Function Reference

Formula One™ for Java®

Powerful spreadsheet application for end users.

Robust development tool for Java developers and Webmasters.

Version 7.0

Tidestone Technologies, Inc.

This program is not fault-tolerant and is not designed, manufactured or intended for use or resale in the on-line control of nuclear facilities, aircraft navigation or communication system, air traffic control, direct life support machines or weapons systems in which the failure of the Program could lead directly to death, personal injury or severe physical or environmental damage. Specifications subject to change without notice.

Published by
Tidestone Technologies, Inc.
12980 Metcalf Avenue, Suite 300
Overland Park, Kansas 66213
phone 913-851-2200
toll-free 1-800-884-8665
fax 913-851-1390

www.tidestone.com
www.f1j.com

Printed in the United States of America
99/700.8

Contents

Tidestone

PREFACE

Overview

Computer spreadsheets — tools that help us understand, model, and manipulate relationships between sets of numerical data — further empower us by providing the ability to manipulate, extrapolate, interpret, graph, and display numeric data with the speed and convenience of the computer. Formula One for Java unites all of the powerful utility and familiar interface of the computer spreadsheet with the speed, power, and the universal presence of the Internet.

Designed for use in Java development environments, Formula One for Java provides the tools you need to design, create, and distribute custom spreadsheets over the Internet as part of your application, applet, or JavaBean. Using Formula One for Java, you may also embed high-quality spreadsheets in a Web page with just a few lines of HTML code.

Formula One for Java also works as a standalone spreadsheet application that operates with the speed and functionality of the best desktop spreadsheet.

About The Function Reference

This Guide, the *Formula One for Java Function Reference*, provides:

- a general introduction to using functions in spreadsheets
- lists and short descriptions of functions categorized by types
- a complete reference record for each function.

The complete documentation for Formula One for Java consists of three separate publications:

1. *Formula One for Java User's Guide*

   A basic guide to using electronic spreadsheets in general and using Formula One for Java on a desktop or over a network as an end user.

2. *Formula One for Java Technical Guide*

   The Technical Guide includes more advanced concepts and uses.

3. *Formula One for Java Function Reference*

Using This Manual

For the most part, this manual assumes you are using a Windows operating environment. Keyboard keys and some graphical user interface elements might vary depending upon your operating environment.

This document along with other useful information about Formula One for Java is also available online at www.tidestone.com.

Technical Support

The Tidestone technical support staff can help you with any problem you encounter installing or using Formula One for Java.

For specific information about support plans for Formula One for Java, visit www.tidestone.com and follow the Support links.

Documentation Conventions

Throughout this documentation, typographic conventions are used to define elements and references to Formula One for Java items.

Recognizing these conventions will help you understand and use the documentation.

Convention example	Description
➤ **To enter a function:**	A series of numbered instructions is preceded by an introductory line. The introductory line begins with an arrowhead.
1.Type `=SUM(A4:B6)`	Numbered instructions provide step-by-step directions for performing tasks. Perform the instructions in the order they are presented. In numbered steps, items you are to enter are shown in Letter Gothic font.
workbook	In general sections, italic text is used for the first occurrence of a new term.
fontname	In reference sections, italic text indicates variable or argument information you must supply.
`=(3+4)*5`	Letter Gothic font is used for examples of cell entries.
F1J7Swing.jar	File names are presented in bold type.
Format > Sheet > Properties	Choose the Properties option on the Sheet submenu of the Format menu.
CTRL + L	Type L while holding down the CTRL key.

New Functions in Version 7.0

The following functions have been added in the 7.0 version of Formula One for Java.

ACCRINT
ACCRINTM
AMORDEGRC
AMORLINC
AREAS
AVEDEV
AVERAGEA
BESSELI
BESSELJ
BESSELK
BESSELY
BETADIST
BETAINV
BIN2DEC
BIN2HEX
BIN2OCT
BINOMDIST
CALL
CELL
CHIDIST
CHIINV
CHITEST
COMBIN
COMPLEX
CONFIDENCE
CONVERT
CORREL
COUNTBLANK
COUPDAYBS
COUPDAYS
COUPDAYSNC
COUPNCD
COUPNUM
COUPPCD
COVAR
CRITBINOM
CUMIPMT
CUMPRINC
DAVERAGE
DCOUNT
DCOUNTA
DEC2BIN
DEC2HEX
DEC2OCT
DEGREES
DELTA
DEVSQ
DGET
DISC
DMAX
DMIN
DOLLARDE
DOLLARFR
DPRODUCT
DSTDEV
DSTDEVP
DSUM
DURATION
DVAR
DVARP
EDATE
EFFECT
EOMONTH
ERF
ERFC
EXPONDIST
FACTDOUBLE
FDIST
FINV
FISHER
FISHERINV
FORECAST
FREQUENCY
FTEST
FVSCHEDULE
GAMMADIST
GAMMAINV
GAMMALN
GCD
GEOMEAN
GESTEP
GETPIVOTDATA
GROWTH
HARMEAN
HEX2BIN
HEX2DEC
HEX2OCT
HYPERLINK
HYPGEOMDIST
IMABS
IMAGINARY
IMARGUMENT
IMCONJUGATE
IMCOS
IMDIV
IMEXP
IMLN
IMLOG10
IMLOG2
IMPOWER
IMPRODUCT
IMREAL
IMSIN
IMSQRT
IMSUB
IMSUM
INFO
INTERCEPT
INTRATE
ISEVEN
ISODD
KURT
LARGE
LCM
LINEST
LOGEST
LOGINV
LOGNORMDIST
MAXA
MDETERM
MDURATION
MEDIAN
MINA
MINVERSE
MMULT
MODE
MROUND
MULTINOMIAL
NEGBINOMDIST
NETWORKDAYS
NOMINAL
NORMDIST
NORMINV
NORMSDIST
NORMSINV
OCT2BIN
OCT2DEC
OCT2HEX
ODDFPRICE
ODDFYIELD

ODDLPRICE
ODDLYIELD
PEARSON
PERCENTILE
PERCENTRANK
PERMUT
POISSON
POWER
PRICE
PRICEDISC
PRICEMAT
PROB
QUARTILE
QUOTIENT
RADIANS
RANDBETWEEN
RANK
RECEIVED
REGISTER.ID
ROMAN
RSQ
SERIESSUM
SKEW
SLOPE
SMALL
SQLREQUEST
SQRTPI
STANDARDIZE
STDEVA
STDEVPA
STEYX
SUBTOTAL
SUMPRODUCT
SUMX2MY2
SUMX2PY2
SUMXMY2
TBILLEQ
TBILLPRICE
TBILLYIELD
TDIST
TINV
TRANSPOSE
TREND
TRIMMEAN
TTEST
VARA
VARPA
WEEKNUM
WEIBULL
WORKDAY
XIRR
XNPV
YEARFRAC
YIELD
YIELDDISC
YIELDMAT
ZTEST

CHAPTER 1

Introduction to Functions

Formula One for Java’s functions allow you to use complex equations within spreadsheets without having to enter the equations themselves. This chapter provides an overview of how to enter and use functions. The following topics are covered.

What is a Function?

A function is a string containing numbers, operators, cell references, and names. Functions calculate and evaluate data and return the result to the cell in which they are entered. Functions are a special kind of formula and follow the rules for entering and working with formulas.

Each function performs a specific calculation. For example, the SQRT function calculates the square root of a number. The following example calculates the square root of 118:

```
=SQRT(118)
```

Function Basics

Functions are composed of three elements: an *equal sign*, a *keyword,* and *arguments*.

- The equal sign tells Formula One for Java that this cell contains a formula or function.
- The keyword identifies the function so that the worksheet knows what type of calculation or evaluation to perform. Each function keyword is unique.
- Arguments provide the data for the function to calculate or evaluate. The arguments for a function immediately follow the function keyword and are enclosed in parentheses. Separate multiple arguments with commas.

Every function requires an equal sign and a keyword, but not all functions require arguments.

Functions can contain as many as 1024 characters.

➤ **To enter a function:**

1. Select the cell where you want to display the function’s result.
2. Type the equals sign.
3. Type the function keyword in lowercase or uppercase characters.

   After the function is entered, the worksheet records the function keyword in uppercase characters, regardless of how it was entered.
4. Type any arguments required by the function, enclosed in parentheses and separated by commas.

   Functions that do not require arguments still require a set of empty parentheses () after the function keyword.

   Some functions contain optional arguments. If you omit an optional argument, a default value is assumed for the argument.
5. Press ENTER.

The result of the function calculation will appear in the cell.

If Formula One for Java finds a syntax error in the function you entered, it will display an error message. You must correct the error or delete the function before proceeding. For a list of syntax errors, see “Understanding Errors” on page 19.

Function Arguments

A function is a special kind of formula, so a function’s arguments must follow the rules for entering formulas. The arguments for a function can be:

- Numbers
- Text strings
- Logical values
- Error values
- Cell or range references
- Array constants

A function may require its first argument to be a number, the second argument a text string, and the third argument a cell reference. Some arguments can be of more than one type; for example, the SUM function’s argument can be a set of numbers, range references, or array constants. Refer to the documentation on the individual function to determine the type of data required for the function you are entering.

Numbers in Arguments

Usually you can enter a number in an argument by simply typing the number. Do not use commas to separate thousands, because the comma will be interpreted as separating two arguments.

Negative numbers are entered using the negative sign (-). Do not use parentheses: the function will interpret the value as a different argument.

Fractions can be entered by using a slash for the fraction separator (`3/4` is evaluated as .75).

Percentages can be entered using the percent sign (`34%` is evaluated as .34).

Scientific notation can be used for very large or very small numbers (`34E+09` is evaluated as 34,000,000,000).

Exponents can be entered using the caret sign (`3^4` is evaluated as three to the fourth power, or 81).

Dates and times may be entered in two ways: as serial numbers, or in one of the conventional formats (e.g., mm/dd/yy) enclosed in quotation marks. Dates and times entered in this way are considered text, but Formula One for Java recognizes them as dates and internally converts them to their serial number values. For example, `"10/10/94"-"10/1/94"` is interpreted as 9.

Text Strings in Arguments

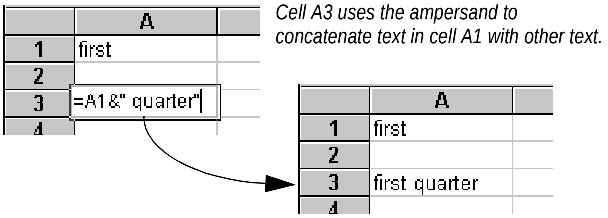
Text strings in an argument list must be enclosed in quotation marks. Without the quotation marks, the function will return the #NAME? error. When entering multiple text arguments, be sure the closing quotation mark comes *before* the comma that separates two arguments.

When the argument is a cell reference pointing to a cell containing a text string, the text in the cell does *not* have to be enclosed in quotes.

If a number is encountered when text is expected, the number is converted to text. `"The number is "&3` is interpreted as The number is 3. If text is encountered when a number is expected, the text is converted to a number (`1 + "3"` is interpreted as 4). If the text cannot be converted to a valid number (e.g., `1 + "Text"`), the function will return the #VALUE! error.

Concatenation

You may use the ampersand (&) character to concatenate text strings when a text argument from two different sources is required. For example, in the spreadsheet below, cell A3 uses concatenation to create the text `first quarter`.



Cell A3 uses the ampersand to concatenate text in cell A1 with other text.

	A
1	first
2	
3	=A1&" quarter"
4	

	A
1	first
2	
3	first quarter
4	

Logical Values in Arguments

The logical value `TRUE` converts to 1, while `FALSE` converts to 0.

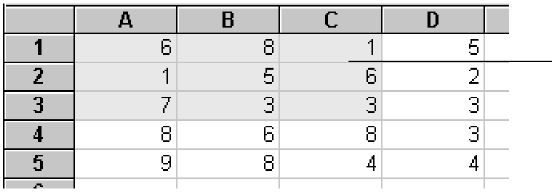
If a number is encountered when a logical value is expected, `0` is evaluated as FALSE and all other numbers are evaluated as TRUE. If text is encountered when a logical value is expected, `"TRUE"` is evaluated as TRUE, `"FALSE"` is evaluated as FALSE, and all other text returns the #VALUE! error.

Cell References in Arguments

For most arguments, you can substitute a cell or range reference for the data required by an argument. A reference is a cell's address. It identifies a cell or range of cells by referring to the column letter and row number of the cell(s). For example, `A1` refers to the cell at the intersection of column A and row 1.

The reference tells Formula One for Java to use the contents of the referenced cell(s) as the function's argument. For example, if an argument requires a number, you can substitute a reference to a cell that contains a number. The number in the referenced cell is used in the calculation of the function. The referenced cell must contain the appropriate data for the argument that uses it.

You specify a range of cells by placing a colon (:) between two cell references. For example, `A1:C3` refers to the range anchored by cells A1 and C3. The range includes all cells in columns A, B, and C of rows 1, 2, and 3.



	A	B	C	D
1	6	8	1	5
2	1	5	6	2
3	7	3	3	3
4	8	6	8	3
5	9	8	4	4

The range reference `A1:C3` *refers to all the grayed cells in this worksheet.*

Some functions may take more than one cell as an argument. For example, when the AVERAGE function has a range reference as its argument, the function averages the data in all the cells in the range.

Empty Cells

Most functions ignore any empty cells found in a range referenced in an argument. However, cells that appear empty and are not may affect the results of the function. For example, cells containing empty text or text consisting only of spaces may be treated as text. Cells that are formatted not to display zero values may contain hidden zeroes that will be treated as numeric values. If your function displays unexpected results, check for empty cells and cells that appear empty.

Entering Cell References

You can enter cell references in arguments in three ways:

- Type in the cell or range address.
- Type in the name of a named cell or range. For information on defining names, see “Using Names” on page 9.
- Use the mouse to click and drag on cells and ranges. Formula One for Java automatically enters a relative reference identifying the cell(s) you select.

Absolute and Relative References

There are two types of cell references: relative and absolute.

- **Relative references** point to a cell based on its position relative to the current cell. When the cell containing the reference is moved or copied, the reference is adjusted to point to a new cell with the same relative offset as the originally referenced cell.

  For example, suppose the function `SUM(B1:B3)` is located in cell A1. When you copy the function and paste it down two rows into cell A3, the function will be adjusted down two rows, to `SUM(B3:B5)`.

  Relative references will be adjusted whenever you cut or copy and paste a function or when you use the Edit > Fill commands to fill a range with a copy of a function.

- **Absolute references** point to a cell at an exact location. When a cell containing a formula with absolute references is moved or copied, the reference does not change. Absolute references have a dollar sign ($) in front of the row number and/or column letter.

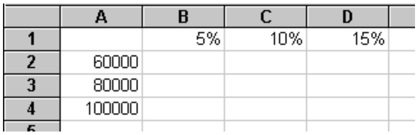
References that are part absolute and part relative are called mixed references. The following table lists the reference types.

Reference	Type
A1	Relative reference pointing to cell A1.
$A$1	Absolute reference pointing to cell A1.
$A1	Absolute column reference, relative row reference pointing to cell A1.
A$1	Relative column reference, absolute row reference pointing to cell A1.

Using Relative and Absolute References

You can copy and paste absolute, relative, and mixed references to create worksheets that are easy to update and that are smaller than worksheets where each formula is created separately.

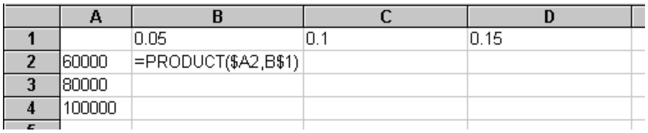
For example, in the following worksheet, the values in column A need to be multiplied by the percentages in row 1.



	A	B	C	D
1		5%	10%	15%
2	60000			
3	80000			
4	100000			

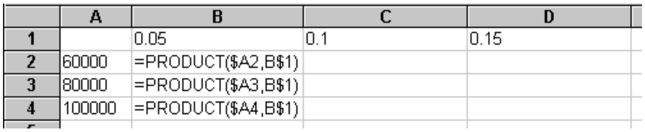
To do these calculations, you could enter the function `PRODUCT(A2,B1)` in cell B2, `PRODUCT(A3,B1)` in cell B3, `PRODUCT(A4,B1)` in cell B4, etc. Besides the fact that this would be a lot of typing, this solution would require Formula One for Java to keep nine separate formulas in memory.

A better way to do it would be to enter the function `PRODUCT($A2,B$1)` in cell B2, and use the worksheet's Edit > Fill command to fill cells B2 through D4 with copies of that function. When the function is copied in this manner, its relative references change, but the absolute references stay the same.



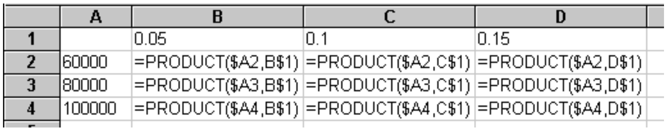
	A	B	C	D
1		0.05	0.1	0.15
2	60000	=PRODUCT($A2,B$1)		
3	80000			
4	100000			

Enter the function...



	A	B	C	D
1		0.05	0.1	0.15
2	60000	=PRODUCT($A2,B$1)		
3	80000	=PRODUCT($A3,B$1)		
4	100000	=PRODUCT($A4,B$1)		

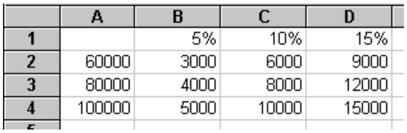
Edit > Fill > Down to fill column B...



	A	B	C	D
1		0.05	0.1	0.15
2	60000	=PRODUCT($A2,B$1)	=PRODUCT($A2,C$1)	=PRODUCT($A2,D$1)
3	80000	=PRODUCT($A3,B$1)	=PRODUCT($A3,C$1)	=PRODUCT($A3,D$1)
4	100000	=PRODUCT($A4,B$1)	=PRODUCT($A4,C$1)	=PRODUCT($A4,D$1)

Edit > Fill > Right to fill columns C and D.

The resulting worksheet calculates all the figures using multiple copies of that one function. Only one function must be kept in memory.



	A	B	C	D
1		5%	10%	15%
2	60000	3000	6000	9000
3	80000	4000	8000	12000
4	100000	5000	10000	15000

If you change the percentages in row 1 or the figures in column A, the calculations in the worksheet will automatically change because of the absolute references to those cells. This makes the worksheet easy to update.

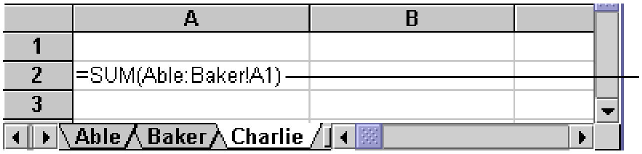
References to Other Worksheets

You can reference cells in other worksheets in the same workbook by placing an exclamation mark between the sheet name and the reference. The sheet name is the name found on the worksheet tab. For example, `Data!A1` refers to the top left cell in a sheet called Data.

Sheet names with spaces. If the sheet name contains spaces, you must enclose the name in single quotes: `‘1994 sales’!B17`.

Cells on two worksheets. You can make a reference to cells on two or more different worksheets by placing a colon between the two sheet names. For example, `Sheet1:Sheet2!A1` refers to two cells: cell A1 in Sheet1 and cell A1 in Sheet2.

Order of sheet names. References to more than one worksheet must list the worksheets in the order in which they appear in the workbook.



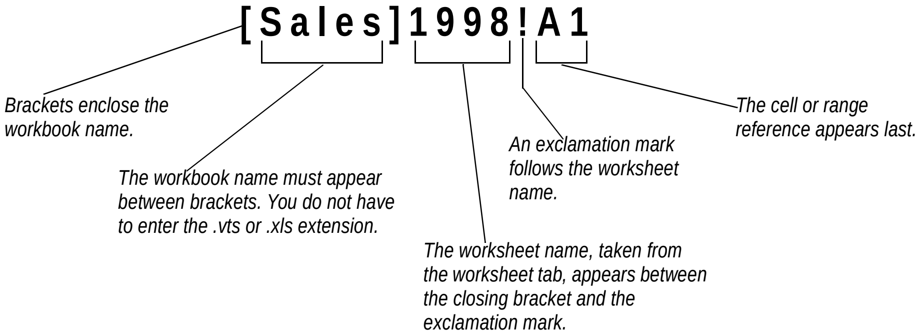
=SUM(Able:Baker!A1)

References to cells in both Able and Baker must list Able first.

References to Other Workbooks

References that point to cells on worksheets in other workbooks are called external references.

An external reference is created by placing the workbook name in brackets, followed by the worksheet name, an exclamation point, and finally a cell or range reference.


[Sales]1998!A1
Brackets enclose the workbook name.
The workbook name must appear between brackets. You do not have to enter the .vts or .xls extension.
An exclamation mark follows the worksheet name.
The worksheet name, taken from the worksheet tab, appears between the closing bracket and the exclamation mark.
The cell or range reference appears last.

External references will work only if both workbooks are open in the workbook designer. If the referenced workbook is not open when you create the external reference, an Invalid Formula Syntax error message will appear.

Developers who want to use external references when they are not using the Workbook Designer control must create a group of workbooks using the setGroup method in the View class. All workbooks that refer to each other must be added to the same group in order for the external references to work.

The following are examples of external references using absolute, relative, and mixed references.

Reference	Type
[Sales]1987!A1	Relative reference pointing to cell A1 in a worksheet titled 1987 of a workbook titled Sales.
[FY91]January!$A$1	Absolute reference pointing to cell A1 in a worksheet titled January of a workbook titled FY91.
[Q1]Sheet1:Sheet2!$A1	Absolute column reference, relative row reference pointing to cell A1 in the first and second worksheets of a workbook titled Q1.
[Store1]Sheet1:Sheet4!A1:F1	Relative row and column reference pointing to the range A1 to F1 in the first four worksheets of a workbook titled Store1.

Paths in External References

After you enter an external reference, Formula One for Java will change the format of the reference to show the absolute path to the workbook you referenced. For example, say you entered this reference to a workbook named September in the Payroll directory on your C drive:

```
[September]Payroll!C2:C420
```

After you enter that reference, if you return to the cell where the reference was entered, you will note that Formula One for Java has changed it to:

```
‘[C:\Payroll\September.vts]Payroll’!C2:C420
```

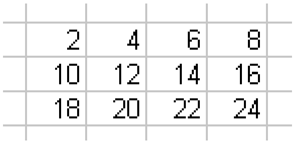
This absolute path is recorded in the worksheet. If you later move the September workbook, the external reference should still work, as long as you open September.vts in the Workbook Designer at the same time as the workbook that references it.

Array Constants in Arguments

For many arguments, you can substitute an array constant for the data required. An array constant is a list of numbers enclosed in curly brackets { }. The function treats each item in an array constant as an individual bit of data, just as it treats the data in each cell in a range reference argument individually.

You can enter an array constant that functions like a range reference, with cells and columns. Use commas to separate individual cells on the same row; use semicolons to separate rows.

For example, the array constant `{2,4,6,8;10,12,14,16;18,20,22,24}` is equivalent to this range reference:



2	4	6	8
10	12	14	16
18	20	22	24

You cannot use array constants as arguments in database functions.

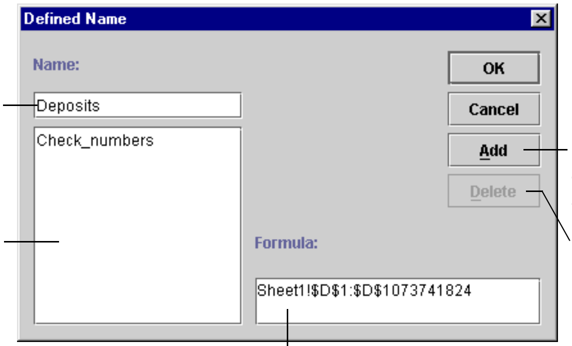
Using Names

User-defined names are an easy way to identify a cell, a range of cells, a value, or a formula. For example, the formula `=Sales-Expenses` is much clearer than `=A10-A6`. When you create a name for a range of cells, you can use that name (without quotation marks) in formulas.

You can also use names to identify constants and formula expressions. For example, you might define the name `LtSp` as 186000. You could then use the name `LtSp` in all your formulas.

➤ To define names:

1. If you are naming a range, select it.
2. Select Insert > Name. The Defined Name dialog, shown below, will appear.



Name: Deposits

Check_numbers

Formula: Sheet1!$D$1:$D$1073741824

Enter the name of the range, formula, or constant here. Do not use spaces in the name.

All the user-defined names in the workbook appear here.

Click here to add a new name.

Click here to delete the selected name.

If you selected a cell or range, its reference appears here. You may edit it. You may also enter a formula or constant value that you want to name.

3. When you are finished, click OK.

Do not use the names `Print_Area` or `Print_Titles`. Formula One for Java reserves those names for its own use.

Identifying Functions on Worksheets

Formula One for Java gives you two tools to help you identify cells that contain function formulas: type markers and view options.

Type Markers

Type markers are colored borders around cells that identify the type of data in the cell. By default, type markers are not displayed.

Type marker color	Data in cell
Green	Values
Red	Formulas and functions
Blue	Empty, formatted cells

➤ To display type markers:

1. Choose Tools > Options and click the General tab, if necessary.
2. Check the Type Markers box, then click OK.

View Options

By default, the workbook designer displays the results of function calculations in the cells where functions are entered. You may choose to display the function formula instead.

➤ **To display function formulas instead of the results of the functions:**

1. Select the worksheet(s) on which you want to see function formulas.
2. Choose Format > Sheet > Properties and click the View tab, if necessary.
3. Check the Formulas box, then click OK.

When you choose to view function formulas, Formula One for Java automatically doubles the column widths of the worksheet(s) to provide space for the formulas. It also left-aligns all cells on that worksheet and displays all numbers in the General format. Returning to normal view will shrink the columns back to their original size and return the alignment and formatting to their original settings.

Nesting Functions

When you use a function as an argument for another function, you are *nesting* functions. The nested function must return the appropriate type of data for the function in which it is nested. You must also provide the necessary arguments for the nested function.

In the following example, the AVERAGE function is used as an argument for the SUM function. In this case, AVERAGE is nested in SUM.

```
=SUM(5.23,6.82,AVERAGE(2.45,5.62,7.74),8.95,9.01)
```

Creating Formulas with Functions

You may use functions with numbers, cell references, names, and mathematical operators to create complex formulas.

The following operators let you specify the type of calculation or evaluation to be performed in the formula.

Operator Type	Operator	Description
Arithmetic	+	Addition
	-	Subtraction
	/	Division
	*	Multiplication
	%	Percentage
	^	Exponentiation
Text	&	Concatenation

Operator Type	Operator	Description
Comparison	=	Equal to
	>	Greater than
	<	Less than
	>=	Greater than or equal to
	<=	Less then or equal to
	<>	Not equal to
Reference	:, .., .	In function arguments: Decimal.
		In range references: Range. Produces a reference that includes all the cells between the two references (e.g., A1:A5 includes cells A1, A2, A3, A4, and A5).
	,	In function arguments: Argument separator.
		In range references: Union. Produces one reference that includes the two references (e.g., A1:A10,C1:C10).

When more than one of the operators appears in a formula, Formula One for Java uses a specific order of precedence to calculate the formula. The operators listed first in the following table are evaluated before the operators below them.

Operator	Description
( )	Parentheses
: .. .	Range
,	Union
-	Negation (when used in front of a constant or variable)
%	Percentage
^	Exponentiation
* /	Multiplication and Division
+ -	Addition and Subtraction (when used between two constants or variables)
&	Text concatenation
= < > <= >= <>	Comparison

When two or more operators on the same line in the table above appear in a formula, Formula One for Java evaluates them from left to right.

Use parentheses to change the order of evaluation. The following example illustrates how the result of a formula can be altered by adding parentheses to change the order of precedence.

Formula	Result
=1+2*37	75
=(1+2)*37	111

Changing Formula Evaluation Rules

In general, Formula One for Java evaluates formulas the way Microsoft Excel does. The Lotus 1-2-3 spreadsheet program evaluates formulas in a slightly different way. Lotus 1-2-3's rules always interpret the logical value TRUE as 1, FALSE as 0. Also, text in cells referred to by formulas and in function arguments is always evaluated as 0 (zero).

You can use Lotus 1-2-3's evaluation rules instead of Microsoft Excel's on any worksheet. You can even have different worksheets in the same workbook use different evaluation rules.

➤ **To use Lotus 1-2-3 evaluation rules:**

1. Select the worksheet(s) whose formulas you want to evaluate using Lotus 1-2-3 rules.
1. Choose Format > Sheet > Properties and click the General tab.
2. Check the Lotus Style Formula Evaluation check box.
3. Click OK.

Array Functions

Most functions return a calculated value in the same cell in which the function was entered. Array functions are a special kind of function that return data to a range of cells instead of to just one cell.

For example, the TRANSPOSE function, which takes a range of cells as its argument, returns a new range of cells in which the rows from the original range are now columns and the original columns are now rows.

When discussing array functions, we distinguish between two different types of ranges: the *argument range* is the range of cells that appear between the parentheses of an array function. The *results range* is the range of cells in which the data computed by the function will be displayed.

The following functions are array functions:

FREQUENCY
GROWTH
INDEX (array type)
LINEST
LOGEST
MMULT
TRANSPOSE
TREND

Entering Array Functions

➤ **To enter an array function:**

1. Select a results range. Some functions require the results range to contain a certain number of columns or rows. A too-small results range will not show all of the function results; a too-large results range will display error messages in some cells. Check the documentation on the specific function for information about the required size and shape of the results range.
2. In the results range you selected, type in the equals sign, the array function keyword, and the argument(s) or argument range(s), in parentheses.
3. To enter the function, press ENTER while holding down CONTROL and SHIFT. (If you simply press ENTER, the array function will return the #VALUE! error.)

Most normal functions can be used as array functions to return the same calculation to a range of cells. To do this, select a range, type in the function, and hold down CONTROL and SHIFT while pressing ENTER.

About Results Ranges

All the cells in the results range of an array function will contain the calculated array function. When you select one of these cells, the function displayed in the formula bar will be enclosed in curly brackets {}. Do not enter these brackets yourself.

Note Results ranges for array functions must be edited as a unit. After you enter an array function, you can change the format of individual cells in the results range, but you cannot edit the data in individual cells. If you try, Formula One for Java will display an error message. To change any of the data in a results range, you must select all the cells and change them all at once.

Database Functions

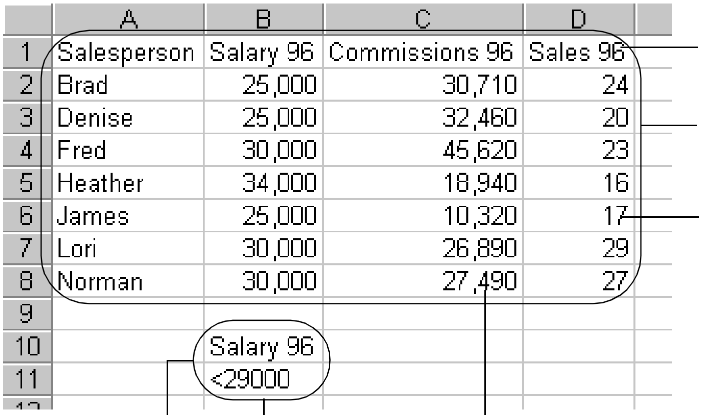
The 12 database functions let you find and perform calculations on specific pieces of data in a database. All of the database functions search a specified database for records that match specified criteria. Some of the database functions then perform calculations on data in a specified field of the matching records.

The following are the database functions.

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

Database Function Example

The following worksheet showing salary, commission, and sales information for a company’s salespeople can be used as a database.



	A	B	C	D
1	Salesperson	Salary 96	Commissions 96	Sales 96
2	Brad	25,000	30,710	24
3	Denise	25,000	32,460	20
4	Fred	30,000	45,620	23
5	Heather	34,000	18,940	16
6	James	25,000	10,320	17
7	Lori	30,000	26,890	29
8	Norman	30,000	27,490	27
9				
10		Salary 96		
11		<29000		

The top database row contains labels.

Database

Each data row is called a record.

Criteria

The data in each data column is called a **field**.

The criteria label(s) must match the database labels.

You can use the DAVERAGE database function to find the average commissions earned in 1996 by all salespeople who made less than $29,000 in salary that year. The arguments for DAVERAGE — *database*, *field*, and *criteria* — are shown in the illustration above. The function searches *database* for records that comply with *criteria*, then finds the average of *field* in the records that pass the test.

This function returns 24,497, the average commissions of the three salespeople (Brad, Denise, and James) whose salaries are less than $29,000:

```
DAVERAGE (A1:D8,"Commissions 96",B10:B11)
```

All database functions have the same three arguments: *database*, *field*, and *criteria*.

The *database* Argument

Database is a reference to a range of cells containing data that the database function searches. Each row of data in the database is called a *record*.

The top row must contain unique text labels identifying the data in the columns. To use numbers or dates as labels, identify them as text by entering a leading apostrophe (’). Since the labels must match the labels in *field* and *criteria*, it’s best to make them simple and descriptive, with no extra spaces. In general, names should be free of punctuation marks and other symbols.

The *field* Argument

Field is the column within *database* containing the data on which you want the function to perform the calculation. You may enter the column’s label enclosed in quotation marks or a number corresponding to the column’s position within the database: the first column is 1, the second column is 2, etc. You may also enter a reference to a cell that contains the column’s label or number.

The *criteria* Argument

Criteria is a reference to a range of cells, usually separate from *database*, containing search criteria. The top row contains text labels that match the labels in *database*. The cells beneath the labels contain criteria that data in the appropriate column of *database* must pass in order for that record to be selected for the function's calculation.

The criteria can be numbers, text, or blank cells. You can enter the criteria directly in the cell, or you can enter formulas and functions that calculate the criteria. You may also enter cell references.

Note Microsoft Excel does not support formulas or functions in database criteria. If your worksheets must be compatible with Microsoft Excel, do not use formulas or functions in your criteria.

Numbers

Numbers will be compared to the numbers in *database*. Only exact matches pass the test. An = before the number is optional.

To test for relationships between the criteria and the numbers in *database*, precede the criteria with the operators <, >, <=, >=, and <> (not equal). For example, `<500` will find all numbers less than 500.

Text

Text will be compared to text in database. Only exact matches pass the test. The function does not consider the case of the letters, so the criteria `DENISE` will find `Denise` and `denise`. Do not use quotation marks (unless you want the database function to find matches containing the quotation marks).

To test for words close to a particular word, use the wildcard characters ? and *. The ? stands in for exactly one character; * stands for zero or more characters. You may use more than one wildcard character in a cell. Any characters after the * wildcard must be located at the end of the cell text. For example, the criteria `JA*S` will find `JAMES` and `JARS`.

To test for text that is alphabetized before or after the criteria, use the operators <, >, <=, and >=. For example, the criteria `>BRAD` will find all cells that start with text that is alphabetized after the word BRAD. To test for text that is not the same as the criteria, use the operator <> (not equal). For example, the criteria `<>FRED` will find all cells that do not contain just the word FRED.

Blank cells

A blank cell indicates that no tests should be performed. A blank row in the criteria range indicates all records should be selected.

Using Functions in Criteria

Functions in criteria let you create sophisticated search strategies that are easy to manipulate. For example, to find all dates that are earlier than today, you can enter the following function as criteria:

```
<TODAY()
```

➤ **Avoid using the following types of functions in criteria ranges:**

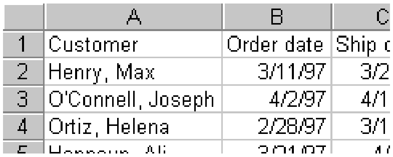
- Functions such as ROW and COLUMN that return information about the current cell. That information will change as the function steps through the records in the database.
- Functions such as RAND that do not return consistent results.
- Complex functions, if the user plans to use SQL to link to databases outside of Formula One for Java. (This capability is not currently part of Formula One for Java, but it may be added later as an enhancement.)

Using Cell References in Criteria

When creating criteria, you can use cell references. For example, to find all dates that match the date in cell H14, enter the following criteria:

```
=$H$14
```

You can also create complex criteria that refer to cells in the database. For example, say you have a database that contains customer names in column A and merchandise order dates in column B:



	A	B	C
1	Customer	Order date	Ship d
2	Henry, Max	3/11/97	3/2
3	O'Connell, Joseph	4/2/97	4/1
4	Ortiz, Helena	2/28/97	3/1

To search for all records with order dates within the last month, and to ensure that the formula you enter will always be updated to the current date, you can enter the following criteria:

```
=AND(B2<TODAY(),B2>=TODAY()-30)
```

The function will perform the test on cell B2, the topmost data cell in the order dates field. It will then perform the test on B3, B4, and so on.

You can use multiple nested functions and refer to one or more database column to create complex selection criteria.

➤ **When your criteria contains cell references, follow these guidelines:**

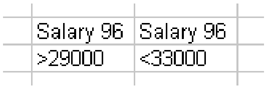
- Criteria that tests against the data in one particular cell (generally outside the database) should use an absolute cell reference.

- Criteria that contains formulas that test against each record in the database should use a relative reference to the first row of data in its field.
- Since criteria formulas that test against each record in the database may contain relative references to more than one cell in the database, the label above the formula's cell can match any of the database labels.

Multiple Criteria

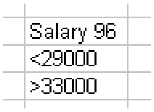
A criteria range that contains more than one row or column has multiple criteria.

When more than one cell in the first criteria row contains criteria, the function finds only the records that pass all the tests. You may use labels more than once in a criteria range to establish multiple criteria on a single field. For example, the following criteria range would find database records in which the 1996 salaries are between $29,000 and $33,000:



Salary 96	Salary 96
>29000	<33000

You may also enter more than one row of criteria. If a record passes a test for one criteria row, it is picked regardless of whether it passes the tests in the other criteria rows. For example, the following criteria range would find all database records in which the 1996 salaries are either below $29,000 or above $33,000:



Salary 96
<29000
>33000

Financials Functions

Many of the Financials functions have the same arguments. The following information expands on the information in the field descriptions of the functions' documentation.

The *frequency* Argument

The *frequency* argument indicates the number of interest payments per year. The options are:

Frequency	Description
1	Annual payments.
2	Semi-annual payments.
4	Quarterly payments.

If you enter any values other than 1, 2, or 4, the function will return the #NUM! error. Decimal values will be truncated to integers.

The *calendar_type* Argument

The *calendar_type* argument lets you pick from five different methods of counting days, usually for computing interest. The options are:

Calendar type	Description
0	30-day months, 360-day years, American method.
1	Actual months, actual years.
2	Actual months, 360-day years.
3	Actual months, 365-day years.
4	30-day months, 360-day years, European method.

When *calendar_type* is 0 or 4, the DAYS360 function is used to calculate number of days. It has special rules for when the start and/or end dates fall on the 30th and/or 31st of the month. See “DAYS360” on page 86 for more information.

If you omit the *calendar_type* argument, 0 is used. If you enter any values other than 0, 1, 2, 3, or 4, the function will return the #NUM! error. Decimal values will be truncated to integers.

Understanding Errors

When a formula cannot be properly calculated, an error is returned in the cell. The following table lists the errors that can be generated and their causes.

Error	Cause
#DIV/0!	Divide by zero. May be caused by a reference to a blank cell or a cell containing zero.
#FORMULA!	Formula cannot be calculated. May be caused when opening a workbook from another file format.
#N/A	No value is available. May be caused by inappropriate values in the formula or a reference to a cell containing the #N/A value.
#NAME?	Name is not recognized. May be caused because a user-defined name is not defined, a function name is misspelled, or you are using an add-in function whose JAR file is not in your class path.
#NULL!	Null intersection. An intersection of two ranges was defined that does not intersect.
#NUM!	Number problem. May be caused by inappropriate numbers in functions, an iteration that cannot solve for a value, or a formula that results in a number too large or too small to represent.
#REF!	Reference error. May be caused by referring to a cell that was deleted.
#VALUE!	Wrong argument type. May be caused by entering text where a number was expected, or supplying a range to an operator or function that was expecting a single value.

Tidestone

CHAPTER 2

Functions by Category

This chapter lists the Formula One for Java functions by category and gives a short description of each function. The categories are:

Database Functions

Function	Description
DAVERAGE	Uses specified criteria to select records from a database, then computes the average of the numeric values in a specified field of the selected records.
DCOUNT	Uses specified criteria to select records from a database, then counts the number of selected records.
DCOUNTA	Uses specified criteria to select records from a database, then counts the number of selected records.
DGET	Uses specified criteria to select a single record from a database, then displays the data found in a specified field of the selected record.
DMAX	Uses specified criteria to select records from a database, then displays the highest number value found in a specified field of the selected records.
DMIN	Uses specified criteria to select records from a database, then displays the lowest number value found in a specified field of the selected records.
DPRODUCT	Uses specified criteria to select records from a database, then multiplies the numeric values in a specified field of the selected records.
DSTDEV	Uses specified criteria to select records from a database, then computes the sample standard deviation of the numeric values in a specified field of the selected records.
DSTDEVP	Uses specified criteria to select records from a database, then computes the population standard deviation of the numeric values in a specified field of the selected records.
DSUM	Uses specified criteria to select records from a database, then adds the numeric values in a specified field of the selected records.
DVAR	Uses specified criteria to select records from a database, then computes the sample variance of the numeric values in a specified field of the selected records.
DVARP	Uses specified criteria to select records from a database, then computes the population variance of the numeric values in a specified field of the selected records.

Date and Time Functions

Function	Description
DATE	Returns the serial number of the supplied date.
DATEDIF	Calculates the number of days, months, or years between two specified dates.
DATEVALUE	Returns the serial number of a date supplied as a text string.
DAY	Returns the day of the month that corresponds to the date represented by the supplied number.

Function	Description
DAYS360	Returns the number of days between two dates, based on a 360-day year (twelve 30-day months).
EDATE	Finds a date a specified number of months before or after a given date.
EOMONTH	Finds the last date in the month that is a specified number of months before or after a given date.
HOUR	Returns the hour component of the specified time, in 24-hour format.
MINUTE	Returns the minute that corresponds to the supplied date.
MONTH	Returns the month that corresponds to the supplied date.
NETWORKDAYS	Computes the number of whole working days between the specified start date and the specified end date.
NOW	Returns the current date and time as a serial number.
SECOND	Returns the second that corresponds to the supplied date.
TIME	Returns a serial number for the supplied time.
TIMEVALUE	Returns a serial number for the supplied text representation of time.
TODAY	Returns the current date as a serial number.
WEEKDAY	Returns the day of the week that corresponds to the supplied date.
WORKDAY	Returns the date that is a specified number of days before or after a specified date, not counting weekends and specified holidays.
YEAR	Returns the year that corresponds to the supplied date.
YEARFRAC	Computes the fraction of a year between two specified dates.

Engineering Functions

Function	Description
BESSELI	Computes the value of the *n*th order modified Bessel function of the first kind evaluated at *x*.
BESSELJ	Computes the value of the *n*th order Bessel function of the first kind evaluated at *x*.
BESSELK	Computes the value of the *n*th order Bessel function of the second kind evaluated at *x*.
BESSELY	Computes the value of the *n*th order modified Bessel function of the second kind evaluated at *x*.
BIN2DEC	Converts a binary number (base 2) to a decimal number (base 10).
BIN2HEX	Converts a binary number (base 2) to a hexadecimal number (base 16).
BIN2OCT	Converts a binary number (base 2) to a octal number (base 8).
COMPLEX	Creates a complex number.
CONVERT	Converts a numeric value from one set of units to another.
DEC2BIN	Converts a decimal number (base 10) to a binary number (base 2).

