



LibreOffice
The Document Foundation

Calc Guide

Appendix B

Description of Functions

Copyright

This document is Copyright © 2005–2011 by its contributors as listed below. You may distribute it and/or modify it under the terms of either the GNU General Public License (<http://www.gnu.org/licenses/gpl.html>), version 3 or later, or the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0/>), version 3.0 or later.

All trademarks within this guide belong to their legitimate owners.

Contributors

Barbara Duprey
Hal Parker
Simon Brydon

Feedback

Please direct any comments or suggestions about this document to:
documentation@libreoffice.org

Acknowledgments

This appendix is based on Appendix B of the *OpenOffice.org 3.3 Calc Guide*. The contributors to that appendix are:

Magnus Adielsson	Richard Barnes	Peter Kupfer
Iain Roberts	Jean Hollis Weber	

Publication date and software version

Published 5 May 2011. Based on LibreOffice 3.3.

Note for Mac users

Some keystrokes and menu items are different on a Mac from those used in Windows and Linux. The table below gives some common substitutions for the instructions in this appendix. For a more detailed list, see the application Help.

<i>Windows/Linux</i>	<i>Mac equivalent</i>	<i>Effect</i>
Tools → Options menu selection	LibreOffice → Preferences	Access setup options
<i>Right-click</i>	<i>Control+click</i>	Open context menu
<i>Ctrl (Control)</i>	⌘ (<i>Command</i>)	Used with other keys
<i>F5</i>	<i>Shift+⌘+F5</i>	Open the Navigator
<i>F11</i>	⌘+T	Open Styles & Formatting window

Contents

Copyright.....	2
Note for Mac users.....	2
Functions available in Calc.....	4
Terminology: numbers and arguments.....	4
Mathematical functions.....	4
Financial analysis functions.....	8
A note about dates.....	8
A note about interest rates.....	8
Statistical analysis functions.....	18
Date and time functions.....	25
Logical functions.....	28
Informational functions.....	28
Database functions.....	30
Array functions.....	32
Spreadsheet functions.....	33
Text functions.....	37
Add-in functions.....	40

Functions available in Calc

Calc provides all of the commonly used functions found in modern spreadsheet applications. Since many of Calc's functions require very specific and carefully calculated input arguments, the descriptions in this appendix should not be considered complete references for each function. Refer to the application Help or the LibreOffice wiki for details and examples of all functions. On the wiki, start with http://help.libreoffice.org/Calc/Functions_by_Category

Over 300 standard functions are available in Calc. More can be added through extensions to Calc (see Chapter 14). The following tables list Calc's functions organized into eleven categories.

Note

Functions whose names end with **_ADD** are provided for compatibility with Microsoft Excel functions. They return the same results as the corresponding functions in Excel (without the suffix), which though they may be correct, are not based on international standards.

Terminology: numbers and arguments

Some of the descriptions in this appendix define limitations on the number of values or arguments that can be passed to the function. Specifically, functions that refer to the following arguments may lead to confusion.

- **Number_1; number_2;... number_30**
- Number 1 to 30
- a list of up to 30 numbers

There is a significant difference between a *list of numbers* (or integers) and the *number of arguments* a function will accept. For, example the *SUM* function will only accept a maximum of 30 arguments. This limit does NOT mean that you can only sum 30 numbers, but that you can only pass 30 separate arguments to the function.

Arguments are values separated by semi-colons, and can include ranges which often refer to multiple values. Therefore one argument can refer to several values, and a function that limits input to 30 arguments may in fact accept more than 30 separate numerical values.

This appendix attempts to clarify this situation by using the term **arguments**, rather than any of the other phrases.

Mathematical functions

Table 1: Mathematical functions

Syntax	Description
ABS(number)	Returns the absolute value of the given number .
ACOS(number)	Returns the inverse cosine of the given number in radians.
ACOSH(number)	Returns the inverse hyperbolic cosine of the given number in radians.
ACOT(number)	Returns the inverse cotangent of the given number in radians.
ACOTH(number)	Returns the inverse hyperbolic cotangent of the given number in radians.
ASIN(number)	Returns the inverse sine of the given number in radians.

Syntax	Description
ASINH(number)	Returns the inverse hyperbolic sine of the given number in radians.
ATAN(number)	Returns the inverse tangent of the given number in radians.
ATAN2(number_x; number_y)	Returns the inverse tangent of the specified x and y coordinates. Number_x is the value for the x coordinate. Number_y is the value for the y coordinate.
ATANH(number)	Returns the inverse hyperbolic tangent of the given number. (Angle is returned in radians.)
CEILING(number; significance; mode)	Rounds the given number to the nearest integer or multiple of significance. Significance is the value to whose multiple of ten the value is to be rounded up (.01, .1, 1, 10, etc.). Mode is an optional value. If it is indicated and non-zero and if the number and significance are negative, rounding up is carried out based on that value.
COMBIN(count_1; count_2)	Returns the number of combinations for a given number of objects. Count_1 is the total number of elements. Count_2 is the selected count from the elements. This is the same as the nCr function on a calculator.
COMBINA(count_1; count_2)	Returns the number of combinations for a given number of objects (repetition included). Count_1 is the total number of elements. Count_2 is the selected count from the elements.
CONVERT(value; "text"; "text")	Converts a currency value of a European currency into Euros. Value is the amount in the currency to be converted. Text is the official abbreviation for the currency in question (for example, "EUR"). The first Text parameter gives the source value to be converted; the second Text parameter gives the destination value. Both text arguments must be within quotes.
COS(number)	Returns the cosine of the given number (angle in radians).
COSH(number)	Returns the hyperbolic cosine of the given number (angle in radians).
COT(number)	Returns the cotangent of the given number (angle in radians).
COTH(number)	Returns the hyperbolic cotangent of the given number (angle in radians).
COUNTBLANK(range)	Returns the number of empty cells. Range is the cell range in which the empty cells are counted.
COUNTIF(range; criteria)	Returns the number of elements that meet certain criteria within a cell range. Range is the range to which the criteria are to be applied. Criteria indicates the criteria in the form of a number, a regular expression, or a character string by which the cells are counted.
DEGREES(number)	Converts the given number in radians to degrees.
EVEN(number)	Rounds the given number up to the nearest even integer.
EXP(number)	Returns e raised to the power of the given number .
FACT(number)	Returns the factorial of the given number .

Syntax	Description
FLOOR(number; significance; mode)	Rounds the given number down to the nearest multiple of significance . Significance is the value to whose multiple of ten the number is to be rounded down (.01, .1, 1, 10, etc.). Mode is an optional value. If it is indicated and non-zero and if the number and significance are negative, rounding up is carried out based on that value.
GCD(numbers)	Returns the greatest common divisor of one or more integers. Numbers is a list of up to 30 numbers whose greatest common divisor is to be calculated, separated by semi-colons.
GCD_ADD(numbers)	Returns the greatest common divisor of a list of numbers. Numbers is a list of up to 30 numbers separated by semi-colons.
INT(number)	Rounds the given number down to the nearest integer.
ISEVEN(value)	Returns TRUE if the given value is an even integer, or FALSE if the value is odd. <i>If the value is not an integer, the function evaluates only the integer part of the value.</i>
ISODD(value)	Returns TRUE if the given value is an odd integer, or FALSE if the value is even. <i>If the value is not an integer, the function evaluates only the integer part of the value.</i>
LCM(integer_1; integer_2; ... integer_30)	Returns the least common multiple of one or more integers. Integer_1; integer_2;... integer_30 are integers whose lowest common multiple is to be calculated.
LCM_ADD(numbers)	Numbers is a list of up to 30 numbers separated by semi-colons. The result is the lowest common multiple of a list of numbers.
LN(number)	Returns the natural logarithm based on the constant e of the given number .
LOG(number; base)	Returns the logarithm of the given number to the specified base. Base is the base for the logarithm calculation.
LOG10(number)	Returns the base-10 logarithm of the given number .
MOD(dividend; divisor)	Returns the remainder after a number is divided by a divisor. Dividend is the number which will be divided by the divisor. Divisor is the number by which to divide the dividend.
MROUND(number; multiple)	The result is the nearest integer multiple of the number .
MULTINOMIAL (number(s))	Returns the factorial of the sum of the arguments divided by the product of the factorials of the arguments. Number(s) is a list of up to 30 numbers separated by semi-colons.
ODD(number)	Rounds the given number up to the nearest odd integer.
PI()	Returns the value of PI to fourteen decimal places.
POWER(base; power)	Returns the result of a number raised to a power. Base is the number that is to be raised to the given power. Power is the exponent by which the base is to be raised.

