV, 128
- CHISQ.INV.RT, 129
- CHISQ.TEST, 130
- CHITEST, 131
- CHOOSE, 132
- CLEAN, 133
- CODE, 134
- COLUMN, 135
- COLUMNS, 136
- COMBIN, 137
- COMBINA, 138
- COMPLEX, 139
- Complex Numbers in Engineering Functions, 40
- CONCAT, 140
- CONCATENATE, 141
- CONFIDENCE, 142
- CONFIDENCE.NORM, 143
- CONFIDENCE.T, 144

- Contacting Us, 20**
- CONVERT, 145-147**
- CORREL, 148**
- COS, 149**
- COSH, 150**
- COT, 151**
- COTH, 152**
- COUNT, 153**
- COUNTA, 154**
- COUNTBLANK, 155**
- COUNTIF, 156**
- COUNTIFS, 157**
- COUNTIFS function, 157**
- COUPDAYBS, 158**
- COUPDAYS, 159**
- COUPDAYSNC, 160**
- COUPNCD, 161**
- COUPNUM, 162**
- COUPPCD, 163**
- COVAR, 164**
- COVARIANCE.P, 165**
- COVARIANCE.S, 166**
- CRITBINOM, 167**
- CSC, 168**
- CSCH, 169**
- Custom Functions, 58**
- Custom Functions in Formulas, 58**
- Custom Names in Formulas, 59**
- Data Types Using Formulas, 57**
- database, 37**
- Database Functions, 37**
- date, 38, 171**
- Date and Time Functions, 38**
- DATEDIF, 172**
- DATEVALUE, 173**
- DAVERAGE, 174**
- DAY, 175**
- Day Count Basis, 42**
- DAYS, 176**
- DAYS360, 177-178**
- DB, 179-180**
- DBCS, 181**
- DCOUNT, 182**

- DCOUNTA, 183**
- DDB, 184**
- DEC2BIN, 185**
- DEC2HEX, 186**
- DEC2OCT, 187**
- DECIMAL, 188**
- DEGREES, 189**
- DELTA, 190**
- DEVSQ, 191**
- DGET, 192**
- DISC, 193**
- DMAX, 194**
- DMIN, 195**
- DOLLAR, 196**
- DOLLARDE, 197**
- DOLLARFR, 198**
- DPRODUCT, 199**
- DSTDEV, 200**
- DSTDEVP, 201**
- DSUM, 202**
- DURATION, 203**
- DVAR, 204**
- DVARP, 205**
- Dynamic Array Formulas, 56**
- EDATE, 206**
- EFFECT, 207**
- ENCODEURL, 208**
- engineering, 39**
- Engineering Functions, 39**
- EOMONTH, 209**
- ERF, 210-211**
- ERF.PRECISE, 212**
- ERFC, 213**
- ERFC.PRECISE, 214**
- ERROR.TYPE, 215**
- ERRORTYPE, 216**
- EUROCONVERT, 217-218**
- EVEN, 219**
- EXACT, 220**
- EXP, 221**
- EXPON.DIST, 222-223**
- EXPONDIST, 224-225**
- F.DIST, 226**

- F.DIST.RT, 227**
- F.INV, 228**
- F.INV.RT, 229**
- F.TEST, 230**
- FACT, 231**
- FACTDOUBLE, 232**
- FALSE, 233**
- FDIST, 234**
- FILTER, 235-236**
- FILTERXML, 237**
- financial, 41**
- Financial Functions, 41**
- FIND, 238**
- FINDB, 239**
- FINV, 240**
- FISHER, 241**
- FISHERINV, 242**
- FIXED, 243**
- FLOOR, 244**
- FLOOR.MATH, 245**
- FLOOR.PRECISE, 246**
- FORECAST, 247**
- FORECAST.LINEAR, 248**
- Formula Functions, 61-64**
- Formula Overview, 22**
- Formula Reference, 1**
- formulas**
 - array, 54
- FORMULATEXT, 249**
- FREQUENCY, 250**
- FTEST, 251**
- Functions A to C, 65**
- Functions D to G, 170**
- Functions H to L, 266**
- Functions in a Formula, 35**
- Functions M to Q, 349**
- Functions R to S, 437**
- Functions T to Z, 509**
- FV, 252**
- FVSCCHEDULE, 253**
- GAMMA, 254**
- GAMMA.DIST, 255**
- GAMMA.INV, 256**