Function	Description
DEC2HEX	Converts a decimal number (base 10) to an hexadecimal number (base 16).
DEC2OCT	Converts a decimal number (base 10) to an octal number (base 8).
DELTA	Compares two specified values and returns 1 if they are equal, 0 if they are not.
ERF	Computes the “error” function.
ERFC	Computes the complementary “error” function.
GESTEP	Compares two specified values and returns 1 if the first value is greater than or equal to the second value, 0 if the first value is less than the second value.
HEX2BIN	Converts a hexadecimal number (base 16) to a binary number (base 2).
HEX2DEC	Converts a hexadecimal number (base 16) to a decimal number (base 10).
HEX2OCT	Converts a hexadecimal number (base 16) to an octal number (base 8).
IMABS	Computes the absolute value of a complex number.
IMAGINARY	Returns the coefficient of the imaginary portion of a complex number as a real number.
IMARGUMENT	Computes the argument θ as an angle expressed in radians.
IMCONJUGATE	Computes the complex conjugate of the argument.
IMCOS	Computes the complex cosine of the argument.
IMDIV	Divides one complex number by another.
IMEXP	Computes the complex exponential of the specified complex number.
IMLN	Computes the complex natural logarithm of the specified complex number.
IMLOG10	Computes the complex common logarithm of the specified complex number.
IMLOG2	Computes the complex logarithm base-2 of the specified complex number.
IMPOWER	Computes the real power of a complex number.
IMPRODUCT	Computes the product of up to 29 complex numbers.
IMREAL	Returns the coefficient of the real portion of a specified complex number.
IMSIN	Computes the complex sine of the specified complex number.
IMSQRT	Computes the complex square root of the specified complex number.
IMSUB	Subtracts one complex number from another.
IMSUM	Computes the sum of up to 29 complex numbers.
OCT2BIN	Converts an octal number (base 8) to a binary number (base 2).
OCT2DEC	Converts an octal number (base 8) to a decimal number (base 10).
OCT2HEX	Converts an octal number (base 8) to a hexadecimal number (base 16).
SQRTPI	Calculates the square root of the product of the specified number and π.

Financial Functions

Function	Description
ACCRINT	Computes the accrued interest for a security that pays periodic interest, from the last coupon date to the settlement date.
ACCRINTM	Computes the accrued interest for a security that pays interest at maturity.
AMORDEGRC	Computes depreciation for the French accounting system, making use of a special French tax rule that allows over-depreciation of assets during the early years of ownership.
AMORLINC	Computes depreciation for the French accounting system.
COUPDAYBS	Computes the number of days from the coupon date previous to the settlement date and the settlement date.
COUPDAYS	Computes the number of days in the coupon period containing the settlement date.
COUPDAYSNC	Computes the number of days from the settlement date to the next coupon date.
COUPNCD	Computes the coupon date that follows the settlement date.
COUPNUM	Computes the number of coupons between settlement and maturity.
COUPPCD	Computes the last coupon date before the settlement date.
CUMIPMT	Computes the cumulative interest for the specified period.
CUMPRINC	Computes the cumulative principal paid for the specified period.
DB	Returns the real depreciation of an asset for a specific period of time using the fixed-declining balance method.
DDB	Returns the depreciation of an asset for a specific period of time using the double-declining balance method or a declining balance factor you supply.
DISC	Computes the discounted rate for a security.
DOLLARDE	Converts a dollar figure from fractional form to decimal form.
DOLLARFR	Converts a dollar figure from decimal form to fractional form.
DURATION	Computes the Macaulay duration for a security, in years.
EFFECT	Computes the effective annual interest rate, which adjusts the nominal rate to show the effect of compounding.
FV	Returns the future value of an annuity based on regular payments and a fixed interest rate.
FVSCHEDULE	Computes the future value of an investment after applying a series of compounded interest rates.
INTRATE	Computes the interest rate of a fully invested security, given a specified investment dollar amount and redemption value.
IPMT	Returns the interest payment of an annuity for a given period, based on regular payments and a fixed periodic interest rate.
IRR	Returns internal rate of return for a series of periodic cash flows.

Function	Description
MIRR	Returns the modified internal rate of return for a series of periodic cash flows.
MDURATION	Computes the modified duration for a security.
MINVERSE	Returns the modified internal rate of return for a series of periodic cash flows.
NOMINAL	Computes the nominal annual interest rate, given the effective interest rate.
NPER	Returns the number of periods of an investment based on regular periodic payments and a fixed interest rate.
NPV	Returns the net present value of an investment based on a series of periodic payments and a discount rate.
ODDFPRICE	Computes the price of a security purchased during a first coupon period that is shorter or longer than the other coupon periods.
ODDFYIELD	Computes the yield of a security per $100 face value when the first coupon period is shorter or longer than the other coupon periods.
ODDLPRICE	Computes the price of a security purchased during a last coupon period that is shorter or longer than the other coupon periods.
ODDLYIELD	Computes the yield of a security per $100 face value when the last coupon period is shorter or longer than the other coupon periods.
PMT	Returns the periodic payment of an annuity, based on regular payments and a fixed periodic interest rate.
PPMT	Returns the principle paid on an annuity for a given period.
PRICE	Computes the price of a security with specified rate, redemption value, and yield.
PRICEDISC	Computes the price of a discounted security per $100 face value, given a specified discount rate and redemption value.
PRICEMAT	Computes the price of a security that pays interest only at maturity, given specified dates, interest rate, and yield.
PV	Returns the present value of an annuity, considering a series of constant payments made over a regular payment period.
RATE	Returns the interest rate per period of an annuity, given a series of constant cash payments over a regular payment period.
RECEIVED	Computes the amount received at maturity for a fully invested security.
SLN	Returns the depreciation of an asset for a specific period of time using the straight-line balance method.
SYD	Returns the depreciation of an asset for a specified period using the sum-of-years method.
TBILLEQ	Computes the bond-equivalent yield for a treasury bill.
TBILLPRICE	Computes the price per $100 face value for a treasury bill.
TBILLYIELD	Computes the yield for a treasury bill.

Function	Description
VDB	Returns the depreciation of an asset for a specified period using a variable method of depreciation.
XIRR	Computes the internal rate of return for an investment with flexible periods.
XNPV	Computes the net present value of an investment with flexible periods.
YIELD	Computes the annual yield of a security that pays periodic interest.
YIELDDISC	Computes the annual yield of a discounted security, given a specified price and redemption value.
YIELDMAT	Computes the annual yield on a security that pays interest only at maturity, given specified dates, interest rate, and price.

Information Functions

Function	Description
CELL	Returns the specified type of information about a specified cell.
COUNTBLANK	Counts the number of empty cells in a specified range.
ERROR.TYPE	Returns a number corresponding to an error.
INFO	Returns the specified type of information about the current workbook and system.
ISBLANK	Determines if the specified cell is blank.
ISERR	Determines if the specified expression returns an error value.
ISERROR	Determines if the specified expression returns an error value.
ISEVEN	Determines whether the specified expression returns an even value.
ISLOGICAL	Determines if the specified expression returns a logical value.
ISNA	Determines if the specified expression returns the value not available error.
ISNONTEXT	Determines if the specified expression is not text.
ISNUMBER	Determines if the specified expression is a number.
ISODD	Determines whether the specified expression returns an odd value.
ISREF	Determines if the specified expression is a range reference.
ISTEXT	Determines if the specified expression is text.
N	Tests the supplied value and returns the value if it is a number.
NA	Returns the error value #N/A, which represents “not available.”
TYPE	Returns the argument type of the given expression.

Logical Functions

Function	Description
AND	Returns True if all arguments are true; returns False if at least one argument is false.
FALSE	Returns the logical value False.
IF	Tests the condition and returns the specified value.
NOT	Returns a logical value that is the opposite of its value.
OR	Returns True if at least one of a series of logical arguments is true.
TRUE	Returns the logical value True.

Lookup and Reference Functions

Function	Description
ADDRESS	Creates a cell address as text.
AREAS	Counts the number of areas in a range reference.
CHOOSE	Returns a value from a list of numbers based on the index number supplied.
COLUMN	Returns the column number of the supplied reference.
COLUMNS	Returns the number of columns in a range reference.
HLOOKUP	Searches the top row of a table for a value and returns the contents of a cell in that table that corresponds to the location of the search value.
HYPERLINK	Sets up a hyperlink to a file or a web page.
INDEX (array type)	Returns a specified value from an array.
INDEX (non-array type)	Returns the contents of a cell from a specified range.
INDIRECT	Returns the contents of the cell referenced by the specified cell.
LOOKUP	Searches for a value in one range and returns the contents of the corresponding position in a second range.
MATCH	Compares a specified value against values in a range and returns the position of the matching value in the search range.
OFFSET	Returns the contents of a range that is offset from a starting point in the spreadsheet.
ROW	Returns the row number of the supplied reference.
ROWS	Returns the number of rows in a range reference.
TRANSPOSE	Places a copy of the contents of a range into a new range in which the original rows are now columns and the original columns are now rows.
VLOOKUP	Searches the first column of a table for a value and returns the contents of a cell in that table that corresponds to the location of the search value.

Math and Trigonometry Functions

Function	Description
ABS	Returns the absolute value of a number.
ACOS	Returns the arc cosine of a number.
ACOSH	Returns the inverse hyperbolic cosine of a number.
ASIN	Returns the arcsine of a number.
ASINH	Returns the inverse hyperbolic sine of a number.
ATAN	Returns the arctangent of a number.
ATAN2	Returns the arctangent of the specified coordinates.
ATANH	Returns the inverse hyperbolic tangent of a number.
CEILING	Rounds a number up to the nearest multiple of a specified significance.
COS	Returns the cosine of an angle.
COSH	Returns the hyperbolic cosine of a number.
COUNTIF	Returns the number of cells within a range that meet the given criteria.
DEGREES	Converts a value in radians to degrees.
EVEN	Rounds the specified number up to the nearest even integer.
EXP	Returns e raised to the specified power.
FACT	Returns the factorial of a specified number.
FACTDOUBLE	Returns the double-factorial of a specified number.
FLOOR	Rounds a number down to the nearest multiple of a specified significance.
GCD	Computes the greatest common divisor of a specified set of values.
INT	Rounds the supplied number down to the nearest integer.
LCM	Computes the least common multiple of a specified set of values.
LN	Returns the natural logarithm (based on the constant e) of a number.
LOG	Returns the logarithm of a number to the specified base.
LOG10	Returns the base-10 logarithm of a number.
MDETERM	Computes the determinant of a square matrix.
MINVERSE	Computes the inverse of a square matrix.
MMULT	Performs matrix multiplication on two arrays.
MOD	Returns the remainder after dividing a number by a specified divisor.
MROUND	Rounds a specified number to an even integral multiple of a specified factor.
MULTINOMIAL	Computes the multinomial of up to 29 numbers.
ODD	Rounds the specified number up to the nearest odd integer.
PI	Returns the value of pi.
POWER	Raises the specified base number to the specified power.
PRODUCT	Multiplies a list of numbers and returns the result.

Function	Description
QUOTIENT	Computes the quotient of two numbers, truncated to an integer (toward 0).
RADIANS	Converts a value in degrees to radians.
RAND	Returns a number selected randomly from a uniform distribution greater than or equal to 0 and less than 1.
RANDBETWEEN	Returns an integer selected randomly from between two specified limits, inclusive, each time the sheet is recalculated.
ROMAN	Converts an arabic number to a Roman numeral.
ROUND	Rounds the given number to the supplied number of decimal places.
ROUNDDOWN	Rounds the given number down.
ROUNDUP	Rounds the given number up to the specified number of decimal places.
SERIESSUM	Returns the sum of a power series.
SIGN	Determines the sign of the specified number.
SIN	Returns the sine of the supplied angle.
SINH	Returns the hyperbolic sine of the specified number.
SQRT	Returns the square root of the specified number.
SQRTPI	Calculates the square root of the product of the specified number and π.
SUBTOTAL	Performs one of 11 functions on the specified data ranges or other cell references.
SUM	Returns the sum of the supplied numbers.
SUMIF	Returns the sum of the specified cells based on the given criteria.
SUMPRODUCT	Multiplies the corresponding cells in the given ranges, then returns the sum of those products.
SUMSQ	Squares each of the supplied numbers and returns the sum of the squares.
SUMX2MY2	Computes $x^2 - y^2$ for two sets of numbers, *arrayX* and *arrayY*, paired up one-to-one.
SUMX2PY2	Computes $x^2 - y^2$ for two sets of numbers, *arrayX* and *arrayY*, paired up one-to-one.
SUMXMY2	Computes $x^2 - y^2$ for two sets of numbers, *arrayX* and *arrayY*, paired up one-to-one.
TAN	Returns the tangent of the specified angle.
TANH	Returns the hyperbolic tangent of a number.
TRUNC	Truncates the given number to an integer.

Statistical Functions

Function	Description
AVEDEV	Computes the average deviation of a list of numbers.
AVERAGE	Returns the average of the supplied numbers. The result of AVERAGE is also known as the arithmetic mean.
AVERAGEA	Computes the average of a list of numbers, considering textual values.
BETADIST	Computes the cumulative beta distribution.
BETAINV	Computes the inverse of the BETADIST function.
BINOMDIST	Computes the binomial distribution, which is used to determine probabilities on repeated tests where each test is independent of every other test.
CHIDIST	Computes the “complementary” version of the chi-square distribution.
CHIINV	Computes the inverse of the “complementary” chi-square distribution.
CHITEST	Computes the probability of the distribution against expected values.
CONFIDENCE	Computes the confidence interval, assuming a normal distribution.
CORREL	Computes the correlation coefficient for two sets of numbers, paired up one-to-one.
COUNT	Returns the number of values in the supplied list.
COUNTA	Returns the number of nonblank values in the supplied list.
COVAR	Computes the covariance for pairwise numbers in two arrays
CRITBINOM	Computes the smallest number of successes that will result in the BINOMDISTCUM function giving an answer larger than a specified criteria value.
DEVSQ	Computes the square of the deviation of the numbers.
EXPONDIST	Computes the exponential distribution.
FDIST	Computes the F-distribution, which is used to compare data from two different populations.
FINV	Computes the inverse of the complementary F-distribution (that is, the inverse of the FDIST function).
FISHER	Computes Fisher’s z-transformation.
FISHERINV	Computes the inverse of Fisher’s z-transformation.
FORECAST	Constructs the least squares regression line through the data given, then computes the predicted *y* value for the requested *x*.
FREQUENCY	Distributes the specified array of numbers, sorted in ascending order, into specified bins.
FTEST	Computes an F-test probability for two specified distributions.
GAMMADIST	Computes the gamma distribution for a given value.
GAMMAINV	Computes the inverse of the cumulative gamma distribution.
GAMMALN	Computes the logarithm of the gamma function.
GEOMEAN	Computes the geometric mean of a list of positive numbers.

Function	Description
GROWTH	Computes result values from the fitted curve of a multiple exponential regression for a group of specified observations relative to a group of specified independent variables.
HARMEAN	Computes the harmonic mean of a list of positive numbers.
HYPGEOMDIST	Computes the hypergeometric distribution.
INTERCEPT	Computes the y-intercept of the least squares linear regression line through the given data.
KURT	Computes the kurtosis or of a list of numbers.
LARGE	Computes the *X*th largest value in a list.
LINEST	Computes multiple linear regression for a group of observations relative to a number of independent variables.
LOGEST	Computes multiple exponential regression for a group of observations relative to a number of independent variables.
LOGINV	Computes the inverse of the cumulative lognormal distribution of *x*, where the logarithm of *x* is normally distributed.
LOGNORMDIST	Computes the cumulative lognormal distribution for *x*, where the logarithm of *x* is normally distributed.
MAX	Returns the largest value in the specified list of numbers.
MAXA	Returns the largest value in the specified list of numbers, considering textual values.
MEDIAN	Computes the median or middle value of a list of numbers arranged in ascending order.
MIN	Returns the smallest value in the specified list of numbers.
MINA	Returns the smallest value in the specified list of numbers, considering textual values.
MODE	Computes the most frequent value in a list of numbers (the number that appears most often in the list).
NEGBINOMDIST	Computes the negative binomial distribution.
NORMDIST	Computes the normal distribution.
NORMINV	Computes the inverse of the cumulative normal distribution.
NORMSDIST	Computes the standard normal cumulative distribution (often called “the bell curve”).
NORMSINV	Computes the inverse of the standard normal distribution.
PEARSON	Computes the correlation coefficient for two sets of numbers, paired up one-to-one.
PERCENTILE	Computes the value corresponding to the specified percentile of a specified range of numbers.
PERCENTRANK	Computes the percent rank of a specified value in a specified array of numbers.
PERMUT	Computes the number of combinations possible by taking *k* items at a time from a pool of *n*, where order in the sample taken is important.

Function	Description
POISSON	Computes the Poisson distribution, which is usually used to determine the probability of a certain number of repeated events taking place over time.
PROB	Given a set of numbers and a probability associated with each number, computes the probability corresponding to a specified number.
QUARTILE	Computes the value corresponding to the specified quartile in a specified array of numbers.
RANK	Computes the rank of a specified value in a specified array of numbers.
RSQ	Computes the square of the correlation coefficient for two sets of numbers, paired up one-to-one.
SKEW	Computes the skewness of a set of numbers.
SLOPE	Computes the slope of the least squares linear regression line through the given data.
SMALL	Computes the *X*th smallest value in a list.
STANDARDIZE	Computes the proper argument for the function that calculates the standard normal cumulative distribution, given specified values for x, mean, and standard deviation.
STDEV	Returns the standard deviation of a population based on a sample of supplied values.
STDEVA	Returns the standard deviation of a population based on a sample of supplied values, considering textual values.
STDEVP	Returns the standard deviation of a population based on an entire population of values.
STDEVPA	Returns the standard deviation of a population based on an entire population of values, considering textual values.
STEYX	Computes the standard error of the predicted *y*-value for each *x* in the regression.
TDIST	Computes the complementary student's T-distribution.
TINV	Computes the x value corresponding to a specified probability value, given a specified number of degrees of freedom.
TREND	Computes result values from the fitted curve of a multiple linear regression for a group of specified observations relative to a group of specified independent variables.
TRIMMEAN	Computes the mean of a list of numbers after reducing a specified percent of the members of the list.
TTEST	Computes the student's T-distribution from the data in two specified arrays, then computes the probability.
VAR	Returns the variance of a population based on a sample of values.
VARA	Returns the variance of a population based on a sample of values, considering textual values.
VARP	Returns the variance of a population based on an entire population of values.

Function	Description
VARPA	Returns the variance of a population based on the entire population of values, considering textual values.
WEIBULL	Computes the Weibull distribution.
ZTEST	Computes the two-tailed probability of a z-test, which is a test of a specified value against a specified set of numbers.

Text Functions

Function	Description
CHAR	Returns a character that corresponds to the supplied Unicode value.
CLEAN	Removes all non-printable characters from the supplied text.
CODE	Returns the Unicode value of the first character of the supplied string.
CONCATENATE	Joins several text items into one item.
DOLLAR	Returns the specified number as text, using the local currency format and the supplied precision.
EXACT	Compares two expressions for identical, case-sensitive matches and returns True if identical, False if not.
FIND	Searches for a string of text within another text string and returns the character position at which the search string first occurs.
FIXED	Rounds a number to the supplied precision, formats the number in decimal format, and returns the result as text.
LEFT	Returns the leftmost character from the specified text string.
LEN	Returns the number of characters in the supplied text string.
LOWER	Changes the characters in the specified string to lowercase characters.
MID	Returns a specified number of characters from a text string, beginning with the specified starting position
PROPER	Returns the specified string in proper-case format.
REPLACE	Replaces part of a text string with another text string.
REPT	Repeats a text string the specified number of times.
RIGHT	Returns the rightmost character(s) from the given text string.
SEARCH	Locates the position of the first character of a specified text string within another string.
SUBSTITUTE	Replaces a specified part of a text string with another text string.
T	Tests the supplied value and returns the value if it is text.
TEXT	Returns the given number as text, using the specified formatting.
TRIM	Removes all spaces from text except single spaces between words.

Function	Description
UPPER	Changes the characters in the specified string to uppercase characters.
USDOLLAR	Returns the specified number as text using the US dollar format and the supplied precision.
VALUE	Returns the specified text as a number.

Tidestone

CHAPTER 3

A-Z Function Reference

This chapter provides a complete alphabetical reference for the Formula One for Java worksheet functions.

ABS

Description Returns the absolute value of a number.

Syntax ABS (*number*)

Argument	Description
number	Any number.

Remarks An absolute value does not display a positive or negative sign.

Examples These functions both return 1:

```
ABS(-1)
```

```
ABS(1)
```

See Also SIGN

ACCRINT

Description Computes the accrued interest for a security that pays periodic interest, from the last coupon date to the settlement date.

Syntax ACCRINT (*issue, first_interest*, *settlement*, *rate* [, *par*], *frequency* [, *calendar_type*])

Argument	Description
issue	The date the security was issued and began accumulating interest. Dates in the argument list must be in the form of a serial number or text. Decimal values will be truncated to integers.
first_interest	The date on which the first interest payment is due. Dates in the argument list must be in the form of a serial number or text. Decimal values will be truncated to integers.
settlement	The date when the security is traded to the buyer. It must be later than *issue*. Dates in the argument list must be in the form of a serial number or text. Decimal values will be truncated to integers.
rate	The security's annual coupon rate. The coupons pay at *rate* divided by *frequency.*
[par]	Optional. The security's face value or par value. If you omit this argument, 1000 is used.
frequency	The number of interest payments per year. See "The frequency Argument" on page 18 for more information.
[calendar_type]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Remarks

Use ACCRINT to compute the portion of the security's price that is interest at settlement.

Equation

$$par \times \frac{rate}{frequency} \times \sum_{j=1}^{\mathrm{NC}} \left(\frac{\mathrm{Aj}}{\mathrm{NLj}}\right)$$

...where *j* is the number of pseudo-periods in the odd period. The other codes are explained in the following table.

Code	Meaning
Aj	Number of accrued days for the *j*th pseudo-period within the odd period.
NC	Number of pseudo-periods that fit in the odd period (fractional values are raised to integers).
NLj	Number of days in the *j*th pseudo-period within the odd period.

Example

This function returns 51.50685:

```
ACCRINT("9/12/92","9/30/92","12/15/92",0.2,1000,1,1)
```

See Also

PRICE, PRICEDISC, PRICEMAT, YIELD, YIELDDISC, YIELDMAT

ACCRINTM

Description

Computes the accrued interest for a security that pays interest at maturity. Note that this function does not calculate compounded interest.

Syntax

ACCRINTM (*issue_date*, *maturity_date*, *rate* [, *par_value*] [, *calendar_type*])

Argument	Description
issue_date	The date the security begins earning interest. Dates in the argument list must be in the form of a serial number or text. Decimal values will be truncated to integers.
maturity_date	The date the security is repaid and interest quits accumulating. It must be later than *issue_date*. Dates in the argument list must be in the form of a serial number or text. Decimal values will be truncated to integers.
rate	The security's annual interest rate.
[*par_value*]	Optional. The face value of the security. If this argument is omitted, $1000 is used.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Equation

$$\frac{\text{par value} \times \text{rate} \times \text{accrued days}}{\text{days in year}}$$

Examples

This function returns 91.667

```
ACCRINTM("3/1/97","2/1/98",.1)
```

This function returns 183.333

```
ACCRINTM("3/1/97","2/1/98",.2)
```

See Also

ACCRINT

ACOS

Description

Returns the arc cosine of a number.

Syntax

ACOS (*number*)

Argument	Description
number	The cosine of the angle. The cosine can range from 1 to –1.

Remarks

The resulting angle is returned in radians (from 0 to PI). To convert the resulting radians to degrees, multiply the radians by 180/PI( ).

Examples

This function returns 1.05:

```
ACOS(.5)
```

This function returns 1.77:

```
ACOS(-.2)
```

See Also

COS

ACOSH

Description

Returns the inverse hyperbolic cosine of a number.

Syntax

ACOSH (*number*)

Argument	Description
number	Any number equal to or greater than 1.

Examples

This function returns .62:

```
ACOSH(1.2)
```

This function returns 1.76:

```
ACOSH (3)
```

See Also

ASINH, ATANH, COSH

ADDRESS

Description

Creates a cell address as text.

Syntax

ADDRESS (*row*, *column*, *ref_type* [, *a1*] [, *sheet*])

Argument	Description
row	The row number for the cell address.
column	The column number for the cell address.

Argument	Description
ref_type	The cell reference type. Following are the valid values for this argument. 1 Absolute 2 Absolute row, relative column 3 Relative row, absolute column 4 Relative
[*a1*]	Optional. The reference format. This argument must be TRUE( ) to represent an A1 reference format; Formula One does not support the R1C1 reference format.
[*sheet*]	Optional. The name of an external worksheet view control. If this argument is omitted, it is assumed that the reference exists in the current spreadsheet.

Examples

This function returns $F$5:

```
ADDRESS(5, 6, 1)
```

This function returns SALES!F5:

```
ADDRESS(5, 6, 4, TRUE(), SALES.)
```

See Also

COLUMN, OFFSET, ROW

AMORDEGRC

Description

Computes depreciation for the French accounting system, making use of a special French tax rule that allows over-depreciation of assets during the early years of ownership.

Syntax

AMORDEGRC (*cost, date, first_period, salvage, period, rate* [, *calendar_type*])

Argument	Description
cost	The cost of the asset.
date	The date the asset was purchased. Dates in the argument list must be entered as text.
first_period	The date marking the end of the first period of depreciation. Dates in the argument list must be entered as text.
salvage	The salvage value of the asset at the end of depreciation.
period	The number of the period you want to compute depreciation for. Enter 0 to compute depreciation for the period in which the asset was purchased.

Argument	Description
rate	The rate of depreciation. Because of the special rules applying to this function, it may either be a number between 0.25 and 0.333 or a number equal to or smaller than 0.2. Other values will return the #NUM! error.
[*calendar_type*]	Optional. One of four methods of counting days for computing interest. The options are: 0 30-day months, 360-day years, American method. 1 Actual months, actual years. 3 Actual months, 365-day years. 4 30-day months, 360-day years, European method. When *calendar_type* is 0 or 4, the DAYS360 function is used to calculate number of days. It has special rules for when the start and/or end dates fall on the 30th and/or 31st of the month. See “DAYS360” on page 86 for more information. If you omit this argument, 0 is used. If you enter any values other than 0, 1, 3, or 4, the function will return the #NUM! error. Decimal values will be truncated to integers. Note that while other functions that use the *calendar_type* argument allow an option 2, this one does not.

Remarks

To calculate normal depreciation under the French system, use AMORLINC.

Equations

The equations require a value for depreciation coefficient, which is different depending on the number of years in the life of the asset being depreciated. The life of asset value is the inverse of *rate*.

rate	**Life of asset**	**Depreciation coefficient**
0.25 - 0.333333	3 - 4	1.5
0.2 - 0.166666	5 - 6	2.0
<0.166666	>6	2.5

For period 0:
cost x *rate* x *depreciation coefficient* x YEARFRAC(*date, first_period, calendar_type*)

For periods 1 through life of asset - 1:
[*cost* - value from period 0] / 2 x *rate* x *depreciation coefficient*

For the last period:
[*cost* - value from period 0] x *rate* x *depreciation coefficient*

Examples

This function returns 58:

```
AMORDEGRC(1000,"5/5/95","7/1/95",100,0,0.15)
```

See Also

AMORLINC, DURATION, MDURATION

AMORLINC

Description Computes depreciation for the French accounting system.

Syntax AMORLINC (*cost, date, first_period, salvage, period, rate* [, *calendar_type*])

Argument	Description
cost	The cost of the asset.
date	The date the asset was purchased. Dates in the argument list must be entered as text.
first_period	The date marking the end of the first period of depreciation. Dates in the argument list must be entered as text.
salvage	The salvage value of the asset at the end of depreciation.
period	The number of the period you want to compute depreciation for. Enter 0 to compute depreciation for the period in which the asset was purchased.
rate	The rate of depreciation.
[*calendar_type*]	Optional. One of four methods of counting days for computing interest. The options are: 0 30-day months, 360-day years, American method. 1 Actual months, actual years. 3 Actual months, 365-day years. 4 30-day months, 360-day years, European method. When *calendar_type* is 0 or 4, the DAYS360 function is used to calculate number of days. It has special rules for when the start and/or end dates fall on the 30th and/or 31st of the month. See “DAYS360” on page 86 for more information. If you omit this argument, 0 is used. If you enter any values other than 0, 1, 3, or 4, the function will return the #NUM! error. Decimal values will be truncated to integers. Note that while other functions that use the *calendar_type* argument allow an option 2, this one does not.

Remarks To make use of a special French tax rule that allows over-depreciation of assets during the early years of ownership, use AMORDEGRC.

Equations For period 0: *cost* × *rate* × YEARFRAC(*date, first_period, calendar_type*)

For the remaining periods: *cost* × *rate*

In the final period(s), *cost* is adjusted so that it never goes below *salvage*.

Examples This function returns 23.333:

```
AMORLINC(1000,"5/5/95","7/1/95",100,0,0.15)
```

See Also AMORDEGRC, DURATION, MDURATION

AND

Description Returns True if all arguments are true; returns False if at least one argument is false.

Syntax AND (*logical_list*)

Argument	Description
logical_list	A list of conditions separated by commas. You can include as many as 30 conditions in the list. The list can contain logical values or a reference to a range containing logical values. Text and empty cells are ignored. If there are no logical values in the list, the error #VALUE! is returned.

Examples This function returns True because both arguments are true:

```
AND(1+1=2, 5+5=10)
```

This function returns False:

```
AND(TRUE(), FALSE())
```

See Also NOT, OR, ROW

AREAS

Description Counts the number of areas in a range reference.

Syntax AREAS (*range*)

Argument	Description
range	One or more range references or the name(s) of named range references. If you enter more than one range reference, enclose them in an extra set of parentheses. Use commas or spaces to separate different ranges: spaces will find the number of areas in which all of the ranges intersect, while commas will find the number of ranges entered, regardless of the intersections. You can use multiple and nested parentheses. Areas for ranges in the innermost parentheses are determined first.

Examples

This function returns 1:

```
AREAS(B2:C3)
```

This function returns 2:

```
AREAS((B2:D3,C3))
```

This function returns 1:

```
AREAS((B2:D3 C3))
```

ASIN

Description

Returns the arcsine of a number.

Syntax

ASIN (*number*)

Argument	Description
number	The sine of the resulting angle, ranging from –1 to 1.

Remarks

The resulting angle is returned in radians (ranging from -PI/2 to PI/2). To convert the resulting radians to degrees, multiply the radians by 180/PI().

Examples

This function returns 1.57:

```
ASIN(1)
```

This function returns .41:

```
ASIN(.4)
```

See Also

ASINH, PI, SIN

ASINH

Description

Returns the inverse hyperbolic sine of a number.

Syntax

ASINH (*number*)

Argument	Description
number	Any number.

Examples

This function returns 2.37:

```
ASINH(5.3)
```

This function returns –2.09:

```
ASINH(-4)
```

See Also ACOSH, ASIN, ATANH, SINH

ATAN

Description Returns the arctangent of a number.

Syntax ATAN (*number*)

Argument	Description
number	The tangent of the angle.

Remarks The resulting angle is returned in radians, ranging from -PI/2 to PI/2. To convert the resulting radians to degrees, multiply the radians by 180/PI( ).

Examples This function returns 1.29:

```
ATAN(3.5)
```

This function returns -1.33:

```
ATAN(4)
```

See Also ATAN2, ATANH, PI, TAN

ATAN2

Description Returns the arctangent of the specified coordinates.

Syntax ATAN2 (*x, y*)

Argument	Description
x	The x coordinate.
y	The y coordinate.

Remarks The arctangent is the angle from the x axis to a line with end points at the origin (0, 0) and a point with the given coordinates (*x*, *y*). The angle is returned in radians, ranging from –PI to PI, excluding –PI.

Examples This function returns 1.11:

```
ATAN2(3, 6)
```

This function returns 3.04:

```
ATAN2(-1, .1)
```

See Also ATAN, ATANH, PI, TAN

ATANH

Description Returns the inverse hyperbolic tangent of a number.

Syntax ATANH (*number*)

Argument	Description
number	A number between –1 and 1, excluding –1 and 1.

Examples This function returns .55:

```
ATANH(.5)
```

This function returns –.26:

```
ATANH(-.25)
```

See Also ACOS, ASINH, TANH

AVEDEV

Description Computes the average deviation of a list of numbers.

Syntax AVEDEV(*number_list*)

Argument	Description
number_list	A list of up to 30 arguments separated by commas. The list may contain numeric values, cell references, range references, or array constants. Text and logical values in range references and array constants are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Equation

$$\frac{1}{N}\sum_{j=1}^{N}|\chi_j - \bar{\chi}|$$

Examples This function returns 0.5:

```
AVEDEV(1,2)
```

See Also AVERAGE, COUNT, DEVSQ, KURT, SKEW

AVERAGE

Description Returns the average of the supplied numbers. The result of AVERAGE is also known as the arithmetic mean.

Syntax AVERAGE (*number_list*)

Argument	Description
number_list	A list of up to 30 numbers separated by commas. The list may contain numeric values, cell references, range references, or array constants. Text and logical values in range references and array constants are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Examples This function returns 8.25:

```
AVERAGE(5, 6, 8, 14)
```

This function returns 134, the average of the values in the range C15:C17:

```
AVERAGE(C15:C17)
```

See Also AVERAGEA, MIN, MAX

AVERAGEA

Description Computes the average of a list of numbers in a Lotus-compatible fashion. This function is equivalent to the AVERAGE function, but its implementation treats text and logical values in cell and range references differently.

Syntax AVERAGEA(*number_list*)

Argument	Description
number_list	A list of up to 30 arguments separated by commas. The list may contain numeric values, cell references, range references, or array constants. Text in cells referenced by this function is treated as the number 0 (this includes zero-length text). Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Text and logical values in arrays are ignored. Logical values referenced in cells or entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Examples

This function returns 1.5:

```
AVERAGEA(1,2)
```

See Also

AVERAGE, COUNTA

BESSELI

Description

Computes the value of the *n*th order modified Bessel function of the first kind evaluated at *x*.

Syntax

BESSELI (*number, order*)

Argument	Description
number	Any real number *x* at which to evaluate the function.
order	Any positive whole number or 0. Values are truncated to integers.

Equation

$$I_n(x) = (-i)^n J_n(ix)$$

Examples

This function returns 3.953370217:

```
BESSELI(3,1)
```

See Also

BESSELJ, BESSELK, BESSELY

BESSELJ

Description

Computes the value of the *n*th order Bessel function of the first kind evaluated at *x*.

Syntax

BESSELJ (*number, order*)

Argument	Description
number	Any real number *x* at which to evaluate the function.
order	Any positive whole number or 0. Values are truncated to integers.

Equations

$$J_v(z) = \left(\frac{1}{2}z\right)^v \sum_{k=0}^{\infty} \frac{\left(-\frac{1}{4}z^2\right)^k}{k!\Gamma(v+k+1)}$$

$$\text{where } \Gamma(n+k+1) = \int_0^{\infty} x^{n+k} e^{-x} dx$$

Examples

This function returns 0.497094103:

```
BESSELJ(2.5,1)
```

See Also

BESSELI, BESSELK, BESSELY

BESSELK

Description

Computes the value of the *n*th order Bessel function of the second kind evaluated at *x*.

Syntax

BESSELK (*number, order*)

Argument	Description
number	Any real number *x* at which to evaluate the function.
order	Any positive whole number or 0. Values are truncated to integers.

Equation

$$Kn(x) = \frac{\pi}{2} i^{n+1}[J_n(ix) + iY_n(ix)]$$

Examples

This function returns 9.758562803:

```
BESSELK(5,10)
```

This function returns 180713288.5:

```
BESSELK(1,10)
```

See Also

BESSELI, BESSELJ, BESSELY

BESSELY

Description

Computes the value of the *n*th order modified Bessel function of the second kind evaluated at *x*.

Syntax

BESSELY (*number, order*)

Argument	Description
number	Any real number *x* at which to evaluate the function.
order	Any positive whole number or 0. Values are truncated to integers.

Equation

$$Y_\nu(x) = \frac{J_\nu(x)\mathrm{COS}(\nu\pi) - J_{-\nu}(x)}{\mathrm{SIN}(\nu\pi)}$$

Examples

This function returns 0.376850015:

```
BESSELY(3,0)
```

This function returns -25.12910983:

```
BESSELY(5,10)
```

See Also

BESSELI, BESSELJ, BESSELK

BETADIST

Description

Computes the cumulative beta distribution.

Syntax

BETADIST (*x*, *alpha*, *beta* [, *A*] [, *B*])

Argument	Description
x	The value at which the function will be evaluated. It must be a number larger than *A* and smaller than *B*.
alpha	The parameter to the beta function. It must be a number greater than zero.
beta	A second parameter to the beta function. It must be a number greater than zero.
[*A*]	Optional. The lower bound to the interval of *x*. It must be a number smaller than *x* and *B*. If this argument is omitted, 0 is used.
[*B*]	Optional. The upper bound to the interval of *x*. It must be a number larger than *x* and *A*. If this argument is omitted, 1 is used.

Equation

$$\frac{1}{B(a,b)}\int_0^x t^{a-1}(1-t)^{b-1}dt \text{ where } B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

Examples

This function returns 0.999023437:

```
BETADIST(0.5, 1, 10)
```

This function returns 0.000976563:

```
BETADIST(0.5, 10, 1)
```

This function returns 1:

```
BETADIST(0.5, 1000, 2000)
```

This function returns 0.685470581:

```
BETADIST(2, 8, 10, 1, 3)
```

See Also

BINOMDIST, CHIDIST, COMBIN, CRITBINOM, EXPONDIST, GAMMADIST, NORMDIST, POISSON

BETAINV

Description

Computes the inverse of the BETADIST function.

Syntax

BETAINV (*probability, alpha, beta* [, *A*] [, *B*])

Argument	**Description**
probability	The probability, as obtained from the BETADIST function. It must be a number between zero and 1 that represents a probability.
alpha	The parameter to the beta function. It must be a number greater than zero.
beta	A second parameter to the beta function. It must be a number greater than zero.
[*A*]	Optional. The lower bound to the interval of *x*. It must be a number smaller than *x* and *B*. If this argument is omitted, 0 is used.
[*B*]	Optional. The upper bound to the interval of *x*. It must be a number larger than *x* and *A*. If this argument is omitted, 1 is used.

Examples

This function returns 2:

```
BETAINV(0.685470581, 8, 10, 1, 3)
```

This function returns 0.5:

```
BETAINV(0.685470581, 8, 10)
```

This function returns 0.5:

```
BETAINV(0.5, 1, 1)
```

See Also BETADIST

BIN2DEC

Description Converts a binary number (base 2) to a decimal number (base 10).

Syntax BIN2DEC (*binary_number)*

Argument	Description
binary_number	Any number expressed in the binary number system (represented by the digits 1 or 0). Data may be entered in text or numerical form. A maximum of 9 value bits (or digits) is allowed in representing positive values. To represent negative values, enter exactly 10 digits--the first digit must be a 1. Positive values may be zero-padded on the left up to a maximum of 10 digits.

Remarks Note that the result is a numeric value.

Examples This function returns 1:

```
BIN2DEC(0000000001)
```

This function returns 128:

```
BIN2DEC(10000000)
```

See Also BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

BIN2HEX

Description Converts a binary number (base 2) to a hexadecimal number (base 16).

Syntax BIN2HEX (*binary_number* [, *places*])

Argument	Description
binary_number	Any number expressed in the binary number system (represented by the digits 1 or 0). Data may be entered in text or numerical form. A maximum of 9 value bits (or digits) is allowed in representing positive values. To represent negative values, enter exactly 10 digits--the first digit must be a 1. Positive values may be zero-padded on the left up to a maximum of 10 digits.
places	Optional number of characters to use in the output. If *places* is omitted, the function returns the number of spaces required to display the result. The *places* argument can be used to 0-pad a positive number. If the number is negative, *places* is ignored and 10 characters are returned.

Remarks Note that the result of this function is a text string.

Examples This function returns 3F:

```
BIN2HEX(111111)
```

This function returns FF:

```
BIN2HEX(11111111)
```

See Also BIN2DEC, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

BIN2OCT

Description Converts a binary number (base 2) to a octal number (base 8).

Syntax BIN2OCT (*binary_number* [, *places*])

Argument	Description
binary_number	Any number expressed in the binary number system (represented by the digits 1 or 0). Data may be entered in text or numerical form. A maximum of 9 value bits (or digits) is allowed in representing positive values. To represent negative values, enter exactly 10 digits--the first digit must be a 1. Positive values may be zero-padded on the left up to a maximum of 10 digits.
places	Optional number of characters to use in the output. If *places* is omitted, the function returns the number of spaces required to display the result. The *places* argument can be used to 0-pad a positive number. If the number is negative, *places* is ignored and 10 characters are returned.

Remarks Note that the result of this function is a text string.

Examples This function returns 164:

```
BIN2OCT(1110100)
```

This function returns 00777:

```
BIN2OCT(111111111,5)
```

See Also BIN2DEC, BIN2HEX, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

BINOMDIST

Description Computes the binomial distribution, which is used to determine probabilities on repeated tests where each test is independent of every other test (that is, the probability is the same for each test).

Syntax BINOMDIST (*successes, trials, probability, cumulative*)

Argument	Description
successes	An integer describing the number of successes in the test. It must be a number larger than or equal to 0 and smaller than *trials*.

Argument	Description
trials	An integer describing the total number of attempts in the test. It must be a number larger than *successes* and smaller than the largest positive integer the machine can handle.
probability	A value between 0 and 1 that represents the chance of success on an individual attempt. For example, the probability of heads in a coin toss is considered to be 0.5; the probability of getting “snake eyes” (two ones) when throwing a pair of dice is 1/36 or 0.02778.
cumulative	A boolean value indicating the type of calculation you want. Enter FALSE to calculate the possibility of the success case (for example, getting exactly 3 heads in 4 coin tosses). Enter TRUE to calculate the sum of all cases from 0 to the success case (for example, for the probability of getting 0 or 1 or 2 or 3 heads in 4 coin tosses).

Remarks

Coin tosses or die rolls are examples of uses for BINOMDIST. For cases of sampling without replacement, use HYPGEOMDIST.

BINOMDIST is similar to NEGBINOMDIST, only in BINOMDIST the number of successes is variable and the number of trials is fixed, while in NEGBINOMDIST the number of successes is fixed and the number of trials is variable.

Equations

When *cumulative* is FALSE: $f(x) = \binom{n}{x} p^x q^{n-x}$

When *cumulative* is TRUE: $\sum_{j=k}^{n} \binom{n}{j} p^j (1-p)^{n-j} = I_p(k, n-k+1)$

Examples

The following examples return the probabilities of getting a 6 on a die roll.

This function calculates the possibility of getting a 6 exactly once in 6 trials. It returns 0.401878:

```
BINOMDIST(1,6,0.166666667,FALSE)
```

This function calculates the possibility of getting a 6 zero or 1 time in 6 trials. It returns 0.736776:

```
BINOMDIST(1,6,0.166666667,TRUE)
```

See Also

BETADIST, CHIDIST, COMBIN, CRITBINOM, EXPONDIST, GAMMADIST, NEGBINOMDIST, NORMSDIST, POISSON

CALL

Description

Returns the #N/A error message.

Syntax

CALL (*arguments*)

Argument	Description
arguments	Multiple arguments of varying types are allowed.

Remarks

This function was included in Formula One for Java only for compatibility with Microsoft Excel. In Excel, CALL is used to call procedures in dynamic linked libraries or code resources, neither of which is supported in Java.

This function is intended only for users or developers who want to import Excel worksheets that contain Excel’s CALL function. To make those worksheets work properly, developers should convert these calls to add-in functions, which they can write in Java and set up to be automatically loaded.

For more information, see the chapter on creating add-in functions in the *Formula One for Java Technical Guide*.

CELL

Description

Returns the specified type of information about a specified cell. If no cell is specified, the function will return information about the cell the function was entered in.

Syntax

CELL (*info_type* [, *cell*])

Argument	Description	
info_type	The type of information you want about *cell*. It must be text, surrounded by quotation marks. The following are the valid entries and the types of information they return, listed in alphabetical order:	
	“across”	Returns 1 if the cell is formatted for long text centered across several cells. Otherwise, it returns 0.
	“address”	Returns an absolute reference to *cell*.
	“backgroundcolor”	Returns the index number of the fill color of *cell*.
	“bold”	Returns 1 if the font in *cell* is bold. Otherwise, it returns 0.
	“bottomborder”	Returns a code corresponding to the type of the bottom border of *cell*. The codes are: 0 for none, 1 for thin, 2 for medium, 3 for dashed, 4 for dotted, 5 for thick, 6 for double, and 7 for hairline.