Syntax	Description
PRODUCT(number 1 to 30)	Multiplies all the numbers given as arguments and returns the product. Number 1 to number 30 are up to 30 arguments whose product is to be calculated, separated by semi-colons.
QUOTIENT(numerator; denominator)	Returns the integer result of a division operation. Numerator is the number that will be divided. Denominator is the number the numerator will be divided by.
RADIANS(number)	Converts the given number in degrees to radians.
RAND()	Returns a random number between 0 and 1. This number will recalculate every time data is entered or F9 is pressed.
RANDBETWEEN (bottom; top)	Returns an integer random number between bottom and top (inclusive). This number will recalculate when the <i>Control+Shift+F9</i> key combination is pressed.
ROUND(number; count)	Rounds the given number to a certain number of decimal places according to valid mathematical criteria. Count (optional) is the number of the places to which the value is to be rounded. If the count parameter is negative, only the whole number portion is rounded. It is rounded to the place indicated by the count.
ROUNDDOWN(number; count)	Rounds the given number . Count (optional) is the number of digits to be rounded down to. If the count parameter is negative, only the whole number portion is rounded. It is rounded to the place indicated by the count.
ROUNDUP(number; count)	Rounds the given number up. Count (optional) is the number of digits to which rounding up is to be done. If the count parameter is negative, only the whole number portion is rounded. It is rounded to the place indicated by the count.
SERIESSUM(x; n; m; coefficients)	Returns a sum of powers of the number x in accordance with the following formula: $\text{SERIESSUM}(x;n;m;\text{coefficients}) = \text{coefficient}_1 * x^n + \text{coefficient}_2 * x^{(n+m)} + \text{coefficient}_3 * x^{(n+2m)} + \dots + \text{coefficient}_i * x^{(n+(i-1)m)}$ x is the number as an independent variable. n is the starting power. m is the increment. Coefficients is a series of coefficients. For each coefficient the series sum is extended by one section. You can only enter coefficients using cell references.
SIGN(number)	Returns the sign of the given number . The function returns the result 1 for a positive sign, -1 for a negative sign, and 0 for zero.
SIN(number)	Returns the sine of the given number (angle in radians).
SINH(number)	Returns the hyperbolic sine of the given number (angle in radians).
SQRT(number)	Returns the positive square root of the given number . The value of the number must be positive.
SQRTPI(number)	Returns the square root of the product of the given number and PI.

Syntax	Description
SUBTOTAL(function; range)	Calculates subtotals. If a range already contains subtotals, these are not used for further calculations. Function is a value that stands for another function such as Average, Count, Min, Sum, Var. Range is the range whose cells are included.
SUM(number_1; number_2; ... number_30)	Adds all the numbers in a range of cells. Number_1; number_2;... number_30 are up to 30 arguments whose sum is to be calculated. You can also enter a range using cell references.
SUMIF(range; criteria; sum_range)	Adds the cells specified by a given criteria. The search supports regular expressions. Range is the range to which the criteria are to be applied. Criteria is the cell in which the search criterion is shown, or the search criterion itself. Sum_range is the range from which values are summed; if it has not been indicated, the values found in the Range are summed.
SUMSQ(number_1; number_2; ... number_30)	Calculates the sum of the squares of numbers (totaling up of the squares of the arguments) Number_1; number_2;... number_30 are up to 30 arguments, the sum of whose squares is to be calculated.
TAN(number)	Returns the tangent of the given number (angle in radians).
TANH(number)	Returns the hyperbolic tangent of the given number (angle in radians).
TRUNC(number; count)	Truncates a number to an integer by removing the fractional part of the number according to the precision specified in Tools → Options → LibreOffice Calc → Calculate . Number is the number whose decimal places are to be cut off. Count is the number of decimal places which are not cut off.

Financial analysis functions

A note about dates

Date values used as parameters for Calc's financial functions must be entered in a specific manner. For example, a date (entered in the US form) must be surrounded by quotes and with periods separating each value. To represent August 6, 2004, or 8/6/04, you would enter "08.06.2004". If you do not enter the date values as required by the function, you will not get the correct results. Date formats are locale specific; check the Help for the acceptable formatting.

A note about interest rates

You can enter interest rates in either of two ways:

- As a decimal. To enter an interest rate as a decimal, divide it by 100 before entering it into a function. For example, to compute a loan with a 3.25% interest rate, enter .0325 into the function.
- As a percentage. To enter an interest rate as a percentage, type in the interest rate followed by the % key. For example, to compute a loan with a 3.25% interest rate, enter 3.25% into the function.

If you enter it as 3.25, the function will treat it as a 325% interest rate.

Accounting systems vary in the number of days in a month or a year used in calculations. The following table gives the integers used for the **basis** parameter used in some of the financial analysis functions.

Table 2: Basis calculation types

Basis	Calculation
0 or missing	US method (NASD), 12 months of 30 days each.
1	Exact number of days in months, exact number of days in year.
2	Exact number of days in month, year has 360 days.
3	Exact number of days in month, year has 365 days.
4	European method, 12 months of 30 days each.

Table 3: Financial analysis functions

Syntax	Description
ACCRINT(issue; first_interest; settlement; rate; par; frequency; basis)	Calculates the accrued interest of a security in the case of periodic payments. Issue is the issue date of the security. First_interest is the first interest date of the security. Settlement is the maturity date. Rate is the annual nominal rate of interest (coupon interest rate). Par is the par value of the security. Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.
ACCRINTM(issue; settlement; rate; par; basis)	Calculates the accrued interest of a security in the case of one-off payment at the settlement date. Issue is the issue date of the security. Settlement is the maturity date. Rate is the annual nominal rate of interest (coupon interest rate). Par is the par value of the security. Basis indicates how the year is to be calculated.
AMORDEGRC(cost; date_purchased; first_period; salvage; period; rate; basis)	Calculates the amount of depreciation for a settlement period as degressive amortization. Unlike AMORLINC, a depreciation coefficient that is independent of the depreciable life is used here. Cost is the acquisition cost. Date_purchased is the date of acquisition. First_period is the end date of the first settlement period. Salvage is the salvage value of the capital asset at the end of the depreciable life. Period is the settlement period to be considered. Rate is the rate of depreciation. Basis indicates how the year is to be calculated.

Syntax	Description
AMORLINC(cost; date_purchased; first_period; salvage; period; rate; basis)	Calculates the amount of depreciation for a settlement period as linear amortization. If the capital asset is purchased during the settlement period, the proportional amount of depreciation is considered. Cost is the acquisition cost. Date_purchased is the date of acquisition. First_period is the end date of the first settlement period. Salvage is the salvage value of the capital asset at the end of the depreciable life. Period is the settlement period to be considered. Rate is the rate of depreciation. Basis indicates how the year is to be calculated.
COUPDAYBS(settlement; maturity; frequency; basis)	Returns the number of days from the first day of interest payment on a security until the settlement date. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.
COUPDAYS(settlement; maturity; frequency; basis)	Returns the number of days in the current interest period in which the settlement date falls. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.
COUPDAYSNC(settlement; maturity; frequency; basis)	Returns the number of days from the settlement date until the next interest date. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.
COUPNCD(settlement; maturity; frequency; basis)	Returns the date of the first interest date after the settlement date, and formats the result as a date. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.
COUPNUM(settlement; maturity; frequency; basis)	Returns the number of coupons (interest payments) between the settlement date and the maturity date. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.
COUPPCD(settlement; maturity; frequency; basis)	Returns the date of the interest date prior to the settlement date, and formats the result as a date. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.