GAMMADIST, 257
GAMMAINV, 258
GAMMALN, 259
GAMMALN.PRECISE, 260
GAUSS, 261
GCD, 262
GEOMEAN, 263
GESTEP, 264
Getting Technical Support, 21
GROWTH, 265
HARMEAN, 267
HEX2BIN, 268
HEX2DEC, 269
HEX2OCT, 270
HLOOKUP, 271
HOUR, 272
HYPERLINK, 273
HYPGEOM.DIST, 274
HYPGEOMDIST, 275
IF, 276
IFERROR, 277
IFERROR function, 277
IFNA, 278
IFS, 279
IMABS, 280
IMAGINARY, 281
IMARGUMENT, 282
IMCONJUGATE, 283
IMCOS, 284
IMCOSH, 285
IMCOT, 286
IMCSC, 287
IMCSCH, 288
IMDIV, 289
IMEXP, 290
IMLN, 291
IMLOG10, 292
IMLOG2, 293
IMPOWER, 294
IMPRODUCT, 295
IMREAL, 296
IMSEC, 297
IMSECH, 298

- IMSIN, 299**
- IMSINH, 300**
- IMSQRT, 301**
- IMSUB, 302**
- IMSUM, 303**
- IMTAN, 304**
- INDEX, 305**
- INDIRECT, 306**
- INFO, 307**
- information, 43**
- Information Functions, 43**
- INT, 308**
- INTERCEPT, 309**
- INTRATE, 310**
- IPMT, 311**
- IRR, 312-313**
- ISBLANK, 314**
- ISERR, 315**
- ISERROR, 316**
- ISEVEN, 317**
- ISFORMULA, 318**
- ISLOGICAL, 319**
- ISNA, 320**
- ISNONTEXT, 321**
- ISNUMBER, 322**
- ISO.CEILING, 323**
- ISODD, 324**
- ISOWEEKNUM, 325**
- ISPMT, 326**
- ISREF, 327**
- ISTEXT, 328**
- JIS, 329**
- KURT, 330**
- LARGE, 331**
- LCM, 332**
- LEFT, 333**
- LEFTB, 334**
- LEN, 335**
- LENB, 336**
- LINEST, 337**
- LN, 338**
- LOG, 339**
- LOG10, 340**

- LOGEST, 341**
- logical, 44**
- Logical Functions, 44**
- LOGINV, 342**
- LOGNORM.DIST, 343**
- LOGNORM.INV, 344**
- LOGNORMDIST, 345**
- lookup, 45 , 346-347 , 350**
- Lookup Functions, 45**
- LOWER, 348**
- MATCH, 350**
- MATCH function, 350**
- math, 46**
- Math and Trigonometry Functions, 46**
- MAX, 351**
- MAXA, 352**
- MAXIFS, 353**
- MDETERM, 354**
- MDURATION, 355**
- MEDIAN, 356**
- MID, 357**
- MIDB, 358**
- MIN, 359**
- MINA, 360**
- MINIFS, 361**
- MINUTE, 362**
- MINVERSE, 363**
- MIRR, 364**
- Missing Arguments, 52**
- MMULT, 365**
- MOD, 366**
- MODE, 367**
- MODE.MULT, 368**
- MODE.SNGL, 369**
- MONTH, 370**
- MROUND, 371**
- MULTINOMIAL, 372**
- MUNIT, 373**
- N, 374**
- NA, 375**
- NEGBINOM.DIST, 376**
- NEGBINOMDIST, 377**
- NETWORKDAYS, 378**

- NETWORKDAYS.INTL, 379-380**
- NOMINAL, 381**
- NORM.DIST, 382**
- NORM.INV, 383**
- NORM.S.DIST, 384**
- NORM.S.INV, 385**
- NORMDIST, 386**
- NORMINV, 387**
- NORMSDIST, 388**
- NORMSINV, 389**
- NOT, 390**
- NOW, 391**
- NPER, 392**
- NPV, 393-394**
- NUMBERVALUE, 395**
- OCT2BIN, 396**
- OCT2DEC, 397**
- OCT2HEX, 398**
- ODD, 399**
- ODDFPRICE, 400**
- ODDFYIELD, 401**
- ODDLPRICE, 402**
- ODDLYIELD, 403**
- OFFSET, 404**
- operators, 32**
- Operators in a Formula, 32**
- Optional Arguments, 51**
- OR, 405**
- Order of Precedence, 33**
- PDURATION, 406**
- PEARSON, 407**
- PERCENTILE, 408**
- PERCENTILE.EXC, 409**
- PERCENTILE.EXE, 409**
- PERCENTILE.INC, 410**
- PERCENTRANK, 411**
- PERCENTRANK.EXC, 412**
- PERCENTRANK.INC, 413**
- PERMUT, 414**
- PERMUTATIONA, 415**
- PHI, 416**
- PHONETIC, 417**
- PI, 418**