Argument	Description	
info_type (continued)	*"bottombordercolor"*	Returns the index number of the color of *cell*'s bottom border.
	"col"	Returns the column number of *cell*. The result is the same as the result of the COLUMN function.
	"color"	Returns #N/A.
	"contents"	Returns the contents of *cell*.
	"coord"	Returns an absolute reference to *cell*, including the book and sheet names if they are different from the cell containing this function.
	"datatype"	Returns a code corresponding to the type of data found in *cell*. The codes are: b for a blank cell; v for a number, a formula that returns a number, a formula that returns a logical value, or a formula that returns a date/time value; l for text or a formula that returns text; e for any error value except #N/A; and n for the #N/A error value.
	"filedate"	Returns #N/A.
	"filename"	Returns #N/A.
	"fontface"	Returns the name of the font in *cell*.
	"fontsize"	Returns the size of the font in *cell*, in points.
	"format"	Returns #N/A.
	"formulatype"	Returns a code corresponding to the type of data and/or formula found in *cell*. The codes are: b for a blank cell, v for a number, fv for a formula that returns a number, l for text, fl for a formula that returns text, e for any error value except #N/A, fe for a formula that returns any error value except #N/A, n for the #N/A error value, and fn for a formula that returns the #N/A error value.
	"halign"	Returns a code corresponding to *cell*'s horizontal alignment type. The codes are: 0 for general, 1 for left, 2 for centered, 3 for right, 4 for fill (the fill string is repeated to fill the width of *cell*), 5 for justified, and 6 for centered across cells.
	"height"	Returns the row height of *cell*'s row, in points.
	"italic"	Returns 1 if the font in *cell* is italic. Otherwise, it returns 0.
	"leftborder"	Returns a code corresponding to the type of the left border of *cell*. The codes are: 0 for none, 1 for thin, 2 for medium, 3 for dashed, 4 for dotted, 5 for thick, 6 for double, and 7 for hairline.
	"leftbordercolor"	Returns the index number of the color of *cell*'s left border.
	"orientation"	Returns #N/A.
	"parentheses"	Returns #N/A.

Argument	Description	
info_type (continued)	*"pattern"*	Returns the index number of *cell*'s fill pattern, or 0 if *cell* has no fill pattern.
	"patterncolor"	Returns the index number of the foreground color used in *cell*'s fill pattern.
	"prefix"	Returns a code corresponding to *cell*'s alignment. The codes are: single tic (') for left-justified text, quotation marks (") for right-justified text, caret (^) for centered text, and backslash (\) for fill-aligned text. Otherwise it returns empty text (" ").
	"protect"	Returns #N/A.
	"rightborder"	Returns a code corresponding to the type of the right border of *cell*. The codes are: 0 for none, 1 for thin, 2 for medium, 3 for dashed, 4 for dotted, 5 for thick, 6 for double, and 7 for hairline.
	"rightbordercolor"	Returns the index number of the color of *cell*'s right border.
	"rotation"	Returns #N/A.
	"row"	Returns the row number of *cell*.
	"sheet"	Returns a letter corresponding to the position of *cell*'s worksheet: A is the leftmost worksheet, B is the next worksheet, etc.
	"sheetname"	Returns the name of *cell*'s worksheet, if it is named; otherwise, returns a letter corresponding to the position of *cell*'s worksheet: A is the leftmost worksheet, B is the next worksheet, etc.
	"textcolor"	Returns the index number of *cell*'s text color. Note that number formats that display text in specific colors may override this setting.
	"topborder"	Returns a code corresponding to the type of the top border of *cell*. The codes are: 0 for none, 1 for thin, 2 for medium, 3 for dashed, 4 for dotted, 5 for thick, 6 for double, and 7 for hairline.
	"topbordercolor"	Returns the index number of the color of *cell*'s top border.
	"type"	Returns b if *cell* is empty, l if *cell* contains text, and v if *cell* contains anything other than text.
	"underline"	Returns 1 if the font in *cell* is underlined. Otherwise, it returns 0.
	"valign"	Returns a code corresponding to *cell*'s vertical alignment. The codes are: 0 for top, 1 for center, 2 for bottom.
	"width"	Returns the column width of *cell*, truncated to an integer. Each unit of column width is equal to the width of one character in the default font size.

Argument	Description	
info_type (continued)	*“wrap”*	Returns 1 if the text in *cell* wraps, 0 if it doesn’t.
cell	The cell you want information about. If you enter a range reference, CELL will return information about the top left cell in the range. If you omit this argument, the function will return information about the cell the function is entered in.	

Remarks

This function is included in Formula One for Java in order to be compatible with Excel, which in turn included it in order to be compatible with other worksheet formats, notably Lotus 1-2-3.

Developers will find it more efficient to extract this type of information using Java direct calls.

Example

This function returns $D$57:

```
CELL("address",D57)
```

See Also

INFO

CEILING

Description

Rounds a number up to the nearest multiple of a specified significance.

Syntax

CEILING (*number*, *significance*)

Argument	Description
number	The value to round.
significance	The multiple to which to round.

Remarks

Regardless of the sign of the number, the value is rounded up, away from zero. If *number* is an exact multiple of *significance*, no rounding occurs. If *number* or *significance* is non-numeric, the error #VALUE! is returned. When the arguments have opposite signs, the error #NUM! is returned.

Examples

This function returns 1.25:

```
CEILING(1.23459, .05)
```

This function returns 150:

```
CEILING(148.24, 2)
```

See Also

EVEN, FLOOR, INT, ODD, ROUND, TRUNC

CHAR

Description Returns a character that corresponds to the supplied Unicode value.

Syntax CHAR (*number*)

Argument	Description
number	A value between 1 and 65535 that specifies an Unicode character.

Remarks Numeric code and associated text are defined according to the operating system’s native character set. Since the Java platform is 100% Unicode, the characters returned by CHAR come from the Unicode character set regardless of the underlying platform.

In non-Java applications such as Formula One ActiveX and Microsoft Excel, CHAR often returns different results from those returned by Formula One for Java. Java applications consistently return results from Unicode, its native character set; non-Java applications may access and return results from character sets other than Unicode.

When CHAR is called with strings containing what is commonly called “plain ASCII” text (characters with values from 32 to 126), the results are the same in Formula One for Java, Fomula One ActiveX, and Microsoft Excel.

CHIDIST

Description Computes the “complementary” version of the chi-square distribution.

Syntax CHIDIST (*x*, *df*)

Argument	Description
x	The value at which the function will be evaluated. It must be larger than zero and smaller than 1e10.
df	An integer indicating the number of degrees of freedom. It must be larger than zero. Decimal values will be rounded down to the nearest integer.

Equation

$$\frac{1}{2^{v/2}\Gamma(v/2)}\int_{0}^{x^2} t^{(v)/(2-1)}e^{(-t)/2}dt = P(v/2, x^2/2)$$

Examples This function returns 0.476258754:

```
CHIDIST(9.6,10)
```

This function returns 0.951229425:

```
CHIDIST(0.1, 2)
```

See Also BETADIST, BINOMDIST, CHIINV, CHITEST, COMBIN, CRITBINOM, EXPONDIST, GAMMADIST, NORMSDIST, POISSON

CHIINV

Description Computes the inverse of the “complementary” chi-square distribution.

Syntax CHIINV (*probability, df*)

Argument	Description
probability	The probability, as obtained from the CHIDIST function. It must be a number between zero and 1 that represents a probability.
df	An integer indicating the number of degrees of freedom. It must be larger than zero and smaller than 1e10.

Examples This function returns 11.0705:

```
CHIINV(0.05,5)
```

See Also CHIDIST, CHITEST

CHITEST

Description Computes the probability of the distribution against expected values.

Syntax CHITEST (*actual, expected*)

Argument	Description
actual	The actual test values, in the form of a range reference or array constant.
expected	The expected test values, in the form of a range reference or array constant. *Actual* and *expected* must contain the same potential numbers of values. Text and logical values are ignored, along with the number they are paired up with. That is, when a number from *actual* is paired up with text from *expected*, the entire pair is ignored.

Remarks Before calculating the probability, CHITEST must determine the number of degrees of freedom (df), which is used to calculate the probability outcome. To calculate df, CHITEST uses the *actual* argument:

- When *actual* is an array constant, df is one less than the number of elements in the array.
- When *actual* is a range reference, df is the product of (rows - 1)(columns - 1).

You can see that empty cells (or cells with text or logical values) in the *actual* argument’s range reference will affect df, which in turn affects the outcome of CHITEST.

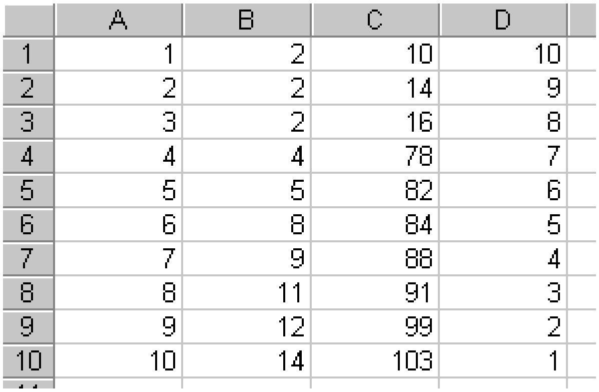
Equation

$$\chi^2 = \sum_i \frac{(N_i - n_i)^2}{n_i}$$

where N_i is the actual value and n_i is the expected value.

Examples

The following examples use this worksheet.



	A	B	C	D
1	1	2	10	10
2	2	2	14	9
3	3	2	16	8
4	4	4	78	7
5	5	5	82	6
6	6	8	84	5
7	7	9	88	4
8	8	11	91	3
9	9	12	99	2
10	10	14	103	1

This function returns 0.71:

```
CHITEST(B1:B10, A1:A10)
```

This function returns 0:

```
CHITEST(C1:C10, A1:A10)
```

See Also

FTEST, TTEST, ZTEST

CHOOSE

Description

Returns a value from a list of numbers based on the index number supplied.

Syntax

CHOOSE (*index*, *item_list*)

Argument	Description
index	A number that refers to an item in *item_list*.
item_list	A list of numbers, formulas, or text separated by commas. This argument can also be a range reference. You can specify as many as 29 items in the list.

Remarks

Index can be a cell reference; *index* can also be a formula that returns any value from 1 to 29. If *index* is less than 1 or greater than the number of items in *item_list*, #VALUE! is returned. If *index* is a fractional number, it is truncated to an integer.

Examples

This function returns Q2:

```
CHOOSE(2,"Q1", "Q2", "Q3", "Q4")
```

This function returns the average of the contents of range A1:A10:

```
AVERAGE(CHOOSE(1, A1:A10, B1:B10, C1:C10))
```

See Also

INDEX (non-array type)

CLEAN

Description

Removes all nonprintable characters from the supplied text.

Syntax

CLEAN (*text*)

Argument	Description
text	Any worksheet information.

Remarks

Text that is imported from another environment may require this function.

Examples

This function returns Payments Due because the character returned by CHAR (8) is nonprintable:

```
CLEAN("Payments " & CHAR(8) & "Due")
```

See Also

CHAR, TRIM

CODE

Description

Returns a Unicode value of the first character of the supplied string.

Syntax

CODE (*text*)

Argument	Description
text	Any string.

Remarks

Numeric code and associated text are defined according to the operating system’s native character set. Since the Java platform is 100% Unicode, the numbers returned by CODE come from the Unicode character set regardless of the underlying platform.

In non-Java applications such as Formula One ActiveX and Microsoft Excel, CODE will often return different results from those returned by Formula One for Java. Java applications consistently return results from Unicode, its native character set; non-Java applications may access and return results from character sets other than Unicode.

When CODE is called with strings containing what is commonly called “plain ASCII” text (characters with values from 32 to 126), the results will be the same in Formula One for Java, Fomula One ActiveX, and Microsoft Excel.

COLUMN

Description Returns the column number of the supplied reference.

Syntax COLUMN (*reference*)

Argument	Description
reference	A reference to a cell or range. Omitting the argument returns the number of the column in which COLUMN is placed.

Examples This function returns 2:

```
COLUMN(B3)
```

This function returns 4 if the function is entered in cell D2:

```
COLUMN()
```

See Also COLUMNS, ROW

COLUMNS

Description Returns the number of columns in a range reference.

Syntax COLUMNS (*range*)

Argument	Description
range	A reference to a range of cells.

Example This function returns 4:

```
COLUMNS(A1:D5)
```

See Also COLUMN, ROWS

COMBIN

Description

Computes the number of combinations possible by taking *k* items at a time from a pool of *n*, when order in the sample taken is not important.

Syntax

COMBIN (*number, chosen*)

Argument	Description
number	A positive integer representing the total number of items in the pool of items. Decimal numbers are rounded down to the next-lower integer. Text in the argument list is interpreted as numeric, if possible; otherwise, the function returns the #NAME? error. Logical values in the argument list are interpreted as 1 for TRUE and 0 for FALSE. Text and logical values in range references return the #VALUE! error.
chosen	A positive integer representing the number of items taken from the pool at a time. It must be less than *number*. Decimal values are rounded down to the next-lower integer. Text in the argument list is interpreted as numeric, if possible; otherwise, the function returns the #NAME? error. Logical values in the argument list are interpreted as 1 for TRUE and 0 for FALSE. Text and logical values in range references return the #VALUE! error.

Remarks

For an example of the use of this function, say we have a box containing four different items. The number of combinations possible taking out 2 items at a time is 6; that is, the six combinations of items 1-2, 1-3, 1-4, 2-3, 2-4, and 3-4.

This function is similar to the PERMUT function, except that PERMUT requires the samples to be ordered, while COMBIN takes samples in any order.

Equation

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\ldots(n-k+1)}{1 \times 2 \ldots k}$$

where *n* is the total number of items and *k* is the number of items taken at a time.

Examples

This function returns 6:

```
COMBIN(4,2)
```

This function returns 252:

```
COMBIN(10, 5)
```

See Also

GAMMALN, PERMUT

COMPLEX

Description

Creates a complex number. The result is text in the form “a+bi” or “a+bj”) where “a” and “b” are from the first two arguments.

Syntax

COMPLEX (*real_number, imaginary_number* [, *suffix*])

Argument	Description
real_number	Any number. The function returns this number as the real number “a” in the equation *z* = a + b*i*.
imaginary_ number	Any number. The function returns this number as the imaginary number “b” in the equation *z* = a + b*i*.
[*suffix*]	The letter respresenting the imaginary number. You may enter i or j. Uppercase versions of these characters will cause a #VALUE! error. If this argument is omitted, i is used. Any function that accepts two or more complex arguments must have the same suffix (i or j) on all arguments to that function, or a #VALUE error will result.

Examples

This function returns 2+3i:

```
COMPLEX(2,3)
```

This function returns 2+3j:

```
COMPLEX(2,3,"j")
```

See Also

IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

CONCATENATE

Description

Joins several text items into one item.

Syntax

CONCATENATE (*text1*, *text2*, ...)

Argument	Description
text1, *text2*, ...	Up to 30 text items to be joined into a single text item. The text items can be strings, numbers, or single-cell references.

Remarks

The “&” operator can be used instead of CONCATENATE to join text items.

Examples

The following example returns “Sale Price.” It is the same as typing “Sale”& “ ” & “Price”:

```
CONCATENATE ("Sale ", "Price")
```

Suppose in an inventory worksheet, C2 contains “extruder1”, C5 contains “gaskets”, and C8 contains the number 15. The following example returns “Inventory currently holds 15 gaskets for extruder1.”:

```
CONCATENATE ("Inventory currently holds ", C8, " ", C5," for ", C2)
```

See Also

COLUMN, ROWS

CONFIDENCE

Description

Computes the confidence interval, assuming a normal distribution. The confidence interval is the portion of a value that appears after the +/-, for example, 10 +/- 2.1.

Syntax

CONFIDENCE (α , σ , *n*)

Argument	Description
α	A number between zero and one indicating confidence level. For example, 0.05 represents a 95% confidence level. The relation of confidence level to α is: Confidence level (as a percent) = 100 * (1 - α *).*
σ	A number representing the population standard deviation for the range. It must be larger than zero.
n	An integer larger than 1 indicating the sample size.

Equation

$$K = c \times \frac{\sigma}{\sqrt{n}} \text{ where } c = \mathrm{NORMSINV}\left(1 - \frac{\alpha}{2}\right)$$

Examples

This function returns 0.69295089:

```
CONFIDENCE(0.05, 2.5, 50)
```

See Also

NORMDIST, NORMINV, NORMSDIST, NORMSINV, STANDARDIZE

CONVERT

Description

Converts a numeric value from one set of units to another.

Syntax

CONVERT (*number, from_units, to_units*)

Argument	Description
number	The value to be converted, in terms of *from_units*.

Argument	Description
from_units	String identifying the units of *number*. The lists of acceptable unit abbreviations are shown below. May have optional prefix from prefix list below. Case is important.
to_units	String identifying units for *number* after conversion. The lists of acceptable unit abbreviations are shown below. May have optional prefix from prefix list below. Case is important.

Remarks

Basic values are taken from the NIST *Guide for the Use of the International System of Units (SI)*, available on the Internet at http://physics.nist.gov/cuu/Units/index.html.

For the US method of distance measurement, the “international foot” is used. One “international foot” = 0.999998 “survey foot.”

Length measurements are based on the “international” system where 1 inch = 2.540 centimeters. The international system replaces the older “survey” system where 1.0 meter = 39.370 inches. The difference amounts to about 3 millimeters per mile.

Microsoft Excel uses the term “pica” as a unit of length where “computer point” is actually meant. “Computer point” also differs from a “printer’s point,” which is approximately 1/72 inch. According to international standards, a “pica” is a unit of length that equals 1/6 inch.

Year is based on nominal average year of 365.25 days (1 leap year every 4), not the true solar year (which is slightly shorter). To maintain Excel compatibility, related conversions involving years are factored based on the nominal average year.

Units tables

These tables show the abbreviations you can use in the *from_units* and *to_units* arguments.

Weight and mass abbreviations	
Gram	g
Slug	sg
Pound mass (avoirdupois)	lbm
U (atomic mass unit)	u
Ounce mass (avoirdupois)	ozm

Distance abbreviations	
Meter	m
Statute mile	mi
Nautical Mile	Nmi
Inch	in
Foot	ft
Yard	yd
Angstrom	ang
Pica (1/72 in.)	Pica

Time abbreviations	
Year	yr
Day	day
Hour	hr
Minute	mn
Second	sec

Pressure abbreviations	
Pascal	Pa
Atmosphere	atm
mm of Mercury	mmHg

Force abbreviations	
Newton	N
Dyne	dyn
Pound force	lbf

Energy abbreviations	
Joule	J
Erg	e
Thermodynamic calorie	c
IT calorie	cal
Electron volt	eV
Horsepower-hour	HPh
Watt-hour	Wh
Foot-pound	flb
BTU	BTU

Power abbreviations	
Horsepower	HP
Watt	W

Magnetism abbreviations	
Tesla	T
Gauss	ga

Temperature abbreviations

Degree Celsius	C
Degree Fahrenheit	F
Degree Kelvin	K

Liquid measure abbreviations

Teaspoon	tsp
Tablespoon	tbs
Fluid ounce	oz
Cup	cup
Pint	pt
Quart	qt
Gallon	gal
Liter	l

Prefix character abbreviations

Exa-	E
Peta-	P
Tera-	T
Giga-	G
Mega-	M
Kilo-	k
Hecto-	h
Dekao-	e
Deci-	d
Centi-	c
Milli-	m
Micro-	u
Nano-	n
Pico-	p
Femto-	f
Atto-	a

Examples

This function returns1.093513298:

```
CONVERT(1,"m","yd")
```

This function returns 0.621371192237334:

```
CONVERT(1,"km","mi")
```

CORREL

Description

Computes the correlation coefficient for two sets of numbers, paired up one-to-one.

Note The CORREL function is exactly the same as the PEARSON function. We provide both in order to be compatible with all the Microsoft Excel functions.

Syntax

CORREL (*array1, array2*)

Argument	Description
array1 and *array2*	Two range references or array constants containing numeric values. *Array1* and *array2* must contain the same potential number of values. The function will return the error value #DIV/0! if *array1* or *array2* contains non-numeric data (text, logical values, or blank cells).

Remarks

The correlation coefficient is a number between -1 and 1 (inclusive) that measures the “relatedness” of the numbers in the samples. A coefficient of 1 indicates a direct relationship in which all points are linearly related on a line with positive slope. A coefficient of -1 indicates an inverse relationship in which a large value in the first argument pairs with a small value in the second argument. A coefficient of 0 indicates no relationship between the pairs of values, or complete randomness.

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

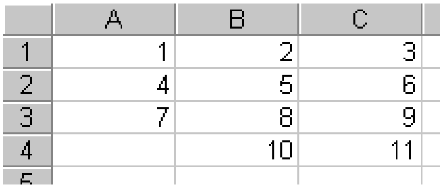
Equation

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_i - \bar{y})^2}}$$

where $\bar{x}$ is the mean of the x_i ‘s and $\bar{y}$ is the mean of they_i ’s.

Examples

The following examples use this worksheet.



	A	B	C
1	1	2	3
2	4	5	6
3	7	8	9
4		10	11
5			

This function returns 1:

```
CORREL(A1:A3,B1:B3)
```

This function returns 0.991117:

```
CORREL(A1:B2,C1:C4)
```

See Also

COVAR, PEARSON

COS

Description

Returns the cosine of an angle.

Syntax

COS (*number*)

Argument	Description
number	The angle in radians. If the angle is in degrees, convert the angle to radians by multiplying the angle by PI()/180.

Examples

This function returns .126:

```
COS(1.444)
```

This function returns .28:

```
COS(5)
```

See Also

ACOS, ASINH, ATANH, COSH, PI

COSH

Description

Returns the hyperbolic cosine of a number.

Syntax

COSH (*number*)

Argument	Description
number	Any number.

Examples

This function returns 4.14:

```
COSH(2.10)
```

This function returns 1.03:

```
COSH(.24)
```

See Also

ASINH, ATANH, COS

COUNT

Description

Returns the number of values in the supplied list.

Syntax

COUNT (*value_list*)

Argument	Description
value_list	A list of values. The list can contain as many as 30 values.

Remarks

COUNT only numerates numbers or numerical values such as logical values, dates, or text representations of dates. If you supply a range, only numbers and numerical values in the range are counted. Empty cells, logical values, text, and error values in the range are ignored.

Examples

This function returns 2:

```
COUNT(5, 6, "Q2")
```

This function returns 3:

```
COUNT("03/06/94", "06/21/94", "10/19/94")
```

See Also

AVERAGE, COUNTA, SUM

COUNTA

Description Returns the number of nonblank values in the supplied list.

Syntax COUNTA (*expression_list*)

Argument	Description
expression_list	A list of expressions. As many as 30 expressions can be included in the list.

Remarks COUNTA returns the number of cells that contain data in a range. Null values (“ ”) are counted, but references to empty cells are ignored.

Examples This function returns 4:

```
COUNTA(32, 45, "Earnings", "")
```

This function returns 0 when the specified range contains empty cells:

```
COUNTA(C38:C40)
```

See Also AVERAGE, COUNT, PRODUCT, SUM

COUNTBLANK

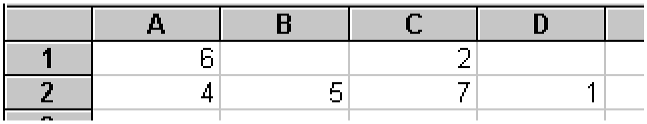
Description Counts the number of empty cells in a specified range.

Syntax COUNTBLANK (*range*)

Argument	Description
range	A range reference or the name of a named range reference.

Remarks COUNTBLANK will *not* count cells containing hidden zero values, spaces, and other entries that make the cell appear empty. This function counts all cells that COUNTA does not count.

Examples The following examples use the worksheet below.



	A	B	C	D
1	6		2	
2	4	5	7	1

This function returns 2:

```
COUNTBLANK(A1:D2)
```

See Also COUNT, COUNTA

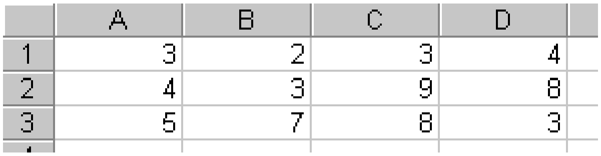
COUNTIF

Description Returns the number of cells within a range that meet the given criteria.

Syntax COUNTIF (*range*, *criteria*)

Argument	**Description**
range	Range of cells you want to count.
criteria	Number, expression, or text that defines which cells are counted.

Example The following example uses this worksheet.



	A	B	C	D
1	3	2	3	4
2	4	3	9	8
3	5	7	8	3

This function returns 4:

```
COUNTIF(A1:D3, 3)
```

See Also AVERAGE, COUNTA, SUM, SUMIF

COUPDAYBS

Description Computes the number of days from the coupon date previous to the settlement date and the settlement date.

Syntax COUPDAYBS (*settlement*, *maturity*, *frequency* [, *calendar_type*])

Argument	**Description**
settlement	The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Dates in the argument list must be in the form of a serial number or text.
frequency	The number of interest payments per year. See “The frequency Argument” on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Remarks This function truncates all arguments to integers.

Examples

This function returns 145:

```
COUPDAYBS("1/25/93","8/31/94",2,0)
```

This function returns 147:

```
COUPDAYBS("1/25/93","8/31/94",2,1)
```

See Also

COUPDAYS, COUPDAYSNC, COUPNCD, COUPNUM, COUPPCD

COUPDAYS

Description

Computes the number of days in the coupon period containing the settlement date.

Syntax

COUPDAYS (*settlement, maturity, frequency* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Dates in the argument list must be in the form of a serial number or text.
frequency	The number of interest payments per year. See “The frequency Argument” on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Remarks

This function truncates all arguments to integers.

Examples

This function returns 180:

```
COUPDAYS("1/25/93","8/31/94",2,0)
```

This function returns 181:

```
COUPDAYS("1/25/93","8/31/94",2,1)
```

See Also

COUPDAYBS, COUPDAYSNC, COUPNCD, COUPNUM, COUPPCD

COUPDAYSNC

Description

Computes the number of days from the settlement date to the next coupon date.

Syntax

COUPDAYSNC (*settlement, maturity, frequency* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Dates in the argument list must be in the form of a serial number or text.
frequency	The number of interest payments per year. See “The frequency Argument” on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Remarks

This function truncates all arguments to integers.

Examples

This function returns 35:

```
COUPDAYSNC("1/25/93","8/31/94",2,0)
```

This function returns 34:

```
COUPDAYSNC("1/25/93","8/31/94",2,1)
```

See Also

COUPDAYBS, COUPDAYS, COUPNCD, COUPNUM, COUPPCD

COUPNCD

Description

Computes the coupon date that follows the settlement date. If the coupon date falls on the settlement date, this function will display the next coupon date.

Syntax

COUPNCD (*settlement*, *maturity, frequency* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Dates in the argument list must be in the form of a serial number or text.
frequency	The number of interest payments per year. See “The frequency Argument” on page 18 for more information.

Argument	Description
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Remarks

This function displays the date as a serial number unless the cell is formatted to display dates. This function truncates all arguments to integers.

Examples

This function returns 2/28/93:

```
COUPNCD("1/25/93","8/31/94",2,0)
```

This function returns 8/31/93:

```
COUPNCD("1/25/93","8/31/94",1,0)
```

See Also

COUPDAYBS, COUPDAYS, COUPDAYSNC, COUPNUM, COUPPCD

COUPNUM

Description

Computes the number of coupons between settlement and maturity. Maturity counts as a coupon date.

Syntax

COUPNUM (*settlement*, *maturity*, *frequency* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Dates in the argument list must be in the form of a serial number or text.
frequency	The number of interest payments per year. See “The frequency Argument” on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Remarks

This function rounds results up to the nearest integer, so if it finds 3 coupon periods plus 10 days, the result will be 4. This function truncates all arguments to integers.

Examples

This function returns 4:

```
COUPNUM("1/25/93","8/31/94",2,0)
```

This function returns 7:

```
COUPNUM("1/25/93","8/31/94",4,0)
```

See Also COUPDAYBS, COUPDAYS, COUPDAYSNC, COUPNCD, COUPPCD

COUPPCD

Description Computes the last coupon date before the settlement date. If the coupon date falls on the settlement date, this function will display the settlement date.

Syntax COUPPCD (*settlement*, *maturity*, *frequency* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Dates in the argument list must be in the form of a serial number or text.
frequency	The number of interest payments per year. See “The frequency Argument” on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Remarks This function displays the date as a serial number unless the cell is formatted to display dates. This function truncates all arguments to integers.

Examples This function returns 8/31/92:

```
COUPPCD("1/25/93","8/31/94",2,0)
```

This function returns 11/30/92:

```
COUPPCD("1/25/93","8/31/94",4,0)
```

See Also COUPDAYBS, COUPDAYS, COUPDAYSNC, COUPNCD, COUPNUM

COVAR

Description Computes the covariance for pairwise numbers in two arrays.

Syntax COVAR (*arrayX*, *arrayY*)

Argument	Description
arrayX and *arrayY*	Two range references or array constants containing numeric values. *ArrayX* and *arrayY* must contain the same potential number of values. Text, logical values, and blank cells are ignored, along with the number they are paired up with. That is, when a number from *arrayX* is paired up with text from *arrayY*, the entire pair is ignored.

Remarks

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Equation

$$\text{COVAR(x,y)} = \frac{1}{n}\sum_{j=1}^{n}(x_j - \bar{x})(y_j - \bar{y})$$

Please note that different statistics methods divide by *n* or by *n* - 1. We chose to divide by *n* in order to be consistent with Microsoft Excel.

Examples

This function returns 4:

```
COVAR({1,2,3,4,5},{2,4,6,8,10})
```

See Also

CORREL

CRITBINOM

Description

Computes the smallest number of successes that will result in the cumulative BINOMDIST function giving an answer larger than a specified criteria value.

Syntax

CRITBINOM (*trials, probability, alpha*)

Argument	Description
trials	An integer larger than 0 describing the total number of attempts in the test.
probability	A value between 0 and 1 that represents the chance of success on an individual attempt. For example, the probability of heads in a coin toss is considered to be 0.6; the probability of getting “snake eyes” (two ones) when throwing a pair of dice is 1/36.
alpha	A value between 0 and 1 that represents the overall criteria value. When the result of CRITBINOM is put into BINOMDIST, the value obtained from BINOMDIST will be equal to or larger than *alpha*.

Examples

This function returns 492:

```
CRITBINOM(1000,0.5,0.3)
```

This function returns 500:

```
CRITBINOM(1000,0.5,0.5)
```

See Also BINOMDIST, COMBIN

CUMIPMT

Description Computes the cumulative interest for the specified period.

Syntax CUMIPMT(*rate, n_periods, PV, start_period, end_period, type*)

Argument	Description
rate	The interest rate. The *rate* should be adjusted to reflect the period length. For example, if you have a mortgage with an annual rate of 9%, but pay monthly, the appropriate rate is .0075 (.09/12 months).
n_periods	The total number of payment periods. Decimal values will be truncated to integers
PV	The present value or principal amount of the loan.
start_period	The first period for accumulating interest; payment periods start with 1. Decimal values will be truncated to integers.
end_period	The last period of the calculation. If the calculation is based on a 30 year mortgage paid monthly, the maximum number would be 360. Decimal values will be truncated to integers.
type	Indicates whether payment is due at the beginning or end of the period. Enter 0 for the end of the period (typical for most consumer loans and mortgages) or 1 for the beginning of the period.

Remarks This function is useful for calculating the anticipated interest on a home mortgage in a calendar year for tax purposes.

The result (payment to interest) displays as a negative number because it reflects “cash flow out” in Present Value nomenclature.

Examples This function returns -5177.27:

```
CUMIPMT(0.009166667,60,17000,1,60,0)
```

This function returns -309.71:

```
CUMIPMT(0.009166667,60,17000,1,2,0)
```

See Also FV, CUMPRINC, IPMT, NPER, PMT, PPMT, PV, RATE

CUMPRINC

Description Computes the cumulative principal paid for the specified period.

Syntax CUMPRINC (*rate, n_periods, PV, start_period, end_period, type*)

Argument	Description
rate	The interest rate. The *rate* should be adjusted to reflect the period length. For example, if you have a mortgage with an annual rate of 9%, but pay monthly, the appropriate rate is .0075 (.09/12 months).
n_periods	The total number of payment periods. Decimal values will be truncated to integers
PV	The present value or principal amount of the loan.
start_period	The first period for accumulating interest; payment periods start with 1. Decimal values will be truncated to integers.
end_period	The last period of the calculation. If the calculation is based on a 30 year mortgage paid monthly, the maximum number would be 360. Decimal values will be truncated to integers.
type	Indicates whether payment is due at the beginning or end of the period. Enter 0 for the end of the period (typical for most consumer loans and mortgages) or 1 for the beginning of the period.

Remarks The result (payment for principal) displays as a negative number because it reflects "cash flow out" in Present Value nomenclature.

Examples This function returns -8483.764676:

```
CUMIPMT(0.009166667,60,17000,1,34,0)
```

See Also FV, CUMIPMT, IPMT, NPER, PMT, PPMT, PV, RATE

DATE

Description Returns the serial number of the supplied date.

Syntax DATE (*year, month, day*)

Argument	Description
year	A number from 1900 to 2078. If *year* is between 1920 to 2019, you can specify two digits to represent the year; otherwise specify all four digits.

Argument	Description
month	A number representing the month (for example, 12 represents December). If a number greater than 12 is supplied, the number is added to the first month of the specified year.
day	A number representing the day of the month. If the number you specify for *day* exceeds the number of days in that month, the number is added to the first day of the specified month.

Examples

This function returns 34506:

```
DATE(94, 6, 21)
```

This function returns 36225:

```
DATE(99, 3, 6)
```

See Also

DATEVALUE, DAY, MONTH, NOW, TIMEVALUE, TODAY, YEAR

DATEDIF

Description

Calculates the number of days, months, or years between two specified dates.

Syntax

DATEDIF (*start*, *end*, *unit*)

Argument	Description
start	The start date. Dates in the argument list must be entered as text, enclosed in quotation marks.
end	The end date. It must be after *start*. Dates in the argument list must be entered as text, enclosed in quotation marks.
unit	A letter representing the time units you want the returned value in. The options are:

Y	The number of complete years between *start* and *end*.
M	The number of complete months between *start* and *end*.
D	The number of days between *start* and *end*.
MD	The difference between the days in *start* and *end*, ignoring the months and years of the dates.
YM	The difference between the months in *start* and *end*, ignoring the days and years of the dates.
YD	The difference between the years in *start* and *end*, ignoring the days and months of the dates.

Examples

This function returns 247:

```
DATEDIF("3/12/96","11/14/96,"d")
```

This function returns 2:

```
DATEDIF("3/12/96","11/14/96,"md")
```

See Also DAY, DAYS360, EDATE, EOMONTH, MONTH, NETWORKDAYS, WORKDAY, YEAR

DATEVALUE

Description Returns the serial number of a date supplied as a text string.

Syntax DATEVALUE (*text*)

Argument	Description
text	A date in text format between January 1, 1900, and December 31, 2078. If you omit the year, the current year is used.

Examples This function returns 34399:

```
DATEVALUE("3/6/94")
```

This function returns 35058:

```
DATEVALUE("12/25/95")
```

See Also NOW, TIMEVALUE, TODAY

DAVERAGE

Description Uses specified criteria to select records from a database, then computes the average of the numeric values in a specified field of the selected records.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax DAVERAGE (*database, field, criteria*)

Argument	Description
database	A reference to a range containing data that the function searches.
field	The column within *database* that contains the data you want to average.
criteria	A reference to a range containing search criteria.

Examples

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 20.33:

```
DAVERAGE(A1:D8,"Sales 96",B10:B11)
```

See Also

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

DAY

Description

Returns the day of the month that corresponds to the date represented by the supplied number.

Syntax

DAY (*serial_number*)

Argument	Description
serial_number	A date represented as a serial number or as text (for example, 06-21-94 or 21-Jun-94).

Examples

This function returns 6:

```
DAY(34399)
```

This function returns 21:

```
DAY("06-21-94")
```

See Also

NOW, HOUR, MINUTE, MONTH, SECOND, TODAY, WEEKDAY, YEAR

DAYS360

Description

Returns the number of days between two dates based on a 360-day year (twelve 30-day months). Use this function to help compute payments if your accounting system is based on twelve 30-day months.

Syntax

DAYS360 (*start_date, end_date* [, *method*])

Argument	Description
start_date, end_date	The two dates between which you want to know the number of days.

Argument	Description
[*method*]	Optional. A logical value that specifies whether the European or US method should be used in the calculation. If False (or omitted), the US (NASD) method is used. If True, the European method is used. The default is based on the local translation. It should be correct for your location.

Remarks

Start_date and *end_date* can be text strings using numbers to represent the month, day, and year (for example, "1/30/93" or "1-30-93"), or they can be serial numbers representing the dates.

If *start_date* occurs after *end_date*, DAYS360 returns a negative number.

If *method* is set to False and *start_date* is the 31st of a month, it becomes equal to the 30th of the same month. If *end_date* is the 31st of a month and *start_date* is less than the 30th of a month, the ending date becomes equal to the 1st of the next month, otherwise the ending date becomes equal to the 30th of the same month.

If *method* is set to True, *start_dates* or *end_dates* which occur on the 31st of a month become equal to the 30th of the same month.

Note To determine the number of days between two dates in a normal year, you can use normal subtraction. For example, "12/31/93"-"1/1/93" equals 364.

Example

This function returns 1:

```
DAYS360("1/30/93", "2/1/93")
```

DB

Description

Returns the real depreciation of an asset for a specific period of time using the fixed-declining balance method.

Syntax

DB (*cost*, *salvage*, *life*, *period* [, *months*])

Argument	Description
cost	The initial cost of the asset.
salvage	The salvage value of the asset.
life	The number of periods in the useful life of the asset.
period	The period for which to calculate the depreciation. The time units used to determine *period* and *life* must match.
[*months*]	Optional. The number of months in the first year of the item's life. If this argument is omitted, 12 is used.

Example

This function returns 1451.52:

```
DB(10000, 1000, 7, 3)
```

See Also

DDB, SLN, SYD, VDB

DCOUNT

Description

Uses specified criteria to select records from a database, then counts the number of selected records. Can also count the number of selected records that have a number in the specified field.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax

DCOUNT (*database,* [*field*,] *criteria*)

Argument	Description
database	A reference to a range containing data that the function searches.
[*field*]	Optional. The column within *database* that contains the data you want to count. The function will only count the selected records that have a number in *field*. If this argument is omitted, the function will count all selected records.
criteria	A reference to a range containing search criteria.

Examples

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 3:

```
DCOUNT(A1:D8,"Sales 96",B10:B11)
```

This function returns 0:

```
DCOUNT(A1:D8,"Salesperson",B10:B11)
```

See Also

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

DCOUNTA

Description

Uses specified criteria to select records from a database, then counts the number of selected records. Can also count the number of selected records for which a specified field is not blank.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax

DCOUNTA (*database*, [*field*,] *criteria*)

Argument	Description
database	A reference to a range containing data that the function searches.
[*field*]	Optional. The column within *database* that contains the data you want to count. The function will only count the selected records that have a number in *field*. If this argument is omitted, the function will count all selected records.
criteria	A reference to a range containing search criteria.

Examples

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 3:

```
DCOUNTA(A1:D8,"Salesperson",B10:B11)
```

See Also

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

DDB

Description

Returns the depreciation of an asset for a specific period of time using the double-declining balance method or a declining balance factor you supply.

Syntax

DDB (*cost*, *salvage*, *life*, *period* [, *factor*])

Argument	Description
cost	The initial cost of the asset.
salvage	The salvage value of the asset.
life	The number of periods in the useful life of the asset.
period	The period for which to calculate the depreciation. The time units used to determine *period* and *life* must match.

Argument	Description
[*factor*]	Optional. The rate at which the balance declines. Of this argument is omitted, a default factor of 2 (the double-declining balance factor) is used.

Remarks

The double-declining balance method uses an accelerated rate where the highest depreciation occurs in the first period, decreasing in successive periods.

All arguments for this function must be positive numbers.

Example

This function returns 1457.73:

```
DDB(10000,1000, 7, 3)
```

See Also

DB, SLN, SYD, VDB

DEC2BIN

Description

Converts a decimal number (base 10) to a binary number (base 2).

Syntax

DEC2BIN (*integer* [, *places*])

Argument	Description
integer	Any negative or positive whole number or zero. For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between -512 and 511 may be used.
places	Optional number of characters to use in the output. If *places* is omitted, the function returns the number of spaces required to display the result. The *places* argument can be used to 0-pad a positive number. If the number is negative, *places* is ignored and 10 characters are returned.

Remarks

The result of this function is a text string.

Examples

This function returns 100000000:

```
DEC2BIN(256)
```

This function returns 1000000000:

```
DEC2BIN(-512)
```

This function returns 01010

```
DEC2BIN(10,5)
```

See Also BIN2DEC, BIN2HEX, BIN2OCT, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

DEC2HEX

Description Converts a decimal number (base 10) to an hexadecimal number (base 16).

Syntax DEC2HEX (*integer* [, *places*])

Argument	Description
integer	Any negative or positive whole number or zero. For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between -549,755,813,888 to 549,755,813,887 may be used.
places	Optional number of characters to use in the output. If *places* is omitted, the function returns the number of spaces required to display the result. The *places* argument can be used to 0-pad a positive number. If the number is negative, *places* is ignored and 10 characters are returned.

Remarks The result of this function is a text string.

Examples This function returns 0000A:

```
DEC2HEX(10,5)
```

This function returns FFFFFFFFF6:

```
DEC2HEX(-10,5)
```

See Also BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

DEC2OCT

Description Converts a decimal number (base 10) to an octal number (base 8).

Syntax DEC2OCT (*integer* [, *places*])

Argument	Description
integer	Any negative or positive whole number or zero. For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between -536,870,912 to 536,870,911 may be used.
places	Optional number of characters to use in the output. If *places* is omitted, the function returns the number of spaces required to display the result. The *places* argument can be used to 0-pad a positive number. If the number is negative, *places* is ignored and 10 characters are returned.

Remarks Note that the result of this function is a text string.

Examples This function returns 144:

```
DEC2OCT(100)
```

This function returns 7777777634:

```
DEC2OCT(-100)
```

This function returns 007:

```
DEC2OCT(7,3)
```

See Also BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

DEGREES

Description Converts a value in radians to degrees.

Syntax DEGREES (*radians*)

Argument	Description
radians	A number in radians that you want to convert to degrees.

Equation

$$radians\left(\frac{180}{\pi}\right)$$

Examples This function returns 360:

```
DEGREES(6.283185307)
```

This function returns 90:

```
DEGREES(1.570796327)
```

See Also PI, RADIANS

DELTA

Description

Compares two specified values and returns 1 if they are equal, 0 if they are not.

Syntax

DELTA (*number1* [, *number2*])

Argument	**Description**
number1	A number you want to compare.
[*number2*]	Optional. This is the number you want to compare to *number1*. If you omit this argument, the function will compare *number1* to 0.

Remarks

This function can be used in place of the equals (=) operator when a numeric rather than logical result is needed (for example, when using aggregate functions such as SUM and COUNT).

This function is sometimes referred to as the Kronecker Delta function.

Examples

This function returns 0:

```
DELTA(6,7)
```

This function returns 1:

```
DELTA(0)
```

See Also

IF, GESTEP

DEVSQ

Description

Computes the square of the deviation of the numbers.

Syntax

DEVSQ (*number_list*)

Argument	**Description**
number_list	A list of up to 30 numbers separated by commas. The list may contain numeric values, cell references, range references, or array constants. Text and logical values in cell references and arrays are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Equation

$\sum_i (x_i - \bar{x})^2$ where $\bar{x}$ is the mean of the x_i 's.

Examples

This function returns 0.5:

```
DEVSQ(1, 2)
```

See Also

AVEDEV, AVERAGE, COUNT, SKEW, KURT

DGET

Description

Uses specified criteria to select a single record from a database, then displays the data found in a specified field of the selected record.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax

DGET (*database, field, criteria*)

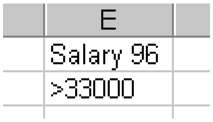
Argument	Description
database	A reference to a range containing data that the function searches.
field	The column within *database* that contains the data you want to display.
criteria	A reference to a range containing search criteria.

Remarks

If the function finds 0 records that match the criteria, it will display the #VALUE! error. If the function finds more than one record that match the criteria, it will display the #NUM! error.