Syntax	Description
CUMIPMT(rate; NPER; PV; S; E; type)	Calculates the cumulative interest payments (the total interest) for an investment based on a constant interest rate. Rate is the periodic interest rate. NPER is the payment period with the total number of periods. NPER can also be a non-integer value. The rate and NPER must refer to the same unit, and thus both must be calculated annually or monthly. PV is the current value in the sequence of payments. S is the first period. E is the last period. Type is the due date of the payment at the beginning (1) or end (0) of each period.
CUMIPMT_ADD(rate; NPER; PV; start_period; end_period; type)	Calculates the accumulated interest for a period. Rate is the interest rate for each period. NPER is the total number of payment periods. The rate and NPER must refer to the same unit, and thus both must be calculated annually or monthly. PV is the current value. Start_period the first payment period for the calculation. End_period the last payment period for the calculation. Type is the due date of the payment at the beginning (1) or end (0) of each period.
CUMPRINC(rate; NPER; PV; S; E; type)	Returns the cumulative interest paid for an investment period with a constant interest rate. Rate is the periodic interest rate. NPER is the payment period with the total number of periods. NPER can also be a non-integer value. The rate and NPER must refer to the same unit, and thus both must be calculated annually or monthly. PV is the current value in the sequence of payments. S is the first period. E is the last period. Type is the due date of the payment at the beginning (1) or end (0) of each period.
CUMPRINC_ADD(rate; NPER; PV; start_period; end_period; type)	Calculates the cumulative redemption of a loan in a period. Rate is the interest rate for each period. NPER is the total number of payment periods. The rate and NPER must refer to the same unit, and thus both must be calculated annually or monthly. PV is the current value. Start period is the first payment period for the calculation. End period is the last payment period for the calculation. Type is the due date of the payment at the beginning (1) or end (0) of each period.
DB(cost; salvage; life; period; month)	Returns the depreciation of an asset for a specified period using the double-declining balance method. Cost is the initial cost of an asset. Salvage is the value of an asset at the end of the depreciation. Life defines the period over which an asset is depreciated. Period is the length of each period. The life must be entered in the same date unit as the depreciation period. Month (optional) denotes the number of months for the first year of depreciation.

Syntax	Description
DDB(cost; salvage; life; period; factor)	Returns the depreciation of an asset for a specified period using the arithmetic-declining method. Note that the book value will never reach zero under this calculation type. Cost fixes the initial cost of an asset. Salvage fixes the value of an asset at the end of its life. Life is the number of periods defining how long the asset is to be used. Period defines the length of the period. The period must be entered in the same time unit as the life. Factor (optional) is the factor by which depreciation decreases.
DISC(settlement; maturity; price; redemption; basis)	Calculates the allowance (discount) of a security as a percentage. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Price is the price of the security per 100 currency units of par value. Redemption is the redemption value of the security per 100 currency units of par value. Basis indicates how the year is to be calculated.
DOLLARDE(fractional _dollar; fraction)	Converts a quotation that has been given as a decimal fraction into a decimal number. Fractional_dollar is a number given as a decimal fraction. (In this number, the decimal value is the numerator of the fraction.) Fraction is a whole number that is used as the denominator of the decimal fraction.
DOLLARFR(decimal _dollar; fraction)	Converts a quotation that has been given as a decimal number into a mixed decimal fraction. The decimal of the result is the numerator of the fraction that would have Fraction as the denominator. Decimal_dollar is a decimal number. Fraction is a whole number that is used as the denominator of the decimal fraction.
DURATION(rate; PV; FV)	Calculates the number of periods required by an investment to attain the desired value. Rate (a constant) is the interest rate to be calculated for the entire duration. Entering the interest rate divided by the periods per year, can calculate the interest after each period. PV is the present value. FV is the desired future value of the investment.
DURATION_ADD (settlement; maturity; coupon; yield; frequency; basis)	Calculates the duration of a fixed interest security in years. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Coupon is the annual coupon interest rate (nominal rate of interest). Yield is the annual yield of the security. Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.
EFFECT_ADD(nominal _rate; Npery)	Calculates the effective annual rate of interest on the basis of the nominal interest rate and the number of interest payments per annum. Nominal interest refers to the amount of interest due at the end of a calculation period. Nominal_rate is the annual nominal rate of interest. Npery is the number of interest payments per year.

Syntax	Description
EFFECTIVE(NOM; P)	Calculates the effective annual rate of interest on the basis of the nominal interest rate and the number of interest payments per annum. Nominal interest refers to the amount of interest due at the end of a calculation period. NOM is the nominal interest. P is the number of interest payment periods per year.
FV(rate; NPER; PMT; PV; type)	Returns the future value of an investment based on periodic, constant payments and a constant interest rate. Rate is the periodic interest rate. NPER is the total number of periods. PMT is the annuity paid regularly per period. PV (optional) is the present cash value of an investment. Type (optional) defines whether the payment is due at the beginning (1) or the end (0) of a period.
FVSCHEDULE(principal; schedule)	Calculates the accumulated value of the starting capital for a series of periodically varying interest rates. Principal is the starting capital. Schedule is a series of interest rates. Schedule has to be entered with cell references.
INTRATE(settlement; maturity; investment; redemption; basis)	Calculates the annual interest rate that results when a security (or other item) is purchased at an investment value and sold at a redemption value with no interest being paid. Settlement is the date of purchase of the security. Maturity is the date on which the security is sold. Investment is the purchase price. Redemption is the selling price. Basis indicates how the year is to be calculated.
IPMT(rate; period; NPER; PV; FV; type)	Calculates the periodic amortization for an investment with regular payments and a constant interest rate. Rate is the periodic interest rate. Period is the period for which the compound interest is calculated. NPER is the total number of periods during which annuity is paid. Period=NPER , if compound interest for the last period is calculated. PV is the present cash value in sequence of payments. FV (optional) is the desired value (future value) at the end of the periods. Type (optional) defines whether the payment is due at the beginning (1) or the end (0) of a period.
IRR(values; guess)	Calculates the internal rate of return for an investment. The values represent cash flow values at regular intervals; at least one value must be negative (payments), and at least one value must be positive (income). Values is an array containing the values. Guess (optional) is the estimated value. If you can provide only a few values, you should provide an initial guess to enable the iteration.
ISPMT(rate; period; total_periods; invest)	Calculates the level of interest for unchanged amortization installments. Rate sets the periodic interest rate. Period is the number of installments for calculation of interest. Total_periods is the total number of installment periods. Invest is the amount of the investment.

Syntax	Description
MDURATION(settlement; maturity; coupon; yield; frequency; basis)	Calculates the modified Macauley duration of a fixed interest security in years. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Coupon is the annual nominal rate of interest (coupon interest rate) Yield is the annual yield of the security. Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.
MIRR(values; investment; reinvest_rate)	Calculates the modified internal rate of return of a series of investments. Values corresponds to the array or the cell reference for cells whose content corresponds to the payments. Investment is the rate of interest of the investments (the negative values of the array) Reinvest_rate is the rate of interest of the reinvestment (the positive values of the array).
NOMINAL(effective_rate; Npery)	Calculates the yearly nominal interest rate, given the effective rate and the number of compounding periods per year. Effective_rate is the effective interest rate Npery is the number of periodic interest payments per year.
NOMINAL_ADD(effective_rate; Npery)	Calculates the yearly nominal rate of interest, given the effective rate and the number of compounding periods per year. Effective_rate is the effective annual rate of interest. Npery is the number of interest payments per year.
NPER(rate; PMT; PV; FV; type)	Returns the number of periods for an investment based on periodic, constant payments and a constant interest rate. Rate is the periodic interest rate. PMT is the constant annuity paid in each period. PV is the present value (cash value) in a sequence of payments. FV (optional) is the future value, which is reached at the end of the last period. Type (optional) defines whether the payment is due at the beginning (1) or the end (0) of a period.
NPV(Rate; value_1; value_2; ... value_30)	Returns the net present value of an investment based on a series of periodic cash flows and a discount rate. Rate is the discount rate for a period. Value_1; value_2;... value_30 are values representing deposits or withdrawals.
ODDFPRICE(settlement; maturity; issue; first_coupon; rate; yield; redemption; frequency; basis)	Calculates the price per 100 currency units par value of a security, if the first interest date falls irregularly. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Issue is the date of issue of the security. First_coupon is the first interest date of the security. Rate is the annual rate of interest. Yield is the annual yield of the security. Redemption is the redemption value per 100 currency units of par value. Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.
ODDFYIELD(settlement; maturity; issue; first_coupon; rate; price; redemption; frequency; basis)	DDB(cost; salvage; life; period; factor)