PMT, 419
POISSON, 420-421
POISSON.DIST, 422-423
POWER, 424
PPMT, 425
PRICE, 426
PRICEDISC, 427
PRICEMAT, 428
PROB, 429
PRODUCT, 430
PROPER, 431
PV, 432
QUARTILE, 433
QUARTILE.EXC, 434
QUARTILE.INC, 435
QUOTIENT, 436
R1C1 (Number-Number) Notation, 27
RADIANS, 438
RAND, 439
RANDARRAY, 440-441
RANDBETWEEN, 442
RANK, 443
RANK.AVG, 444
RANK.EQ, 445
RATE, 446
RECEIVED, 447
references, 29 , 30-31
Relative and Absolute, 28
REPLACE, 448
REPLACEB, 449
REPT, 450
Resultant Error Values, 60
RIGHT, 451
RIGHTB, 452
ROMAN, 453
ROUND, 454
ROUNDDOWN, 455
ROUNDUP, 456
ROW, 457
ROWS, 458
RRI, 459
RSQ, 460
RTD, 461

- Sample Formula, 24**
- Scope of Cell References, 29**
- SEARCH, 462**
- SEARCHB, 463**
- SEC, 464**
- SECH, 465**
- SECOND, 466**
- SECOND function, 466**
- SEQUENCE, 468**
- SERIESSUM, 467**
- SHEET, 469**
- Sheet References in a Formula, 30-31**
- SHEETS, 470**
- SIGN, 471**
- SIN, 472**
- SINGLE, 474**
- SINH, 473**
- SKEW, 475**
- SKEW.P, 476**
- SLN, 477**
- SLOPE, 478**
- SMALL, 479**
- SORT, 480-482**
- SORTBY, 483-484**
- SQRT, 485**
- SQRTPI, 486**
- STANDARDIZE, 487**
- statistical, 47-48**
- Statistical Functions, 47-48**
- STDEV, 488**
- STDEV.P, 489**
- STDEV.S, 490**
- STDEVA, 491**
- STDEV.P, 492**
- STDEVPA, 493**
- STEYX, 494**
- SUBSTITUE, 495**
- SUBSTITUTE, 495**
- SUBTOTAL, 496-497**
- SUM, 498-499**
- SUMIF, 500**
- SUMIFS, 501**
- SUMIFS function, 501**

SUMPRODUCT, 502
SUMSQ, 503
SUMX2MY2, 504
SUMX2PY2, 505
SUMXMY2, 506
support, 21
SWITCH, 507
SYD, 508
T, 510
T.DIST, 511
T.DIST.2T, 512
T.DIST.RT, 513
T.INV, 514
T.INV.2T, 515
T.TEST, 516
TAN, 517
TANH, 518
TBILLEQ, 519
TBILLPRICE, 520
TBILLYIELD, 521
TDIST, 522
text, 49 , 523
TEXT function, 523
Text Functions, 49
TEXTJOIN, 524
time, 38 , 525
TIMEVALUE, 526
TINV, 527
TODAY, 528
TRANSPOSE, 529
TREND, 530
trigonometry, 46
TRIM, 531
TRIMMEAN, 532
TRUE, 533
TRUNC, 534
TTEST, 535
TYPE, 536
Types of Functions, 36
UNICHAR, 537
UNICODE, 538
UNIQUE, 539-540
UPPER, 541

USDOLLAR, 542

Using Operators with Dates and Times, 34

VALUE, 543

VAR, 544-545

VAR.P, 546

VAR.S, 547

VARA, 548-549

VARP, 550

VARPA, 551-552

VDB, 553

VLOOKUP, 554

volatile, 53

Volatile Functions, 53

Web Functions, 50

WEBSERVICE, 555

WEEKDAY, 556

WEEKNUM, 557

WEIBULL, 558

WEIBULL.DIST, 559

What is a Formula?, 23

WORKDAY, 560

WORKDAY.INTL, 561-562

XIRR, 563

XNPV, 564

XOR, 565

YEAR, 566

YEARFRAC, 567

YIELD, 568

YIELDDISC, 569

YIELDMAT, 570

Z.TEST, 571

ZTEST, 572