Examples

This example uses the example database shown in “Database Functions” on page 14 and the criteria range below.



	E	
	Salary 96	
	>33000	

This function returns 18940:

```
DGET(A1:D8,"Commissions 96",E1:E2)
```

See Also

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

DISC

Description

Computes the discounted rate for a security.

Syntax

DISC (*settlement, maturity, price, redemption* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text. Decimal values are truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Dates in the argument list must be in the form of a serial number or text. Decimal values are truncated to integers.
price	The amount paid per $100 face value.
redemption	The security's redemption value per $100 face value. This is the amount paid at *maturity* that is not part of any final coupon payment.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Remarks

DISC understates the equivalent yield normally quoted on CDs and coupon bonds. See YIELDDISC for the yield rate comparison.

DISC, not YIELDDISC, is the complementary function to PRICEDISC.

Equation

$$\frac{(\mathit{redemption} - \mathit{price})}{\mathit{redemption} * \text{YEARFRAC}(\mathit{settlement}, \mathit{maturity}, \mathit{calendar\_type})}$$

Examples

This function returns 0.02024:

```
DISC("7/15/92","12/30/95",93,100)
```

See Also

PRICE, PRICEDISC, PRICEMAT, YIELD, YIELDDISC, YIELDMAT

DMAX

Description

Uses specified criteria to select records from a database, then displays the highest number value found in a specified field of the selected records.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see "Database Functions" on page 14.

Syntax

DMAX (*database, field, criteria*)

Argument	Description
database	A reference to a range containing data that the function searches.
field	The column within *database* that contains the maximum value you want to find.
criteria	A reference to a range containing search criteria.

Remarks

If there are no numeric values in *field* in the selected records, DMAX will display 0.

Examples

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 32460:

```
DMAX(A1:D8,"Commissions 96",B10:B11)
```

See Also

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

DMIN

Description

Uses specified criteria to select records from a database, then displays the lowest number value found in a specified field of the selected records.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax

DMIN (*database, field, criteria*)

Argument	Description
database	A reference to a range containing data that the function searches.
field	The column within *database* that contains the minimum value you want to find.
criteria	A reference to a range containing search criteria.

Remarks

If there are no numeric values in *field* in the selected records, DMIN will display 0.

Examples

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 10320:

```
DMIN(A1:D8,"Commissions 96",B10:B11)
```

See Also

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

DOLLAR

Description

Returns the specified number as text, using the local currency format and the supplied precision.

Syntax

DOLLAR (*number* [, *precision*])

Argument	Description
number	A number, a formula that evaluates to a number, or a reference to a cell that contains a number.
[*precision*]	Optional. A value representing the number of decimal places to the right of the decimal point. If this argument is omitted, 2 is used.

Note “Local” currency refers to the currency format for the current system.

Remarks

Dollar will return the specified number format as text using currency format for the current system. If you wish to always convert to the US Dollar format, regardless of the language of your system, then use the USDOLLAR worksheet function.

US Example

When using a US setting in Windows, this function returns $1023.79:

```
DOLLAR(1023.789)
```

This function returns $500:

```
DOLLAR(495.301, -2)
```

UK Example

When using a British setting in Windows, this function returns £1023.8:

```
DOLLAR(1023.789)
```

This function returns £500:

```
DOLLAR(495.301, -2)
```

German Example

When using a German setting in Windows, this function returns 1023,8 DM

```
DOLLAR(1023.789)
```

This function returns 500 DM:

```
DOLLAR(495.301, -2)
```

See Also

FIXED, TEXT, VALUE, USDOLLAR

DOLLARDE

Description Converts a dollar figure from fractional form to decimal form.

Syntax DOLLARDE (*dollar*, *denominator*)

Argument	Description
dollar	A dollar amount expressed as a fraction in the following manner: The digits to the left of the decimal represent the number of whole dollars. The digits to the right of the decimal point represent the numerator of the fractional portion of the dollar
denominator	The denominator of the fractional portion of the dollar.

Remarks This function is useful for converting dollar amounts involved in securities dealings, where dollar amounts are often expressed in fractions.

Examples This function returns 25.75:

```
DOLLARDE(25.3,4)
```

See Also DOLLARFR

DOLLARFR

Description Converts a dollar figure from decimal form to fractional form.

Syntax DOLLARFR (*dollar, denominator*)

Argument	Description
dollar	A dollar amount expressed in the normal decimal form.
denominator	The denominator you want for the fractional portion of the dollar. Decimal values are truncated to integers.

Remarks This function is useful for converting dollar amounts involved in securities dealings, where dollar amounts are often expressed in fractions.

Examples This function returns 25.1:

```
DOLLARFR(25.25,4)
```

See Also DOLLARDE

DPRODUCT

Description

Uses specified criteria to select records from a database, then multiplies the numeric values in a specified field of the selected records.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax

DPRODUCT (*database, field, criteria*)

Argument	Description
database	A reference to a range containing data that the function searches.
field	The column within *database* that contains the data you want to multiply.
criteria	A reference to a range containing search criteria.

Remarks

If there are no numeric values in *field* in the selected records, DPRODUCT will display 0.

Examples

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 8160:

```
DPRODUCT(A1:D8,"Sales 96",B10:B11)
```

See Also

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

DSTDEV

Description

Uses specified criteria to select records from a database, then computes the sample standard deviation of the numeric values in a specified field of the selected records.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax

DSTDEV (*database, field, criteria*)

Argument	Description
database	A reference to a range containing data that the function searches.

Argument	Description
field	The column within *database* that contains the data you want to find the standard deviation of.
criteria	A reference to a range containing search criteria.

Remarks At least two values are required to compute sample standard deviation, so if there are fewer than two numeric values in *field* in the selected records, DSTDEV will display the #DIV/0! error.

Examples This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 3.5119:

```
DSTDEV(A1:D8,"Sales 96",B10:B11)
```

See Also DAVERAGE, DCOUNT, DCOUNTA, DGET, DMAX, DMIN, DPRODUCT, DSTDEVP, DSUM, DVAR, DVARP

DSTDEVP

Description Uses specified criteria to select records from a database, then computes the population standard deviation of the numeric values in a specified field of the selected records.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax DSTDEVP (*database, field, criteria*)

Argument	Description
database	A reference to a range containing data that the function searches.
field	The column within *database* that contains the data you want to find the standard deviation of.
criteria	A reference to a range containing search criteria.

Remarks At least one value is required to compute population standard deviation, so if there are fewer than one numeric values in *field* in the selected records, DSTDEVP will display the #DIV/0! error.

Examples This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 2.8674:

```
DSTDEVP(A1:D8,"Sales 96",B10:B11)
```

See Also

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

DSUM

Description

Uses specified criteria to select records from a database, then adds the numeric values in a specified field of the selected records.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax

DSUM (*database, field, criteria*)

Argument	Description
database	A reference to a range containing data that the function searches.
field	The column within *database* that contains the data you want to add.
criteria	A reference to a range containing search criteria.

Remarks

If there are no numeric values in *field* in the selected records, DSUM will display 0.

Examples

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 73490:

```
DSUM(A1:D8,"Commissions 96",B10:B11)
```

See Also

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

DURATION

Description

Computes the Macaulay duration for a security, in years.

Syntax

DURATION (*settlement, maturity, rate, yield, frequency* [, *calendar_type*]

Argument	Description
settlement	The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text. Numbers will be truncated to integers.

Argument	Description
maturity	The date the security expires and the remaining amount is paid to the investor. Dates in the argument list must be in the form of a serial number or text. Numbers will be truncated to integers.
rate	The security’s annual coupon rate.
yield	The security’s annual yield.
frequency	The number of interest payments per year. See “The frequency Argument” on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Equation

$$\frac{\dfrac{\left(\mathrm{N}-1+\dfrac{\mathrm{DSC}}{\mathrm{E}}\right)\times 100}{\left(1+\dfrac{yield}{freq}\right)^{\left(\mathrm{N}-1+\dfrac{\mathrm{DSC}}{\mathrm{E}}\right)}}+\displaystyle\sum_{k\,=\,1}^{\mathrm{N}}\left(\left(\dfrac{100\times rate}{freq\times\left(1+\dfrac{yield}{freq}\right)^{\left(k-1+\dfrac{\mathrm{DSC}}{\mathrm{E}}\right)}}\right)\times\left(k-1+\dfrac{\mathrm{DSC}}{\mathrm{E}}\right)\right)}{\dfrac{100}{\left(1+\dfrac{yield}{freq}\right)^{\left(\mathrm{N}-1+\dfrac{\mathrm{DSC}}{\mathrm{E}}\right)}}+\displaystyle\sum_{k\,=\,1}^{\mathrm{N}}\left(\dfrac{100\times rate}{freq\times\left(1+\dfrac{yield}{freq}\right)^{\left(k-1+\dfrac{\mathrm{DSC}}{\mathrm{E}}\right)}}\right)}\times\frac{1}{freq}$$

...where the codes correspond to values you can compute using other functions, as shown in the following table.

Code	Meaning	Function
DSC	Number of days from *settlement* to the next coupon period.	COUPDAYSNC
E	Number of days in the coupon period.	COUPDAYS
N	Number of coupons payable between *settlement* and *maturity.*	COUPNUM

Examples

This function returns 7.24649:

```
DURATION("3/17/89","3/17/99",0.07,0.08,2,1)
```

See Also

MDURATION

DVAR

Description

Uses specified criteria to select records from a database, then computes the sample variance of the numeric values in a specified field of the selected records.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax

DVAR (*database, field, criteria*)

Argument	Description
database	A reference to a range containing data that the function searches.
field	The column within *database* that contains the data you want to find the variance of.
criteria	A reference to a range containing search criteria.

Remarks

At least two values are required to compute sample variance, so if there are fewer than two numeric values in *field* in the selected records, DVAR will display the #DIV/0! error.

Examples

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 12.333:

```
DVAR(A1:D8,"Sales 96",B10:B11)
```

See Also

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

DVARP

Description

Uses specified criteria to select records from a database, then computes the population variance of the numeric values in a specified field of the selected records.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax

DVARP (*database, field, criteria*)

Argument	Description
database	A reference to a range containing data that the function searches.
field	The column within *database* that contains the data you want to find the variance of.
criteria	A reference to a range containing search criteria.

Remarks

At least one value is required to compute population variance, so if there are fewer than one numeric values in *field* in the selected records, DVARP will display the #DIV/0! error.

Examples

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 8.2222:

```
DVARP(A1:D8,"Sales 96",B10:B11)
```

See Also

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

EDATE

Description

Finds a date a specified number of months before or after a given date. This is useful for computing dates associated with securities.

Syntax

EDATE (*start_date, months*)

Argument	Description
start_date	Any date. Dates in the argument list must be in the form of a serial number or text.
months	The number of months after or before *start_date*. Enter a negative number for months before, a positive number for months after. Decimal values are truncated to integers.

Examples

This function returns 4/15/99:

```
EDATE("2/15/99",2)
```

See Also

EOMONTH

EFFECT

Description

Computes the effective annual interest rate, which adjusts the nominal rate to show the effect of compounding.

Syntax

EFFECT (*nominal_rate, periods*)

Argument	Description
nominal_rate	The nominally quoted interest rate.
periods	The number of compounding periods per year. Decimal values are truncated to integers.

Equation

$$\left(\frac{1 + nominal\_rate}{periods}\right)^{periods} - 1$$

Examples

This function returns 0.0696:

```
EFFECT(0.0675,12)
```

This function returns 0.0686:

```
EFFECT(0.0675,2)
```

See Also

NOMINAL

EOMONTH

Description

Finds the last date in the month that is a specified number of months before or after a given date. This is useful for computing dates associated with securities.

Syntax

EOMONTH (*start_date, months*)

Argument	**Description**
start_date	Any date. Dates in the argument list must be in the form of a serial number or text.
months	The number of months after or before *start_date*. Enter a negative number for months before, a positive number for months after. Decimal values are truncated to integers.

Examples

This function returns 4/30/99:

```
EOMONTH("2/15/99",2)
```

See Also

EDATE

ERF

Description

Computes the “error” function.

Syntax

ERF (*lower_limit* [, *upper_limit*])

Argument	**Description**
lower_limit	The lower limit of the values across which you want to evaluate the function. Must be a real number.
upper_limit	The optional upper limit of the values across which you want to evaluate the function. Must be a real number. Omitting the upper limit calculates the error function between 0 and the lower limit.

Remarks

Called the “error” function because it follows the standard distribution of “errors” around the mean (the “bell-shaped curve”).

This function forms the basis of “normal” distribution functions.

Equation

$$\mathrm{ERF}(a, b) = \frac{2}{\pi}\int_{a}^{b} e^{-t^{2}}\,dt$$

Examples

This function returns 0.995322265:

```
ERF(2)
```

This function returns 0.004655645:

```
ERF(2,3)
```

See Also

ERFC, NORMDIST, NORMSDIST

ERFC

Description

Computes the complementary “error” function.

Syntax

ERFC (*x*)

Argument	Description
x	The value at which the function will be evaluated.

Remarks

Used to compute errors in larger cases of *x*, where the ERF function would lose digits while approaching 1 as a limit.

Equation

$$\mathrm{ERFC}(a) = \frac{2}{\pi}\int_{a}^{\infty} e^{-t^{2}}\,dt = 1 - \mathrm{ERFC}(a)$$

Examples

This function returns 0.157272:

```
ERFC(1)
```

See Also

ERF, NORMDIST, NORMSDIST

ERROR.TYPE

Description

Returns a number corresponding to an error.

Syntax

ERROR.TYPE (*error_ref*)

Argument	Description
error_ref	A cell reference.

Remarks

The following error text or numbers can be returned by this function.

Number	Description
1	#NULL!
2	#DIV/0!
3	#VALUE!
4	#REF!
5	#NAME?
6	#NUM!
7	#N/A
#N/A	Other

Example

This function returns 2 if the formula in cell A1 attempts to divide by zero:

```
ERROR.TYPE(A1)
```

See Also

ISERR, ISERROR

EVEN

Description

Rounds the specified number up to the nearest even integer.

Syntax

EVEN (*number*)

Argument	Description
number	Any number, a formula that evaluates to a number, or a reference to a cell that contains a number.

Examples

This function returns 4:

```
EVEN(2.5)
```

This function returns 2032:

```
EVEN(2030.45)
```

See Also

CEILING, FLOOR, INT, ODD, ROUND, TRUNC

EXACT

Description

Compares two expressions for identical, case-sensitive matches. True is returned if the expressions are identical; False is returned if they are not.

Syntax

EXACT (*expression1*, *expression2*)

Argument	Description
expression1	Any text.
expression2	Any text.

Examples

This function returns True:

```
EXACT("Match", "Match")
```

This function returns False:

```
EXACT("Match", "match")
```

See Also

LEN, SEARCH

EXP

Description

Returns e raised to the specified power. The constant e is 2.71828182845904 (the base of the natural logarithm).

Syntax

EXP (*number*)

Argument	Description
number	Any number.

Examples

This function returns 12.18:

```
EXP(2.5)
```

This function returns 20.09:

```
EXP(3)
```

See Also

LINEST, LOG

EXPONDIST

Description

Computes the exponential distribution, which can be used to determine the amount of time between random events.

Syntax

EXPONDIST (*x*, *lambda*, *cumulative*)

Argument	Description
x	The value at which the function will be evaluated. It must be larger than 0.
lambda	A parameter value for distribution that determines the shape of the curve. The value must be larger than 0.
cumulative	A logical value that determines whether EXPONDIST returns a singular or cumulative value. Use False to calculate the distribution for the *x* value only. Use True to calculate the cumulative distribution.

Equation

When *cumulative* is FALSE: $\lambda e^{-\lambda x}$

When *cumulative* is TRUE: $1 - e^{-\lambda x}$

Examples

This function returns 0.6065306597:

```
EXPONDIST(0.5,1,FALSE)
```

This function returns 0.3934693403:

```
EXPONDIST(0.5,1,TRUE)
```

See Also

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, GAMMADIST, NORMDIST, POISSON

FACT

Description

Returns the factorial of a specified number.

Syntax

FACT (*number*)

Argument	Description
number	Any non-negative integer. If you supply a real number, FACT truncates the number to an integer before calculation.

Examples

This function returns 2:

```
FACT(2.5)
```

This function returns 720:

```
FACT(6)
```

See Also

FACT, FACTDOUBLE, PRODUCT

FACTDOUBLE

Description Returns the double-factorial of a specified number.

Syntax FACTDOUBLE (*number*)

Argument	Description
number	Any non-negative integer. If you supply a real number, FACTDOUBLE truncates the number to an integer before calculation.

Remarks Double-factorials differ from factorials in that the decrement value (the amount of decrease in the variable) is 2 instead of one, denoted by the double exclamation point: $6!! = 6 \bullet 4 \bullet 2 = 48$ and $5!! = 5 \bullet 3 \bullet 1 = 15$.

Examples This function returns 2027025:

```
FACTDOUBLE(15)
```

This function returns 3840:

```
FACTDOUBLE(10)
```

See Also FACT, FACTDOUBLE, PRODUCT

FALSE

Description Returns the logical value False. This function always requires the trailing parentheses.

Syntax FALSE ( )

See Also TRUE

FDIST

Description Computes the complementary F-distribution, which is used to compare data from two different populations.

Syntax FDIST (*X*, *df1*, *df2*)

Argument	Description
X	The value at which the function will be evaluated. It must be larger than or equal to 0.

Argument	Description
df1	A positive integer indicating degrees of freedom in the numerator. Decimal values will be rounded down to the nearest integer.
df2	A positive integer indicating degrees of freedom in the denominator. Decimal values will be rounded down to the nearest integer.

Equation

$I_w(d_1/2, d_2/2)$ where $w = \dfrac{d_1 X}{d_2 + d_1 X}$ and *I* refers to the beta distribution.

Examples

This function returns 0.256387198:

```
FDIST(2, 3, 4)
```

This function returns 0.047619048:

```
FDIST(20, 2, 2)
```

See Also

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FINV, FTEST, GAMMADIST, NORMDIST, POISSON

FIND

Description

Searches for a string of text within another text string and returns the character position at which the search string first occurs.

Syntax

FIND (*search_text, text* [, *start_position*])

Argument	Description
search_text	The text to find. If you specify an empty string (“”), FIND matches the first character in text.
text	The text to be searched.
[*start_position*]	Optional. The character position in *text* where the search begins. The first character in *text* is character number 1. If this argument is omitted, 1 is used.

Remarks

FIND is case-sensitive. You cannot use wildcard characters in the *search_text*.

Examples

This function returns 12:

```
FIND("time", "There’s no time like the present")
```

This function returns 19:

```
FIND("4", "Aisle 4, Part 123-4-11", 9)
```

See Also

EXACT, LEN, MID, SEARCH

FINV

Description

Computes the inverse of the complementary F-distribution (that is, the inverse of the FDIST function.) It is used when comparing data from two different populations.

Syntax

FINV (*probability, df1, df2*)

Argument	Description
probability	The probability, as obtained from the FDIST function. It must be a number between 0 and 1 that represents a probability.
df1	A positive integer indicating degrees of freedom in the numerator. Decimal values will be rounded down to the nearest integer.
df2	A positive integer indicating degrees of freedom in the denominator. Decimal values will be rounded down to the nearest integer.

Remarks

This function can be used to compute the critical value needed to test the “null hypothesis” in many statistical experiments. For example, if there is to be less than a 1% chance that results are due to random effects for two populations, you can run FINV with 0.01 as the *probability* argument and the number of degrees of freedom you choose. Values achieved in the experiment must be higher than the resultant critical value in order to be significant.

Examples

This function returns 7.71:

```
FINV(0.05, 1, 4)
```

This function returns 9.01:

```
FINV(0.05, 5, 3)
```

See Also

FDIST, FTEST, TDIST, TINV, TTEST

FISHER

Description

Computes Fisher’s z-transformation. Note that this is a two-tailed test.

Syntax

FISHER (*x*)

Argument	Description
x	The correlation coefficient value for which you want the transformation. It must be a number between -1 and 1.

Equation

$$z = \frac{1}{2} ln\left(\frac{1+x}{1-x}\right)$$

Examples

This function returns 0.549306144:

```
FISHER (0.5)
```

This function returns -0.255412812:

```
FISHER (-0.25)
```

See Also

CORREL, FISHERINV, FTEST, PEARSON

FISHERINV

Description

Computes the inverse of Fisher's z-transformation.

Syntax

FISHERINV (*z*)

Argument	Description
z	A value from FISHER for which you want the inverse transformation.

Examples

This function returns .9999999:

```
FISHERINV(10)
```

This function returns 0.244918662:

```
FISHERINV(0.25)
```

See Also

CORREL, FISHER, FTEST, PEARSON

FIXED

Description

Rounds a number to the supplied precision, formats the number in decimal format, and returns the result as text.

Syntax

FIXED (*number* [, *precision*][, *no_commas*])

Argument	Description
number	Any number.
[*precision*]	Optional. The number of digits that appear to the right of the decimal place. If you specify negative precision, *number* is rounded to the left of the decimal point. You can specify a precision as great as 127 digits. If this argument is omitted, 2 is used.
[*no_commas*]	Optional. Determines if thousands separators (commas) are used in the result. Use 1 to exclude commas in the result. Use 0 to include thousands separators (for example, 1,000.00). If the argument is omitted, 0 is used.

Examples This function returns 2,000.500:

```
FIXED(2000.5, 3)
```

This function returns 2010:

```
FIXED(2009.5, -1, 1)
```

See Also DOLLAR, ROUND, TEXT, VALUE

FLOOR

Description Rounds a number down to the nearest multiple of a specified significance.

Syntax FLOOR (*number, significance*)

Argument	Description
number	The value to round.
significance	The multiple to which to round.

Remarks Regardless of the sign of the *number*, the value is rounded towards zero. If *number* is an exact multiple of *significance*, no rounding occurs.

If *number* or *significance* is non-numeric, #NAME? is returned. When the arguments have opposite signs, #NUM! is returned.

Examples This function returns 1.2:

```
FLOOR(1.23459, .05)
```

This function returns –148:

```
FLOOR(-148.24, -2)
```

See Also CEILING, EVEN, INT, ODD, ROUND, TRUNC

FORECAST

Description Constructs the least squares regression line through the data given, then computes the predicted *y* value for the requested *x*.

Syntax FORECAST (*x, arrayY, arrayX*)

Argument	Description
x	The *x* value at which you are requesting the corresponding *y* value for this regression.

Argument	Description
arrayY and *arrayX*	A range reference or an array constant containing numeric values. *ArrayX* and *arrayY* must contain the same potential number of values. Text, logical values, and empty cells referenced by this function are ignored, along with the number they are paired up with. That is, when a number from *arrayX* is paired up with text from *arrayY*, the entire pair is ignored.

Remarks

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Equations

FORECAST uses the following formulas for slope and intercept to draw the line used to generate the forecast of *y* at a given *x*.

$$\text{slope } b = \frac{n\sum xy - \left(\sum x\right)\left(\sum y\right)}{n\sum x^2 - \left(\sum x\right)^2}$$

$$\text{intercept } a = \frac{\sum y}{n} - b\left(\frac{\sum y}{n}\right)$$

where *n* is the number of members in *arrayY* and *arrayX*.

Examples

This function returns 1:

```
FORECAST(0.5, {1, 2, 4, 6, 7, 9}, {0, 2, 4, 5, 7, 8})
```

See Also

INTERCEPT, SLOPE

FREQUENCY

Description

Distributes the specified array of numbers, sorted in ascending order, into specified bins.

Note This is an array function. For information, see “Array Functions” on page 13.

Syntax

FREQUENCY (*array, bins*)

Argument	Description
array	A range reference or the name of a named range reference.

Argument	Description
bins	A range reference or the name of a named range reference. Empty cells in the range are ignored. A repeated number will cause 0 to appear in the bin for the second occurrence of the value. It’s generally best for the values in *bins* to be in ascending order. If not, the results appear in an order that may be confusing. See Remarks and Examples.

Results range

Before entering this function, select a results range one column wide with one more row than the number of entries in *bins*. If you select too few cells, some cells will be truncated. If you select too many cells, the extra cells will contain the #N/A error.

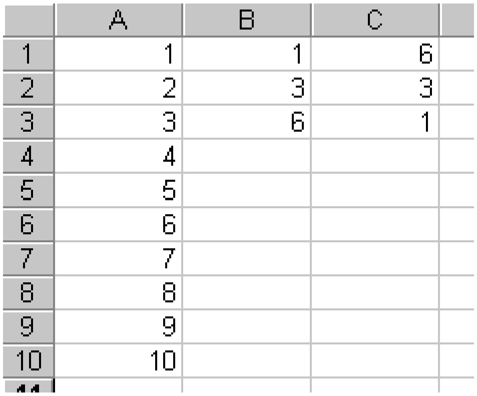
Remarks

When the values in *bins* are in ascending order, FREQUENCY sorts *array* in ascending order. It then calculates the number of members of *array* that are smaller than or equal to the smallest value in *bins* and places that value in the first cell of the results range. It then calculates the number of members of *array* that are larger than the smallest value in *bins* but smaller than or equal to the next larger member of *bins*, and places that value in the second cell of the results range. The last cell of the results range will contain the number of members of *array* that are larger than all the values in *bins*.

When the values in *bins* are not in ascending order, FREQUENCY determines the number to appear in each bin in the same manner. However, it displays the numbers in the order that their bins were entered. The last cell of the results range will always contain the number of members of *array* that are larger than all the values in *bins*. See examples.

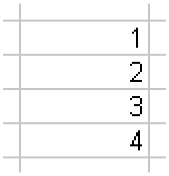
Examples

The examples use the following worksheet.



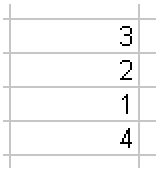
	A	B	C
1	1	1	6
2	2	3	3
3	3	6	1
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	10		

`FREQUENCY(A1:A10,B1:B3)` returns the following range:



1
2
3
4

`FREQUENCY(A1:A10,C1:C3)` returns the following range:



3
2
1
4

FREQUENCY computed the same results in the two examples, but displayed them in a different order in the results range. Also note that the last value is the same regardless of the order of the values in *bins*.

See Also PROB

FTEST

Description Computes an F-test probability for two specified distributions.

Syntax FTEST (*array1*, *array2*)

Argument	Description
array1	A set of at least 2 values. It must be a range reference or array constant containing numeric values. Text, logical values, and blank cells referenced by this function are ignored.
array2	A set of at least 2 values. It must be a range reference or array constant containing numeric values. Text, logical values, and blank cells referenced by this function are ignored. The number of values in *array2* may be different than in *array1*.

Remarks To arrive at a result, this function first computes the variances of the two distributions, then an F ratio based on the two variances. It also computes the degrees of freedom by adding the counts of each sample and subtracting 1 from each count. Finally, it uses the FDIST function internally, with F ratio and degrees of freedom as arguments, to calculate the end result.

If the variance of *array1* or *array2* is 0, this function will return the #DIV/0! error.

Examples

This function returns 0.167:

```
FTEST({51,45,41,27},{91,37,89,82})
```

See Also

CHITEST, TTEST, ZTEST

FV

Description

Returns the future value of an annuity based on regular payments and a fixed interest rate.

Syntax

FV (*interest*, *nper*, *payment* [, *pv*] [, *type*])

Argument	**Description**
interest	The fixed interest rate.
nper	The number of payments in an annuity.
payment	The fixed payment made each period.
[*pv*]	Optional. The present value, or the lump sum amount, the annuity is currently worth. If this argument is omitted, 0 is used.
[*type*]	Optional. Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. If you omit this argument, 0 is used.

Remarks

The units used for *interest* must match those used for *nper.* For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for *interest* and 5*12 for *nper.*

Cash paid out, such as a payment, is shown as a negative number. Cash received, such as a dividend check, is shown as a positive number.

Examples

This function returns 4,774.55:

```
FV(5%, 8, -500)
```

This function returns 531,550.86:

```
FV(10%/12, 240, -700, 1)
```

See Also

CUMIPMT, CUMPRINC, FVSCHEDULE, IPMT, NPER, PMT, PPMT, PV, RATE

FVSCHEDULE

Description

Computes the future value of an investment after applying a series of compounded interest rates.

Syntax

FVSCHEDULE (*value*, *schedule*)

Argument	Description
value	The starting value of the investment (the principal).
schedule	A list of interest rates for the compounding periods. This may be a range reference or an array constant. The number of values in *schedule* is the number of periods for compounding interest. An empty cell or 0 value will indicate no interest earned in that period.

Examples

This function returns 1367.52:

```
FVSCHEDULE(1000,{0.2,0.21,0.22})
```

See Also

FV

GAMMADIST

Description Computes the gamma distribution for a given value.

Syntax GAMMADIST (*x, alpha, beta, cumulative*)

Argument	Description	
x	The value at which you want to evaluate the function. It must be greater than or equal to 0.	
alpha	The alpha parameter. It must be greater than 0.	
beta	The beta parameter. It must be greater than 0.	
cumulative	A logical value indicating the type of calculation.	
	TRUE	Computes the cumulative area from 0 to *x*.
	FALSE	Computes the value at *x*.

Equations

$$f(x;\alpha,\beta) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)}\int_0^x x^{\alpha-1}e^{-x/\beta}\,dx$$ ...when *cumulative* is TRUE.

$$f(x;\alpha,\beta) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)}x^{\alpha-1}e^{-x/\beta}$$ ...when *cumulative* is FALSE.

Examples This function returns 0.247:

```
GAMMADIST (12, 3, 7, TRUE)
```

This function returns 0.038:

```
GAMMADIST (12, 3, 7, FALSE)
```

See Also BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, GAMMAINV, NORMDIST, POISSON

GAMMAINV

Description Computes the inverse of the cumulative gamma distribution. This is the inverse of the GAMMADIST function when *cumulative* is TRUE.

Syntax GAMMAINV (*probability, alpha, beta*)

Argument	Description
probability	A number between 0 and 1 indicating the probability.
alpha	The alpha parameter. It must be greater than 0.
beta	The beta parameter. It must be greater than 0.

Examples

This function returns 5.812:

```
GAMMAINV (0.01, 8, 2)
```

This function returns 7.962:

```
GAMMAINV (0.05, 8, 2)
```

See Also

GAMMADIST

GAMMALN

Description

Computes the logarithm of the gamma function.

Syntax

GAMMALN (*number*)

Argument	Description
number	A positive number. The function will return the #VALUE! error if *number* is nonnumeric and the #NUM! error if *number* is negative.

Remarks

The gamma function is a generalization of the factorial function. For positive integers, the factorial of *n* is the same as the gamma function of *n* + 1. The gamma function is a building block used in worksheet functions like COMBIN and PERMUT.

The inverse of the GAMMALN function is the EXP function with the GAMMALN function as the argument.

Equation

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt$$

where *z* is the argument.

As noted above, when z is an integer, the equation can be simplified to $n! = \Gamma(n+1)$.

Examples

This function returns 3.17805383

```
GAMMALN(5)
```

See Also

COMBIN, FACT, PERMUT

GCD

Description

Computes the greatest common divisor of a specified set of values. This number will divide all the numbers in the set evenly, with no fraction or remainder.

Syntax

GCD (*number_list*)

Argument	Description
number_list	A list of as many as 29 arguments in the form of numbers, cell references, range references, and array constants. The values in the arguments must be positive integers. Decimal values will be truncated to integers. Non-numeric arguments will return errors.

Remarks This function uses Euclid's algorithm to find the greatest common divisor.

Examples This function returns 3:

```
GCD({0,3,33,333})
```

This function returns 30:

```
GCD({1234567890,3000})
```

See Also LCM

GEOMEAN

Description Computes the geometric mean of a list of positive numbers.

Syntax GEOMEAN (*number_list*)

Argument	Description
number_list	A list of between 1 and 30 arguments consisting of numbers, cell references, range references, and/or array constants. The arguments must evaluate to positive numbers or 0. Text and logical values in arrays or cells referenced by this function is ignored. Empty cells are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Remarks The geometric mean is an alternative form of calculating a mean for a sample with strong positive skewedness and no negative values. It is obtained from a list of *n* numbers by taking the *n*th root of the product of the numbers.

Example This function returns 6.931:

```
GEOMEAN(1, 3, 5, 16, 42, 11)
```

This function returns 12:

```
GEOMEAN(24, 6)
```

See Also AVERAGE, COUNT, HARMEAN, SKEW

GESTEP

Description

Compares two specified values and returns 1 if the first value is greater than or equal to the second value, 0 if the first value is less than the second value.

Syntax

DELTA (*number1* [, *number2*])

Argument	Description
number1	A number you want to compare.
[*number2*]	Optional. This is the number you want to compare to *number1*. If you omit this argument, the function will compare *number1* to 0.

Remarks

This function can be used in place of the equals (=) operator when a numeric rather than logical result is needed (for example, when using aggregate functions such as SUM and COUNT).

GE stands for "greater than or equal to."

Examples

This function returns 0:

```
DELTA(6,7)
```

This function returns 1:

```
DELTA(6)
```

See Also

DELTA, IF

GETPIVOTDATA

Description

Returns the #N/A error message.

Syntax

GETPIVOTDATA (*pivot_table*, *name*)

Argument	Description
pivot_table	The name of the pivot table.
name	The name of the cell.

Remarks

This function was included only for compatibility with Microsoft Excel, which uses GETPIVOTDATA to return data from a pivot table. Pivot tables are not supported in Formula One for Java.

This function is intended only for users or developers who want to import Excel worksheets that contain Excel's GETPIVOTDATA function. Such worksheets can be opened and saved in Formula One for Java, but the GETPIVOTDATA functionality will not be included.

GROWTH

Description

Computes result values from the fitted curve of a multiple exponential regression for a group of specified observations relative to a group of specified independent variables.

This type of regression is often applied to exponential growth situations. Because of the type of formula and the often limited amount of data the formula is based on, using these formulas to estimate values outside of the range of data is dangerous, since they are likely to give exponentially wrong results.

Another problematic aspect of this function is that it assumes the range of data in question fits an exponential model. While this assumption is appropriate in some situations (for example, radioactive decay), in many other situations it is misleading, particularly in regards to economic data.

Note If you want GROWTH to return more than one result value, you must enter it as an array function. For information, see “Array Functions” on page 13.

Syntax

GROWTH (*known_y’s* [, *known_x’s*] [, *new_x’s*] [, *constant*])

Argument	Description
known_y’s	A list of observed values for the dependent variable. This must be a range reference or an array constant containing all numeric values and no empty cells. For a regression with a **single independent variable**, enter any type of range. For a regression with **multiple independent variables**, enter a single row or column of values. If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array.

Argument	Description
[*known_x's*]	Optional. A list of observed values for the independent variable(s). This must be a range reference or an array constant containing all numeric values and no empty cells. For a regression with a **single independent variable**, when *known_y's* is a single row or column, enter a range that exactly matches the size and shape of the *known_y's* range. When *known_y's* is more than one row and column, *known_x's* must be a range containing the same number of values as the *known_y's* range, although the two ranges may be different shapes. For a regression with **multiple independent variables**, when *known_y's* is a single row, enter a contiguous row of values for each independent variable, The number of columns must match the number of columns in *known_y's*. Similarly, when *known_y's* is a single column, enter a contiguous column of values for each independent variable, ensuring that the number of rows matches *known_y's*. If you are using array constants, see "Array Constants in Arguments" on page 9 for information on how to enter the array. To fit the regression to a formula that uses polynomials, see "Polynomials," below. When you omit this argument, the function assumes a single independent variable of the sequence {1, 2, 3, ...}.
[*new_x's*]	Optional. A list of values for the independent variable(s) that will compute the results of this function. This must be a range reference or an array constant containing all numeric values and no empty cells. You may enter any number of values in *new_x's*. The shape of the range only matters if this is a regression with multiple independent variables, in which case it must have the same number of rows or colunms of independent variables as *known_x's*. If you omit this argument, the function uses the specified or default values in *known_x's*.
[*constant*]	Optional. A logical value that determines whether or not a constant is to be included. The constant allows the regression line to intercept the Y axis at a point other than (0,0). Use True (the default value if this argument is omitted) to include the constant. Use False to force the line to intercept the Y axis at 0.

Remarks

This function uses LOGEST internally to compute the regression's coefficients. The resulting formula is then used to compute the requested values.

Results range

Before entering this function, select a results range according to the following criteria:

- If there is a single independent variable, the results range should match the size and shape of the *new_x's* argument.

- If there are multiple independent variables, the results range depends on the number of data sets in *new_x’s*. If there is only one data set, the results range should be a single cell. For multiple data sets, the results range should be a column if *known_y’s* is a column, a row if *known_y’s* is a row, with the same number of cells as data sets entered in *new_x’s*. If you omitted the *new_x’s* argument, the results range must match the size and shape of the *known_y’s* argument.

Equation

The regression computed by GROWTH attempts to fit the following formula:

$$y = C_1^{x_1} \times C_2^{x_2} \times \ldots \times C_n^{x_n} \times b$$

where *C* is the coefficient determined internally by the LOGEST function, *n* is the number of independent variables, and *b* is the constant.

This formula can be converted to a linear form if logarithms are taken of both sides. The formula then becomes:

$$y = lC_1x_1+lC_2x_2+ \ldots + lC_nX_n+lb$$

This is the same formula fitted by the LINEST function. LOGEST use LINEST internally to fit the linearized form of the equation. LOGEST then takes the LINEST coefficients (lC_1, lC_2 ,... lC_n) and converts them to the LOGEST coefficients (C_1, C_2, ... C_n) by applying the EXP function: C_1 = EXP(lC_1). The same conversion is done with the constant, *b*.

Example

This example of exponential regression uses the following data on the growth of the national debt of a small country.

Year	**National debt, in millions**
1971	0.5
1972	1.1
1973	1.4
1974	2.0
1975	4.4
1976	20.0
1977	445.0

In this case, the debt figures are the *known_y’s* and we can use default values for the *known_x’s*, since the years increment by one.

We want to use this data to find what the national debt would have been in June of 1973. If the national debt figures fill range A2:A8, this function returns 2.958273:

```
GROWTH(A2:A8,,3.5,TRUE)
```

See Also

INTERCEPT, LINEST, LOGEST, SLOPE, TREND

HARMEAN

Description Computes the harmonic mean of a list of positive numbers.

Syntax HARMEAN (*number_list*)

Argument	Description
number_list	A list of between 1 and 30 arguments consisting of numbers, cell references, range references, and/or array constants. The arguments must evaluate to positive numbers or 0. Text and logical values in arrays or cells referenced by this function is ignored. Empty cells are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Remarks This is an alternative form of calculating a mean for a sample with strong positive skewedness and no negative values. It is obtained by taking the reciprocal of the arithmetic mean of the reciprocals of a list of numbers.

The result of HARMEAN is smaller than the geometric mean calculated by GEOMEAN.

Example This function returns 9.731:

```
HARMEAN(5,4,25,60,14,26)
```

See Also AVERAGE, COUNT, GEOMEAN, SKEW

HLOOKUP

Description Searches the top row of a table for a value and returns the contents of a cell in that table that corresponds to the location of the search value.

Syntax HLOOKUP (*search_item*, *search_range*, *row_index*)

Argument	Description
search_item	A value, text string, or reference to a cell containing a value that is matched against data in the top row of search_range.
search_range	A reference to the range (table) to be searched. The cells in the first row of *search_range* can contain numbers, text, or logical values. The contents of the first row must be in ascending order (for example, –2, –1, 0, 2 ... A through Z, False, True). Text searches are not case-sensitive.

Argument	Description
row_index	The row in *search_range* from which the matching value is returned. *row_index* can be a number from 1 to the number of rows in *search_range*. If *row_index* is less than 1, the error #VALUE! is returned. When *row_index* is greater than the number of rows in the table, the error #REF! is returned.

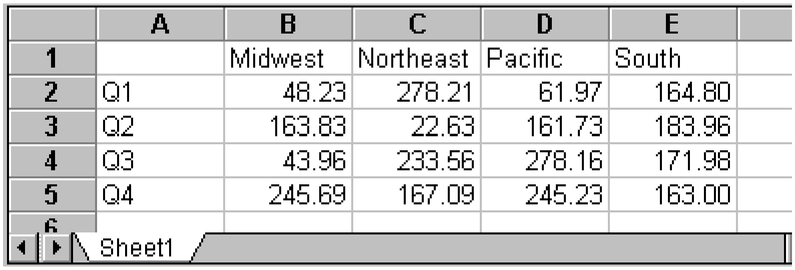
Remarks

HLOOKUP compares the information in the top row of *search_range* to the supplied *search_item*. When a match is found, information located in the same column and supplied row (*row_index*) is returned.

If *search_item* cannot be found in the top row of *search_range*, the largest value that is less than *search_item* is used. When *search_item* is less than the smallest value in the first row of the *search_range*, the error #REF! is returned.

Examples

The following examples use this worksheet.



	A	B	C	D	E
1		Midwest	Northeast	Pacific	South
2	Q1	48.23	278.21	61.97	164.80
3	Q2	163.83	22.63	161.73	183.96
4	Q3	43.96	233.56	278.16	171.98
5	Q4	245.69	167.09	245.23	163.00
6					

This function returns 22.63:

```
HLOOKUP("Northeast", B1:E5, 3)
```

This function returns #REF!:

```
HLOOKUP("Pacific", B1:E5, 7)
```

See Also

INDEX (non-array type), LOOKUP, MATCH, VLOOKUP

HEX2BIN

Description

Converts a hexadecimal number (base 16) to a binary number (base 2).

Syntax

HEX2BIN (*integer* [, *places*])

Argument	Description
integer	Any negative or positive whole number or zero. For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between FFFFFFFE00 to 1FF may be used.

Argument	Description
places	Optional number of characters to use in the output. If *places* is omitted, the function returns the number of spaces required to display the result. The *places* argument can be used to 0-pad a positive number. If the number is negative, *places* is ignored and 10 characters are returned.

Remarks

Note that the result of this function is a text string.

Examples

This function returns 10000:

```
HEX2BIN(10)
```

This function returns 1111110000:

```
HEX2BIN("FFFFFFFFF0")
```

This function returns 000001000:

```
HEX2BIN(10,10)
```

See Also

BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

HEX2DEC

Description

Converts a hexadecimal number (base 16) to a decimal number (base 10).

Syntax

HEX2DEC (*integer*)

Argument	Description
integer	Any negative or positive whole number or zero. For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between 8,000,000,000 and 7FFFFFFFFF may be used.

Remarks

Note that the result of this function is a number.

Examples

This function returns 16:

```
HEX2DEC(10)
```

This function returns 5.49756E + 11:

```
HEX2DEC(7FFFFFFFFF)
```

See Also

BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

HEX2OCT

Description Converts a hexadecimal number (base 16) to an octal number (base 8).

Syntax HEX2OCT(*integer* [, *places*])

Argument	Description
integer	Any negative or positive whole number or zero. For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between FFE0000000 to 1FFFFFF may be used.
places	Optional number of characters to use in the output. If *places* is omitted, the function returns the number of spaces required to display the result. The *places* argument can be used to 0-pad a positive number. If the number is negative, *places* is ignored and 10 characters are returned.

Remarks Note that the result of this function is a text string.

Examples This function returns 400:

```
HEX2OCT(100)
```

This function returns 007:

```
HEX2OCT(7,3)
```

See Also BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, OCT2BIN, OCT2DEC, OCT2HEX

HOUR

Description Returns the hour component of the specified time in 24-hour format.

Syntax HOUR (*serial_number*)

Argument	Description
serial_number	The time as a serial number. The decimal portion of the number represents time as a fraction of the day.

Remarks The result is an integer ranging from 0 (12:00 AM) to 23 (11:00 PM).

Examples This function returns 9:

```
HOUR(34259.4)
```

This function returns 23:

```
HOUR(34619.976)
```

See Also DAY, MINUTE, MONTH, NOW, SECOND, WEEKDAY, YEAR

HYPERLINK

Description Returns the #N/A error message.

Syntax HYPERLINK (*link_location* [, *cell_content*]*)*)

Argument	Description
link_location	The path and filename to the document to be opened as text in a browser.
[*cell_content*]	Optional argument. Text to display in cell.

Remarks This function was included only for compatibility with Microsoft Excel. In Excel, HYPERLINK sets up a hyperlink to a file or web page, which is not supported in Java.