Syntax	Description
ODDLPRICE(settlement; maturity; last_interest; rate; yield; redemption; frequency; basis)	Calculates the price per 100 currency units par value of a security, if the last interest date falls irregularly. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Last_interest is the last interest date of the security. Rate is the annual rate of interest. Yield is the annual yield of the security. Redemption is the redemption value per 100 currency units of par value. Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.
ODDLYIELD(settlement; maturity; last_interest; rate; price; redemption; frequency; basis)	Calculates the yield of a security if the last interest date falls irregularly. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Last_interest is the last interest date of the security. Rate is the annual rate of interest. Price is the price of the security. Redemption is the redemption value per 100 currency units of par value. Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.
PMT(rate; NPER; PV; FV; type)	Returns the periodic payment for an annuity with constant interest rates. Rate is the periodic interest rate. NPER is the number of periods in which annuity is paid. PV is the present value (cash value) in a sequence of payments. FV (optional) is the desired value (future value) to be reached at the end of the periodic payments. Type (optional) defines whether the payment is due at the beginning (1) or the end (0) of a period.
PPMT(rate; period; NPER; PV; FV; type)	Returns for a given period the payment on the principal for an investment that is based on periodic and constant payments and a constant interest rate. Rate is the periodic interest rate. Period is the amortization period. NPER is the total number of periods during which annuity is paid. PV is the present value in the sequence of payments. FV (optional) is the desired (future) value. Type (optional) defines whether the payment is due at the beginning (1) or the end (0) of a period.
PRICE(settlement; maturity; rate; yield; redemption; frequency; basis)	Calculates the market value of a fixed interest security with a par value of 100 currency units as a function of the forecast yield. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Rate is the annual nominal rate of interest (coupon interest rate). Yield is the annual yield of the security. Redemption is the redemption value per 100 currency units of par value. Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.

Syntax	Description
PRICEDISC(settlement; maturity; discount; redemption; basis)	Calculates the price per 100 currency units of par value of a non-interest-bearing security. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Discount is the discount of a security as a percentage. Redemption is the redemption value per 100 currency units of par value. Basis indicates how the year is to be calculated.
PRICEMAT(settlement; maturity; issue; rate; yield; basis)	Calculates the price per 100 currency units of par value of a security, that pays interest on the maturity date. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Issue is the date of issue of the security. Rate is the interest rate of the security on the issue date. Yield is the annual yield of the security. Basis indicates how the year is to be calculated.
PV(rate; NPER; PMT; FV; type)	Returns the present value of an investment resulting from a series of regular payments. Rate defines the interest rate per period. NPER is the total number of payment periods. PMT is the regular payment made per period. FV (optional) defines the future value remaining after the final installment has been made. Type (optional) defines whether the payment is due at the beginning (1) or the end (0) of a period.
RATE(NPER; PMT; PV; FV; type; guess)	Returns the constant interest rate per period of an annuity. NPER is the total number of periods, during which payments are made (payment period). PMT is the constant payment (annuity) paid during each period. PV is the cash value in the sequence of payments. FV (optional) is the future value, which is reached at the end of the periodic payments. Type (optional) defines whether the payment is due at the beginning (1) or the end (0) of a period. Guess (optional) determines the estimated value of the interest with iterative calculation.
RECEIVED(settlement; maturity; investment; discount; basis)	Calculates the amount received that is paid for a fixed-interest security at a given point in time. Settlement is the date of purchase of the security. Maturity is the date on which the security matures. Investment is the purchase sum. Discount is the percentage discount on acquisition of the security. Basis indicates how the year is to be calculated.
RRI(P; PV; FV)	Calculates the interest rate resulting from the profit (return) of an investment. P is the number of periods needed for calculating the interest rate. PV is the present value (must be >0). FV is determines what is desired as the cash value of the deposit.
SLN(cost; salvage; life)	Returns the straight-line depreciation of an asset for one period. The amount of the depreciation is constant during the depreciation period. Cost is the initial cost of an asset. Salvage is the value of an asset at the end of the depreciation. Life is the depreciation period determining the number of periods in the depreciation of the asset.

Syntax	Description
SYD(cost; salvage; life; period)	Returns the arithmetic-declining depreciation rate. Use this function to calculate the depreciation amount for one period of the total depreciation span of an object. Arithmetic declining depreciation reduces the depreciation amount from period to period by a fixed sum. Cost is the initial cost of an asset. Salvage is the value of an asset after depreciation. Life is the period fixing the time span over which an asset is depreciated. Period defines the period for which the depreciation is to be calculated.
TBILLEQ(settlement; maturity; discount)	Calculates the annual return on a treasury bill. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). (The settlement and maturity date must be in the same year.) Discount is the percentage discount on acquisition of the security.
TBILLPRICE(settlement; maturity; discount)	Calculates the price of a treasury bill per 100 currency units. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Discount is the percentage discount upon acquisition of the security.
TBILLYIELD(settlement; maturity; price)	Calculates the yield of a treasury bill. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Price is the price (purchase price) of the treasury bill per 100 currency units of par value.
VDB(cost; salvage; life; start; end; factor; type)	Returns the depreciation of an asset for a specified or partial period using a variable declining balance method. Cost is the initial value of an asset. Salvage is the value of an asset at the end of the depreciation. Life is the depreciation duration of the asset. Start is the start of the depreciation entered in the same date unit as the life. End is the end of the depreciation. Factor (optional) is the depreciation factor. FA=2 is double rate depreciation. Type (optional) defines whether the payment is due at the beginning (1) or the end (0) of a period.
XIRR(values; dates; guess)	Calculates the internal rate of return for a list of payments which take place on different dates. The calculation is based on a 365 days per year basis, ignoring leap years. If the payments take place at regular intervals, use the IRR function. Values and dates are a series of payments and the series of associated date values entered as cell references. Guess (optional) is a guess for the internal rate of return. The default is 10%.
XNPV(rate; values; dates)	Calculates the capital value (net present value) for a list of payments which take place on different dates. The calculation is based on a 365 days per year basis, ignoring leap years. If the payments take place at regular intervals, use the NPV function. Rate is the internal rate of return for the payments. Values and dates are a series of payments and the series of associated date values entered as cell references.

Syntax	Description
YIELD(settlement; maturity; rate; price; redemption; frequency; basis)	Calculates the yield of a security. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Rate is the annual rate of interest. Price is the price (purchase price) of the security per 100 currency units of par value. Redemption is the redemption value per 100 currency units of par value. Frequency is the number of interest payments per year (1, 2 or 4). Basis indicates how the year is to be calculated.
YIELDDISC(settlement; maturity; price; redemption; basis)	Calculates the annual yield of a non-interest-bearing security. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Price is the price (purchase price) of the security per 100 currency units of par value. Redemption is the redemption value per 100 currency units of par value. Basis indicates how the year is to be calculated.
YIELDMAT(settlement; maturity; issue; rate; price; basis)	Calculates the annual yield of a security, the interest of which is paid on the date of maturity. Settlement is the date of purchase of the security. Maturity is the date on which the security matures (expires). Issue is the date of issue of the security. Rate is the interest rate of the security on the issue date. Price is the price (purchase price) of the security per 100 currency units of par value. Basis indicates how the year is to be calculated.

Statistical analysis functions

Calc includes over 70 statistical functions which enable the evaluation of data from simple arithmetic calculations, such as averaging, to advanced distribution and probability computations. Several other statistics-based functions are available through the Add-ins which are noted at the end of this appendix.

Table 4: Statistical analysis functions

Syntax	Description
AVEDEV(number1; number2; ... number_30)	Returns the average of the absolute deviations of data points from their mean. Displays the diffusion in a data set. Number_1; number_2; ... number_30 are values or ranges that represent a sample. Each number can also be replaced by a reference.
AVERAGE(number_1; number_2; ... number_30)	Returns the average of the arguments. Number_1; number_2; ... number_30 are numerical values or ranges. Text is ignored.
AVERAGEA(value_1; value_2; ... value_30)	Returns the average of the arguments. The value of a text is 0. Value_1; value_2; ... value_30 are values or ranges.
B(trials; SP; T_1; T_2)	Returns the probability of a sample with binomial distribution. Trials is the number of independent trials. SP is the probability of success on each trial. T_1 defines the lower limit for the number of trials. T_2 (optional) defines the upper limit for the number of trials.

Syntax	Description
BETADIST(number; alpha; beta; start; end)	Returns the cumulative beta probability density function. Number is the value between Start and End at which to evaluate the function. Alpha is a parameter to the distribution. Beta is a parameter to the distribution. Start (optional) is the lower bound for number . End (optional) is the upper bound for number .
BETAINV(number; alpha; beta; start; end)	Returns the inverse of the cumulative beta probability density function. Number is the value between Start and End at which to evaluate the function. Alpha is a parameter to the distribution. Beta is a parameter to the distribution. Start (optional) is the lower bound for number . End (optional) is the upper bound for number .
BINOMDIST(X; trials; SP; C)	Returns the individual term binomial distribution probability. X is the number of successes in a set of trials. Trials is the number of independent trials. SP is the probability of success on each trial. C = 0 calculates the probability of a single event and C = 1 calculates the cumulative probability.
CHIDIST(number; degrees_freedom)	Returns the probability value that a hypothesis will be confirmed from the indicated chi square. The probability determined by CHIDIST can also be determined by CHITEST. Number is the chi-square value of the random sample used to determine the error probability. Degrees_freedom is the degrees of freedom of the experiment.
CHIINV(number; degrees_freedom)	Returns the inverse of the one-tailed probability of the chi-squared distribution. Number is the value of the error probability. Degrees_freedom is the degrees of freedom of the experiment.
CHITEST(data_B; data_E)	Returns the chi-square distribution from a random distribution of two test series based on the chi-square test for independence. The probability determined by CHITEST can also be determined with CHIDIST, in which case the chi square of the random sample must then be passed as a parameter instead of the data row. Data_B is the array of the observations. Data_E is the range of the expected values.
CONFIDENCE(alpha; STDEV; size)	Returns the (1-alpha) confidence interval for a normal distribution. Alpha is the level of the confidence interval. STDEV is the standard deviation for the total population. Size is the size of the total population.
CORREL(data_1; data_2)	Returns the correlation coefficient between two data sets. Data_1 is the first data set. Data_2 is the second data set.
COUNT(value_1; value_2; ... value_30)	Counts how many numbers are in the list of arguments. Text entries are ignored. Value_1; value_2; ... value_30 are values or ranges which are to be counted.

Syntax	Description
COUNTA(value_1; value_2; ... value_30)	Counts how many values are in the list of arguments. Text entries are also counted, even when they contain an empty string of length 0. If an argument is an array or reference, empty cells within the array or reference are ignored. value_1; value_2; ... value_30 are up to 30 arguments representing the values to be counted.
COVAR(data_1; data_2)	Returns the covariance of the product of paired deviations. Data_1 is the first data set. Data_2 is the second data set.
CRITBINOM(trials; SP; alpha)	Returns the smallest value for which the cumulative binomial distribution is less than or equal to a criterion value. Trials is the total number of trials. SP is the probability of success for one trial. Alpha is the threshold probability to be reached or exceeded.
DEVSQ(number_1; number_2; ... number_30)	Returns the sum of squares of deviations based on a sample mean. Number_1; number_2; ... number_30 are numerical values or ranges representing a sample.
EXPONDIST(number; lambda; C)	Returns the exponential distribution. Number is the value of the function. Lambda is the parameter value. C is a logical value that determines the form of the function. C = 0 calculates the density function, and C = 1 calculates the distribution.
FDIST(number; degrees_freedom_1; degrees_freedom_2)	Calculates the values of an F probability distribution. Number is the value for which the F distribution is to be calculated. Degrees_freedom_1 is the degrees of freedom in the numerator in the F distribution. Degrees_freedom_2 is the degrees of freedom in the denominator in the F distribution.
FINV(number; degrees_freedom_1; degrees_freedom_2)	Returns the inverse of the F probability distribution. Number is probability value for which the inverse F distribution is to be calculated. Degrees_freedom_1 is the number of degrees of freedom in the numerator of the F distribution. Degrees_freedom_2 is the number of degrees of freedom in the denominator of the F distribution.
FISHER(number)	Returns the Fisher transformation for the given number and creates a function close to a normal distribution.
FISHERINV(number)	Returns the inverse of the Fisher transformation for the given number and creates a function close to a normal distribution.
FORECAST(value; data_Y; data_X)	Extrapolates future values based on existing x and y values. Value is the x value, for which the y value of the linear regression is to be returned. Data_Y is the array or range of known Ys. Data_X is the array or range of known Xs. Does not work for exponential functions.
FTEST(data_1; data_2)	Returns the result of an F test. Data_1 is the first record array. Data_2 is the second record array.

Syntax	Description
GAMMADIST(number; alpha; beta; C)	Returns the values of a Gamma cumulative distribution. Number is the value for which the Gamma distribution is to be calculated. Alpha is the parameter Alpha of the Gamma distribution. Beta is the parameter Beta of the Gamma distribution. C = 0 calculates the density function, and C = 1 calculates the distribution.
GAMMAINV(number; alpha; beta)	Returns the inverse of the Gamma cumulative distribution. This function allows you to search for variables with different distribution. Number is the probability value for which the inverse Gamma distribution is to be calculated. Alpha is the parameter Alpha of the Gamma distribution. Beta is the parameter Beta of the Gamma distribution.
GAMMALN(number)	Returns the natural logarithm of the Gamma function, G(x), for the given number .
GAUSS(number)	Returns the standard normal cumulative distribution for the given number .
GEOMEAN(number_1; number_2; ... number_30)	Returns the geometric mean of a sample. Number_1; number_2; ... number_30 are numerical arguments or ranges that represent a random sample.
HARMEAN(number_1; number_2; ... number_30)	Returns the harmonic mean of a data set. Number_1; number_2; ... number_30 are values or ranges that can be used to calculate the harmonic mean.
HYPGEOMDIST(X; n_sample; successes; n_population)	Returns the hypergeometric distribution. X is the number of results achieved in the random sample. N_sample is the size of the random sample. Successes is the number of possible results in the total population. N_population is the size of the total population.
INTERCEPT(data_Y; data_X)	Calculates the y-value at which a line will intersect the y-axis by using known x-values and y-values. Data_Y is the dependent set of observations or data. Data_X is the independent set of observations or data. Names, arrays or references containing numbers must be used here. Numbers can also be entered directly.
KURT(number_1; number_2; ... number_30)	Returns the kurtosis of a data set (at least 4 values required). Number_1; number_2; ... number_30 are numerical arguments or ranges representing a random sample of distribution.
LARGE(data; rank_c)	Returns the Rank_c-th largest value in a data set. Data is the cell range of data. Rank_c is the ranking of the value (2nd largest, 3rd largest, etc.) written as an integer.
LOGINV(number; mean; STDEV)	Returns the inverse of the lognormal distribution for the given Number , a probability value. Mean is the arithmetic mean of the standard logarithmic distribution. STDEV is the standard deviation of the standard logarithmic distribution.