This function is intended only for users or developers who want to import Excel worksheets that contain Excel's HYPERLINK function. To make those worksheets work properly, developers should convert these functions to add-in functions, which they can write in Java and set up to be automatically loaded.

For more information, see the chapter on creating add-in functions in the *Formula One for Java Technical Guide*.

HYPGEOMDIST

Description Computes the hypergeometric distribution. This is used in cases of sampling without replacement, and is especially important for determining the likelihood of finding defective parts in a production run.

For cases of sampling with replacement, use BINOMDIST.

Syntax HYPGEOMDIST (*x, n, M, N*)

Argument	Description
x	A positive integer indicating the number of successes in the sample. It must be smaller than *n* and *M*. Decimal values will be truncated to integers.
n	A positive integer indicating the size of the sample. It must be smaller than *N*. Decimal values will be truncated to integers.

Argument	Description
M	A positive integer indicating the number of successes in the total population. It must be smaller than *N*. Decimal values will be truncated to integers.
N	A positive integer indicating the size of the total population. Decimal values will be truncated to integers.

Equation

$$f(x) = \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}}$$

Examples

Say 10 sweaters on a store rack have been inspected, and 2 of the sweaters have flaws. This function calculates the probability that, if you buy four sweaters, none of them will have flaws.

This function returns 0.33:

```
HYPGEOMDIST (0,4,2,10)
```

See Also

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, NORMDIST, POISSON

IF

Description

Tests the condition and returns the specified value.

Syntax

IF (*condition*, *true_value*, *false_value*)

Argument	Description
condition	Any logical expression.
true_value	The value to be returned if *condition* evaluates to True.
false_value	The value to be returned if *condition* evaluates to False.

Example

This function returns Greater if the contents of A1 is greater than 10 and Less if the contents of A1 is less than 10:

```
IF(A1>10, "Greater", "Less")
```

See Also

AND, FALSE, NOT, OR, TRUE

IMABS

Description Computes the absolute value of a complex number.

Syntax IMABS (*complex_number*)

Argument	Description
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).

Remarks The result can be thought of as the distance from the origin of a graph to the point where the X-axis is the value of the real number and the Y-axis is the value of the imaginary number.

Example This function returns 5:

```
IMABS("3+4i")
```

See Also COMPLEX, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

IMAGINARY

Description Returns the coefficient of the imaginary portion of a complex number as a real number.

Syntax IMAGINARY (*complex_number*)

Argument	Description
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).

Example This function returns 3:

```
IMAGINARY("2+3i")
```

See Also COMPLEX, IMABS, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

IMARGUMENT

Description Computes the argument θ as an angle expressed in radians.

Syntax IMARGUMENT (*complex_number*)

Argument	Description
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i"))`.

Equation $\mathrm{ATAN}\left(\frac{b}{a}\right)$

Remarks This is the complementary function to IMABS, where the degree of angle (θ) begins at the positive X-axis, and rotates counterclockwise in polar coordinates.

An alternate form of complex notation is r(cos θ + *i* sin θ) where "r" is the result of IMABS.

Note The range of θ is -π < θ <= π.

Example This function returns 0.785398163:

```
IMARGUMENT("1+1i")
```

This function returns 3.14159265, the value π:

```
IMARGUMENT(-1)
```

See Also COMPLEX, IMABS, IMAGINARY, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

IMCONJUGATE

Description Computes the complex conjugate of the argument.

Syntax IMCONJUGATE (*complex_number*)

Argument	**Description**
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).

Remarks This is simply the same number with the sign changed on the imaginary component.

Example This function returns 2-3*i*:

```
IMCONJUGATE("2+3i")
```

This function returns 2+3*i*:

```
IMCONJUGATE("2-3i")
```

See Also COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

IMCOS

Description Computes the complex cosine of the argument.

Syntax IMCOS (*complex_number*)

Argument	**Description**
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).

Equation $\mathrm{COS}(a+bi) = \mathrm{COS}(a)\mathrm{COSH}(b) - i\mathrm{SIN}(a)\mathrm{SINH}(b)$

Example This function returns -0.64214812471552 - 1.06860742138278*i*

```
IMCOS("2+i")
```

This function returns -0.9899992496600445:

```
IMCOS(3)
```

See Also COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

IMDIV

Description Divides one complex number by another.

Syntax IMDIV (*complex_number, complex_number*$_2$)

Argument	Description
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).
complex_number$_2$	Another complex number (a + b*i*). Any function that accepts two or more complex arguments must have the same suffix (i or j) on all arguments to that function, or a #VALUE error will result.

Equation $\frac{z_1}{z_2} = \frac{a + bi}{c + di} + \frac{-a + bi}{c + di}$ ...where z is a complex number.

Example This function returns 2 + 6*i*:

```
IMDIV("-10+10i","1+2i")
```

See Also COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

IMEXP

Description Computes the complex exponential of the specified complex number. (The constant, *e*, is raised to the power equal to the specified complex number.)

Syntax IMEXP (*complex_number*)

Argument	Description
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).

Equation $e^z = e^{(a+bi)} = e^a(\mathrm{COS}(b) + i\mathrm{SIN}(b))$ ...where *z* is a complex number.

Example This function returns -7.3151100949011 + 1.0427436562359*i*:

```
IMEXP("2+3i")
```

See Also COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

IMLN

Description Computes the complex natural logarithm of the specified complex number.

Syntax IMLN (*complex_number*)

Argument	Description
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).

Equation $\mathrm{LN}z = \mathrm{LN}(|z| + \mathrm{IMARG}z) = \mathrm{LN}\sqrt{a^2 + b^2} + \mathrm{IMARG}(a + bi)$

...where *z* is a complex number.

Example This function returns 1.28247467873077 + 0.982793723247329*i*:

```
IMLN("2+3i")
```

See Also COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

IMLOG10

Description Computes the complex common logarithm of the specified complex number.

Syntax IMLOG10 (*complex_number*)

Argument	Description
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).

Equation $\mathrm{LOG}_{10}e \cdot \mathrm{LN}(a + bi)$

Example This function returns 0.556971676153418 + 0.426821890855467*i*:

```
IMLOG10("2+3i")
```

See Also COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

IMLOG2

Description Computes the complex logarithm base-2 of the specified complex number.

Syntax IMLOG2 (*complex_number*)

Argument	Description
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).

Equation $\mathrm{LOG}_{2}e \cdot \mathrm{LN}(a + bi)$

Example This function returns 1.85021985921295 + 1.41787163085485*i*:

```
IMLOG2("2+3i")
```

See Also COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

IMPOWER

Description Computes the real power of a complex number (raises a complex number to a specified power).

Syntax IMPOWER (*complex_number* , *number*)

Argument	Description
complex_number	A complex number in the form (a + b*i*), or a cell or range reference containing complex numbers. You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).
number	Any real number

Equation

$$z^n = r^n[\mathrm{COS}(n\theta) + i\mathrm{SIN}(n\theta)]$$

...where z is a complex number, n is an integer greater than 0, r = imabs(z), and θ = imargument(z).

Example This function returns -3 + 4*i*:

```
IMPOWER("1+2i",2)
```

See Also COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

IMPRODUCT

Description Computes the product of up to 29 complex numbers.

Syntax IMPRODUCT (*complex_number*, *complex_number*$_2$)

Argument	Description
complex_number	A complex number in the form (a + b*i*), or a cell or range reference containing complex numbers. You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).
complex_number$_2$	Another complex number Any function that accepts two or more complex arguments must have the same suffix (i or j) on all arguments to that function, or a #VALUE error will result.

Equation

$$z_1z_2 = (a_1 + ib_1) \cdot (a_2 + ib_2) = (a_1a_2 - b_1b_2) + i(a_1b_2 + a_2b_1)$$

...where z is a complex number.

Examples

This function returns 3 + 29*i*:

```
IMPRODUCT("3+4i","5+3i")
```

This function returns 30 + 60*i*:

```
IMPRODUCT("1+2i",30)
```

This function returns -1650 + 1050*i*:

```
IMPRODUCT("3+4i","5+3i","1+2i",30)
```

See Also

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

IMREAL

Description

Returns the coefficient of the real portion of a specified complex number.

Syntax

IMREAL (*complex_number*)

Argument	Description
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).

Example

This function returns 2:

```
IMREAL("2+3i")
```

See Also

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMSIN, IMSQRT, IMSUB, IMSUM

IMSIN

Description

Computes the complex sine of the specified complex number.

Syntax

IMSIN (*complex_number*)

Argument	Description
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).

Equation

$$\mathrm{SIN}(a+bi) = \mathrm{SIN}(a)\mathrm{COSH}(b) + i\mathrm{COS}(a)\mathrm{SINH}(b)$$

Example

This function returns: 9.15449914691143 - 4.16890695996656*i*:

```
IMSIN("2+3i")
```

See Also

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSQRT, IMSUB, IMSUM

IMSQRT

Description

Computes the complex square root of the specified complex number.

Syntax

IMSQRT (*complex_number*)

Argument	Description
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).

Equation

$$\sqrt{z} = \sqrt{r} \cdot \left(\mathrm{COS}\frac{\theta}{2} + i\,\mathrm{SIN}\frac{\theta}{2} \right)$$

where z is a complex number, r = IMABS(z), and θ = IMARGUMENT(z).

Example

This function returns 1.67414922803554 + 0.895977476129838*i*:

```
IMSQRT("2+3i")
```

See Also

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSUB, IMSUM

IMSUB

Description

Subtracts one complex number from another.

Syntax

IMSUB (*complex_number*, *complex_number*$_2$)

Argument	Description
complex_number	A complex number in the form (a + b*i*). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).
complex_number$_2$	Another complex number Any function that accepts two or more complex arguments must have the same suffix (i or j) on all arguments to that function, or a #VALUE error will result.

Equation

$$z_1 - z_2 = (a_1 - a_2) + \iota(b_1 - b_2)$$

...where z is a complex number.

Example

This function returns -1 - *i*:

```
IMSUB("2+3i","3+4i")
```

See Also

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUM

IMSUM

Description Computes the sum of up to 29 complex numbers.

Syntax IMSUM (*complex_number*, *complex_number*$_2$)

Argument	Description
complex_number	A complex number in the form (a + b*i*), or a cell or range reference containing complex numbers. You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., `IMABS("1+1i")` or `IMABS("i")`).
complex_number$_2$	Another complex number Any function that accepts two or more complex arguments must have the same suffix (i or j) on all arguments to that function, or a #VALUE error will result.

Equation

$$z_1 + z_2 = (a_1 + a_2) + i(b_1 + b_2)$$

...where z is a complex number.

Examples This function returns 8 + 7*i*:

```
IMSUM("2+3i","3+4i")
```

This function returns 36 + 9*i*:

```
IMSUM("1+2i","2+3i","3+4i",30)
```

See Also COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB

INDEX (non-array type)

Description Returns the contents of a cell from a specified range.

Syntax INDEX (*reference* [, *row*] [, *column*] [, *range_number*])

Argument	Description
reference	A reference to one or more ranges. If *reference* specifies more than one range, separate each reference with a comma and enclose *reference* in parentheses. For example, `INDEX((A1:C6,B7:E14,F4),3,2,2)`.

Argument	Description
[*row*]	Optional. The row number in *reference* from which to return data.
[*column*]	Optional. Column number in *reference* from which to return data.
[*range_number*]	Optional. Specifies the range from which data is returned if *reference* contains more than one range. For example, if *reference* is `(A1:A10, B1:B5, D14:E23)`, A1:A10 is *range_number* 1, B1:B5 is *range_number* 2, and D14:E23 is *range_number* 3.

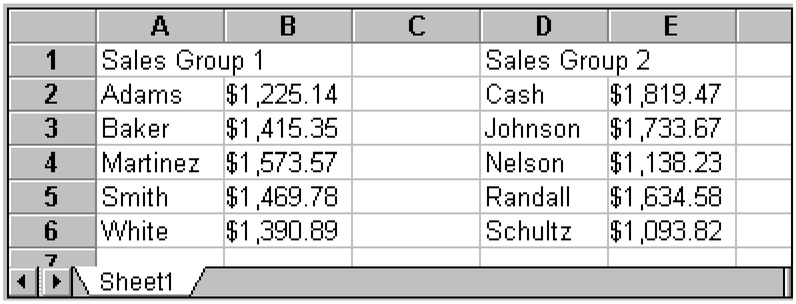
Remarks

If each range in *reference* contains only one row or column, you can omit the *row* or *column* argument. For example, if *reference* is A1:A15, you can omit the column argument like so: `INDEX(A1:A15,3,,1)`.

If *row*, *column*, and *range_number* do not point to a cell within *reference*, #REF! is returned. If *row* and *column* are omitted, INDEX returns the range in *reference* specified by *range_number.*

Examples

The following examples use this worksheet.



	A	B	C	D	E	
1	Sales Group 1			Sales Group 2		
2	Adams	$1,225.14		Cash	$1,819.47	
3	Baker	$1,415.35		Johnson	$1,733.67	
4	Martinez	$1,573.57		Nelson	$1,138.23	
5	Smith	$1,469.78		Randall	$1,634.58	
6	White	$1,390.89		Schultz	$1,093.82	
7						

This function returns `$1415.35:`

```
INDEX(A2:B6,2,2)
```

This function returns $1634.58:

```
INDEX((A2:B6,D2:E6),4,2,2)
```

See Also

CHOOSE, HLOOKUP, LOOKUP, MATCH, VLOOKUP

INDEX (array type)

Description

Returns a specified value from an array.

Note This is an array function. For information, see “Array Functions” on page 13.

Syntax

INDEX (*array* [, *row*] [, *column*])

Argument	Description
array	An array constant or range reference. For information on array constants, see “Array Constants in Arguments” on page 9. If it is a range, it must be one contiguous range.
[*row*]	Optional. The row number in *array* from which to return data.
[*column*]	Optional. The column number in *array* from which to return data.

Remarks

You can omit either the *row* or *column* argument. If you omit one or the other, the function returns all the values in the specified row or column. See examples.

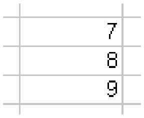
If *row*, *column*, and *range_number* do not point to a value within *array*, #REF! is returned.

Examples

When entered in a single cell, this function returns 7:

```
INDEX({1,4,7;2,5,8;3,6,9},,3)
```

When entered as an array function, the same function returns the following range:



7
8
9

See Also

CHOOSE, HLOOKUP, LOOKUP, MATCH, VLOOKUP

INDIRECT

Description

Returns the contents of the cell referenced by the specified cell.

Syntax

INDIRECT (*ref_text* [, *a1*])

Argument	Description
ref_text	A reference to a cell that references a third cell. If *ref_text* is not a valid reference, the error #REF! is returned.
[*a1*]	Optional. The reference format. This argument must be TRUE() to represent an A1 reference format; Formula One does not support the R1C1 reference format.

Example

This function returns the contents of the cell that C1 references. If C1 contains “D1,” then the contents of D1is returned:

```
INDIRECT(C1)
```

See Also

OFFSET

INFO

Description Returns the specified type of information about the current workbook and system.

Syntax INFO (*info_type*)

Argument	Description
info_type	The type of information you want. It must be text, surrounded by quotation marks. The following are the valid entries and the types of information they return, in alphabetical order:

“author”	Returns #N/A.
“creation-date”	Returns #N/A.
“editing-time”	Returns #N/A.
“dbreturncode”	Returns #N/A.
“dbdrivermessage”	Returns #N/A.
“dbrecordcount”	Returns #N/A.
“directory”	Returns the path of the current directory or folder.
“last-revision-by”	Returns #N/A.
“last-revision-date”	Returns #N/A.
“macro-step”	Returns No.
“macro-trace”	Returns No.
“memavail”	Returns the amount of memory available, in bytes.
“memused”	Returns the amount of memory being used for data.
“mode”	Returns 1.
“numfile”	Returns #N/A.
“origin”	Returns #N/A.
“osreturncode”	Returns 0.
“osversion”	Returns the current operating system version as text.
“recalc”	Returns the current recalculation mode: either Automatic or Manual.
“release”	Returns the release number of the version of Formula One for Java you are using, as text.
“setup-user-name”	Returns #N/A.
“screen-height”	Returns #N/A.
“screen-width”	Returns #N/A.
“selection”	Returns #N/A.
“selection-part”	Returns #N/A.
“selection-type”	Returns #N/A.
“system”	Returns the name of the operating system as text.

Argument	Description	
info_type (continued)	*“totmem”*	Returns the total memory available, including memory in use, in bytes.
	“windir	Returns #N/A.
	“worksheet-number”	Returns the identification number of the current worksheet.
	“worksheet-size”	Returns #N/A.

Remarks

This function is included in Formula One for Java in order to be compatible with Excel, which in turn included it in order to be compatible with other worksheet formats, notably Lotus 1-2-3.

Developers will find it more efficient to extract this type of information using Java direct calls.

Example

This function returns 7.0:

```
INFO("release")
```

See Also

CELL

INT

Description

Rounds the supplied number down to the nearest integer.

Syntax

INT (*number*)

Argument	Description
number	Any real number.

Examples

This function returns 10:

```
INT(10.99)
```

This function returns –11:

```
INT(-10.99)
```

See Also

CEILING, FLOOR, MOD, ROUND, TRUNC

INTERCEPT

Description

Computes the y-intercept of the least squares linear regression line through the given data.

Syntax

INTERCEPT (*arrayY, arrayX*)

Argument	Description
arrayY and *arrayX*	A range reference or an array constant containing numeric values. *ArrayX* and *arrayY* must contain the same potential number of values. Text, logical values, and empty cells referenced by this function are ignored, along with the number they are paired up with. That is, when a number from *arrayX* is paired up with text from *arrayY*, the entire pair is ignored.

Remarks

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Equation

$$\text{intercept } a = \frac{\sum y}{n} - b\left(\frac{\sum y}{n}\right)$$

where *b* is the slope and *n* is the number of members in *arrayY* and *arrayX*.

Examples

This function returns 2.142857143:

```
INTERCEPT({1,5,8,9,12,13},{0,1,2,3,4,5})
```

See Also

FORECAST, SLOPE

INTRATE

Description

Computes the interest rate of a fully invested security, given a specified investment dollar amount and redemption value.

Syntax

INTRATE (*settlement, maturity, investment, redemption* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Decimal values will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be the same day or later than *settlement*. Decimal values will be truncated to integers.
investment	The dollar amount invested in the security.
redemption	The amount paid at *maturity.*

Argument	Description
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Equation

$$\frac{\textit{redemption - investment}}{\textit{investment} \times \text{YEARFRAC}\ (\textit{settlement, maturity, calendar\_type})}$$

Examples

This function returns 0.0244:

```
INTRATE("10/23/94","7/7/95",98.31,100)
```

See Also

DISC, PRICE, PRICEDISC, PRICEMAT, YIELD, YIELDDISC, YIELDMAT

IPMT

Description

Returns the interest payment of an annuity for a given period, based on regular payments and a fixed periodic interest rate.

Syntax

IPMT (*interest*, *per, nper, pv* [, *fv*] [, *type*])

Argument	Description
interest	The fixed periodic interest rate.
per	The period for which to return the interest payment. This number must be between 1 and *nper.*
nper	The number of payments.
pv	The present value, or the lump sum amount the annuity is currently worth.
[*fv*]	Optional. The future value, or the value after all payments are made. If this argument is omitted, 0 is used.
[*type*]	Optional. Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. If this argument is omitted, 0 is used.

Remarks

The units used for *interest* must match those used for *nper.* For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for *interest* and 5*12 for *nper.*

Cash paid out, such as a payment, is shown as a negative number. Cash received, such as a dividend check, is shown as a positive number.

Examples

This function returns –117.87:

```
IPMT(8%/12, 2, 48, 18000)
```

This function returns –117.09:

```
IPMT(8%/12, 2, 48, 18000, 0, 1)
```

See Also FV, CUMIPMT, CUMPRINC, NPER, PMT, PPMT, PV, RATE

IRR

Description

Returns the internal rate of return for a series of periodic cash flows.

Syntax

IRR (*cash_flow* [, *guess*])

Argument	Description
cash_flow	A reference to a range that contains values for which to calculate the internal rate of return. The values must contain at least one positive and one negative value. During calculation, IRR uses the order in which the values appear to determine the order of the cash flow. Text, logical values, and empty cells in the range are ignored.
[*guess*]	Optional. The estimate of the internal rate of return. If this argument is omitted, 10 is used.

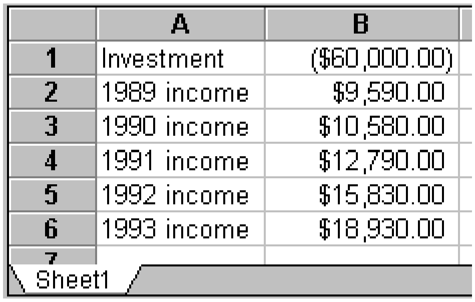
Remarks

The internal rate of return is the interest rate received for an investment consisting of payments (specified by negative numbers) and investments (specified by positive numbers).

IRR is calculated iteratively, cycling through the calculation until the result is accurate to .00001 percent. If the result cannot be found after 20 iterations, #NUM! is returned. When this occurs, supply a different value for *guess*.

Examples

The following examples use this worksheet.



	A	B
1	Investment	($60,000.00)
2	1989 income	$9,590.00
3	1990 income	$10,580.00
4	1991 income	$12,790.00
5	1992 income	$15,830.00
6	1993 income	$18,930.00
7		

This function returns 3.72 percent:

```
IRR(B1:B6)
```

This function returns –49.26 percent:

```
IRR(B1:B3, -20%)
```

See Also MINVERSE, NPV, RATE

ISBLANK

Description Determines if the specified cell is blank.

Syntax ISBLANK (*reference*)

Argument	Description
reference	A reference to any cell.

Remarks If the referenced cell is blank, True is returned. False is returned if the cell is not blank.

Example This function returns True if A1 is a blank cell:

```
ISBLANK(A1)
```

See Also ISERR, ISERROR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISREF, ISTEXT

ISERR

Description Determines if the specified expression returns an error value.

Syntax ISERR (*expression*)

Argument	Description
expression	Any expression.

Remarks If the expression returns any error except #N/A!, True is returned. Otherwise, False is returned.

Example This function returns True if A1 contains a formula that returns an error such as #NUM!:

```
ISERR(A1)
```

See Also ISBLANK, ISERROR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISREF, ISTEXT

ISERROR

Description Determines if the specified expression returns an error value.

Syntax ISERROR (*expression*)

Argument	Description
expression	Any expression.

Remarks If *expression* returns any error value, such as #N/A!, #VALUE!, #REF!, #DIV/0!, #NUM!, #NAME?, or #NULL!, True is returned. Otherwise, False is returned.

Examples This function returns True:

```
ISERROR(4/0)
```

This function returns False if A1 contains a formula that does not return an error.

```
ISERROR(A1)
```

See Also ISBLANK, ISERR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISREF, ISTEXT

ISEVEN

Description Determines whether the specified number is even.

Syntax ISEVEN (*number*)

Argument	Description
number	Number value to be analyzed. Decimal values will be truncated to integers.

Examples This function returns FALSE:

```
ISEVEN(5.8)
```

This function returns TRUE:

```
ISEVEN(4.5)
```

See Also ISODD

ISLOGICAL

Description Determines if the specified expression returns a logical value.

Syntax ISLOGICAL (*expression*)

Argument	Description
expression	Any expression.

Remarks If *expression* returns a logical value, True is returned. Otherwise, False is returned.

Example This function returns True because ISBLANK returns a logical value:

```
ISLOGICAL(ISBLANK(A1))
```

See Also ISBLANK, ISERR, ISERROR, ISNA, ISNONTEXT, ISNUMBER, ISREF, ISTEXT

ISNA

Description Determines if the specified expression returns the value not available error.

Syntax ISNA (*expression*)

Argument	Description
expression	Any expression.

Remarks If *expression* returns the #N/A! error, True is returned. Otherwise, False is returned.

Example This function returns True if cell A1 contains the NA ( ) function or returns the error value #N/A!:

```
ISNA(A1)
```

See Also ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNONTEXT, ISNUMBER, ISREF, ISTEXT

ISNONTEXT

Description Determines if the specified expression is not text.

Syntax ISNONTEXT (*expression*)

Argument	Description
expression	Any expression.

Remarks If *expression* returns any value that is not text, True is returned. Otherwise, False is returned.

Examples This function returns True if cell F3 contains a number or is a blank cell:

```
ISNONTEXT(F3)
```

This function returns False:

```
ISNONTEXT("text")
```

See Also ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNA, ISNUMBER, ISREF, ISTEXT

ISNUMBER

Description Determines if the specified expression is a number.

Syntax ISNUMBER (*expression*)

Argument	Description
expression	Any expression.

Remarks If *expression* returns a number, True is returned. Otherwise, False is returned. If *expression* returns a number represented as text (for example, “12”), False is returned.

Examples This function returns True:

```
ISNUMBER(123.45)
```

This function returns False:

```
ISNUMBER("123")
```

See Also ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNA, ISNONTEXT, ISREF, ISTEXT

ISODD

Description Determines whether the specified number is odd.

Syntax ISODD (*number*)

Argument	Description
number	Number value to be analyzed. Decimal values will be trucated to integers.

Examples

This function returns TRUE:

```
ISODD(5.8)
```

This function returns False:

```
ISODD(4.5)
```

See Also

ISEVEN

ISREF

Description

Determines if the specified expression is a range reference.

Syntax

ISREF (*expression*)

Argument	Description
expression	Any expression.

Remarks

If *expression* returns a range reference, True is returned. Otherwise, False is returned.

Example

This function returns True:

```
ISREF(A3)
```

See Also

ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISTEXT

ISTEXT

Description

Determines if the specified expression is text.

Syntax

ISTEXT (*expression*)

Argument	Description
expression	Any expression.

Remarks

If *expression* returns text, True is returned. Otherwise, False is returned.

Example

This function returns True:

```
ISTEXT("2nd Quarter")
```

See Also

ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISREF

KURT

Description Computes the kurtosis of a list of numbers.

Syntax KURT (*number_list*)

Argument	Description
number_list	A list of between 1 and 30 arguments consisting of values, cell references, range references, and/or array constants. The argument(s) must contain at least 4 numbers. All numeric values, including 0, are used. Text and logical values in array constants and cells referenced by this function are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Remarks Kurtosis measures the peakedness of the curve around its mean. A kurtosis of zero is approximately the same as the normal distribution; a larger kurtosis indicates a more pronounced central peak, while a smaller kurtosis indicates a flattened distribution around the peak.

Kurtosis is also known as the fourth movement, the first three being mean, standard deviation, and skewness.

Equation

$$\frac{n(n+1)}{(n-1)(n-2)(n-3)}\sum\left(\frac{x_j-\bar{x}}{s}\right)^4-\frac{3(n-1)^2}{(n-2)(n-3)}$$

where *s* is the sample standard deviation.

Example This function returns -1.2:

```
KURT(1,2,3,4,5,6,7)
```

See Also AVEDEV, AVERAGE, COUNT, DEVSQ, SKEW

LARGE

Description Computes the *x*th largest value in a list. If you set *x* at 5, LARGE will sort the values in ascending order and return the 5th largest value.

Syntax LARGE (*numbers, x*)

Argument	Description
numbers	A range reference or array constant containing between 2 and 30 numbers. Text and logical values in cell references and arrays are ignored. Empty cells are ignored.
x	The index of the number to return after sorting the list in ascending order. *X* must be larger than zero and smaller than the number of values in *numbers*. The logical value TRUE is evaluated as 1, but FALSE will return the #NUM! error. Text will be evaluated as a number, if possible; otherwise the function will return the #VALUE! error.

Example

This function returns 7:

```
LARGE({1,2,3,4,5,6,7,8,9,15},4)
```

See Also

MAX, MIN, SMALL

LCM

Description

Computes the least common multiple of a specified set of values. This number is the smallest that can be divided evenly by all the numbers in the set, with no fraction or remainder.

Syntax

LCM (*number_list*)

Argument	Description
number_list	A list of as many as 29 arguments in the form of numbers, cell references, range references, and array constants. The values in the arguments must be positive integers. Decimal values will be truncated to integers. Non-numeric arguments will return errors.

Examples

This function returns 16:

```
LCM(2,4,8,16)
```

This function returns 60

```
LCM(15,20)
```

See Also

GCD

LEFT

Description Returns the leftmost characters from the specified text string.

Syntax LEFT (*text* [, *num_chars*])

Argument	Description
text	Any text string.
[*num_chars*]	Optional. The number of characters to return. This value must be greater than or equal to zero. If *num_chars* is greater than the number of characters in *text*, the entire string is returned. If this argument is omitted, 1 is used.

Examples This function returns 2:

```
LEFT("2nd Quarter")
```

This function returns 2nd:

```
LEFT("2nd Quarter", 3)
```

See Also MID, RIGHT

LEN

Description Returns the number of characters in the supplied text string.

Syntax LEN (*text*)

Argument	Description
text	Any text string. Spaces in the string are counted as characters.

Examples This function returns 11:

```
LEN("3rd Quarter")
```

This function returns 3:

```
LEN("1-3")
```

See Also EXACT, SEARCH

LINEST

Description

Computes multiple linear regression for a group of observations relative to a number of independent variables. Also optionally computes statistics related to the regression.

Note This is an array function. For information, see “Array Functions” on page 13.

Syntax

LINEST (*known_y’s* [, *known_x’s*] [, *constant*] [, *statistics*]

Argument	**Description**
known_y’s	A list of observed values for the dependent variable. This must be a range reference or an array constant containing all numeric values and no empty cells. For a regression with a **single independent variable**, enter any type of range. For a regression with **multiple independent variables**, enter a single row or column of values. If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array.
[*known_x’s*]	Optional. A list of observed values for the independent variable(s). This must be a range reference or an array constant containing all numeric values and no empty cells. For a regression with a **single independent variable**, when *known_y’s* is a single row or column, enter a range that exactly matches the size and shape of the *known_y’s* range. When *known_y’s* is more than one row and column, *known_x’s* must be a range containing the same number of values as the *known_y’s* range, although the two ranges may be different shapes. For a regression with **multiple independent variables**, when *known_y’s* is a single row, enter a contiguous row of values for each independent variable, The number of columns must match the number of columns in *known_y’s*. Similarly, when *known_y’s* is a single column, enter a contiguous column of values for each independent variable, ensuring that the number of rows matches *known_y’s*. If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array. To fit the regression to a formula that uses polynomials, see “Polynomials,” below. When you omit this argument, the function assumes a single independent variable of the sequence {1, 2, 3, ...}.

Argument	Description
[*new_x's*]	Optional. A list of values for the independent variable(s) that will compute the results of this function. This must be a range reference or an array constant containing all numeric values and no empty cells. You may enter any number of values in *new_x's*. The shape of the range only matters if this is a regression with multiple independent variables, in which case it must have the same number of rows or colunms of independent variables as *known_x's*. If you omit this argument, the function uses the specified or default values in *known_x's*.
[*constant*]	Optional. A logical value that determines whether or not a constant is to be included. The constant allows the regression line to intercept the Y axis at a point other than (0,0). Use True (the default value if this argument is omitted) to include the constant. Use False to force the line to intercept the Y axis at 0.

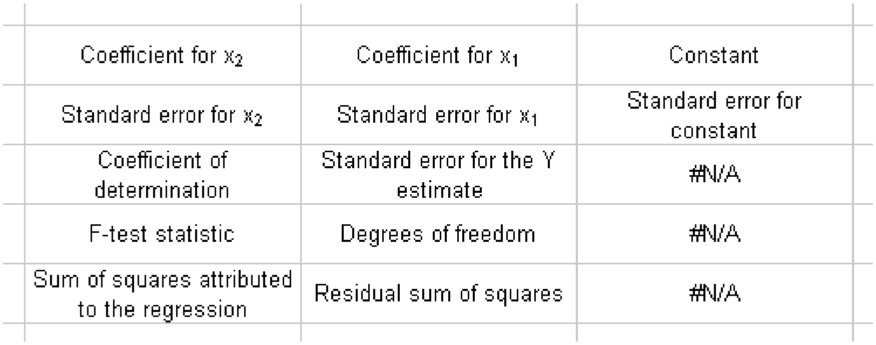
Results range

Before entering this function, select a results range of the following size:

- It must have the same number of columns as your regression has independent variables, plus one. (If your regression has 3 independent variables, your results range must have 4 columns.)
- It must have 1 row if *statistics* is False, 5 rows if *statistics* is true.

Results map

This map shows how LINEST fills the results range with data and statistics.



Coefficient for x_2	Coefficient for x_1	Constant
Standard error for x_2	Standard error for x_1	Standard error for constant
Coefficient of determination	Standard error for the Y estimate	#N/A
F-test statistic	Degrees of freedom	#N/A
Sum of squares attributed to the regression	Residual sum of squares	#N/A

In the last three statistics rows, for results ranges wider than three columns, #N/A will appear in the third and subsequent columns. This is correct.

Coefficients

The coefficients computed by LINEST are displayed in the first row of the results range. For a multiple regression, the coefficients are displayed in the reverse order of the independent variables. (If your regression has 3 independent variables, the coefficient associated with the 1st variable is displayed in the 3rd column, the coefficient associated with the 2nd variable in the 2nd column, and the coefficient associated with the 3rd variable in the 1st column.)

The last cell of the first row of the results range displays the constant value, which will be 0 if *constant* is False.

If there is only one independent variable and *constant* is True, then the two coefficients will equal the results of the SLOPE and INTERCEPT functions when calculated on the same data.

Coefficients of 0 indicate either that there are not enough data points for the number of independent variables, or that some of the independent variables are too closely related.

Equation

The regression computed by LINEST attempts to fit the following formula:

$$y = C_1x_1+C_2x_2+ ... + C_nX_n+b$$

where *C* is the coefficient, *n* is the number of independent variables, and *b* is the constant.

Statistics

If *statistics* is True, the results range displays the following statistics in the following locations.

Standard Error Values row 2	The error values correspond to the coefficients displayed directly above them. This is computed by dividing up the standard error for the Y estimate (see below) into components for each independent variable. It is an estimate of how much the errors in individual independent variable measurements contribute to the overall error.
Coefficient of determination row 3, col 1	This statistic is similar in concept to the correlation coefficient computed by the RSQ function. It is a number between 0 and 1 that measures goodness of fit: low numbers indicate a poor fit, high numbers a good fit.
Standard error for the Y estimate row 3, col 2	This is computed by taking the square root of the residual sum of squares (see below) divided by the degrees of freedom (see below).
F-test statistic row 4, col 1	This statistic can be used with FINV to do null hypothesis testing on the overall goodness of fit of the regression.

Degrees of Freedom row 4, col 2	This is determined by subtracting the number of independent variables from the number of values in *known_y's*, less 1 if *constant* is True. This statistic is used when running FINV and TDIST on the data to judge goodness of fit of the formula.
Sum of squares attributed to the regression row 5, col 1	This is determined by subtracting the residual sum of squares (below) from the total sum of squares of the differences between the values in *known_y's* and $\bar{y}$. This statistic helps determine how well the formula fits the data.
Residual sum of squares row 5, col 2	This is determined by summing the squares of the differences between the values in *known_y's* and the Y values computed by the formula you are trying to fit. It is a measure of the goodness of fit.

Using Statistics

A user who wishes to determine if the formula returned by LINEST is a good fit for the data in the regression should perform the following two tests.

- **Overall goodness of fit.** To determine if the overall regression is a good fit, you can do null hypothesis testing using the F-test statistic returned by LINEST. Compare the F-test statistic to the F-rejection value calculated by running FINV with the probability of your choice (usually .01 or .05), and a denominator value of the degrees of freedom statistic returned by LINEST.

  If the F-test statistic is larger than the F-rejection value, then the null hypothesis is rejected and this is a good fit.

- **Goodness of fit of each independent variable.** After determining that the overall regression is a good fit, test each independent variable to determine whether it adds to the soundness of the regression as a whole. Do this by finding the T-distribution of the T-observed statistic. The T-observed statistic is computed by dividing the coefficient of the variable you want to test by its standard error.

  Find the T-distribution by running TDIST using the absolute value of T-observed, the degrees of freedom statistic returned by LINEST, and a one-tailed distribution.

  If the probability returned by TDIST is larger than the rejection criteria (usually 0.01 or 0.05), then the variable you tested may not be contributing to the regression.

Polynomials

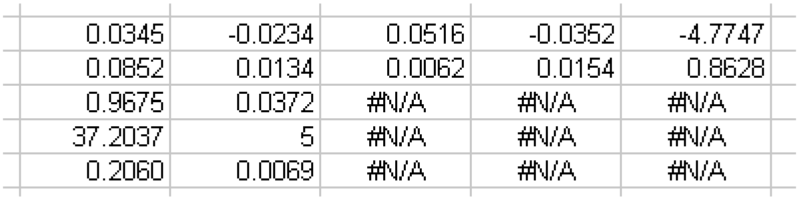
To create a polynomial regression, the data for the independent variable must be set up carefully. Create a worksheet in which column A represents *x* and contains the appropriate data or formulas. For column B, which will contain x^2, fill the cells with a formula such as `=A1^COLUMN()`, which will raise the data in column A to the 2nd power. Fill column C with the same formula to raise the data in column A to the 3rd power.

Example

The following example of multiple linear regression attempts to correlate physiological data on preadolescent boys with their maximal oxygen uptake. The data are given in the table below.

Maximal O2 Uptake	Age (years)	Height (cm)	Weight (kilos)	Chest Depth (cm)
y	x_1	x_2	x_3	x_4
1.54	8.4	132.0	29.1	14.4
1.74	8.7	135.5	29.7	14.5
1.32	8.9	127.7	28.4	14.0
1.50	9.9	131.1	28.8	14.2
1.46	9.0	130.0	25.9	13.6
1.35	7.7	127.6	27.6	13.9
1.53	7.3	129.9	29.0	14.0
1.71	9.9	138.1	33.6	14.6
1.27	9.3	126.6	27.7	13.9
1.50	8.1	131.8	30.8	14.5

The data were entered in a worksheet, the LINEST function was run, and the following results range was returned.



0.0345	-0.0234	0.0516	-0.0352	-4.7747
0.0852	0.0134	0.0062	0.0154	0.8628
0.9675	0.0372	#N/A	#N/A	#N/A
37.2037	5	#N/A	#N/A	#N/A
0.2060	0.0069	#N/A	#N/A	#N/A

See Also

GROWTH, INTERCEPT, LOGEST, SLOPE, TREND

LN

Description

Returns the natural logarithm (based on the constant *e*) of a number.

Syntax

LN (*number*)

Argument	Description
number	Any positive real number.

Remarks

LN is the inverse of the EXP function.

Examples

This function returns 2.50:

```
LN(12.18)
```

This function returns 3.00:

```
LN(20.09)
```

See Also

EXP, LOG, LOG10

LOG

Description

Returns the logarithm of a number to the specified base.

Syntax

LOG (*number* [, *base*])

Argument	Description
number	Any positive real number.
[*base*]	Optional. The base of the logarithm. If this argument is omitted, 10 is used.

Examples

This function returns 0:

```
LOG(1)
```

This function returns 1:

```
LOG(10)
```

See Also

EXP, LN, LOG10

LOG10

Description

Returns the base-10 logarithm of a number.

Syntax

LOG10 (*number*)

Argument	Description
number	Any positive real number.

Examples

This function returns 2.41:

```
LOG10(260)
```

This function returns 2:

```
LOG10(100)
```

See Also EXP, LN, LOG

LOGEST

Description

Computes multiple exponential regression for a group of observations relative to a number of independent variables. Also optionally computes statistics related to the regression.

This type of regression is often applied to growth situations. Because of the type of formula and the often limited amount of data the formula is based on, using these formulas to estimate values outside of the range of data is dangerous, since they are likely to give exponentially wrong results.

Another problematic aspect of this function is that it assumes the range of data in question fits an exponential model. While this assumption is appropriate in some situations (for example, radioactive decay), in many other situations it is misleading, particularly in regards to economic data.

Note This is an array function. For information, see “Array Functions” on page 13.

Syntax

LOGEST (*known_y’s* [, *known_x’s*] [, *constant*] [, *statistics*]

Argument	**Description**
known_y’s	A list of observed values for the dependent variable. This must be a range reference or an array constant containing all numeric values and no empty cells. For a regression with a **single independent variable**, enter any type of range. For a regression with **multiple independent variables**, enter a single row or column of values. If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array.

Argument	Description
[*known_x's*]	Optional. A list of observed values for the independent variable(s). This must be a range reference or an array constant containing all numeric values and no empty cells. For a regression with a **single independent variable**, when *known_y's* is a single row or column, enter a range that exactly matches the size and shape of the *known_y's* range. When *known_y's* is more than one row and column, *known_x's* must be a range containing the same number of values as the *known_y's* range, although the two ranges may be different shapes. For a regression with **multiple independent variables**, when *known_y's* is a single row, enter a contiguous row of values for each independent variable, The number of columns must match the number of columns in *known_y's*. Similarly, when *known_y's* is a single column, enter a contiguous column of values for each independent variable, ensuring that the number of rows matches *known_y's*. If you are using array constants, see "Array Constants in Arguments" on page 9 for information on how to enter the array. To fit the regression to a formula that uses polynomials, see "Polynomials," below. When you omit this argument, the function assumes a single independent variable of the sequence {1, 2, 3, ...}.
[*new_x's*]	Optional. A list of values for the independent variable(s) that will compute the results of this function. This must be a range reference or an array constant containing all numeric values and no empty cells. You may enter any number of values in *new_x's*. The shape of the range only matters if this is a regression with multiple independent variables, in which case it must have the same number of rows or colunms of independent variables as *known_x's*. If you omit this argument, the function uses the specified or default values in *known_x's*.
[*constant*]	Optional. A logical value that determines whether or not a constant is to be included. The constant allows the regression line to intercept the Y axis at a point other than (0,0). Use True (the default value if this argument is omitted) to include the constant. Use False to force the line to intercept the Y axis at 0.

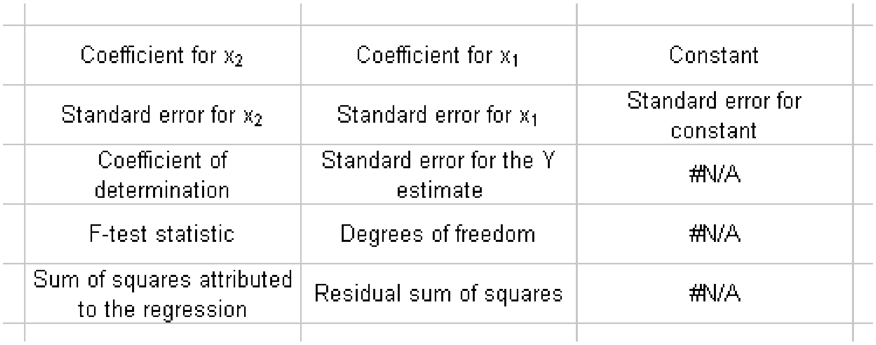
Results range

Before entering this function, select a results range of the following size:

- It must have the same number of columns as your regression has independent variables, plus one. (If your regression has 3 independent variables, your results range must have 4 columns.)
- It must have 1 row if *statistics* is False, 5 rows if *statistics* is true.

Results map

This map shows how LOGEST fills the results range with data and statistics.



Coefficient for x_2	Coefficient for x_1	Constant
Standard error for x_2	Standard error for x_1	Standard error for constant
Coefficient of determination	Standard error for the Y estimate	#N/A
F-test statistic	Degrees of freedom	#N/A
Sum of squares attributed to the regression	Residual sum of squares	#N/A

In the last three statistics rows, for results ranges wider than three columns, #N/A will appear in the third and subsequent columns. This is correct.

Coefficients

The coefficients computed by LOGEST are displayed in the first row of the results range. For a multiple regression, the coefficients are displayed in the reverse order of the independent variables. (If your regression has 3 independent variables, the coefficient associated with the 1st variable is displayed in the 3rd column, the coefficient associated with the 2nd variable in the 2nd column, and the coefficient associated with the 3rd variable in the 1st column.)

The last cell of the first row of the results range displays the constant value, which will be 1 if *constant* is False.

Coefficients of 1 indicate either that there are not enough data points for the number of independent variables, or that some of the independent variables are too closely related.