Syntax	Description
LOGNORMDIST(number; mean; STDEV)	Returns the cumulative lognormal distribution for the given Number , a probability value. Mean is the mean value of the standard logarithmic distribution. STDEV is the standard deviation of the standard logarithmic distribution.
MAX(number_1; number_2; ... number_30)	Returns the maximum value in a list of arguments. Number_1; number_2; ... number_30 are numerical values or ranges.
MAXA(value_1; value_2; ... value_30)	Returns the maximum value in a list of arguments. Unlike MAX, text can be entered. The value of the text is 0. Value_1; value_2; ... value_30 are values or ranges.
MEDIAN(number_1; number_2; ... number_30)	Returns the median of a set of numbers. Number_1; number_2; ... number_30 are values or ranges, which represent a sample. Each number can also be replaced by a reference.
MIN(number_1; number_2; ... number_30)	Returns the minimum value in a list of arguments. Number_1; number_2; ... number_30 are numerical values or ranges.
MINA(value_1; value_2; ... value_30)	Returns the minimum value in a list of arguments. Here text can also be entered. The value of the text is 0. Value_1; value_2; ... value_30 are values or ranges.
MODE(number_1; number_2; ... number_30)	Returns the most common value in a data set. Number_1; number_2; ... number_30 are numerical values or ranges. If several values have the same frequency, it returns the smallest value. An error occurs when a value does not appear twice.
NEGBINOMDIST(X; R; SP)	Returns the negative binomial distribution. X is the value returned for unsuccessful tests. R is the value returned for successful tests. SP is the probability of the success of an attempt.
NORMDIST(number; mean; STDEV; C)	Returns the normal distribution for the given Number in the distribution. Mean is the mean value of the distribution. STDEV is the standard deviation of the distribution. C = 0 calculates the density function, and C = 1 calculates the distribution.
NORMINV(number; mean; STDEV)	Returns the inverse of the normal distribution for the given Number in the distribution. Mean is the mean value in the normal distribution. STDEV is the standard deviation of the normal distribution.
NORMSDIST(number)	Returns the standard normal cumulative distribution for the given Number .
NORMSINV(number)	Returns the inverse of the standard normal distribution for the given Number , a probability value.
PEARSON(data_1; data_2)	Returns the Pearson product moment correlation coefficient r . Data_1 is the array of the first data set. Data_2 is the array of the second data set.

Syntax	Description
PERCENTILE(data; alpha)	Returns the alpha-percentile of data values in an array. Data is the array of data. Alpha is the percentage of the scale between 0 and 1.
PERCENTRANK(data; value)	Returns the percentage rank (percentile) of the given value in a sample. Data is the array of data in the sample.
PERMUT(count_1; count_2)	Returns the number of permutations for a given number of objects. Count_1 is the total number of objects. Count_2 is the number of objects in each permutation.
PERMUTATIONA(count_1; count_2)	Returns the number of permutations for a given number of objects (repetition allowed). Count_1 is the total number of objects. Count_2 is the number of objects in each permutation.
PHI(number)	Returns the values of the distribution function for a standard normal distribution for the given Number .
POISSON(number; mean; C)	Returns the Poisson distribution for the given Number . Mean is the middle value of the Poisson distribution. C = 0 calculates the density function, and C = 1 calculates the distribution.
PROB(data; probability: start; end)	Returns the probability that values in a range are between two limits. Data is the array or range of data in the sample. Probability is the array or range of the corresponding probabilities. Start is the start value of the interval whose probabilities are to be summed. End (optional) is the end value of the interval whose probabilities are to be summed. If this parameter is missing, the probability for the Start value is calculated.
QUARTILE(data; type)	Returns the quartile of a data set. Data is the array of data in the sample. Type is the type of quartile. (0 = Min, 1 = 25%, 2 = 50% (Median), 3 = 75% and 4 = Max.)
RANK(value; data; type)	Returns the rank of the given Value in a sample. Data is the array or range of data in the sample. Type (optional) is the sequence order, either ascending (0) or descending (1).
RSQ(data_Y; data_X)	Returns the square of the Pearson correlation coefficient based on the given values. Data_Y is an array or range of data points. Data_X is an array or range of data points.
SKEW(number_1; number_2; ... number_30)	Returns the skewness of a distribution. Number_1; number_2; ... number_30 are numerical values or ranges.
SLOPE(data_Y; data_X)	Returns the slope of the linear regression line. Data_Y is the array or matrix of Y data. Data_X is the array or matrix of X data.
SMALL(data; rank_c)	Returns the Rank_c-th smallest value in a data set. Data is the cell range of data. Rank_c is the rank of the value (2nd smallest, 3rd smallest, etc.) written as an integer.

Syntax	Description
STANDARDIZE(number; mean; STDEV)	Converts a random variable to a normalized value. Number is the value to be standardized. Mean is the arithmetic mean of the distribution. STDEV is the standard deviation of the distribution.
STDEV(number_1; number_2; ... number_30)	Estimates the standard deviation based on a sample. Number_1; number_2; ... number_30 are numerical values or ranges representing a sample based on an entire population.
STDEVA(value_1; value_2; ... value_30)	Calculates the standard deviation of an estimation based on a sample. Value_1; value_2; ... value_30 are values or ranges representing a sample derived from an entire population. Text has the value 0.
STDEV(number_1; number_2; ... number_30)	Calculates the standard deviation based on the entire population. Number_1; number_2; ... number_30 are numerical values or ranges representing a sample based on an entire population.
STDEVPA(value_1; value_2; ... value_30)	Calculates the standard deviation based on the entire population. Value_1; value_2; ... value_30 are values or ranges representing a sample derived from an entire population. Text has the value 0.
STEYX(data_Y; data_X)	Returns the standard error of the predicted y value for each x in the regression. Data_Y is the array or matrix of Y data. Data_X is the array or matrix of X data.
TDIST(number; degrees_freedom; mode)	Returns the t-distribution for the given Number . Degrees_freedom is the number of degrees of freedom for the t-distribution. Mode = 1 returns the one-tailed test, Mode = 2 returns the two-tailed test.
TINV(number; degrees_freedom)	Returns the inverse of the t-distribution, for the given Number associated with the two-tailed t-distribution. Degrees_freedom is the number of degrees of freedom for the t-distribution.
TRIMMEAN(data; alpha)	Returns the mean of a data set without the Alpha proportion of data at the margins. Data is the array of data in the sample. Alpha is the proportion of the marginal data that will not be taken into consideration.
TTEST(data_1; data_2; mode; type)	Returns the probability associated with a Student's t-Test. Data_1 is the dependent array or range of data for the first record. Data_2 is the dependent array or range of data for the second record. Mode = 1 calculates the one-tailed test, Mode = 2 the two-tailed test. Type of t-test to perform: paired (1), equal variance (homoscedastic) (2), or unequal variance (heteroscedastic) (3).
VAR(number_1; number_2; ... number_30)	Estimates the variance based on a sample. Number_1; number_2; ... number_30 are numerical values or ranges representing a sample based on an entire population.

Syntax	Description
VARA(value_1; value_2; ... value_30)	Estimates a variance based on a sample. The value of text is 0. Value_1; value_2; ... value_30 are values or ranges representing a sample derived from an entire population. Text has the value 0.
VARP(Number_1; number_2; ... number_30)	Calculates a variance based on the entire population. Number_1; number_2; ... number_30 are numerical values or ranges representing an entire population.
VARPA(value_1; value_2; .. value_30)	Calculates the variance based on the entire population. The value of text is 0. Value_1; value_2; ... value_30 are values or ranges representing an entire population.
WEIBULL(number; alpha; beta; C)	Returns the values of the Weibull distribution for the given Number . Alpha is the Alpha parameter of the Weibull distribution. Beta is the Beta parameter of the Weibull distribution. C indicates the type of function: C= 0 the form of the function is calculated, C=1 the distribution is calculated.
ZTEST(data; number; sigma)	Returns the two-tailed P value of a z test with standard distribution. Data is the array of the data. Number is the value to be tested. Sigma (optional) is the standard deviation of the total population. If this argument is missing, the standard deviation of the sample is processed.

Date and time functions

Use these functions for inserting, editing, and manipulating dates and times. LibreOffice handles and computes a date/time value as a number. When you assign the number format "Number" to a date or time value, it is displayed as a number. For example, 01/01/2000 12:00 PM, converts to 36526.5. This is just a matter of formatting; the actual value is always stored and manipulated as a number. To see the date or time displayed in a standard format, change the number format (date or time) accordingly.

To set the default date format used by Calc. go to **Tools** → **Options** → **LibreOffice Calc** → **Calculate**.

Caution



When entering dates, slashes or dashes used as date separators may be interpreted as arithmetic operators. To keep dates from being interpreted as parts of formulas, and thus returning erroneous results, always place them in quotation marks, for example, "12/08/52".

Table 5: Data and time functions

Syntax	Description
DATE(year; month; day)	Converts a date written as year, month, day to an internal serial number and displays it in the cell's formatting. Year is an integer between 1583 and 9956 or 0 and 99. Month is an integer between 1 and 12. Day is an integer between 1 and 31.
DATEVALUE("Text")	Returns the internal date number for text in quotes. Text is a valid date expression and must be entered with quotation marks.

Syntax	Description
DAY(number)	Returns the day, as an integer, of the given date value. A negative date/time value can be entered. Number is a time value.
DAYS(date_2; date_1)	Calculates the difference, in days, between two date values. Date_1 is the start date. Date_2 is the end date. If Date_2 is an earlier date than Date_1 , the result is a negative number.
DAYS360(date_1; date_2; type)	Returns the difference between two dates based on the 360 day year used in interest calculations. If Date_2 is earlier than Date_1 , the function will return a negative number. Type (optional) determines the type of difference calculation: the US method (0) or the European method (≠0).
DAYSINMONTH(date)	Calculates the number of days in the month of the given date.
DAYSINYEAR(date)	Calculates the number of days in the year of the given date .
EASTERSUNDAY(integer)	Returns the date of Easter Sunday for the entered year. Year is an integer between 1583 and 9956 or 0 and 99.
EDATE(start_date; months)	The result is a date a number of Months away from the given Start_date . Only months are considered; days are not used for calculation. Months is the number of months.
EOMONTH(start_date; months)	Returns the date of the last day of a month which falls Months away from the given Start_date . Months is the number of months before (negative) or after (positive) the start date.
HOUR(number)	Returns the hour, as an integer, for the given time value. Number is a time value.
ISLEAPYEAR(date)	Determines whether a given date falls within a leap year. Returns either 1 (TRUE) or 0 (FALSE).
MINUTE(number)	Returns the minute, as an integer, for the given time value. Number is a time value.
MONTH(number)	Returns the month, as an integer, for the given date value. Number is a time value.
MONTHS(start_date; end_date; type)	Calculates the difference, in months, between two date values. Date_1 is the start (earlier) date. Date_2 is the end date. Type is one of two possible values, 0 (interval) or 1 (in calendar months). If Date_2 is an earlier date than Date_1 , the result is a negative number.
NETWORKDAYS(start_date; end_date; holidays)	Returns the number of workdays between start_date and end_date . Holidays can be deducted. Start_date is the date from which the calculation is carried out. End_date is the date up to which the calculation is carried out. If the start or end date is a workday, the day is included in the calculation. Holidays (optional) is a list of holidays. Enter a cell range in which the holidays are listed individually.
NOW()	Returns the computer system date and time. The value is updated when your document recalculates. NOW is a function without arguments.

Syntax	Description
SECOND(number)	Returns the second, as an integer, for the given time value. Number is a time value.
TIME(hour; minute; second)	Returns the current time value from values for hours, minutes and seconds. This function can be used to convert a time based on these three elements to a decimal time value. Hour , minute and second must all be integers.
TIMEVALUE(text)	Returns the internal time number from a text enclosed by quotes in a time entry format. The internal number indicated as a decimal is the result of the date system used under LibreOffice to calculate date entries.
TODAY()	Returns the current computer system date. The value is updated when your document recalculates. TODAY is a function without arguments.
WEEKDAY(number; type)	Returns the day of the week for the given number (date value). The day is returned as an integer based on the type. Type determines the type of calculation: type = 1 (default), the weekdays are counted starting from Sunday (Monday = 0); type = 2, the weekdays are counted starting from Monday (Monday = 1); type = 3, the weekdays are counted starting from Monday (Monday = 0).
WEEKNUM(number; mode)	Calculates the number of the calendar week of the year for the internal date number . Mode sets the start of the week and the calculation type: 1 = Sunday, 2 = Monday.
WEEKNUM_ADD(date; return_type)	Calculates the calendar week of the year for a Date . Date is the date within the calendar week. Return_type sets the start of the week and the calculation type: 1 = Sunday, 2 = Monday.
WEEKS(start_date; end_date; type)	Calculates the difference in weeks between two dates, start_date and end_date . Type is one of two possible values, 0 (interval) or 1 (in numbers of weeks).
WEEKSINYEAR(date)	Calculates the number of weeks in a year until a certain date . A week that spans two years is added to the year in which most days of that week occur.
WORKDAY(start_date; days; holidays)	Returns a date number that can be formatted as a date. You then see the date of a day that is a certain number of Workdays away from the start_date . Holidays (optional) is a list of holidays. Enter a cell range in which the holidays are listed individually.
YEAR(number)	Returns the year as a number according to the internal calculation rules. Number shows the internal date value for which the year is to be returned.
YEARFRAC(start_date; end_date; basis)	Returns a number between 0 and 1, representing the fraction of a year between start_date and end_date . Start_date and end_date are two date values. Basis is chosen from a list of options and indicates how the year is to be calculated.
YEARS(tart_date; end_date; type)	Calculates the difference in years between two dates: the start_date and the end_date . Type calculates the type of difference.

Logical functions

Use the logical functions to test values and produce results based on the result of the test. These functions are conditional and provide the ability to write longer formulas based on input or output.

Table 6: Logical functions

Syntax	Description
AND(logical_value_1; logical_value_2; ...logical_value_30)	Returns TRUE if all arguments are TRUE. If any element is FALSE, this function returns the FALSE value. Logical_value_1; logical_value_2; ...logical_value_30 are conditions to be checked. All conditions can be either TRUE or FALSE. If a range is entered as a parameter, the function uses the value from the range that is in the current column or row. The result is TRUE if the logical value in all cells within the cell range is TRUE
FALSE()	Set the logical value to FALSE. The FALSE() function does not require any arguments.
IF(test; then_value; otherwise_value)	Specifies a logical test to be performed. Test is any value or expression that can be TRUE or FALSE. Then_value (optional) is the value that is returned if the logical test is TRUE. Otherwise_value (optional) is the value that is returned if the logical test is FALSE.
NOT(logical_value)	Reverses the logical value. Logical_value is any value to be reversed.
OR(logical_value_1; logical_value_2; ...logical_value_30)	Returns TRUE if at least one argument is TRUE. Returns the value FALSE if all the arguments have the logical value FALSE. Logical_value_1; logical_value_2; ...logical_value_30 are conditions to be checked. All conditions can be either TRUE or FALSE. If a range is entered as a parameter, the function uses the value from the range that is in the current column or row.
TRUE()	Sets the logical value to TRUE. The TRUE() function does not require any arguments.

Informational functions

These functions provide information (or feedback) regarding the results of a test for a specific condition, or a test for the type of data or content a cell contains.

Table 7: Informational functions

Syntax	Description
CELL(info_type; reference)	Returns information on a cell such as its address, formatting or contents of a cell based on the value of the info_type argument. Info_type specifies the type of information to be returned and comes from a predefined list of arguments. Info_type is not case sensitive, but it must be enclosed within quotes. Reference is the address of the cell to be examined. If reference is a range, the cell reference moves to the top left of the range. If reference is missing, Calc uses the position of the cell in which this formula is located.
CURRENT()	Calculates the current value of a formula at the actual position.