Equation

The regression computed by LOGEST attempts to fit the following formula:

$$y = C_1^{x_1} \times C_2^{x_2} \times \ldots \times C_n^{x_n} \times b$$

where *C* is the coefficient, *n* is the number of independent variables, and *b* is the constant.

This formula can be converted to a linear form if logarithms are taken of both sides. The formula then becomes:

$$y = lC_1x_1+lC_2x_2+ \ldots + lC_nX_n+lb$$

This is the same formula fitted by the LINEST function. LOGEST uses LINEST internally to fit the linearized form of the equation. It then takes the LINEST coefficients (lC_1, lC_2 ,... lC_n) and converts them to the LOGEST coefficients (C_1, C_2, ... C_n) by applying the EXP function: C_1 = EXP(lC_1). The same conversion is done with the constant, *b.*

Statistics

If *statistics* is True, the results range displays the following statistics in the following locations. Please note that these statistics apply to the linearized form of the equation, as explained above.

Standard Error Values row 2	The error values correspond to the coefficients displayed directly above them. This is computed by dividing up the standard error for the Y estimate (see below) into components for each independent variable. It is an estimate of how much the errors in individual independent variable measurements contribute to the overall error.
Coefficient of determination row 3, col 1	This statistic is similar in concept to the correlation coefficient computed by the RSQ function. It is a number between 0 and 1 that measures goodness of fit: low numbers indicate a poor fit, high numbers a good fit.
Standard error for the Y estimate row 3, col 2	This is computed by taking the square root of the residual sum of squares (see below) divided by the degrees of freedom (see below).
F-test statistic row 4, col 1	This statistic can be used to do null hypothesis testing on the overall goodness of fit of the regression.
Degrees of Freedom row 4, col 2	This is determined by subtracting the number of independent variables from the number of values in *known_y's*, less 1 if *constant* is True.
Sum of squares attributed to the regression row 5, col 1	This is determined by subtracting the residual sum of squares (below) from the total sum of squares of the differences between the values in *known_y's* and $\bar{y}$. This statistic helps determine how well the formula fits the data.
Residual sum of squares row 5, col 2	This is determined by summing the squares of the differences between the values in *known_y's* and the Y values computed by the formula you are trying to fit. It is a measure of the goodness of fit.

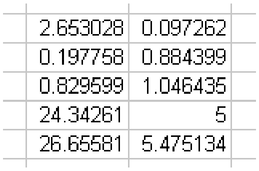
Example

This example of exponential regression uses the following data on the growth of the national debt of a small country.

Year	National debt, in millions
1971	0.5
1972	1.1
1973	1.4
1974	2.0
1975	4.4
1976	20.0
1977	445.0

In this case, the debt figures are the *known_y's* and we can use default values for the *known_x's*, since the years increment by one.

If the national debt figures fill range A2:A8, the function `LOGEST(A2:A8,,TRUE,TRUE)`returns the following results range:



2.653028	0.097262
0.197758	0.884399
0.829599	1.046435
24.34261	5
26.65581	5.475134

See Also GROWTH, INTERCEPT, LINEST, SLOPE, TREND

LOGINV

Description Computes the inverse of the cumulative lognormal distribution of *x*, where the logarithm of *x* is normally distributed.

Syntax LOGINV (*prob, mean, st_dev*)

Argument	Description
prob	The probability, as computed by the LOGNORMDIST function.
mean	The arithmetic mean of the logarithm of *x*.
st_dev	The standard deviation of the logarithm of *x*.

Remarks LOGINV assumes the logarithm of *x* is normally distributed, meaning that, if you graph the distribution with a logarithmic X axis, the bell-shaped curve will appear.

Equation

$$\mathrm{LOGINV}(prob, \mu, \sigma) = \mathrm{EXP}(\mu + \sigma \times \mathrm{NORMSINV}(prob))$$

Example This function returns 16.0002:

```
LOGINV(0.223218,18,20)
```

See Also LOGNORMDIST

LOGNORMDIST

Description Computes the cumulative lognormal distribution for *x*, where the logarithm of *x* is normally distributed.

Syntax LOGNORMDIST (*x, mean, st_dev*)

Argument	Description
x	The value at which you want to evaluate the function. It must be greater than or equal to 0.
mean	The arithmetic mean of the logarithm of *x*.
st_dev	The standard deviation of the logarithm of *x*.

Remarks

LOGNORMDIST assumes the logarithm of *x* is normally distributed, meaning that, if you graph the distribution with a logarithmic X axis, the bell-shaped curve will appear.

Equation

$$\mathrm{LOGNORMDIST}\ (x, \mu, \sigma) = \left(\frac{\mathrm{LN}(x) - \mu}{\sigma}\right)$$

Example

This function returns 0.2232:

```
LOGNORMDIST(16,18,20)
```

See Also

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, GAMMADIST, GAMMAINV, LOGINV, NORMDIST, NORMSDIST, POISSON

LOOKUP

Description

Searches for a value in one range and returns the contents of the corresponding position in a second range.

Syntax

LOOKUP (*lookup_value*, *lookup_range*, *result_range*)

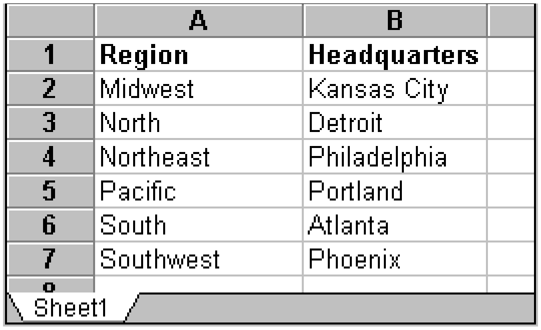
Argument	Description
lookup_value	The value for which to search in the first range.
lookup_range	The first range to search and contains only one row or one column. The range can contain numbers, text, or logical values. To search *lookup_range* correctly, the expressions in the range must be placed in ascending order (for example, –2, –1, 0, 1, 2 ... A through Z, False, True). The search is not case-sensitive.
result_range	A range of one row or one column that is the same size as *lookup_range*.

Remarks

If *lookup_value* does not have an exact match in *lookup_range*, the largest value that is less than or equal to *lookup_value* is found and the corresponding position in *result_range* is returned. When *lookup_value* is smaller than the data in *lookup_range*, #N/A is returned.

Examples

The following examples use this worksheet.



	A	B
1	**Region**	**Headquarters**
2	Midwest	Kansas City
3	North	Detroit
4	Northeast	Philadelphia
5	Pacific	Portland
6	South	Atlanta
7	Southwest	Phoenix

This function returns Detroit:

```
LOOKUP("North", A2:A7, B2:B7)
```

This function returns #N/A:

```
LOOKUP("Alabama", A2:A7, B2:B7)
```

See Also HLOOKUP, INDEX (non-array type), VLOOKUP

LOWER

Description Changes the characters in the specified string to lowercase characters. Numeric characters in the string are not changed.

Syntax LOWER (*text*)

Argument	Description
text	Any string.

Examples This function returns 3rd quarter:

```
LOWER("3rd Quarter")
```

This function returns john doe:

```
LOWER("JOHN DOE")
```

See Also PROPER, UPPER

MATCH

Description

A specified value is compared against values in a range. The position of the matching value in the search range is returned.

Syntax

MATCH (*lookup_value*, *lookup_range*, *comparison*)

Argument	Description
lookup_value	The value against which to compare. It can be a number, text, or logical value or a reference to a cell that contains one of those values.
lookup_range	The range to search and contains only one row or one column. The range can contain numbers, text, or logical values.
comparison	A number that represents the type of comparison to be made between *lookup_value* and the values in *lookup_range*. When you omit this argument, comparison method 1 is assumed. When *comparison* is 1, the largest value that is less than or equal to *lookup_value* is matched. When using this comparison method, the values in *lookup_range* must be in ascending order (for example, ...–2, –1, 0, 1, 2..., A through Z, False, True). When *comparison* is 0, the first value that is equal to *lookup_value* is matched. When using this comparison method, the values in *lookup_range* can be in any order. When *comparison* is –1, the smallest value that is greater than or equal to *lookup_value* is matched. When using this comparison method, the values in *lookup_range* must be in descending order (for example, True, False, Z through A, ...2, 1, 0, –1, –2...).

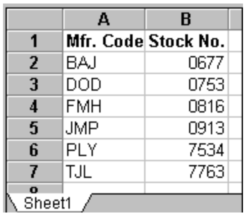
Remarks

When using comparison method 0 and *lookup_value* is text, *lookup_value* can contain wildcard characters. The wildcard characters are * (asterisk), which matches any sequence of characters, and ? (question mark), which matches any single character.

When no match is found for *lookup_value,* #N/A is returned.

Examples

The following examples use this worksheet.



	A	B
1	**Mfr. Code**	**Stock No.**
2	BAJ	0677
3	DOD	0753
4	FMH	0816
5	JMP	0913
6	PLY	7534
7	TJL	7763

This function returns 5:

```
MATCH(7600, B2:B7,1)
```

This function returns 2:

```
MATCH("D*", A2:A7,0)
```

See Also HLOOKUP, INDEX (non-array type), LOOKUP, VLOOKUP

MAX

Description Returns the largest value in the specified list of numbers.

Syntax MAX (*number_list*)

Argument	Description
number_list	A list of up to 30 numbers separated by commas. The list may contain numeric values, cell references, range references, or array constants. Text in and logical values in cell references and arrays are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Examples This function returns 500:

```
MAX(50, 100, 150, 500, 200)
```

This function returns the largest value in the range:

```
MAX(A1:F12)
```

See Also AVERAGE, MIN

MAXA

Description Returns the largest value in the specified list of numbers.

This function is equivalent to the MAX function, but its implementation treats text and logical values in cell and range references differently.

Syntax MAXA (*number_list*)

Argument	Description
number_list	A list of between 1 and 30 arguments consisting of numbers, cell references, range references, and/or array constants. Text in cells referenced by this function is treated as the number 0 (this includes zero-length text). Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Text and logical values in arrays are ignored. Logical values referenced in cells or entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Examples

This function returns 500:

```
MAXA(50,100,150,"500",200)
```

This function returns 1:

```
MAXA(TRUE,FALSE)
```

See Also

AVERAGE, AVERAGEA, MAX, MIN, MINA

MDETERM

Description

Computes the determinant of a square matrix.

Syntax

MDTERM (*array*)

Argument	Description
array	An array of numbers or a range of cells.

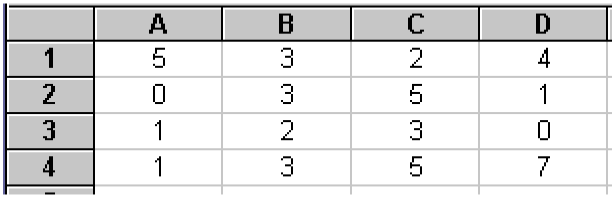
Remarks

The determinant can be used as a measurement of the singularity of a matrix. If the matrix represents the coefficients of a set of simultaneous equations, and the determinant is relatively small, this may indicate that one or more pairs of equations (i.e., rows) are closely related and the solution of the simultaneous equations may not be very accurate (if it is computed at all). A value of 0 indicates that the matrix is singular and not solvable (see MINVERSE).

Note In contrast to the closely related functions MINVERSE and MMULT, the result of MDETERM is a single value. MDETERM is not an array function.

Examples

The first example uses this worksheet.



	A	B	C	D
1	5	3	2	4
2	0	3	5	1
3	1	2	3	0
4	1	3	5	7

This function returns 33:

```
MDETERM(A1:D4)
```

This function returns 33:

```
MDETERM({5,3,2,4;0,3,5,1;1,2,3,0;1,3,5,7})
```

See Also MINVERSE, MMULT

MDURATION

Description Computes the modified duration for a security.

Syntax MDURATION (*settlement, maturity, rate, yield, frequency* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text. Numbers will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Dates in the argument list must be in the form of a serial number or text. Numbers will be truncated to integers.
rate	The security's annual coupon rate. The coupons pay at this rate divided by *frequency.*
yield	The security's annual yield.
frequency	The number of interest payments per year. See "The frequency Argument" on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Remarks Modified duration is a direct measure of the sensitivity of a bond's price to a change in its yield.

Equation

$$\frac{\text{DURATION}\ (\textit{settlement, maturity, rate, yield, frequency, calendar\_type})}{1 + \dfrac{\textit{yield}}{\textit{frequency}}}$$

Examples This function returns 7.2465:

```
MDURATION("3/17/89","3/17/99",0.07,0.08,2,1)
```

See Also DURATION

MEDIAN

Description Computes the median or middle value of a list of numbers arranged in ascending order.

Syntax MEDIAN (*number_list*)

Argument	Description
number_list	A list of between 1 and 30 arguments consisting of numbers, cell references, range references, and/or array constants. All numeric values, including 0, are used. Text and logical values in arrays and cells referenced by this function are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Remarks If there is an odd number of values in *number_list*, the middle value is returned. If there is an even number of values, the average of the two middle values is returned.

Example This function returns 3:

```
MEDIAN(1,2,3,4,5)
```

This function returns 12:

```
MEDIAN(24,15,9,6,2,76,11,13)
```

See Also AVERAGE, MODE

MID

Description Returns the specified number of characters from a text string, beginning with the specified starting position.

Syntax MID (*text*, *start_position*, *num_chars*)

Argument	Description
text	The string from which to return characters.

Argument	Description
start_position	The position of the first character to return from *text*. If *start_position* is 1, the first character in *text* is returned. If *start_position* is greater than the number of characters in *text*, an empty string ("") is returned. If *start_position* is less than 1, #VALUE! is returned.
num_chars	The number of characters to return. If *num_chars* is negative, #VALUE! is returned.

Remarks

If *start_position* plus the number of characters in *num_chars* exceeds the length of *text*, the characters from *start_position* to the end of *text* are returned.

Examples

This function returns Expenses:

```
MID("Travel Expenses", 8, 8)
```

This function returns 45:

```
MID("Part #45-7234", 7, 2)
```

See Also

CODE, FIND, LEFT, RIGHT, SEARCH

MIN

Description

Returns the smallest value in the specified list of numbers.

Syntax

MIN (*number_list*)

Argument	Description
number_list	A list of up to 30 numbers separated by commas. The list may contain numeric values, cell references, range references, or array constants. Text in and logical values in cell references and arrays are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Examples

This function returns 50:

```
MIN(50, 100, 150, 500, 200)
```

This function returns the smallest value in the range:

```
MIN(A1:F12)
```

See Also

AVERAGE, MAX

MINA

Description

Returns the smallest value in the specified list of numbers.

This function is equivalent to the MIN function, but its implementation treats text and logical values in cell and range references differently.

Syntax

MINA (*number_list*)

Argument	Description
number_list	A list of between 1 and 30 arguments consisting of numbers, cell references, range references, and/or array constants. Text in cells referenced by this function is treated as the number 0 (this includes zero-length text). Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Text and logical values in arrays are ignored. Logical values referenced in cells or entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Examples

This function returns 50:

```
MINA(50,100,150,"500",200)
```

This function returns 0:

```
MINA(TRUE,FALSE)
```

See Also

AVERAGE, AVERAGEA, MAX, MAXA, MIN

MINUTE

Description

Returns the minute that corresponds to the supplied date.

Syntax

MINUTE (*serial_number*)

Argument	Description
serial_number	The time as a serial number. The decimal portion of the number represents time as a fraction of the day.

Remarks

The result is an integer ranging from 0 to 59.

Examples

This function returns 36:

```
MINUTE(34506.4)
```

This function returns 48:

```
MINUTE(34399.825)
```

See Also DAY, HOUR, MONTH, NOW, SECOND, WEEKDAY, YEAR

MINVERSE

Description Computes the inverse of a square matrix.

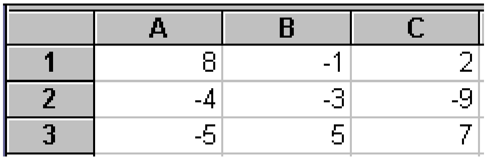
Note This is an array function. For information, see “Array Functions” on page 13.

Syntax MINVERSE (*array*)

Argument	Description
array	An array of numbers or range of cells.

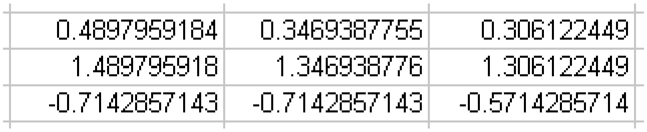
Remarks The inverse is often used to solve series of simultaneous equations. The inverse has the property that when it is multiplied (using MMULT) by the original matrix, the identity matrix is the result--1's on the diagonal, 0's (or very small numbers because of round-off errors in computers) in all other locations.

Examples The first example uses this worksheet.



	A	B	C
1	8	-1	2
2	-4	-3	-9
3	-5	5	7

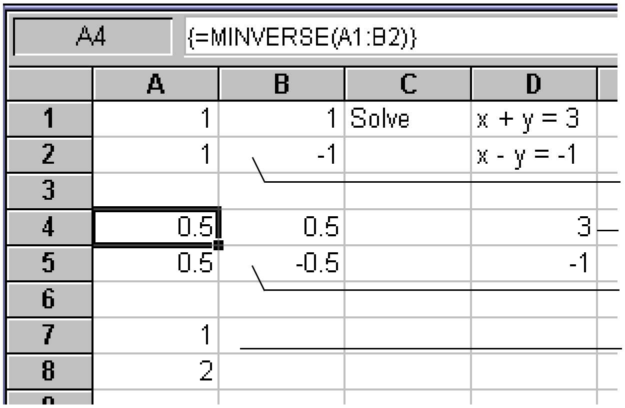
`MINVERSE(A1:C3)` returns the following results range:



0.4897959184	0.3469387755	0.306122449
1.489795918	1.346938776	1.306122449
-0.7142857143	-0.7142857143	-0.5714285714

Example Problem

Given: x + y = 3 and x - y = 2, solve for x and y:



A4 {=MINVERSE(A1:B2)}

	A	B	C	D
1	1	1	Solve	x + y = 3
2	1	-1		x - y = -1
3				
4	0.5	0.5		3
5	0.5	-0.5		-1
6				
7	1			
8	2			

Coefficient Array Matrix

Constant Array Matrix

Inverse of the Coefficient Array Matrix

The result array--product of MMULT(A4:B5,D4:D5)

➤ **To solve using MINVERSE and MMULT:**

1. Enter the coefficient and constant array matrices.
2. Select a four-cell area (A4:B5 in the example) to hold the results of the formula.
3. Type in the formula MINVERSE (A1:B2).
4. Press CTRL + SHIFT + ENTER to enter the array formula so that it applies to all four cells in the selected area.
5. Select a two-cell column area where you want the answer to appear.
6. Multiply the inverse coefficient array matrix and the constant array matrix by typing MMULT(A4:B5,D4:D5), then pressing CTRL + SHIFT + ENTER.

See Also

MDETERM, MMULT

MIRR

Description

Returns the modified internal rate of return for a series of periodic cash flows.

Syntax

MIRR (*cash_flows*, *finance_rate*, *reinvest_rate*)

Argument	Description
cash_flow	A reference to a range that contains values for which to calculate the modified internal rate of return. The values must contain at least one positive and one negative value. During calculation, MIRR uses the order in which the values appear to determine the order of cash flow. Values that represent cash received should be positive; negative values represent cash paid. Text, logical values, and empty cells in the range are ignored.

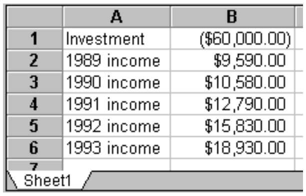
Argument	Description
finance_rate	The interest rate paid on money used in the cash flow.
reinvest_rate	The interest rate received on money reinvested from the cash flow.

Remarks

The modified internal rate of return considers the cost of the investment and the interest received on the reinvestment of cash.

Examples

The following examples use this worksheet.



	A	B
1	Investment	($60,000.00)
2	1989 income	$9,590.00
3	1990 income	$10,580.00
4	1991 income	$12,790.00
5	1992 income	$15,830.00
6	1993 income	$18,930.00
7		

This function returns 5.20 percent:

```
MIRR(B1:B6, 12%, 8%)
```

This function returns –40.93 percent:

```
MIRR(B1:B3, 12%, 8%)
```

See Also

IRR, NPV, RATE

MMULT

Description

Performs matrix multiplication on two arrays.

Note This is an array function. For information, see “Array Functions” on page 13.

Syntax

MMULT (*array1, array2*)

Argument	Description
array1 and *array2*	Two range references. All cells must contain numeric values or the function will return the #VALUE! error. The number of rows in *array2* must be the same as the number of columns in *array1*.

Results range

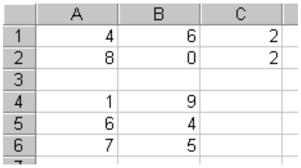
Before entering this function, select a results range with the same number of rows as *array1* and the same number of columns as *array2*. If you select too few cells, some cells will be truncated. If you select too many cells, the extra cells will contain the #N/A error.

Remarks

To fill the results range, MMULT computes SUMPRODUCT (see page 238) with the first row of *array1* and first column of *array2* as the arguments. The result goes in the first row and column of the results range. MMULT then computes SUMPRODUCT on the second row of *array1* and the first column of *array2*, and puts the result in the second row, first column of the results range. The function proceeds in this manner until the results range is filled.

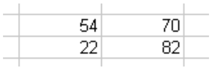
Examples

The following example uses this worksheet.



	A	B	C
1	4	6	2
2	8	0	2
3			
4	1	9	
5	6	4	
6	7	5	

`MMULT(A1:C2,A4:B6)` returns the following results range:



54	70
22	82

MOD

Description

Returns the remainder after dividing a number by a specified divisor.

Syntax

MOD (*number*, *divisor*)

Argument	Description
number	Any number.
divisor	Any nonzero number. If *divisor* is 0, #DIV/0! is returned.

Examples

This function returns 1:

```
MOD(-23, 3)
```

This function returns –2:

```
MOD(-23, -3)
```

See Also

INT, ROUND, TRUNC

MODE

Description

Computes the most frequent value in a list of numbers (the number that appears most often in the list).

Syntax

MODE (*number_list*)

Argument	Description
number_list	A list of between 1 and 30 arguments consisting of numbers, cell references, range references, and/or array constants. All numeric values, including 0, are used. Text, logical values, and empty cells are ignored.

Remarks If no number is repeated in *number_list*, the error value #N/A is returned.

Example This function returns 3:

```
MODE(1,2,3,3,4)
```

See Also AVERAGE, MEDIAN

MONTH

Description Returns the month that corresponds to the supplied date.

Syntax MONTH (*serial_number*)

Argument	Description
serial_number	The date as a serial number or as text (for example, 06-21-94 or 21-Jun-94).

Remarks MONTH returns a number ranging from 1 (January) to 12 (December).

Examples This function returns 6:

```
MONTH("06-21-94")
```

This function returns 10:

```
MONTH(34626)
```

See Also DAY, HOUR, MINUTE, NOW, SECOND, TODAY, WEEKDAY, YEAR

MROUND

Description Rounds a specified number to an even integral multiple of a specified factor.

Syntax MROUND (*number, factor*)

Argument	Description
number	The value to be rounded. *Number* must have the same sign as *factor.*
factor	Lowest factor of desired result after rounding. *Factor* must have the same sign as *number.*

Remarks If the remainder is exactly half of *factor,* the result is rounded away from 0. (See examples.)

Examples This function returns 14:

```
MROUND(13,2)
```

This function returns -3.3:

```
MROUND(-3.8,-1.1)
```

See Also ODD, EVEN, ROUND, ROUNDUP, ROUNDDOWN

MULTINOMIAL

Description Computes the multinomial of up to 29 numbers.

Syntax MULTINOMIAL (*number*)

Argument	Description
number	A list of as many as 29 arguments. The list can contain numbers, ranges containing values, or array constants. Decimal values are truncated to integers.

Equation

$$\text{MULTINOMIAL}(a + b + c) = \frac{(a + b + c)!}{(a! + b! + c!)}$$

Examples This function returns 12600:

```
MULTINOMIAL(1,2,3,4)
```

This function returns 12600:

```
MULTINOMIAL(1.9,2.9,3.9,4.9)
```

See Also FACT

N

Description Tests the supplied value and returns the value if it is a number.

Syntax N (*value*)

Argument	Description
value	A value or a reference to a cell containing a value to test.

Remarks Numbers are returned as numbers, serial numbers formatted as dates are returned as serial numbers, and the logical function TRUE() is returned as 1. All other expressions return 0.

Examples This function returns 32467:

```
N(32467)
```

This function returns 1 if A4 contains the logical function TRUE:

```
N(A4)
```

See Also T, VALUE

NA

Description Returns the error value #N/A, which represents “not available.”

Syntax NA ( )

Remarks Use NA to mark cells that lack data without leaving them empty. Empty cells may not be correctly represented in some calculations.

Although NA does not use arguments, you must supply the empty parentheses to correctly reference the function.

See Also ISNA

NEGBINOMDIST

Description Computes the negative binomial distribution, which is used to determine probabilities on repeated tests where each test is independent of every other test (that is, the probability is the same for each test).

Syntax NEGBINOMDIST (*failures, threshold, probability*)

Argument	Description
failures	The number of failures you expect in the trial. It must be a positive integer or 0.
threshold	The number of successes the trial requires. It must be a positive integer. It may not be 0.
probability	The chance of success on an individual attempt in the trial. It must be a number larger than 0 and smaller than 1. For example, the probability of throwing heads on a coin toss is 0.5.

Remarks

Coin tosses or die rolls are examples of uses for NEGBINOMDIST. The probability returned measures the likelihood that there will be a certain number of failures before a certain number of successes occurs. This can be used to determine how many tests to run to achieve the desired results.

NEGBINOMDIST is similar to BINOMDIST, only in NEGBINOMDIST the number of successes is fixed and the number of trials is variable, while in BINOMDIST the number of successes is variable and the number of trials is fixed.

Equation

$$\binom{f+t-1}{t-1}p^t(1-p)^f$$

...where *f* is *failures*, *t* is *threshold*, and *p* is *probability.*

Example

This function returns 0.1875:

```
NEGBINOMDIST(2,2,0.5)
```

See Also

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, EXPONDIST, GAMMADIST, NORMDIST, POISSON

NETWORKDAYS

Description

Computes the number of whole working days between the specified start date and the specified end date.

Syntax

NETWORKDAYS (*start_date*, *end_date*, [, *holidays*])

Argument	Description
start_date	First date of the period.
end_date	Last date of the period.
holidays	Optional. A list containing numbers, a range of cells, or an array constant representing the dates of holidays.

Remarks

Counts both the first and last days if appropriate.

Excludes weekends and any days in the list of holidays.

Useful for computations that are dependent on the actual hours or days worked in a time period.

Examples This function returns 260:

```
NETWORKDAYS(1,365)
```

See Also WEEKDAY

NOMINAL

Description Computes the nominal annual interest rate, given the effective interest rate. This will remove the effect of yearly compounding on the specified interest rate.

Syntax NOMINAL (*effective_rate, periods*)

Argument	Description
effective_rate	The interest rate that reflects the effect of compounding.
periods	The number of compounding periods per year. Decimal values are truncated to integers.

Examples This function returns 0.0675:

```
NOMINAL(0.0696,12)
```

This function returns 0.0684:

```
NOMINAL(0.0696,2)
```

See Also EFFECT

NORMDIST

Description Computes the normal distribution.

Syntax NORMDIST (*x, mean, st_dev, cumulative*)

Argument	Description
x	The value at which you want to evaluate the function.
mean	The arithmetic mean of the distribution.
st_dev	The standard deviation of the distribution. It must be a number greater than 0.

Argument	Description
cumulative	A logical value indicating the type of computation you want NORMDIST to do. TRUE The function will return the cumulative area from negative infinity to *x*. FALSE The function will return the y value for *x*.

Remarks

You may use the STANDARDIZE function to “translate” the first three arguments of this function into a form that can be used in the NORMSDIST function.

Equation

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Examples

This function returns 0.5:

```
NORMDIST (50, 50, 4, TRUE)
```

See Also

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, GAMMADIST, GAMMAINV, NORMINV, NORMSDIST, POISSON

NORMINV

Description

Computes the inverse of the cumulative normal distribution. This is the inverse of the NORMDIST function when the last argument is TRUE.

Syntax

NORMINV (*probability, mean, st_dev*)

Argument	Description
probability	A number between 0 and 1 that indicates the probability associated with the cumulative normal distribution.
mean	The arithmetic mean of the distribution.
st_dev	The standard deviation of the distribution. It must be a number greater than 0.

Examples

This function returns 21.64:

```
NORMINV (0.95, 20, 1)
```

See Also

NORMDIST

NORMSDIST

Description

Computes the standard normal cumulative distribution (often called “the bell curve”).

Syntax

NORMSDIST (*x*)

Argument	Description
x	The value at which you want to evaluate the function.

Remarks

This is a version of the NORMDIST function that assumes the mean of the distribution is 0 and the standard deviation is 1. You can adjust values from a distribution with a different mean and/or standard deviation by plugging those values into the STANDARD function. The result of the STANDARD function is the argument for NORMSDIST.

Equation

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Examples

This function returns 0.89:

```
NORMSDIST (1.25)
```

See Also

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, LOGNORMDIST, NORMDIST, NORMSINV, POISSON

NORMSINV

Description

Computes the inverse of the standard normal distribution. This is the inverse of the NORMSDIST function.

Syntax

NORMSINV (*probability*)

Argument	Description
probability	A number between 0 and 1 that indicates the probability associated with the standard normal distribution.

Remarks

This

Examples

This function returns 0.25:

```
NORMSINV (0.599)
```

See Also

NORMSDIST

NOT

Description

Returns a logical value that is the opposite of its value.

Syntax

NOT (*logical*)

Argument	Description
logical	An expression that returns a logical value such as True or False.

Remarks

If *logical* is false, NOT returns True. Conversely, if *logical* is true, NOT returns False.

Examples

This function returns False:

```
NOT(TRUE())
```

This function returns False:

```
NOT(MONTH("12/25/94") = 12)
```

See Also

AND, IF, OR

NOW

Description

Returns the current date and time as a serial number.

Syntax

NOW ( )

Remarks

In a serial number, numbers to the left of the decimal point represent the date; numbers to the right of the decimal point represent the time. The result of this function changes only when a recalculation of the worksheet occurs.

See Also

DATE, DAY, HOUR, MINUTE , MONTH, SECOND, TODAY, WEEKDAY, YEAR

NPER

Description

Returns the number of periods of an investment based on regular periodic payments and a fixed interest rate.

Syntax

NPER (*interest*, *pmt*, *pf* [, *fv*] [, *type*])

Argument	Description
interest	The fixed interest rate.

Argument	Description
pmt	The fixed payment made each period. Generally, *pmt* includes the principle and interest, not taxes or other fees.
pf	The present value, the lump-sum amount that a series of future payments is currently worth.
[*fv*]	Optional. The future value, the balance to attain after the final payment. If this argument is omitted, 0 is used.
[*type*]	Optional. Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. If this argument is omitted, 0 is used.

Examples

This function returns 36.67:

```
NPER(12%/12, -350, -300, 16000, 1)
```

This function returns 36.98:

```
NPER(1%, -350, -300, 16000)
```

See Also

FV, CUMIPMT, CUMPRINC, IPMT, PMT, PPMT, PV, RATE

NPV

Description

Returns the net present value of an investment based on a series of periodic payments and a discount rate.

Syntax

NPV (*discount_rate*, *value_list*)

Argument	Description
discount_rate	The rate of discount for one period.
value_list	A list of as many as 29 arguments or a reference to a range that contains values that represent payments and income. During calculation, NPV uses the order in which the values appear to determine the order of cash flow. Numbers, empty cells, and text representations of numbers are included in the calculation. Errors and text that cannot be translated into numbers are ignored. If *value_list* is a range reference, only numeric data in the range is included in the calculation. Other types of data in the range, such as empty cells, logical values, text, and error values, are ignored.

Remarks

The time span NPV uses for calculation begins one period before the first cash flow date and ends when the last cash flow payment is made. This function is based on future cash flows. When your first cash flow occurs at the beginning of the first period, the first value must be added to the NPV result, not supplied as a value in *value_list*.

Example

This function returns 811.57:

```
NPV(8%, -12000, 3000, 3000, 3000, 7000)
```

See Also

FV, IRR, PV

OCT2BIN

Description

Converts an octal number (base 8) to a binary number (base 2).

Syntax

OCT2BIN (*integer* [, *places*])

Argument	Description
integer	Any negative or positive whole number or zero. For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between 0 and 777 may be used.
places	Optional number of characters to use in the output. If *places* is omitted, the function returns the number of spaces required to display the result. The *places* argument can be used to 0-pad a positive number. If the number is negative, *places* is ignored and 10 characters are returned.

Remarks

Note that the result of this function is a text string.

Examples

This function returns 1000:

```
OCT2BIN(10)
```

This function returns 111111111:

```
OCT2BIN(777)
```

See Also

BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2DEC, OCT2HEX

OCT2DEC

Description Converts an octal number (base 8) to a decimal number (base 10).

Syntax OCT2DEC (*integer*)

Argument	Description
integer	Any negative or positive whole number or zero. For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between 0 and 3,777,777,777 may be used.

Remarks Note that the result of this function is a number.

Examples This function returns 8:

```
OCT2DEC(10)
```

This function returns 536870911:

```
OCT2DEC(3777777777)
```

See Also BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2HEX

OCT2HEX

Description Converts an octal number (base 8) to a hexadecimal number (base 16).

Syntax OCT2HEX (*integer* [, *places*])

Argument	Description
integer	Any negative or positive whole number or zero. For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between 0 and 3,777,777,777 may be used.
places	Optional number of characters to use in the output. If *places* is omitted, the function returns the number of spaces required to display the result. The *places* argument can be used to 0-pad a positive number. If the number is negative, *places* is ignored and 10 characters are returned.

Remarks The result of this function is a text string.

Examples This function returns 8:

```
OCT2HEX(10)
```

This function returns 1FFFFFFF:

```
OCT2HEX(3777777777)
```

See Also BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC

ODD

Description Rounds the specified number up to the nearest odd integer.

Syntax ODD (*number*)

Argument	Description
number	Any number, a formula that evaluates to a number, or a reference to a cell that contains a number.

Examples This function returns 5:

```
ODD(3.5)
```

This function returns 7:

```
ODD(6)
```

See Also CEILING, EVEN, FLOOR, INT, ROUND, TRUNC

ODDFPRICE

Description Computes the price of a security purchased during a first coupon period that is shorter or longer than the other coupon periods. In addition to the dates, you must specify the rate, redemption value, and yield expected.

Syntax ODDFPRICE (*settlement, maturity, issue, first_coup, rate, yield, redemption, frequency* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. It must be before *first_coup*. Decimal values will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. Decimal values will be truncated to integers.
issue	The date the security becomes effective.
first_coup	The date of the first interest payment.

Argument	Description
rate	The security’s annual coupon rate.
yield	The annual income produced by the security.
redemption	The security’s redemption value per $100 face value. This is the amount paid at *maturity* that is not part of any final coupon payment.
frequency	The number of interest payments per year. See “The frequency Argument” on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Remarks

This function is useful in cases where the security’s issuer does not pay interest for the first few months or years after issuing the security.

This function should be used only when the security’s first coupon date is a different length than the other coupon dates AND the security was purchased during this odd coupon period.

If all coupon periods are of equal length, use PRICE. If the last coupon period is shorter or longer than the rest, use ODDLPRICE.

Equation

$$\left[\frac{redemption}{\left(1+\frac{yield}{frequency}\right)^{(N-1)+NQ+FQC}}\right]+\left[\sum_{k=1}^{N}\frac{100\times\frac{rate}{frequency}}{\left(1+\frac{yield}{frequency}\right)^{(k-1)+NQ+FQC}}\right]$$

...where the codes correspond to values computed using other functions, as shown in the following table.

Note The software divides the odd first period into non-paying coupon periods in order to correctly calculate the price. These non-paying periods are referred to as pseudo-periods. The variable *i* represents the number of pseudo-periods. The first pseudo-period may be shorter than the rest of the pseudo-periods.

Code	Meaning	Function
N	Number of coupons payable between *first_coup* and *settlement*.	COUPNUM
NQ	Number of whole coupon periods in the pseudo-period. Those having a settlement date on a pseudo-period date are not considered whole, but the fraction (FQC) is 1.0.	COUPDAYSNC
FQC	Fraction of the coupon period between *settlement* and next pseudo-period.	COUPNUM

Examples

This function returns 108.0131:

```
ODDFPRICE("1/15/93","1/1/98","1/1/93","3/1/93",0.07,0.06,100,2)
```

See Also

ODDFYIELD, ODDLPRICE, ODDLYIELD, PRICE, YIELD

ODDFYIELD

Description

Computes the yield of a security (per $100 face value) purchased during a first coupon period that is shorter or longer than the other coupon periods. In addition to the dates, you must specify the rate, redemption value, and price.

Syntax

ODDFYIELD (*settlement, maturity, issue, first_coup, rate, price, redemption, frequency* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. It must be before *first_coup*. Decimal values will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. Decimal values will be truncated to integers.
issue	The date the security becomes effective.
first_coup	The date of the first interest payment.
rate	The security's annual coupon rate.
price	The price paid by the buyer at *settlement*.
redemption	The security's redemption value per $100 face value. This is the amount paid at *maturity* that is not part of any final coupon payment.
frequency	The number of interest payments per year. See "The frequency Argument" on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Remarks

This function is useful in cases where the security's issuer does not pay interest for the first few months or years after issuing the security.

This function should be used only when the security's first coupon date is a different length than the other coupon dates AND the security was purchased during this odd coupon period.

If all coupon periods are of equal length, use YIELD. If the last coupon period is shorter or longer than the rest, use ODDLYIELD.

Examples

This function returns 0.075014:

```
ODDFYIELD("1/15/93","1/1/98","1/1/93","3/1/93",0.07,98,100,2)
```

See Also

ODDFPRICE, ODDLPRICE, ODDLYIELD, PRICE, YIELD

ODDLPRICE

Description

Computes the price of a security purchased during a last coupon period that is shorter or longer than the other coupon periods. In addition to the dates, you must specify the rate, redemption value, and yield expected.

Syntax

ODDLPRICE (*settlement*, *maturity*, *last_coupon*, *rate*, *yield*, *redemption*, *frequency* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. It must be after *last_coupon*. Decimal values will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. Decimal values will be truncated to integers.
last_coupon	The date of the last interest payment.
rate	The security's annual coupon rate. The coupons pay at this rate divided by *frequency*.
yield	The annual income produced by the security.
redemption	The security's redemption value per $100 face value. This is the amount paid at the settlement date that is not part of any final coupon payment.
frequency	The number of interest payments per year. See "The frequency Argument" on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Remarks

This function should be used only when the security's last coupon date is a different length than the other coupon dates AND the security was purchased during this odd coupon period.

If all coupon periods are of equal length, use PRICE. If the first coupon period is shorter or longer than the rest, use ODDFPRICE.

Equation

$$\frac{redemption + fracDCi \times 100 \times \frac{rate}{frequency}}{\left(1 + fracDSCi \times \frac{yield}{freuency}\right) - \left(fracAi \times 100 \times \frac{rate}{frequency}\right)}$$

...where the codes correspond to values you can compute using the coupon functions, as shown in the following table.

Note The software divides the odd last period into non-paying coupon periods in order to correctly calculate the price. These non-paying periods are referred to as pseudo-periods. The variable *i* represents the pseudo-periods.

Code	**Meaning**	**Function**
fracDCi	Number of coupon periods from *last_coupon* to *maturity*. This number may be greater or less than 1.	COUPNUM
fracDSCi	Number of coupon periods from *settlement* to *maturity*. This number may be greater or less than 1.	COUPNUM
fracAi	Number of coupon periods from *last_coupon* to *settlement.* This number may be greater or less than 1.	COUPNUM

Examples

This function returns 1.416517:

```
ODDLPRICE("4/19/97","11/25/97","4/1/97",0.05,99.8,100,4)
```

See Also

ODDFPRICE, ODDFYIELD, ODDLYIELD, PRICE, YIELD

ODDLYIELD

Description

Computes the yield of a security (per $100 face value) purchased during a last coupon period that is shorter or longer than the other coupon periods. In addition to the dates, you must specify the rate, redemption value, and price.

Syntax

ODDLYIELD (*settlement, maturity, last_coupon, rate, price, redemption, frequency* [*, calendar_type*])

Argument	**Description**
settlement	The date when the security is traded to the buyer. It must be after *last_coupon*. Decimal values will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. Decimal values will be truncated to integers.
last_coupon	The date of the last interest payment.
rate	The security’s annual coupon rate.
price	The price paid by the buyer at *settlement.*
redemption	The security’s redemption value per $100 face value. This is the amount paid at *maturity* that is not part of any final coupon payment.
frequency	The number of interest payments per year. See “The frequency Argument” on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Remarks

This function should be used only when the security’s last coupon date is a different length than the other coupon dates AND the security was purchased during this odd coupon period.

If all coupon periods are of equal length, use YIELD. If the first coupon period is shorter or longer than the rest, use ODDFYIELD.

Equation

$$\left[\frac{\left(redemption + fracDCi \times 100 \times \frac{rate}{frequency}\right) - \left(value + fracAi \times 100 \times \frac{rate}{frequency}\right)}{value + fracAi \times 100 \times \frac{rate}{frequency}}\right]$$

$$\times \frac{frequency}{fracDSCi}$$

...where the codes correspond to values you can compute using the coupon functions, as shown in the following table.

Note The software divides the odd last period into non-paying coupon periods in order to correctly calculate the price. These non-paying periods are referred to as pseudo-periods. The variable *i* represents the pseudo-periods.

Code	Meaning	Function
fracDCi	Number of coupon periods from *last_coupon* to *maturity*. This number may be greater or less than 1.	COUPNUM
fracAi	Number of coupon periods from *last_coupon* to *settlement*. This number may be greater or less than 1.	COUPNUM
fracDSCi	Number of coupon periods from *settlement* to *maturity*. This number may be greater or less than 1.	COUPNUM

Examples

This function returns 0.079064:

```
ODDLYIELD("6/1/96","1/1/98","1/1/96",.07,98.4,100,2)
```

See Also

ODDFPRICE, ODDFYIELD, ODDLPRICE, PRICE, YIELD

OFFSET

Description

Returns the contents of a range that is offset from a starting point in the spreadsheet.

Syntax

OFFSET (*reference*, *rows*, *columns* [, *height*] [, *width*])

Argument	Description
reference	A reference to a cell from which the offset reference is based. If you specify a range reference, #VALUE! is returned.

Argument	Description
rows	The number of rows from *reference* that represents the upper-left cell of the offset range. A positive number represents rows below the starting cell; a negative number represents rows above the starting cell. If *rows* places the upper-left cell of the offset range outside the spreadsheet boundary, #REF! is returned.
columns	The number of columns from *reference* that represents the upper-left cell of the offset range. A positive number represents columns right of the starting cell; a negative number represents columns left of the starting cell. If *columns* places the upper-left cell of the offset range outside the spreadsheet boundary, #REF! is returned.
[*height]*	Optional. A positive number representing the number of rows to include in the offset range. If this argument is omitted, 1 is used.
[*width*]	Optional. A positive number representing the number of columns to include in the offset range. If this argument is omitted, 1 is used.

Remarks

OFFSET does not change the current selection in the worksheet. Because it returns a reference, OFFSET can be used in any function that requires or uses a cell or range reference as an argument.

Examples

This function returns the contents of cell D4:

```
OFFSET(B1, 3, 2, 1, 1)
```

This function returns the sum of the values in the range E3:F5:

```
SUM(OFFSET(A1, 2, 4, 3, 2))
```

OR

Description

Returns True if at least one of a series of logical arguments is true.

Syntax

OR (*logical_list*)

Argument	Description
logical_list	A list of conditions separated by commas. You can include as many as 30 conditions in the list. The list can contain logical values or a reference to a range containing logical values. Text and empty cells are ignored. If there are no logical values in the list, the error value #VALUE! is returned.

Example

This function returns True because one of the arguments is true:

```
OR(1 + 1 = 1, 5 + 5 = 10)
```

See Also

AND, IF, NOT

PEARSON

Description

Computes the correlation coefficient for two sets of numbers, paired up one-to-one.

The correlation coefficient is a number between -1 and 1 (inclusive) that measures the “relatedness” of the numbers in the samples. A coefficient of 1 indicates a direct relationship in which all points are linearly related on a line with positive slope. A coefficient of -1 indicates an inverse relationship in which a large value in the first argument pairs with a small value in the second argument. A coefficient of 0 indicates no relationship between the pairs of values, or complete randomness.

Note The PEARSON function is exactly the same as the CORREL function. We provide both in order to be compatible with all the Microsoft Excel functions.

Syntax

PEARSON (*array1*, *array2*)

Argument	Description
array1 and *array2*	A range reference or array constant containing numeric values. *Array1* and *array2* must contain the same potential number of values. The function will return the error value #DIV/0! if *array1* or *array2* contains non-numeric data (text, logical values, or blank cells).

Remarks

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

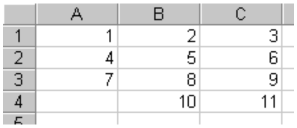
Equation

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_i - \bar{y})^2}}$$

where $\bar{x}$ is the mean of the x_i's and $\bar{y}$ is the mean of they_i's.

Examples

The following examples use this worksheet.



	A	B	C
1	1	2	3
2	4	5	6
3	7	8	9
4		10	11
5			

This function returns 1:

```
PEARSON(A1:A3,B1:B3)
```

This function returns 0.991117:

```
PEARSON(A1:B2,C1:C4)
```

See Also CORREL, COVAR

PERCENTILE

Description Computes the value corresponding to the specified percentile of a specified range of numbers.

Syntax PERCENTILE (*array*, *percentile*)

Argument	Description
array	A list of numbers, in the form of a range reference or array constant. Text and logical values are ignored.
percentile	The percentile for which you want to find the numeric value. It must be a number between 0 and 1. Text in the argument list that converts to a number will be evaluated as a number; otherwise the function will return the #VALUE! error. Text referenced in cells will return the #VALUE! error. Logical values are evaluated as 1 if TRUE and 0 if FALSE.

Remarks To determine the value, the function sorts *array* in ascending order. Then the function assigns a percentile to each number in *array*: the lowest number is the 0th percentile, the highest number is the 100th percentile, and the middle number is 50th percentile.

If *percentile* falls on a value in *array*, the function returns that value.

If *percentile* does not fall on a value in *array*, the function calculates the value that would fall at *percentile*, given the range values above and below that percentile.

This function is the inverse of the PERCENTRANK function.

Examples This function returns 2:

```
PERCENTILE({1,2,3,4,5}, .25)
```

This function returns 2.32:

```
PERCENTILE({1,2,3,4,5}, .33)
```

See Also PERCENTRANK, QUARTILE, RANK

PERCENTRANK

Description

Computes the percent rank of a specified value in a specified array of numbers.

Syntax

RANK (*array, number* [, *digits*])

Argument	Description
array	A list of at least 2 numbers, in the form of a range reference or array constant. Text and logical values are ignored.
number	The number you want ranked. It must be within the range of values in *array.* Text in the argument list that converts to a number will be evaluated as a number; otherwise the function will return the #VALUE! error. Text referenced in cells will return the #VALUE! error. Logical values are evaluated as 1 if TRUE and 0 if FALSE.
[*digits*]	Optional. The number of decimal digits you want the answer to be rounded to. It must be a numeric value between 1 and 308. If this argument is omitted, 3 is used.

Remarks

To determine percent rank, the function sorts *array* in ascending order. Then the function assigns a percent rank to each number in *array*: the lowest number is 0% rank, the highest number is 100% rank, and the middle number is 50% rank.

If *number* is in *array*, the function returns the percent rank assigned to *number.*

If *number* is not in *array*, the function calculates the percent rank that falls proportionately between the percent ranks above and below *number.*

This function is the inverse of the PERCENTILE function.

Examples

This function returns .5:

```
PERCENTRANK ({1, 2, 3, 4, 5}, 3)
```

This function returns .375:

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

See Also

PERCENTILE, QUARTILE, RANK

PERMUT

Description

Computes the number of combinations possible by taking *k* items at a time from a pool of *n*, where order in the sample taken is important.

For example, in a box of four different items, the number of combinations possible taking out 2 items at a time is 12; that is, the twelve combinations of items 1-2, 1-3, 1-4, 2-1, 2-3, 2-4, 3-1, 3-2, 3-4, 4-1, 4-2, and 4-3.

Syntax

PERMUT (*number, chosen*)

Argument	Description
number	A positive integer representing the total number of items in the pool of items. Decimal numbers are truncated to integers.
chosen	A positive integer representing the number of items taken from the pool at a time. It must be less than or equal to *number*. Decimal values are truncated to integers.

Remarks

This function is similar to the COMBIN function, except that PERMUT requires the samples to be ordered, while COMBIN takes samples in any order.

Equation

$$n(n-1)(n-2)\ldots(n-k+1) = \frac{n!}{(n-k)!}$$

where *n* is the total number of items and *k* is the number of items taken at a time.

Examples

This function returns 12:

```
PERMUT(4,2)
```

This function returns 30240:

```
PERMUT(10, 5)
```

See Also

COMBIN, GAMMALN

PI

Description

Returns the value of pi, which is approximately 3.14159265358979 when calculated to 15 significant digits.

Syntax

PI ( )

Remarks

Although PI does not use arguments, you must supply the empty parentheses to correctly reference the function.

See Also

COS, SIN, TAN

PMT

Description

Returns the periodic payment of an annuity, based on regular payments and a fixed periodic interest rate.

Syntax

PMT (*interest*, *nper*, *pv* [, *fv*] [, *type*])

Argument	Description
interest	The fixed periodic interest rate.
nper	The number of periods in the annuity.
pv	The present value, or the amount the annuity is currently worth.
[*fv*]	The future value, or the amount the annuity will be worth. If this argument is omitted, 0 is used.
[*type*]	Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. If this argument is omitted, 0 is used.

Remarks

PMT returns only the principal and interest payment, it does not include taxes or other fees.

The units used for *interest* must match those used for *nper.* For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for *interest* and 5*12 for *nper.*

Cash paid out, such as a payment, is shown as a negative number. Cash received, such as a dividend check, is shown as a positive number.

Examples

This function returns –439.43:

```
PMT(8%/12, 48, 18000)
```

This function returns –436.52:

```
PMT(8%/12, 48, 18000, 0, 1)
```

See Also

FV, CUMIPMT, CUMPRINC, IPMT, NPER, PPMT, PV, RATE

POISSON

Description

Computes the Poisson distribution, which is usually used to determine the probability of a certain number of repeated events taking place over time.

For example, you could use POISSON to calculate the probability that more than the average number of calls will pass through a telephone switching station over an hour's period.

Syntax

POISSON (*x*, *mean*, *cumulative*)

Argument	Description
x	A positive integer indicating the number of events you want to test the probability of happening over a given period. Decimal values will be rounded down to the nearest integer. Logical values are interpreted as 1 for TRUE and 0 for FALSE.
mean	A positive number indicating the average or expected number of events for the given test period. Logical values are interpreted as 1 for TRUE and 0 for FALSE.
cumulative	A logical value. True or 1 indicates you want to find the probability of at least *x* events happening. False or 0 indicates you want to find the probability of exactly *x* events happening.

Equations

when *cumulative* is FALSE: $f(x) = \frac{\mu^x}{x!}e^{-\mu}$

when *cumulative* is TRUE: $\sum_{j=0}^{x} e^{-\mu}\left(\frac{\mu^j}{j!}\right)$

Examples

This function returns .57, or a 57% probability:

```
POISSON(15,15,1)
```

See Also

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, GAMMADIST, NORMDIST

POWER

Description

Raises the specified base number to the specified power.

Syntax

POWER (*base*, *exponent*)

Argument	Description
base	Any number. Nonnumeric values will return the #VALUE! error.
exponent	Any number. Nonnumeric values will return the #VALUE! error. If *base* is negative, *exponent* must be an integer, except in the case when *exponent* is larger than -1, smaller than 1, and corresponds to an odd root (for example, $^1/_3$ and $^1/_5$). If *base* is 0, *exponent* must be larger than 0. If *exponent* is 0, *base* may not be 0.

Examples

This function returns 9:

```
POWER(3,2)
```

This function returns 0.001:

```
POWER(10,-3)
```

PPMT

Description

Returns the principle paid on an annuity for a given period.

Syntax

PPMT (*interest*, *per*, *nper*, *pv*, [*fv*], [*type*])

Argument	**Description**
interest	The fixed periodic interest rate.
per	The period for which to return the principle.
nper	The number of periods in the annuity.
pv	The present value, or the amount the annuity is currently worth.
[*fv*]	The future value, or the amount the annuity will be worth. If this argument is omitted, 0 is used.
[*type*]	Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. If this argument is omitted, 0 is used.

Remarks

The units used for *interest* must match those used for *nper*. For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for *interest* and 5*12 for *nper*.

Examples

This function returns –321.56:

```
PPMT(8%/12, 2, 48, 18000)
```

This function returns –319.43:

```
PPMT(8%/12, 2, 48, 18000, 0, 1)
```

See Also

FV, CUMIPMT, CUMPRINC, IPMT, NPER, PMT, PV, RATE

PRICE

Description Computes the price of a security with specified rate, redemption value, and yield.

Syntax PRICE (*settlement, maturity, rate, yield, redemption, frequency* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Decimal values will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Decimal values will be truncated to integers.
rate	The security's annual coupon rate. The coupons pay at this rate divided by *frequency.*
yield	The annual income produced by the security.
redemption	The security's redemption value per $100 face value. This is the amount paid at the settlement date that is not part of any final coupon payment.
frequency	The number of interest payments per year. See "The frequency Argument" on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Remarks Use this function when all coupon periods between *settlement* and *maturity* are of equal length. If the first coupon period is shorter or longer than the rest, use ODDFPRICE. If the last coupon period is shorter or longer than the rest, use ODDLPRICE.

Equation

$$\left[\frac{redemption}{\left(1+\frac{yield}{frequency}\right)^{\left(\mathrm{NCP}-1+\frac{\mathrm{DSC}}{\mathrm{DIC}}\right)}}\right]+\left[\sum_{\mathrm{DCS}=1}^{\mathrm{NCP}}\frac{100\times\frac{rate}{frequency}}{\left(1+\frac{yield}{frequency}\right)^{\left(\mathrm{DCS}-1+\frac{\mathrm{DSC}}{\mathrm{DIC}}\right)}}\right]$$

$$-\left(100\times\frac{rate}{frequency}\times\frac{\mathrm{DCS}}{\mathrm{DIC}}\right)$$

where the three-letter codes correspond to values you can compute using the coupon functions, as shown in the following table.

Code	Meaning	Function
DCS	Number of days from beginning of the coupon period to *settlement*.	COUPDAYBS
DIC	Number of days in coupon period in which *settlement* falls.	COUPDAYS

Code	Meaning	Function
DSC	Number of days from *settlement* to the next coupon.	COUPDAYSNC
NCP	Number of coupons payable between *settlement* and *redemption*.	COUPNUM

Examples

This function returns 90.35:

```
PRICE("4/19/97","11/25/01",0.05,0.075,100,4)
```

See Also

PRICEDISC, PRICEMAT, YIELD, YIELDDISC, YIELDMAT

PRICEDISC

Description

Computes the price of a discounted security per $100 face value, given a specified discount rate and redemption value.

Syntax

PRICEDISC (*settlement*, *maturity*, *discount*, *redemption* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Decimal values will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Decimal values will be truncated to integers.
discount	The security's annual discount rate.
redemption	The security's redemption value per $100 face value. This is the amount paid at *maturity*.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Equation

$$\textit{redemption} - \textit{discount} \times \textit{redemption} \times \text{YEARFRAC}(\textit{settlement}, \textit{maturity}, \textit{calendar\_type})$$

Examples

This function returns 56.01:

```
PRICEDISC("5/7/92","12/31/99",0.0575,100)
```

See Also

DISC, PRICE, PRICEMAT, YIELD, YIELDDISC, YIELDMAT

PRICEMAT

Description

Computes the price of a security that pays interest only at maturity, given specified dates, interest rate, and yield. The price computed is per $100 face value.

Syntax

PRICEMAT (*settlement, maturity, issue, rate, yield* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Decimal values will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be the same day or later than *settlement*. Decimal values will be truncated to integers.
issue	The date the security was originally issued and began earning interest. Decimal values will be truncated to integers.
rate	The security's annual interest rate at date of issue.
yield	The annual income produced by the security. This can be considered the effective interest rate for the buyer.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Equation

$$\frac{100 + \left(\frac{\text{DIM}}{\text{DIY}} \times rate \times 100\right)}{1 + \left(\frac{\text{DSM}}{\text{DIY}} \times yield\right)} - \left(\frac{\text{DIS}}{\text{DIY}} \times rate \times 100\right)$$

...where the three-letter codes correspond to values you can compute using the YEARFRAC function, as shown in the following table.

Code	Meaning	Function and arguments
DIM/DIY	Days from *issue* to *maturity.*	YEARFRAC (*issue, maturity, calendar_type*)
DIS/DIY	Days from *issue* to *settlement.*	YEARFRAC (*issue, settlement, calendar_type*)
DSM/DIY	Days from *settlement* to *maturity.*	YEARFRAC (*settlement, maturity, calendar_type*)

Examples

This function returns 82.11:

```
PRICEMAT("4/8/93","2/14/99","2/14/89",0.045,0.075)
```

See Also

PRICE, PRICEDISC, YIELD, YIELDDISC, YIELDMAT

PROB

Description

Given a set of numbers and a probability associated with each number, computes the probability corresponding to a specified number. Also computes all the probabilities corresponding to a specified range of numbers.

Syntax

PROB (*arrayX*, *probabilities*, *test* [, *upper*])

Argument	Description
arrayX	A range reference or array constant containing numeric values. There must be the same number of values in *probabilities* as in *arrayX*. The function will return the error value #NUM! if *arrayX* contains non-numeric data (text, logical values, or blank cells).
probabilities	A range reference or array constant containing the probabilities associated with the numbers in *arrayX*. Each probability must be a number between 0 and 1. All values in *probabilities* must add up to 1. There must be the same number of values in *probabilities* as in *arrayX*. The function will return the error value #NUM! if *probabilities* contains non-numeric data (text, logical values, or blank cells).
test	The value from *arrayX* whose probability you want to find. If *test* is not found in *arrayX* and no value is entered for *upper*, the function returns 0.
[*upper*]	Optional. Specifies the upper limit of a range of numbers from *arrayX* whose cumulative probabilities you want to know. *Test* is the lower limit, *upper* the upper limit, inclusive. PROB will return the sum of all values in *probabilities* that correspond to the numbers in *arrayX* that fall between or include *test* and *upper.*

Remarks

When *arrayX* and *probabilities* are range references, the function pairs up the numbers by moving left-to-right through each sequential row.

Examples

This function returns 0.05:

```
PROB ({1,2,3,4,5}, {0.05, 0.1, 0.2, 0.3, 0.35}, 1)
```

This function returns 0.3:

```
PROB ({1,2,3,4,5}, {0.05, 0.1, 0.2, 0.3, 0.35}, 2, 3)
```

This function returns 0.1:

```
PROB ({1,2,3,4,5}, {0.05, 0.1, 0.2, 0.3, 0.35}, 1.5, 2.5)
```

See Also

COVAR, PEARSON

PRODUCT

Description Multiplies a list of numbers and returns the result.

Syntax PRODUCT (*number_list*)

Argument	Description
number_list	A list of as many as 30 numbers, separated by commas. The list can contain numbers, logical values, text representations of numbers, or a reference to a range containing those values. Error values or text that cannot be translated into numbers return errors. If a range reference is included in the list, text, logical expressions, and empty cells in the range are ignored. All numeric values, including 0, are used in the calculation.

Example This function returns 24:

```
PRODUCT(1, 2, 3, 4)
```

See Also FACT, FACTDOUBLE, PRODUCT, SUM

PROPER

Description Returns the specified string in proper-case format.

Syntax PROPER (*text*)

Argument	Description
text	Any string.

Remarks In proper-case format, the first alphabetic character in a word is capitalized. If an alphabetic character follows a number, punctuation mark, or space, it is capitalized. All other alphabetic characters are lowercase. Numbers are not changed by PROPER.

Examples This function returns 3Rd Quarter:

```
PROPER("3rd Quarter")
```

This function returns John Doe:

```
PROPER("JOHN DOE")
```

See Also LOWER, UPPER

PV

Description

Returns the present value of an annuity, considering a series of constant payments made over a regular payment period.

Syntax

PV (*interest*, *nper*, *pmt* [, *fv*] [, *type*])

Argument	Description
interest	The fixed periodic interest rate.
nper	The number of payment periods in the investment.
pmt	The fixed payment made each period.
[*fv*]	The future value, or the amount the annuity will be worth. If this argument is omitted, 0 is used.
[*type*]	Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. If this argument is omitted, 0 is used.

Remarks

The units used for *interest* must match those used for *nper*. For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for *interest* and 5*12 for *nper*.

Cash paid out, such as a payment, is shown as a negative number. Cash received, such as a dividend check, is shown as a positive number.

Examples

This function returns –17999.89:

```
PV(8%/12, 48, 439.43)
```

This function returns 17999.89:

```
PV(8%/12, 48, -439.43)
```

See Also

FV, CUMIPMT, CUMPRINC, IPMT, NPER, PMT, PPMT, RATE

QUARTILE

Description

Computes the value corresponding to the specified quartile in a specified array of numbers.

Syntax

QUARTILE (*array*, *quartilerank*)

Argument	Description
array	A list of numbers, in the form of a range reference or array constant. Text and logical values are ignored.

Argument	Description
quartilerank	An integer in the set {0, 1, 2, 3, 4} that specifies in which of the quartile divisions you want to find a value. Decimal values are rounded down to the nearest integer. Text in the argument list that converts to one of the numbers in the set will be evaluated as a number; otherwise the function will return the #VALUE! error. Text referenced in cells will return the #VALUE! error. Logical values are evaluated as 1 if TRUE and 0 if FALSE.

Remarks

To determine the value, the function sorts *array* in ascending order. Then the function divides *array* into four parts or quartiles. The lowest value in the array is assigned the 0th quartile and the highest value is the 4th quartile. If the array contains an odd number of values, the middle value is assigned the 2nd quartile; otherwise, a value halfway between the two middle values is calculated and assigned the 2nd quartile. Values corresponding to the 1st and 3rd quartiles are either assigned or calculated in the same manner.

The function returns the value it assigned to or calculated for the *quartilerank* value you chose.

Examples

This function returns 57:

```
QUARTILE({12,13,57,89,90}, 2)
```

This function returns 5:

```
QUARTILE({1,1,9,9}, 2)
```

See Also

PERCENTILE, PERCENTRANK

QUOTIENT

Description

Computes the quotient of two numbers, truncated to an integer (toward 0).

Syntax

QUOTIENT (*dividend, divisor*)

Argument	Description
dividend	The number to be divided.
divisor	The number by which to divide. Any number other than zero.

Examples

This function returns 1:

```
QUOTIENT(9,7)
```

See Also

MOD, INT

RADIANS

Description Converts a value in degrees to radians.

Syntax RADIANS (*degrees*)

Argument	Description
degrees	A number describing the angle in degrees that you want to convert to radians. Nonnumeric entries cause a #VALUE! error.

Equation

$$degrees\left(\frac{\pi}{180}\right)$$

Examples This function returns -3.141592654:

```
RADIANS(-180)
```

This function returns 0:

```
RADIANS(0)
```

See Also PI, DEGREES

RAND

Description Returns a number selected randomly from a uniform distribution greater than or equal to 0 and less than 1 each time the worksheet is recalculated.

Syntax RAND ( )

Remarks Although RAND does not use arguments, you must supply the empty parentheses to correctly reference the function.

Formula One for Java will regenerate this random number every time the worksheet is recalculated. Each recalculation will generate only a single random number that is used during all of a recalc. Multiple copies of this function will generate different random numbers.

Example This function returns a random number greater than or equal to 0 and less than 10:.

```
RAND()*10
```

See Also RANDBETWEEN

RANDBETWEEN

Description

Returns an integer selected randomly from between two specified limits, inclusive, each time the sheet is recalculated.

Syntax

RANDBETWEEN (*bottom, top*))

Argument	Description
bottom	A number that marks the bottom of the range of values this function will use. It must be smaller than *top*. If *bottom* is an integer, it may be returned by RANDBETWEEN. Otherwise, only integers above it can be returned.
top	A number that marks the top of the range of values this function will use. It must be larger than *bottom*. If *top* is an integer, it may be returned by RANDBETWEEN. Otherwise, only integers below it can be returned

Remarks

Formula One for Java will regenerate this random number every time the worksheet is recalculated.

Example

This function returns a random integer greater than or equal to 14 and less than or equal to 27:

```
RANDBETWEEN(14,27)
```

See Also

RAND

RANK

Description

Computes the rank of a specified value in a specified array of numbers. By default, the array is sorted in descending order, but you may choose to sort it in ascending order.

Syntax

RANK (*number, array* [,*order*])

Argument	Description
number	The member of *array* that you want ranked. If the function finds more than one of *number* in *array,* it will return the rank of the first instance it finds. If the function doesn’t find *number* in *array,* it will return the #N/A! error. Text in the argument list that converts to a number will be evaluated as a number; otherwise the function will return the #VALUE! error. Text referenced in cells will return the #VALUE! error. Logical values are evaluated as 1 if TRUE and 0 if FALSE.
array	A list of from 2 to 8191 numbers, in the form of a range reference or array constant. Text and logical values are ignored.

Argument	Description
[*order*]	Optional. Specifies whether you want the array sorted in ascending or descending order. Enter 0 to sort the list in descending order. Enter any nonzero number to sort the list in ascending order. If this argument is omitted, 0 is used.

Examples

This function returns 5:

```
RANK (3, {5, 1, 8, 3, 0, 4, 8, 3, 2})
```

This function returns 4:

```
RANK (3, {5, 1, 8, 3, 0, 4, 8, 3, 2},1)
```

See Also

PERCENTILE, PERCENTRANK, QUARTILE

RATE

Description

Returns the interest rate per period of an annuity, given a series of constant cash payments made over a regular payment period.

Syntax

RATE (*nper, pmt, pv* [, *fv*] [, *type*] [, *guess*])

Argument	Description
nper	The number of periods in the annuity.
pmt	The fixed payment made each period. Generally, *pmt* includes only principle and interest, not taxes or other fees.
pv	The present value of the annuity.
[*fv*]	Optional. The future value, or the amount the annuity will be worth. If this argument is omitted, 0 is used.
[*type*]	Optional. Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. If this argument is omitted, 0 is used.
[*guess*]	Optional. Your estimate of the interest rate. If this argument is omitted, 0.1 (10 percent) is used.

Remarks

RATE is calculated iteratively, cycling through the calculation until the result is accurate to .00001 percent. If the result cannot be found after 20 iterations, #NUM! is returned. When this occurs, supply a different value for *guess*.

Example

The following example returns the monthly interest rate of .0067; the annual interest rate (.0067 multiplied by 12) is 8 percent:

```
RATE(48, -439.43, 18000)
```

See Also

FV, CUMIPMT, CUMPRINC, IPMT, NPER, PMT, PPMT, PV

RECEIVED

Description

Computes the amount received at maturity for a fully invested security. Investment amount and discount rate must be specified.

Syntax

RECEIVED (*settlement, maturity, investment, discount* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Decimal values will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Decimal values will be truncated to integers.
investment	The amount invested in the security.
discount	The security's discount rate. Note that this is not the security's yield.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Equation

$$\frac{\textit{investment}}{1\text{- }(\textit{discount} \times \text{YEARFRAC }(\textit{settlement, maturity, calendar\_type}))}$$

Examples

This function returns 63.745:

```
RECEIVED("1/1/94","10/1/97",50,0.0575)
```

See Also

INTRATE, PRICEDISC, YIELD

REGISTER.ID

Description

Returns the #N/A error message.

Syntax

REGISTER.ID (*arguments*)

Argument	Description
arguments	2 to 3 arguments of varying types

Remarks

This function was included only for compatibility with Microsoft Excel. In Excel, REGISTER.ID extracts the identification number from a dynamic linked library or code resource, neither of which is supported in Java.

This function is intended only for users or developers who want to import Excel worksheets that contain Excel's REGISTER.ID function. To make those worksheets work properly, developers should convert these functions to add-in functions, which they can write in Java and set up to be automatically loaded.

For more information, see the chapter on creating add-in functions in the *Formula One for Java Technical Guide*.

REPLACE

Description Replaces part of a text string with another text string.

Syntax REPLACE (*orig_text*, *start_position*, *num_chars*, *repl_text*)

Argument	Description
orig_text	The original text string.
start_position	The character position where the replacement begins. If *start_position* is greater than the number of characters in *orig_text*, *repl_text* is appended to the end of *orig_text*. If *start_position* is less than 1, #VALUE! is returned.
num_chars	The number of characters to replace. If this argument is negative, #VALUE! is returned.
repl_text	The replacement text string.

Examples This function returns “For the year: 1994”:

```
REPLACE("For the year: 1993", 18, 1, "4")
```

See Also MID, SEARCH, TRIM

REPT

Description Repeats a text string the specified number of times.

Syntax REPT (*text*, *number*)

Argument	Description
text	Any text string.
number	The number of times you want *text* to repeat. If *number* is 0, empty text ("") is returned.

Remarks The result of REPT cannot exceed 255 characters.

Example This function returns error-error-error-:

```
REPT("error-", 3)
```

RIGHT

Description Returns the rightmost characters from the given text string.

Syntax RIGHT (*text* [, *num_chars*])

Argument	Description
text	Any text string.
[*num_chars*]	Optional. The number of characters to return. The value must be greater than or equal to zero. If *num_chars* is greater than the number of characters in *text*, the entire string is returned. If this argument is omitted, 1 is used.

Examples This function returns r:

```
RIGHT("2nd Quarter")
```

This function returns Quarter:

```
RIGHT("2nd Quarter", 7)
```

See Also LEFT, MID

ROMAN

Description Converts an arabic number to a Roman numeral.

Syntax ROMAN (*number* [, *type*])

Argument	Description
number	A number between 0 and 3999. Any nonnumeric value will return the #VALUE! error. Decimal values are truncated to integers.
[*type*]	Optional. One of five types or styles of roman numerals. The options are 0, 1, 2, 3, or 4. If this argument is omitted, 0 is used. Type 0 is the classical type taught in schools. The differences between the styles have to do with whether the function allows the letters I (1), V (5), and X (10) to appear before the letters L (50), C (100), D (500), and M (1000). See the examples below. The logical value TRUE is equivalent to type 0; the logical value FALSE is equivalent to type 4.

Examples

This function returns CDXCIX:

```
ROMAN(499)
```

This function returns LDVLIV:

```
ROMAN(499, 1)
```

This function returns XDIX:

```
ROMAN(499, 2)
```

This function returns VDIV:

```
ROMAN(499, 3)
```

This function returns ID:

```
ROMAN(499, 4)
```

See Also BIN2DEC, BIN2HEX, BIN2OCT, OCT2BIN, OCT2DEC, OCT2HEX

ROUND

Description Rounds the given number to the supplied number of decimal places.

Syntax ROUND (*number, precision*)

Argument	Description
number	Any value.
precision	The number of decimal places to which *number* is rounded. When a negative precision is used, the digits to the right of the decimal point are dropped and the absolute number of significant digits specified by *precision* are replaced with zeros. If *precision* is 0, *number* is rounded to the nearest integer.

Example

This function returns 123.46:

```
ROUND(123.456, 2)
```

This function returns 9900:

```
ROUND(9899.435, -2)
```

See Also CEILING, FLOOR, INT, MOD, ROUNDDOWN, ROUNDUP, TRUNC

ROUNDDOWN

Description Rounds a number down.

Syntax ROUNDDOWN (*number*, *number_of_digits*)

Argument	Description
number	Any real number you want to round.
number_of_digits	The number of decimal places to which *number* is rounded. When a negative precision is used, the digits to the right of the decimal point are dropped and the absolute number of significant digits specified by *precision* are replaced with zeros. If *precision* is 0, *number* is rounded down to the nearest integer.

Example This function returns 31.141:

```
ROUNDDOWN(3.14159, 3)
```

This function returns 31.400:

```
ROUNDDOWN(31415.92654, -2)
```

See Also CEILING, FLOOR, INT, MOD, ROUND, ROUNDUP, TRUNC

ROUNDUP

Description Rounds the given number up to the supplied number of decimal places.

Syntax ROUNDUP (*number*, *number_of_digits*)

Argument	Description
number	Any value you want to round up.
number_of_digits	The number of decimal places to which *number* is rounded. When a negative precision is used, the digits to the right of the decimal point are dropped and the absolute number of significant digits specified by *precision* are replaced with zeros. If *precision* is 0, *number* is rounded up to the nearest integer.

Example This function returns 77:

```
ROUNDUP(76.9,0)
```

This function returns 31500:

```
ROUNDUP(31415.92654, -2)
```

See Also CEILING, FLOOR, INT, MOD, ROUND, ROUNDDOWN, TRUNC

ROW

Description Returns the row number of the supplied reference.

Syntax ROW (*reference*)

Argument	Description
reference	A cell or range reference. Omitting this argument returns the row number of the cell in which ROW is entered.

Examples This function returns 3:

```
ROW(B3)
```

See Also COLUMN, ROWS

ROWS

Description Returns the number of rows in a range reference.

Syntax ROWS (*range*)

Argument	Description
range	A reference to a range of cells.

Examples This function returns 5:

```
ROWS(A1:D5)
```

This function returns 6:

```
ROWS(C30:F35)
```

See Also COLUMNS, ROW

RSQ

Description

Computes the square of the correlation coefficient for two sets of numbers, paired up one-to-one.

The correlation coefficient is a number between -1 and 1 (inclusive) that measures the “relatedness” of the numbers in the samples. A coefficient of 1 indicates a direct relationship in which all points are linearly related on a line with positive slope. A coefficient of -1 indicates an inverse relationship in which a large value in the first argument pairs with a small value in the second argument. A coefficient of 0 indicates no relationship between the pairs of values, or complete randomness.

Note The RSQ function simply squares the value found by the PEARSON function. See “PEARSON” on page 201 for the equation and further examples.

Syntax

RSQ (*array1*, *array2*)

Argument	Description
array1 and *array2*	A range reference or array constant containing numeric values. *Array1* and *array2* must contain the same potential number of values. The function will return the error value #DIV/0! if *array1* or *array2* contains non-numeric data (text or logical values or blank cells).

Remarks

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Examples

This function returns 0.75:

```
RSQ({1,2,3,4,5}, {9,9,10,10,10})
```

See Also

CORREL, COVAR, PEARSON

SEARCH

Description

Locates the position of the first character of a specified text string within another text string.

Syntax

SEARCH (*search_text*, *text* [, *start_position*])

Argument	Description
search_text	The text to find. The search string can contain wildcard characters. The available wildcard characters are * (asterisk), which matches any sequence of characters, and ? (question mark), which matches any single character. To search for an asterisk or question mark, include a tilde (~) before the character.
text	The text to be searched.
[*start_position*]	Optional. The character position where the search begins. If the number you specify is less than 0 or greater than the number of characters in text, #VALUE! is returned. If this argument is omitted, 1 is used.

Remarks

Text is searched from left to right, starting at the position specified. The search is not case-sensitive. If *text* does not contain the search string, #VALUE! is returned.

Examples

This function returns 6:

```
SEARCH("?5", "Bin b45")
```

This function returns 5:

```
SEARCH("b", "Bin b45", 4)
```

See Also

FIND, MID, REPLACE, SUBSTITUTE

SECOND

Description

Returns the second that corresponds to the supplied date.

Syntax

SECOND (*serial_number*)

Argument	Description
serial_number	The time as a serial number. The decimal portion of the number represents time as a fraction of the day.

Examples

This function returns 58:

```
SECOND(.259)
```

This function returns 46:

```
SECOND(34657.904)
```

See Also

DAY, HOUR, MINUTE, MONTH, NOW, WEEKDAY, YEAR

SERIESSUM

Description Returns the sum of a power series.

Syntax SERIESSUM (*x*, *n*, *m*, *a*)

Argument	Description
x	The value of the independent variable in the series.
n	The initial power of *x* in the series.
m	The increment of the power of *x* in the series.
a	A list of coefficients for the series. The list can be a range reference or array constant. The number of coefficients determines the number of terms in the series.

Equation

$$a_1 \cdot x^n + a_2 \cdot x^{(n+m)} + a_3 \cdot x^{(n+2m)} + \ldots + a_i \cdot x^{(n+(i-1)m))}$$

...where *i* is the number of coefficients in *a*.

Examples This function returns 11368:

```
SERIESSUM(2,2,3,{2,3,4,5})
```

SIGN

Description Determines the sign of the specified number.

Syntax SIGN (*number*)

Argument	Description
number	Any number.

Remarks SIGN returns 1 if the specified number is positive, –1 if it is negative, and 0 if it is 0.

Examples This function returns –1:

```
SIGN(-123)
```

This function returns 1:

```
SIGN(123)
```

See Also ABS

SIN

Description Returns the sine of the supplied angle.

Syntax SIN (*number*)

Argument	Description
number	The angle in radians. If the angle is in degrees, convert the angle to radians by multiplying the angle by PI( )/180.

Examples This function returns .85:

```
SIN(45)
```

This function returns .89:

```
SIN(90)
```

See Also ASIN, PI

SINH

Description Returns the hyperbolic sine of the specified number.

Syntax SINH (*number*)

Argument	Description
number	Any number.

Examples This function returns 1.18:

```
SINH(1)
```

This function returns 10.02:

```
SINH(3)
```

See Also ASINH, PI

SKEW

Description Computes the skewness of a set of numbers.

Syntax SKEW (*number_list*)

Argument	Description
number_list	A set of at least 3 numbers. This can be between 1 and 30 arguments consisting of numbers, cell references, range references, and/or array constants. All numeric values, including 0, are used. Text and logical values in array constants and cells referenced by this function are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Remarks Skewness, also called the third movement, is a measure of the asymmetry of a distribution around its mean. A large positive skewness indicates that more of the distribution falls closer to the positive numbers on the x axis; a negative skewness indicates more of the distribution is closer to the negative numbers on the x axis.

Equation

$$\frac{n}{(n-1)(n-2)}\sum\left(\frac{x_j-\bar{x}}{s}\right)^3$$

Examples This function returns 0:

```
SKEW(1,2,3,4,5,6,7,8,9)
```

This function returns 0.148190368:

```
SKEW(1,1,1,3,4,5,6,7,8)
```

This function returns -0.245294714:

```
SKEW(-1,-1,-1,-1,-4,-5,-6,-7,-8)
```

See Also AVERAGE, AVEDEV, COUNT, DEVSQ, KURT

SLN

Description Returns the depreciation of an asset for a specific period of time using the straight-line balance method.

Syntax SLN (*cost*, *salvage*, *life*)

Argument	Description
cost	The initial cost of the asset.
salvage	The salvage value of the asset.
life	The number of periods of the useful life of the asset.

Example

This function returns 1285.71:

```
SLN(10000, 1000, 7)
```

See Also

DDB, SYD, VDB

SLOPE

Description

Computes the slope of the least squares linear regression line through the given data.

Syntax

SLOPE (*arrayY, arrayX*)

Argument	Description
arrayY and *arrayX*	A range reference or an array constant containing numeric values. *ArrayX* and *arrayY* must contain the same potential number of values. Text, logical values, and empty cells referenced by this function are ignored, along with the number they are paired up with. That is, when a number from *arrayX* is paired up with text from *arrayY,* the entire pair is ignored.

Remarks

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Equation

$$\text{slope } b = \frac{n\sum xy - \left(\sum x\right)\left(\sum y\right)}{n\sum x^2 - \left(\sum x\right)^2}$$

where *n* is the number of members in *arrayY* and *arrayX*.

Examples

This function returns 2.342857143:

```
SLOPE({1,5,8,9,12,13},{0,1,2,3,4,5})
```

See Also

FORECAST, INTERCEPT

SMALL

Description

Computes the *x*th smallest value in a list. If you set *x* at 5, SMALL will sort the values in descending order and return the 5th smallest value.

Syntax

SMALL (*numbers*, *x*)

Argument	Description
numbers	A range reference or array constant containing numbers. Text and logical values in cell references and arrays are ignored. Empty cells are ignored.
x	The index of the number to return after sorting the list in ascending order. *X* must be larger than zero and smaller than the number of values in *numbers*. The logical value TRUE is evaluated as 1, but FALSE will return the #NUM! error. Text will be evaluated as a number, if possible; otherwise the function will return the #VALUE! error.

Example

This function returns 4:

```
SMALL({1,2,3,4,5,6,7,8,9,15},4)
```

See Also

LARGE, MAX, MIN

SQLREQUEST

Description

Returns the #N/A error message.

Syntax

SQLREQUEST (*arguments*)

Argument	Description
arguments	4 to 5 arguments.

Remarks

This function was included only for compatibility with Microsoft Excel. In Excel, SQLREQUEST connects the worksheet to an external data source, functionality which can be handled more efficiently in Formula One for Java using JDBC support.

This function is intended only for users or developers who want to import Excel worksheets that contain Excel's SQLREQUEST function. To make those worksheets work properly, developers should either convert these functions to add-in functions, which they can write in Java and set up to be automatically loaded, or utilize Formula One for Java's JDBC capabilities.

For more information on add-in functions and JDBC, see the *Formula One for Java Technical Guide*.

SQRT

Description Returns the square root of the specified number.

Syntax SQRT (*number*)

Argument	Description
number	Any positive number. If you specify a negative number, the error #NUM! is returned.

Examples This function returns 3:

```
SQRT(9)
```

This function returns 1.58:

```
SQRT(2.5)
```

See Also SUMSQ

SQRTPI

Description Calculates the square root of the product of the specified number and π.

Syntax SQRTPI (*number*)

Argument	Description
number	Any positive number. If you specify a negative number, the error #NUM! is returned.

Examples This function returns 1.772453851:

```
SQRTPI(1)
```

This function returns 2.506628275:

```
SQRTPI(2)
```

See Also SQRT, PI

STANDARDIZE

Description

Computes the proper argument for the function that calculates the standard normal cumulative distribution, given specified values for x, mean, and standard deviation. This function is used to “translate” the three arguments required for the NORMDIST function into a form that can be used in NORMSDIST.

Syntax

STANDARDIZE (*x*, *mean, st_dev*)

Argument	Description
x	The value at which you want to evaluate the function.
mean	The arithmetic mean of the distribution.
st_dev	The standard deviation of the distribution. It must be greater than 0.

Examples

This function returns 11.25:

```
STANDARDIZE (95, 50, 4)
```

See Also

NORMDIST, NORMSDIST

STDEV

Description

Returns the standard deviation of a population based on a sample of supplied values. The standard deviation of a population represents an average of deviations from the population mean within a list of values.

Syntax

STDEV (*number_list*)

Argument	Description
number_list	A list of up to 30 numbers. The list may contain numeric values, cell references, range references, or array constants. Text and logical values in range references and array constants are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Example

This function returns .56:

```
STDEV(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also

STDEVA, STDEVP, STDEVPA, VAR, VARA, VARP, VARPA

STDEVA

Description

Returns the standard deviation of a population based on a sample of supplied values. The standard deviation of a population represents an average of deviations from the mean within a list of values.

This function is equivalent to the STDEV function, but its implementation treats text and logical values in cell and range references differently.

Syntax

STDEVA (*number_list*)

Argument	Description
number_list	A list of up to 30 numbers. The list may contain numeric values, cell references, range references, or array constants. Text in cells referenced by this function is treated as the number 0 (this includes zero-length text). Text entered in the argument list will be evaluated as a number, if possible; otherwise it causes a #VALUE! error. Text and logical values in arrays are ignored. Logical values referenced in cells or entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Example

This function returns .55634864:

```
STDEV(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also

STDEV, STDEVP, STDEVPA, VAR, VARA, VARP, VARPA

STDEVP

Description

Returns the standard deviation of a population based on an entire population of values. The standard deviation of a population represents an average of deviations from the population mean within a list of values.

Syntax

STDEVP (*number_list*)

Argument	Description
number_list	A list of up to 30 numbers. The list may contain numeric values, cell references, range references, or array constants. Text and logical values in range references and array constants are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Example This function returns .52:

```
STDEVP(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also STDEV, VAR, VARP

STDEVPA

Description Returns the standard deviation of a population based on an entire population of values. The standard deviation of a population represents an average of deviations from the population mean within a list of values

This function is equivalent to the STDEVP function, but its implementation treats text and logical values in cell and range references differently.

Syntax STDEVPA (*number_list*)

Argument	Description
number_list	A list of up to 30 numbers. The list may contain numeric values, cell references, range references, or array constants. Text in cells referenced by this function is treated as the number 0 (this includes zero-length text). Text entered in the argument list will be evaluated as a number, if possible; otherwise it causes a #VALUE! error. Text and logical values in arrays are ignored. Logical values referenced in cells or entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Example This function returns .52:

```
STDEVPA(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also STDEV, STDEVA, STDEVP, VAR, VARA, VARP, VARPA

STEYX

Description Computes the standard error of the predicted *y*-value for each *x* in the regression.

Syntax STEYX (*arrayX*, *arrayY*)

Argument	Description
arrayX and *arrayY*	A range reference or an array constant containing at least three numeric values. *ArrayX* and *arrayY* must contain the same potential number of values or the function will return the error value #N/A! All numeric values, including 0, are used. Text, logical values, and empty cells referenced by this function are ignored, along with the number they are paired up with. That is, when a number from *arrayX* is paired up with text from *arrayY*, the entire pair is ignored. The two ranges do not have to have the same shape or orientation: for example, *arrayX* may be 2 columns wide by 3 rows deep, and *arrayY* may be 1 column wide by 6 rows deep. The product of the number of rows and columns in each argument must be the same.

Remarks

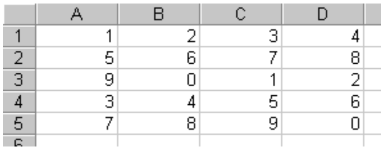
This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Equation

$$\sqrt{\frac{1}{n(n-2)}\left(n\sum y^2-\left(\sum y\right)^2-\frac{\left(n\sum xy-\left(\sum x\right)\left(\sum y\right)\right)^2}{n\sum x^2-\left(\sum x\right)^2}\right)}$$

Examples

The examples are based on the following worksheet.



	A	B	C	D
1	1	2	3	4
2	5	6	7	8
3	9	0	1	2
4	3	4	5	6
5	7	8	9	0
6				

This function returns 3.651484:

```
STEYX(A1:A5, B1:B5)
```

This function returns 2.459675:

```
STEYX(A1:B2,D2:D5)
```

SUBSTITUTE

Description

Replaces a specified part of a text string with another text string.

Syntax

SUBSTITUTE (*text*, *old_text*, *new_text* [, *instance*])

Argument	Description
text	A text string that contains the text to replace. You can also specify a reference to a cell that contains text.
old_text	The text string to be replaced.
new_text	The replacement text.
[*instance*]	Optional. Specifies the occurrence of *old_text* to replace. If this argument is omitted, every instance of *old_text* is replaced.

Examples

This function returns “Second Quarter Results”:

```
SUBSTITUTE("First Quarter Results", "First", "Second")
```

This function returns "Shipment 45, Bin 52":

```
SUBSTITUTE("Shipment 45, Bin 45", "45", "52", 2)
```

See Also

REPLACE, TRIM

SUBTOTAL

Description

Performs one of 11 functions on the specified data ranges or other cell references.

Syntax

SUBTOTAL (*function_index*, *references*)

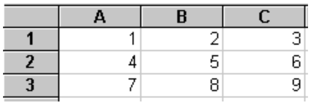
<table>
<tr><th>Argument</th><th>Description</th></tr>
<tr><td><i>function_index</i></td><td>The index number of the function performed on the cells in the specified references, as follows:
<table>
<tr><th>index_number</th><th>function</th></tr>
<tr><td>1</td><td>AVERAGE</td></tr>
<tr><td>2</td><td>COUNT</td></tr>
<tr><td>3</td><td>COUNTA</td></tr>
<tr><td>4</td><td>MAX</td></tr>
<tr><td>5</td><td>MIN</td></tr>
<tr><td>6</td><td>PRODUCT</td></tr>
<tr><td>7</td><td>STDEV</td></tr>
<tr><td>8</td><td>STDEVP</td></tr>
<tr><td>9</td><td>SUM</td></tr>
<tr><td>10</td><td>VAR</td></tr>
<tr><td>11</td><td>VARP</td></tr>
</table></td></tr>
<tr><td><i>references</i></td><td>1-29 data ranges or cell references</td></tr>
</table>

Remarks

SUBTOTAL ignores embedded SUBTOTAL functions. See Examples.

Examples

The examples are based on the following worksheets.



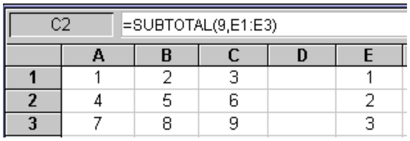
	A	B	C
1	1	2	3
2	4	5	6
3	7	8	9

This function returns 5:

```
SUBTOTAL(1,A1:C3)
```

This function returns 45:

```
SUBTOTAL(9,A1:C3)
```



C2 =SUBTOTAL(9,E1:E3)

	A	B	C	D	E
1	1	2	3		1
2	4	5	6		2
3	7	8	9		3

This function returns 4.875:

```
SUBTOTAL(1,A1:C3)
```

This function returns 39:

```
SUBTOTAL(9,A1:C3)
```

See Also

AVERAGE, COUNT, COUNTA, MAX, MIN, PRODUCT, STDEV, STDEVP, SUM, VAR, VARP

SUM

Description

Returns the sum of the supplied numbers.

Syntax

SUM (*number_list*)

Argument	Description
number_list	A list of as many as 30 numbers. The list can contain numbers, logical values, text representations of numbers, range references, or array constants. Error values or text that cannot be translated into numbers return errors. If a range reference is included in the list, text, logical expressions, and empty cells in the range are ignored.

Examples

This function returns 6000:

```
SUM(1000, 2000, 3000)
```

This function returns 4000 when each cell in the range contains 1000:

```
SUM(A10:D10)
```

See Also AVERAGE, COUNT, COUNTA, PRODUCT, SUMSQ

SUMIF

Description Returns the sum of the specified cells based on the given criteria.

Syntax SUMIF (*range*, *criteria*, *sum_range*)

Argument	Description
range	The range of cells you want evaluated.
criteria	A number, expression, or text that defines which cells are added. For example, *criteria* can be expressed as 15, “15”, “>15”, “cars”.
sum_range	The actual cells to sum. These cells are only summed if their corresponding cells in *range* match the criteria. If this argument is omitted, the cells in *range* are summed.

See Also AVERAGE, COUNT, COUNTA, COUNTIF, PRODUCT, SUM

SUMPRODUCT

Description Multiplies the corresponding cells in the given ranges, then returns the sum of those products.

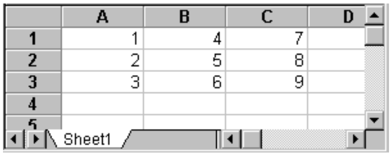
Syntax SUMPRODUCT (*range_list*)

Argument	Description
range_list	Two or more range references that provide the sets of numbers you want to multiply. The values in the upper left cell in each range are multiplied together, then the values in the next cell, etc. All the products are then summed. All the ranges in *range_list* must contain the same number of cells in the same arrangement. That is, if the first range is three rows deep and three columns wide, the second and subsequent ranges must also be three rows deep and three columns wide.

Remarks Excel documentation states that SUMPRODUCT takes “2 to 30” arguments. Both Excel and Formula One for Java support 1 to 30 arguments for this function.

Examples

The following examples use this worksheet.



	A	B	C	D
1	1	4	7	
2	2	5	8	
3	3	6	9	
4				
5				

This function returns 630:

```
SUMPRODUCT(A1:C1,A2:C2,A3:C3)
```

This function returns 50:

```
SUMPRODUCT(A1:A3,C1:C3)
```

See Also

SUM

SUMSQ

Description

Squares each of the supplied numbers and returns the sum of the squares.

Syntax

SUMSQ (*number_list*)

Argument	Description
number_list	A list of as many as 30 numbers. The list can contain numbers, logical values, text representations of numbers, range references, or array constants. Error values or text that cannot be translated into numbers return errors. If a range reference is included in the list, text, logical expressions, and empty cells in the range are ignored.

Example

This function returns 302:

```
SUMSQ(9, 10, 11)
```

See Also

SUM

SUMX2MY2

Description

Computes $x^2 - y^2$ for two sets of numbers, *arrayX* and *arrayY*, paired up one-to-one. The result is part of many statistical computations.

Syntax

SUMX2MY2 (*array1*, *array2*)

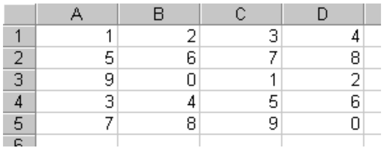
Argument	Description
arrayX and *arrayY*	Two range references or an array constants containing numeric values. Text, logical values, and empty cells referenced by this function are ignored, along with the number they are paired up with. That is, when a number from *arrayX* is paired up with text from *arrayY*, the entire pair is ignored. *ArrayX* and *arrayY* must contain the same potential number of values.

Remarks

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Examples

The examples are based on the following worksheet.



	A	B	C	D
1	1	2	3	4
2	5	6	7	8
3	9	0	1	2
4	3	4	5	6
5	7	8	9	0
6				

This function returns 45:

```
SUMX2MY2(A1:A5, B1:B5)
```

This function returns -42:

```
SUMX2MY2(A1:A3, B2:D2)
```

See Also

SUMX2PY2, SUMXMY2

SUMX2PY2

Description

Computes $x^2 + y^2$ for two sets of numbers, *arrayX* and *arrayY*, paired up one-to-one. The result is part of many statistical computations.

Syntax

SUMX2MY2 (*array1*, *array2*)

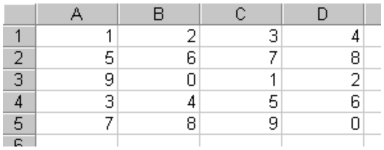
Argument	Description
arrayX and *arrayY*	Two range references or an array constants containing numeric values. Text, logical values, and empty cells referenced by this function are ignored, along with the number they are paired up with. That is, when a number from *arrayX* is paired up with text from *arrayY,* the entire pair is ignored. *ArrayX* and *arrayY* must contain the same potential number of values.

Remarks

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Examples

The examples are based on the following worksheet.



	A	B	C	D
1	1	2	3	4
2	5	6	7	8
3	9	0	1	2
4	3	4	5	6
5	7	8	9	0
6				

This function returns 285:

```
SUMX2PY2(A1:A5, B1:B5)
```

This function returns 256:

```
SUMX2PY2(A1:A3, B2:D2)
```

See Also

SUMX2MY2, SUMXMY2

SUMXMY2

Description

Computes $(x-y)^2$ for two sets of numbers, *arrayX* and *arrayY,* paired up one-to-one. The result is part of many statistical computations.

Syntax

SUMX2MY2 (*array1*, *array2*)

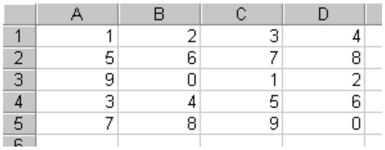
Argument	Description
arrayX and *arrayY*	Two range references or an array constants containing numeric values. Text, logical values, and empty cells referenced by this function are ignored, along with the number they are paired up with. That is, when a number from *arrayX* is paired up with text from *arrayY,* the entire pair is ignored. *ArrayX* and *arrayY* must contain the same potential number of values.

Remarks

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Examples

The examples are based on the following worksheet.



	A	B	C	D
1	1	2	3	4
2	5	6	7	8
3	9	0	1	2
4	3	4	5	6
5	7	8	9	0
6				

This function returns 85:

```
SUMXMY2(A1:A5, B1:B5)
```

This function returns 30:

```
SUMXMY2(A1:A3, B2:D2)
```

See Also

SUMX2MY2, SUMX2PY2

SYD

Description

Returns the depreciation of an asset for a specified period using the sum-of-years method. This depreciation method uses an accelerated rate, where the greatest depreciation occurs early in the useful life of the asset.

Syntax

SYD (*cost*, *salvage*, *life*, *period*)

Argument	**Description**
cost	The initial cost of the asset.
salvage	The salvage value of the asset.
life	The number of periods in the useful life of the asset.
period	The period for which to calculate the depreciation. The time units used to determine *period* and *life* must match.

Example

This function returns 1607.14:

```
SYD(10000, 1000, 7, 3)
```

See Also

DDB, SLN, VDB

T

Description Tests the supplied value and returns the value if it is text.

Syntax T (*value*)

Argument	Description
value	The value to test.

Remarks Empty text ("") is returned for any value that is not text.

Examples This function returns Report:

```
T("Report")
```

This function returns empty text (" ") if A4 contains a number:

```
T(A4)
```

See Also N, VALUE

TAN

Description Returns the tangent of the specified angle.

Syntax TAN (*number*)

Argument	Description
number	The angle in radians. To convert a number expressed as degrees to radians, use the RADIANS function.

Examples This function returns 0.752:

```
TAN(0.645)
```

This function returns 1:

```
TAN(45*PI()/180)
```

See Also ATAN, ATAN2, PI, TANH

TANH

Description Returns the hyperbolic tangent of a number.

Syntax TANH (*number*)

Argument	Description
number	Any number.

Examples This function returns –.96:

```
TANH(-2)
```

This function returns .83:

```
TANH(1.2)
```

See Also ATANH, COSH, SINH, TAN

TBILLEQ

Description Computes the bond-equivalent yield for a treasury bill.

Syntax TBILLEQ (*settlement, maturity, discount*)

Argument	Description
settlement	The date when the security is traded to the buyer. It must be later than the issue date.
maturity	The date the security expires and the remaining amount is paid. It must fall during the 1-year period following *settlement*.
discount	The T-bill’s discount rate, represented as a decimal. It must be larger than 0.

Equation

$$\frac{(365 \times discount)}{360 - (discount \times \mathrm{DSM})}$$

where DSM is the number of days from *settlement* to *maturity*, computed according to the 360-day year, the standard for bond interest computations.

Examples This function returns 0.1026, or 10.26%:

```
TBILLEQ("6/28/91","10/23/91",0.098)
```

See Also TBILLPRICE, TBILLYIELD

TBILLPRICE

Description Computes the price per $100 face value for a treasury bill.

Syntax TBILLPRICE (*settlement, maturity, discount*)

Argument	Description
settlement	The date when the security is traded to the buyer. It must be later than the issue date.
maturity	The date the security expires and the remaining amount is paid. It must fall during the 1-year period following *settlement*.
discount	The T-bill's discount rate, represented as a decimal. It must be larger than 0.

Equation

$$100 \times \left(1 - \frac{discount \times \mathrm{DSM}}{360}\right)$$

where DSM is the number of actual days from *settlement* to *maturity.*

Examples This function returns 96.815:

```
TBILLPRICE("6/28/91","10/23/91",0.098)
```

See Also TBILLEQ, TBILLYIELD

TBILLYIELD

Description Computes the yield for a treasury bill.

Syntax TBILLYIELD (*settlement, maturity, price*)

Argument	Description
settlement	The date when the security is traded to the buyer. It must be later than the issue date.
maturity	The date the security expires and the remaining amount is paid. It must fall during the 1-year period following *settlement*.
price	The T-bill's price per $100 face value. It must be larger than 0.

Remarks The bond-equivalent yield, which is the discount rate quoted for most treasury bills, is calcuated with the TBILLEQ function.

Equation

$$\frac{100 - price}{price} \times \frac{360}{\text{DSM}}$$

where DSM is the number of actual days from *settlement* to *maturity*.

Examples

This function returns 0.0469:

```
TBILLYIELD("6/28/91","10/23/91",98.5)
```

See Also

TBILLEQ, TBILLPRICE

TDIST

Description

Computes the complementary student's T-distribution, one of the methods of determining whether two samples (often the experimental sample and a control group) are statistically different.

This function assumes the value you are testing is part of a normal distribution. You may choose to compute a one-tailed or two-tailed distribution.

Syntax

TDIST (*x, df, tails*)

Argument	Description
x	The value at which you want to evaluate the distribution. It must be a positive number.
df	A positive integer indicating the number of degrees of freedom. Decimal values will be truncated to integers.
tails	Enter 1 for a one-tailed distribution, 2 for two-tailed.

Remarks

This function returns a probability value between zero and one. Values closer to one indicate a high probability that the samples are similar. Lower values indicate a lower probability that the samples are similar.

Equation

$$\frac{\Gamma\left(\frac{k+1}{2}\right)}{\sqrt{k\pi}\,\Gamma\left(\frac{k}{2}\right)}\int_{-\infty}^{t}\left(1+\frac{x^2}{k}\right)^{\frac{-k+1}{2}}dx$$

...where k is the degrees of freedom.

Examples

This function returns 0.089213:

```
TDIST(1.75,3,1)
```

This function returns 0.178425:

```
TDIST(1.75,3,2)
```

See Also

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, GAMMADIST, NORMSDIST, TINV

TEXT

Description Returns the given number as text, using the specified formatting.

Syntax TEXT (*number*, *format*)

Argument	Description
number	Any value, a formula that evaluates to a number, or a reference to a cell that contains a value.
format	A string representing a number format. The string can be any valid format string including “General,” “M/DD/YY,” or “H:MM AM/PM.” The format must be surrounded by a set of double quotation marks. Asterisks cannot be included in *format*.

Examples This function returns 123.620:

```
TEXT(123.62, "0.000")
```

This function returns 10/19/94:

```
TEXT(34626.2, "MM/DD/YY")
```

See Also DOLLAR, FIXED, T, VALUE

TIME

Description Returns a serial number for the supplied time.

Syntax TIME (*hour*, *minute*, *second*)

Argument	Description
hour	A number from 0 to 23.
minute	A number from 0 to 59.
second	A number from 0 to 59.

Examples This function returns .52:

```
TIME(12, 26, 24)
```

This function returns .07:

```
TIME(1, 43, 34)
```

See Also HOUR, MINUTE, NOW, SECOND, TIMEVALUE

TIMEVALUE

Description Returns a serial number for the supplied text representation of time.

Syntax TIMEVALUE (*text*)

Argument	Description
text	A time in text format.

Examples This function returns .07:

```
TIMEVALUE("1:43:43 am")
```

This function returns .59:

```
TIMEVALUE("14:10:07")
```

See Also HOUR, MINUTE, NOW, SECOND, TIME

TINV

Description Computes the x value corresponding to a specified probability value, given a specified number of degrees of freedom. TINV assumes that this is a two-tailed test of a normal distribution curve.

TINV is the inverse of TDIST.

Syntax TINV (*prob, df*)

Argument	Description
prob	A number between 0 and 1 indicating the probability for which you want the corresponding x value.
df	A positive integer indicating the number of degrees of freedom. Decimal values will be truncated to integers.

Remarks This is the complementary version of TINV.

Examples This function returns 9.925:

```
TINV (0.01, 2)
```

This function returns 3.182:

```
TINV (0.05, 3)
```

See Also BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, GAMMADIST, NORMDIST, TDIST

TODAY

Description Returns the current date as a serial number.

Syntax TODAY ( )

Remarks This function is updated only when the worksheet is recalculated.

See Also DATE, DAY, NOW

TRANSPOSE

Description Places a copy of the contents of a range into a new range in which the original rows are now columns and the original columns are now rows.

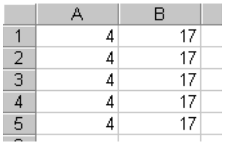
Note This is an array function. For information, see “Array Functions” on page 13.

Syntax TRANSPOSE (*range*)

Argument	Description
range	A range reference or the name of a named range reference.

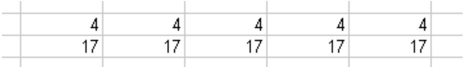
Results range Before entering this function, select a results range with the same number of rows as the argument range has columns and the same number of columns as the argument range has rows. If you select too few cells, some cells will be truncated. If you select too many cells, the extra cells will contain the #N/A error.

Example The following example uses this worksheet.



	A	B
1	4	17
2	4	17
3	4	17
4	4	17
5	4	17

`TRANSPOSE(A1:B5)` returns the following results range:



4	4	4	4	4
17	17	17	17	17

See Also SIGN

TREND

Description

Computes result values from the fitted curve of a multiple linear regression for a group of specified observations relative to a group of specified independent variables. This function is useful for extrapolating information based on a trend in existing data.

Note If you want TREND to return more than one result value, you must enter it as an array function. For information, see “Array Functions” on page 13.

Syntax

TREND (*known_y’s* [, *known_x’s*] [, *new_x’s*] [, *constant*])

Argument	Description
known_y’s	A list of observed values for the dependent variable. This must be a range reference or an array constant containing all numeric values and no empty cells. For a regression with a **single independent variable**, enter any type of range. For a regression with **multiple independent variables**, enter a single row or column of values. If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array.
[*known_x’s*]	Optional. A list of observed values for the independent variable(s). This must be a range reference or an array constant containing all numeric values and no empty cells. For a regression with a **single independent variable**, when *known_y’s* is a single row or column, enter a range that exactly matches the size and shape of the *known_y’s* range. When *known_y’s* is more than one row and column, *known_x’s* must be a range containing the same number of values as the *known_y’s* range, although the two ranges may be different shapes. For a regression with **multiple independent variables**, when *known_y’s* is a single row, enter a contiguous row of values for each independent variable, The number of columns must match the number of columns in *known_y’s*. Similarly, when *known_y’s* is a single column, enter a contiguous column of values for each independent variable, ensuring that the number of rows matches *known_y’s*. If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array. To fit the regression to a formula that uses polynomials, see “Polynomials,” below. When you omit this argument, the function assumes a single independent variable of the sequence {1, 2, 3, ...}.

Argument	Description
[*new_x's*]	Optional. A list of values for the independent variable(s) that will compute the results of this function. This must be a range reference or an array constant containing all numeric values and no empty cells. You may enter any number of values in *new_x's*. The shape of the range only matters if this is a regression with multiple independent variables, in which case it must have the same number of rows or colunms of independent variables as *known_x's*. If you omit this argument, the function uses the specified or default values in *known_x's*.
[*constant*]	Optional. A logical value that determines whether or not a constant is to be included. The constant allows the regression line to intercept the Y axis at a point other than (0,0). Use True (the default value if this argument is omitted) to include the constant. Use False to force the line to intercept the Y axis at 0.

Remarks

This function uses LINEST internally to compute the regression's coefficients. The resulting formula is then used to compute the requested values.

Results range

Before entering this function, select a results range according to the following criteria:

- If there is a single independent variable, the results range should match the size and shape of the *new_x's* argument.
- If there are multiple independent variables, the results range depends on the number of data sets in *new_x's*. If there is only one data set, the results range should be a single cell. For multiple data sets, the results range should be a column if *known_y's* is a column, a row if *known_y's* is a row, with the same number of cells as data sets entered in *new_x's*. If you omitted the *new_x's* argument, the results range must match the size and shape of the *known_y's* argument.

Equation

The regression computed by TREND attempts to fit the following formula:

$$y = C_1x_1+C_2x_2+ \ldots + C_nX_n+b$$

where *C* is the coefficient determined internally by the LINEST function, *n* is the number of independent variables, and *b* is the constant.

Polynomials

To create a polynomial regression, the data for the independent variable must be set up carefully. Create a worksheet in which column A represents *x* and contains the appropriate data or formulas. For column B, which will contain x^2, fill the cells with a formula such as `=A1^COLUMN()`, which will raise the data in column A to the 2nd power. Fill column C with the same formula to raise the data in column A to the 3rd power.

The data for the *new_x's* variable should be set up in the same manner.

Example

The following example of multiple linear regression attempts to correlate physiological data on preadolescent boys with their maximal oxygen uptake. The data are given in the table below.

Maximal O2 Uptake	Age (years)	Height (cm)	Weight (kilos)	Chest Depth (cm)
y	x_1	x_2	x_3	x_4
1.54	8.4	132.0	29.1	14.4
1.74	8.7	135.5	29.7	14.5
1.32	8.9	127.7	28.4	14.0
1.50	9.9	131.1	28.8	14.2
1.46	9.0	130.0	25.9	13.6
1.35	7.7	127.6	27.6	13.9
1.53	7.3	129.9	29.0	14.0
1.71	9.9	138.1	33.6	14.6
1.27	9.3	126.6	27.7	13.9
1.50	8.1	131.8	30.8	14.5

We want to use this data to extrapolate the maximal oxygen uptake for a boy with the following characteristics:

Age (years)	Height (cm)	Weight (kilos)	Chest Depth (cm)
x_1	x_2	x_3	x_4
10	137	33.5	14.5

The data were entered in a worksheet with the row and column arrangement shown here and a *constant* value of True. The TREND function returned a value of 1.66295.

See Also

GROWTH, INTERCEPT, LINEST, LOGEST, SLOPE

TRIM

Description

Removes all spaces from text except single spaces between words.

Syntax

TRIM (*text*)

Argument	Description
text	Any text string or a reference to a cell that contains a text string.

Remarks

Text that is imported from another environment may require this function.

Example

This function returns Level 3, Gate 45:

```
TRIM(" Level 3, Gate 45 ")
```

See Also

CLEAN, MID, REPLACE, SUBSTITUTE

TRIMMEAN

Description

Computes the mean of a list of numbers after reducing a specified percent of the members of the list. The list is sorted in ascending order, then half of *F* percent of the list members are eliminated from the top and half from the bottom of the list, and finally the mean of the remaining members is taken.

This function is useful for eliminating outliers in a sample.

Syntax

TRIMMEAN (*numbers, F*)

Argument	Description
numbers	A range reference or array constant containing numbers. All numeric values, including 0, are used. Text and logical values in cell references and arrays is ignored. Empty cells are ignored.
F	The percent of the members of the list of values that you want to eliminate. The list will be sorted in ascending order and half of this percentage of members will be eliminated from the bottom of the list, half from the top. The percentage will be rounded down. For example, if you chose to eliminate 15% of a 100-member list, 7 members will be deleted from the top and 7 from the bottom of the list, totalling only 14% eliminated.

Example

This function returns 5.5:

```
TRIMMEAN({1,2,3,4,5,6,7,8,9,15},0.2)
```

First, the top 10% and bottom 10% of the list members (1 and 15) are eliminated. Then the mean of the remaining numbers is taken.

See Also

AVERAGE

TRUE

Description

Returns the logical value True. This function always requires the trailing parentheses.

Syntax

TRUE ( )

See Also

FALSE

TRUNC

Description Truncates the given number to an integer.

Syntax TRUNC (*number* [, *precision*])

Argument	Description
number	Any value.
[*precision*]	Optional. The number of decimal places allowed in the truncated number. If this argument is omitted, 0 is used.

Remarks TRUNC removes the fractional part of a number to the specified precision without rounding the number**.**

Examples This function returns 123.45:

```
TRUNC(123.456, 2)
```

This function returns 9800:

```
TRUNC(9899.435, -2)
```

See Also CEILING, FLOOR, INT, MOD, ROUND

TTEST

Description Computes the student’s T-distribution from the data in two specified arrays, then computes the probability.

Syntax TTEST (*array1, array2, tails, type*)

Argument	Description
array1 and *array2*	Two range references or an array constants containing numeric values. Text, logical values, and empty cells are ignored. *Array1* and *array2* must contain the same potential number of values.
tails	Enter 1 for a one-tailed distribution, 2 for two-tailed.
type	The type of T-test you want to perform on the given arrays. The options are: 1 Paired values 2 Two distributions with the same variance 3 Two distributions with different variances

Remarks In *type* option 1, aggregate values for mean, variance, and counts are determined for both arrays. In options 2 and 3, those values are determined for each array independently.

TTEST uses TDIST to compute the T-distribution. TDIST requires that its degrees of freedom argument be an integer. However, in *type* option 3, the computed degrees of freedom will usually not be an integer. Therefore, in this case, the function uses the incomplete beta function instead of TDIST.

Equation

The equations used in this function can be found on pages 616-618 of *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed., William Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, Cambridge University Press, 1992.

Examples

This function returns 0.77:

```
TTEST ({2,4,6,8,10,12}, {8,4,6,1,0,18}, 2, 1)
```

See Also

CHITEST, FTEST, ZTEST

TYPE

Description

Returns the argument type of the given expression.

Syntax

TYPE (*expression*)

Argument	Description
expression	Any expression.

Remarks

The valid values returned by this argument are:

Number	Description
1	Number
2	Text string
4	Logical value
16	Error value

Examples

This function returns 1 if cell A1 contains a number:

```
TYPE(A1)
```

This function returns 2:

```
TYPE("Customer")
```

See Also

ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISREF, ISTEXT

UPPER

Description Changes the characters in the specified string to uppercase characters.

Syntax UPPER (*text*)

Argument	Description
text	Any string.

Remarks Numeric characters in the string are not changed.

Examples This function returns 3RD QUARTER:

```
UPPER("3rd Quarter")
```

This function returns JOHN DOE:

```
UPPER("JOHN DOE")
```

See Also LOWER, PROPER

USDOLLAR

Description Returns the specified number as text using the US Dollar format and the supplied precision. Omitting the precision argument assumes two decimal places.

Syntax USDOLLAR (*number* [, *precision*])

Argument	Description
number	A number, a formula that evaluates to a number, or a reference to a cell that contains a number.
[*precision*]	Optional. A value representing the number of decimal places to the right of the decimal point. If this argument is omitted, 2 is used.

Examples This function returns $1023.79:

```
USDOLLAR(1023.789)
```

This function returns $500:

```
USDOLLAR(495.301, -2)
```

See Also DOLLAR, FIXED, TEXT, VALUE

# VALUE

Description Returns the specified text as a number.

Syntax VALUE (*text*)

Argument	Description
text	Any text string, a formula that evaluates to a text string, or a cell reference that contains a text string. You can also specify a date or time in a recognizable format (for example, M/DD/YY for dates or H:MM AM/PM for time). If the format is not recognized, #VALUE! is returned.

Examples This function returns 9800:

```
VALUE(9800)
```

This function returns 123:

```
VALUE("123")
```

See Also DOLLAR, FIXED, TEXT

VAR

Description Returns the variance of a population based on a sample of values.

Syntax VAR (*number_list*)

Argument	Description
number_list	A list of 1 to 30 numbers separated by commas. The list may contain numeric values, cell references, range references, or array constants. All numeric values, including 0, are used. Text and logical values in cell references and arrays are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Example This function returns .31:

```
VAR(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also STDEV, STDEVP, VARP

VARA

Description

Returns the variance of a population based on a sample of values.

This function is equivalent to the VAR function, but its implementation treats text and logical values in cell and range references differently.

Syntax

VARA (*number_list*)

Argument	Description
number_list	A list of up to 30 arguments separated by commas. The list may contain numeric values, cell references, range references, or array constants. Text in cells referenced by this function is treated as the number 0 (this includes zero-length text). Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Text and logical values in arrays are ignored. Logical values referenced in cells or entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Example

This function returns .30952381:

```
VARA(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also

STDEV, STDEVA, STDEVP, STDEVPA, VAR, VARP, VARPA

VARP

Description

Returns the variance of a population based on an entire population of values.

Syntax

VARP (*number_list*)

Argument	Description
number_list	A list of 1 to 30 numbers separated by commas. The list may contain numeric values, cell references, range references, or array constants. All numeric values, including 0, are used. Text and logical values in cells and arrays are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Example

This function returns .27:

```
VARP(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also

STDEV, STDEVP, VAR

VARPA

Description

Returns the variance of a population based on the entire population of values.

This function is equivalent to the VARP function, but its implementation treats text and logical values in cell and range references differently.

Syntax

VARPA (*number_list*)

Argument	Description
number_list	A list of up to 30 arguments separated by commas. The list may contain numeric values, cell references, range references, or array constants. Text in cells referenced by this function is treated as the number 0 (this includes zero-length text). Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Text and logical values in arrays are ignored. Logical values referenced in cells or entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Example

This function returns .26530612:

```
VARPA(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also

STDEV, STDEVA, STDEVP, STDEVPA, VAR, VARA, VARP

VDB

Description

Returns the depreciation of an asset for a specified period using a variable method of depreciation.

Syntax

VDB (*cost*, *salvage*, *life*, *start_period*, *end_period* [, *factor*] [, *method*])

Argument	Description
cost	The initial cost of the asset.
salvage	The salvage value of the asset.
life	The number of periods in the useful life of the asset.
start_period	The beginning period for which to calculate the depreciation. The time units used to determine *start_period* and *life* must match.
end_period	The ending period for which to calculate the depreciation. The time units used to determine *end_period* and *life* must match.
[*factor*]	Optional. The rate at which the balance declines. If this argument is omitted, 2 (the double-declining balance factor) is used.

Argument	Description
[*method*]	Optional. A logical value that determines if you want to switch to straight-line depreciation when depreciation is greater than the declining balance calculation. Use True to maintain declining balance calculation; use False to switch to straight-line depreciation calculation. If the argument is omitted, False is used.

Example This function returns 1041.23:

```
VDB(10000, 1000, 7, 3, 4)
```

See Also DDB, SLN, SYD

VLOOKUP

Description Searches the first column of a table for a value and returns the contents of a cell in that table that corresponds to the location of the search value.

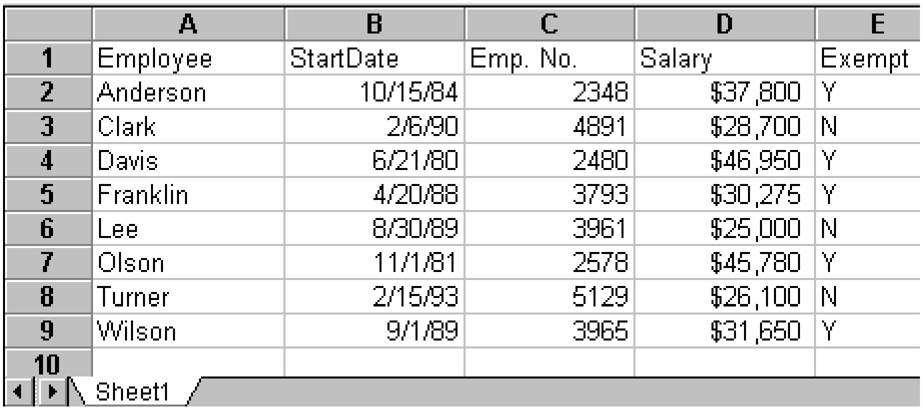
Syntax VLOOKUP (*search_item*, *search_range*, *column_index*)

Argument	Description
search_item	A value, text string, or reference to a cell containing a value that is matched against data in the top row of *search_range*.
search_range	The reference of the range (table) to be searched. The cells in the first column of *search_range* can contain numbers, text, or logical values. The contents of the first column must be in ascending order (for example, –2, –1, 0, 2...A through Z, False, True). Text searches are not case-sensitive.
column_index	The column in the search range from which the matching value is returned. It can be a number from 1 to the number of rows in the search range. If *column_index* is less than 1, #VALUE! is returned. If *column_index* is greater than the number of rows in the table, #REF! is returned.

Remarks VLOOKUP compares the information in the first column of *search_range* to the supplied *search_item*. When a match is found, information located in the same row and supplied column (*column _index*) is returned.

If *search_item* cannot be found in the first column of *search_range*, the largest value that is less than *search_item* is used. When *search_item* is less than the smallest value in the first column of the *search_range*, #REF! is returned.

Examples The following examples use this worksheet.



	A	B	C	D	E
1	Employee	StartDate	Emp. No.	Salary	Exempt
2	Anderson	10/15/84	2348	$37,800	Y
3	Clark	2/6/90	4891	$28,700	N
4	Davis	6/21/80	2480	$46,950	Y
5	Franklin	4/20/88	3793	$30,275	Y
6	Lee	8/30/89	3961	$25,000	N
7	Olson	11/1/81	2578	$45,780	Y
8	Turner	2/15/93	5129	$26,100	N
9	Wilson	9/1/89	3965	$31,650	Y
10					

This function returns $28,700:

```
VLOOKUP("Clark", A2:E9, 4)
```

This function returns 3961:

```
VLOOKUP("Lee", A2:E9, 3)
```

See Also HLOOKUP, INDEX (non-array type), LOOKUP, MATCH

WEEKDAY

Description Returns the day of the week that corresponds to the supplied date.

Syntax WEEKDAY (*serial_number* [,*order*])

Argument	Description
serial_number	The date as a serial number or as text (for example, 06-21-94 or 21-Jun-94).
[*order*]	Optional. This argument determines how the days of the week are numbered for the purposes of this function. If this argument is omitted, 1 is used. The options are: 1 1=Sunday, 2=Monday, 3=Tuesday ... 7=Saturday 2 1=Monday, 2=Tuesday, 3=Wednesday ... 7=Sunday 3 0=Monday, 1=Tuesday, 2=Wednesday ... 6=Sunday

Examples This function returns 1, indicating Sunday:

```
WEEKDAY(34399.92)
```

This function returns 3, indicating Tuesday:

```
WEEKDAY("06/21/94")
```

See Also DAY, NOW, TEXT, TODAY

WEEKNUM

Description Returns a number indicating which week of the year the specified date falls in.

Syntax WEEKNUM (*date* [, *week_begins*])

Argument	Description
date	Any date. Dates in the argument list must be entered as text, with quotation marks.
[*week_begins*]	Optional. Indicates whether Sunday or Monday is used as the starting point for a week. If this argument is omitted, Sunday is used. Decimal values will be truncated to integers. The options are: 1 Sunday starts the week. 2 Monday starts the week.

Remarks Dates in the first week of January will return 1. The last few dates in a year will return 53.

Examples This function returns 20:

```
WEEKNUM("5/9/99",1)
```

This function returns 19:

```
WEEKNUM("5/9/99",2)
```

See Also WEEKDAY

WEIBULL

Description Computes the Weibull distribution.

Syntax WEIBULL (*x*, *alpha*, *beta*, *cumulative*)

Argument	Description
x	The value at which the function will be evaluated. It must be larger than or equal to 0.
alpha	The alpha parameter to the distribution. It must be larger than 0.

Argument	Description
beta	The beta parameter to the distribution. It must be larger than 0.
cumulative	Enter TRUE to calculate the cumulative area from 0 to *x* and FALSE to calculate the value.

Equations

For the cumulative distribution: $F(x;\alpha,\beta) = 1 - e^{-\left(\frac{x}{\beta}\right)^{\alpha}}$

For the probability density function: $f(x;\alpha,\beta) = \frac{\alpha}{\beta^{\alpha}} x^{\alpha - 1} e^{-\left(\frac{x}{\beta}\right)^{\alpha}}$

If *alpha* = 1, WEIBULL returns the exponential distribution with $\lambda = \frac{1}{\beta}$.

Examples

This function returns 0.439:

```
WEIBULL(100,3,120,TRUE)
```

This function returns 0.010:

```
WEIBULL(100,3,120,FALSE)
```

See Also

BINOMDIST, BETADIST, CHIDIST, COMBIN, CRITBINOM, EXPONDIST, FDIST, GAMMADIST, NORMDIST, POISSON

WORKDAY

Description

Returns the date that is a specified number of days before or after a specified date, not counting weekends and specified holidays.

Syntax

WORKDAY (*start_date, days* [, *holidays*])

Argument	Description
start_date	The date from which calculations will start. Dates in the argument list must be entered as text, with quotation marks. Decimal values are truncated to integers.
days	The number of days before or after *start_date*. Enter a negative number to find days before, a positive number to find days after. Decimal values are truncated to integers.
[*holidays*]	Optional. A range reference or array constant containing a list of dates that represent holidays. The dates specified in *holidays* will not be counted as workdays in the function’s computation. Dates in array references must be entered as text, with quotation marks. Decimal values are truncated to integers.

Examples

This function returns 1/14/93:

```
WORKDAY("12/15/92",20,{"12/24/92","12/25/92","1/1/93"})
```

This function returns 6/29/99:

```
WORKDAY("7/14/99",-10,{"7/5/99"})
```

See Also

NETWORKDAYS, WEEKDAY

XIRR

Description

Computes the internal rate of return for an investment with flexible periods.

Syntax

XIRR (*values, dates* [, *guess*])

Argument	Description
values	A range reference or array constant containing a series of payments to the investment. The first value in the range or array constant is the cost at the beginning of the investment; it is usually negative or 0. Positive values represent payments. Empty cells are evaluated as 0. The function requires at least one negative and at least one positive value in *values*. You may use a series of positive and negative values to show payments and costs.
dates	A range reference or array constant containing the dates each of the payments in *values* was made. The first date in the range or array constant is the initial investment date. All the other dates must be later than the initial date, but need not be in chronological order. All dates are truncated to integers. There must be the same number of potential values in *dates* as in *values*.
[*guess*]	Optional. An estimate of the rate of return. If you omit this argument, 0.1 or 10% will be used. Use this argument when the function returns the #NUM! error (see below).

Remarks

XIRR calculates rate of return similarly to IRR, only XIRR allows you to enter the exact dates of the payments. This may be useful in cases where, for example, the institution offering the investment is writing a contract that sets down the exact payment schedule.

All computations are based on a 365-day year.

Unlike other investment functions, XIRR may have a negative rate result.

This is an iterative function, which means that Formula One for Java will attempt to calculate the result over and over to a finer and finer estimate until it reaches a stable value. If after 100 tries the function still has not found a stable value, it returns the #NUM! error. In this case, you can re-enter the function with an estimate in the *guess* argument that may help the iteration converge. The *guess* argument is especially helpful if the data in *values* alternates between positive and negative values.

Equation

$$0 = \sum_{i=0}^{N}\left(\frac{\mathrm{P}_i}{(1 + rate)^{\frac{\mathrm{d}_i - \mathrm{d}_0}{365}}}\right)$$

...where d_i is the *i*th, or last, payment date, d_0 is the 0th payment date, and P_i is the *i*th, or last, payment. The software iterates to a solution to this equation.

Example

This function returns 0.024851, or just above 2%:

```
XIRR({-1000,600,600},{"1/1/85","1/1/90","1/1/95"})
```

See Also

PV, NPV, IRR, XNPV

XNPV

Description

Computes the net present value of an investment with flexible periods.

Syntax

XNPV (*rate, values, dates*)

Argument	Description
rate	The annual interest rate.
values	A range reference or array constant containing a series of payments to the investment. The first value in the range or array constant is the cost at the beginning of the investment; it is usually negative or 0. Positive values represent payments to the investment. Empty cells are evaluated as 0.
dates	A range reference or array constant containing the dates each of the payments in *values* was made. The first date in the range or array constant is the initial investment date. All the other dates must be later than the initial date, but need not be in chronological order. All dates are truncated to integers. There must be the same number of potential values in *dates* as in *values*.

Remarks

XNPV calculates net present value similarly to NPV, only XNPV allows you to enter the exact dates of the payments. This may be useful in cases where, for example, the institution offering the investment is writing a contract that sets down the exact payment schedule.

All computations are based on a 365-day year.

Equation

$$\sum_{i-0}^{N}\frac{\mathrm{P}_i}{(1 + rate)^{\frac{\mathrm{d}_i - \mathrm{d}_0}{365}}}$$

...where d_i is the *i*th, or last, payment date, d_0 is the 0th payment date, and P_i is the *i*th, or last, payment.

Examples This function returns -937.41:

```
XNPV(0.6,{-1000,600,600},{"1/1/85","1/1/90","1/1/95"})
```

See Also PV, NPV, IRR, XIRR

YEAR

Description Returns the year that corresponds to the supplied date.

Syntax YEAR (*serial_number*)

Argument	Description
serial_number	The date as a serial number or as text (for example, 06-21-94 or 21-Jun-94).

Examples This function returns 1993:

```
YEAR(34328)
```

This function returns 1994:

```
YEAR("06/21/94")
```

See Also DAY, NOW, HOUR, MINUTE, MONTH, SECOND, TODAY, WEEKDAY

YEARFRAC

Description Computes the fraction of a year between two specified dates.

Syntax YEARFRAC (*start_date*, *end_date* [, *calendar_type*])

Argument	Description
start_date	The starting date. Dates in the argument list must be in the form of a serial number or text.
end_date	The ending date. Dates in the argument list must be in the form of a serial number or text.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Remarks This function truncates all arguments to integers.

Examples

This function returns 0.344:

```
YEARFRAC("1/11/97","5/15/97")
```

This function returns 1.344:

```
YEARFRAC("1/11/97","5/15/98")
```

See Also

DAYS360

YIELD

Description

Computes the annual yield of a security that pays periodic interest. Rate, redemption value, and price must be specified.

Syntax

YIELD (*settlement*, *maturity*, *rate*, *price*, *redemption*, *frequency* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Decimal values will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Decimal values will be truncated to integers.
rate	The security's annual coupon rate. The coupons pay at this rate divided by *frequency*.
price	The amount the buyer pays, per $100 face value.
redemption	The security's redemption value per $100 face value. This is the amount paid at the settlement date that is not part of any final coupon payment.
frequency	The number of interest payments per year. See "The frequency Argument" on page 18 for more information.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Examples

This function returns 0.0921:

```
YIELD("5/6/97","12/31/99",0.06,0.92.53,100,4)
```

See Also

PRICE, PRICEDISC, PRICEMAT, YIELDDISC, YIELDMAT

YIELDDISC

Description

Computes the annual yield of a discounted security, given a specified price and redemption value.

Syntax

YIELDDISC (*settlement*, *maturity*, *price*, *redemption* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Decimal values will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Decimal values will be truncated to integers.
price	The amount the buyer pays, per $100 face value.
redemption	The security's redemption value per $100 face value. This is the amount paid at *maturity*.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Equation

$$\frac{redemption - price}{price \times \mathrm{YEARFRAC}\ (settlement, maturity, calendar\_type)}$$

Examples

This function returns 0.0244:

```
YIELDDISC("10/23/94","7/7/95",98.31,100)
```

See Also

DISC, PRICE, PRICEDISC, PRICEMAT, YIELD, YIELDMAT

YIELDMAT

Description

Computes the annual yield on a security that pays interest only at maturity, given specified dates, interest rate, and price.

Syntax

YIELDMAT (*settlement*, *maturity*, *issue*, *rate*, *price* [, *calendar_type*])

Argument	Description
settlement	The date when the security is traded to the buyer. Decimal values will be truncated to integers.
maturity	The date the security expires and the remaining amount is paid to the investor. It must be later than *settlement*. Decimal values will be truncated to integers.
issue	The date the security was originally issued and began earning interest. Decimal values will be truncated to integers.

Argument	Description
rate	The security's annual interest rate at date of issue.
price	The amount the buyer pays, per $100 face value.
[*calendar_type*]	Optional. One of five methods of counting days for computing interest. See "The calendar_type Argument" on page 19 for more information.

Equation

$$\frac{1}{\mathrm{DSM}} \times \frac{100 + \mathrm{DIM} \times rate \times 100}{(price + \mathrm{DIS} \times rate \times 100) - 1}$$

where the three-letter codes correspond to values you can compute using the YEARFRAC function, as shown in the following table.

Code	Meaning	Function
DIM	Days from *issue* to *maturity.*	YEARFRAC(*issue, maturity, calendar_type*)
DIS	Days from *issue* to *settlement*.	YEARFRAC(*issue, settlement, calendar_type*)
DSM	Days from *settlement* to *maturity.*	YEARFRAC(*settlement, maturity, calendar_type*)

Example

This function returns 0.046:

```
YIELDMAT("9/19/86","2/28/94","2/28/84",0.0525,100.0154)
```

See Also

PRICE, PRICEDISC, PRICEMAT, YIELD, YIELDDISC

ZTEST

Description

Computes the two-tailed probability of a z-test, which is a test of a specified value against a specified set of numbers. ZTEST returns the number of standard deviations that separate the specified value from the mean of the set of numbers.

Syntax

ZTEST (*array, x* [, σ])

Argument	Description
array	A list of numbers separated by commas. The list can be in the form of a range reference or array constant. Text and logical values are ignored.
x	The value you want to test. A blank cell will be evaluated as 0. Text will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.
[σ]	Optional. The population standard deviation. If this argument is omitted, ZTEST calculates a sample standard deviation based on the values in *array.*

Equation

$$1 - \text{NORMSDIST}\left(\frac{\bar{x} - \mu}{\sigma / \sqrt{n}}\right)$$

Example

This function returns 0.99973395:

```
ZTEST ({1,2,3}, 4)
```

See Also

CHITEST, FTEST, TTEST

Index