Syntax	Description
FORMULA(reference)	Displays the formula of a formula cell at any position. The formula will be returned as a string in the Reference position. If no formula cell can be found, or if the presented argument is not a reference, returns the error value #N/A.
ISBLANK(value)	Returns TRUE if the reference to a cell is blank. This function is used to determine if the content of a cell is empty. A cell with a formula inside is not empty. If an error occurs, the function returns a logical or numerical value. Value is the content to be tested.
ISERR(value)	Returns TRUE if the value refers to any error value except #N/A. You can use this function to control error values in certain cells. If an error occurs, the function returns a logical or numerical value. Value is any value or expression in which a test is performed to determine whether an error value not equal to #N/A is present.
ISERROR(value)	The ISERROR tests if the cells contain general error values. ISERROR recognizes the #N/A error value. If an error occurs, the function returns a logical or numerical value. Value is any value where a test is performed to determine whether it is an error value.
ISEVEN_ADD(number)	Tests for even numbers . Returns TRUE (1) if the number returns a whole number when divided by 2.
ISFORMULA(reference)	Returns TRUE if a cell is a formula cell. If an error occurs, the function returns a logical or numerical value. Reference indicates the reference to a cell in which a test will be performed to determine if it contains a reference.
ISLOGICAL(value)	Returns TRUE if the cell contains a logical number format. The function is used in order to check for both TRUE and FALSE values in certain cells. If an error occurs, the function returns a logical or numerical value. Value is the value to be tested for logical number format.
ISNA(value)	Returns TRUE if a cell contains the #N/A (value not available) error value. If an error occurs, the function returns a logical or numerical value. Value is the value or expression to be tested.
ISNONTEXT(value)	Tests if the cell contents are text or numbers, and returns FALSE if the contents are text. If an error occurs, the function returns a logical or numerical value. Value is any value or expression where a test is performed to determine whether it is a text or numbers or a Boolean value.
ISNUMBER(value)	Returns TRUE if the value refers to a number. If an error occurs, the function returns a logical or numerical value. Value is any expression to be tested to determine whether it is a number or text.
ISODD_ADD(number)	Returns TRUE (1) if the number does not return a whole number when divided by 2. Number is the number to be tested.

Syntax	Description
ISREF(value)	Tests if the content of one or several cells is a reference. Verifies the type of references in a cell or a range of cells. If an error occurs, the function returns a logical or numerical value. Value is the value to be tested, to determine whether it is a reference.
ISTEXT(value)	Returns TRUE if the cell contents refer to text. If an error occurs, the function returns a logical or numerical value. Value is a value, number, Boolean value, or error value to be tested.
N(value)	Returns the number 1, if the parameter is TRUE. Returns the parameter, if the parameter is a number. Returns the number 0 for other parameters. If an error occurs, the function returns a logical or numerical value. Value is the parameter to be converted into a number.
NA()	Returns the error value #N/A.
TYPE(value)	Returns the type of value. If an error occurs, the function returns a logical or numerical value. Value is a specific value for which the data type is determined. Value 1 = number, value 2 = text, value 4 = Boolean value, value 8 = formula, value 16 = error value.

Database functions

This section deals with functions used with data organized as one row of data for one record. The *Database* category should not be confused with the Base database component in LibreOffice. A Calc database is simple a range of cells that comprises a block of related data where each row contains a separate record. There is no connection between a database in LibreOffice and the *Database* category in LibreOffice Calc.

The database functions use the following common arguments:

- **Database** is a range of cells which define the database.
- **Database_field** specifies the column where the function operates on after the search criteria of the first parameter is applied and the data rows are selected. It is not related to the search criteria itself. The number 0 specifies the whole data range. To reference a column by using the column header name, place quotation marks around the header name.
- **Search_criteria** is a cell range containing the search criteria.. Empty cells in the search criteria range will be ignored.

Note

All of the **search-criteria** arguments for the database functions support regular expressions. For example, "all.*" can be entered to find the first location of "all" followed by any characters. To search for text that is also a regular expression, precede every character with a \ character. You can switch the automatic evaluation of regular expressions on and off in **Tools** → **Options** → **LibreOffice Calc** → **Calculate**.

Table 8: Database average

Syntax	Description
DAVERAGE(database; database_field; search_criteria)	Returns the average of the values of all cells (fields) in all rows (database records) that match the specified search_criteria . The search supports regular expressions.
DCOUNT(database; database_field; search_criteria)	Counts the number of rows (records) in a database that match the specified search_criteria and contain numerical values. The search supports regular expressions. For the database_field parameter, enter a cell address to specify the column, or enter the number 0 for the entire database. The parameter cannot be empty.
DCOUNTA(database; database_field; search_criteria)	Counts the number of rows (records) in a database that match the specified search_criteria and contain numeric or alphanumeric values. The search supports regular expressions.
DGET(database; database_field; search_criteria)	Returns the contents of the referenced cell in a database which matches the specified search_criteria . In case of an error, the function returns either #VALUE! for no row found, or Err502 for more than one cell found.
DMAX(database; database_field; search_criteria)	Returns the maximum content of a cell (field) in a database (all records) that matches the specified search_criteria . The search supports regular expressions.
DMIN(database; database_field; search_criteria)	Returns the minimum content of a cell (field) in a database that matches the specified search_criteria . The search supports regular expressions.
DPRODUCT(database; database_field; search_criteria)	Multiplies all cells of a data range where the cell contents match the search_criteria . The search supports regular expressions.
DSTDEV(database; database_field; search_criteria)	Calculates the standard deviation of a population based on a sample, using the numbers in a database column that match the search_criteria . The records are treated as a sample of data. Note that a representative result of a large population can not be obtained from a sample of fewer than one thousand.
DSTDEVP(database; database_field; search_criteria)	Calculates the standard deviation of a population based on all cells of a data range which match the search_criteria . The records from the example are treated as the whole population.
DSUM(database; database_field; search_criteria)	Returns the total of all cells in a database field in all rows (records) that match the specified search_criteria . The search supports regular expressions.
DVAR(database; database_field; search_criteria)	Returns the variance of all cells of a database field in all records that match the specified search_criteria . The records from the example are treated as a sample of data. A representative result of a large population cannot be obtained from a sample population of fewer than one thousand.
DVARP(database; database_field; search_criteria)	Calculates the variance of all cell values in a database field in all records that match the specified search_criteria . The records are from the example are treated as an entire population.

Array functions

Table 9: Array functions

Syntax	Description
FREQUENCY(data; classes)	Calculates the frequency distribution in a one-column-array. The default value supply and the number of intervals or classes are used to count how many values are omitted on the single intervals. Data is the array of, or reference to, the set of values to be counted. Classes is the array of the class set.
GROWTH(data_Y; data_X; new_data_X; function_type)	Calculates the points of an exponential trend in an array. Data_Y is the Y Data array. Data_X (optional) is the X Data array. New_Data_X (optional) is the X data array, in which the values are recalculated. Function_type is optional. If function_type = 0, functions in the form $y = m^x$ are calculated. Otherwise, $y = b \cdot m^x$ functions are calculated.
LINEST(data_Y; data_X; linear_type; stats)	Returns the parameters of a linear trend. Data_Y is the Y Data array. Data_X (optional) is the X Data array. Linear_Type (optional): If the line goes through the zero point, then set Linear_Type = 0. Stats (optional): If Stats=0, only the regression coefficient is calculated. Otherwise, other statistics will be seen.
LOGEST(data_Y; data_X; function_type; stats)	Calculates the adjustment of the entered data as an exponential regression curve ($y=b \cdot m^x$). Data_Y is the Y Data array. Data_X (optional) is the X Data array. Function_type (optional): If function_type = 0, functions in the form $y = m^x$ are calculated. Otherwise, $y = b \cdot m^x$ functions are calculated. Stats (optional). If Stats=0, only the regression coefficient is calculated.
MDETERM(array)	Returns the array determinant of an array. This function returns a value in the current cell; it is not necessary to define a range for the results. Array is a square array in which the determinants are defined.
MINVERSE(array)	Returns the inverse array. Array is a square array that is to be inverted.
MMULT(array; array)	Calculates the array product of two arrays. The number of columns for array 1 must match the number of rows for array 2. The square array has an equal number of rows and columns. Array at first place is the first array used in the array product. Array at second place is the second array with the same number of rows.
MUNIT(dimensions)	Returns the unitary square array of a certain size. The unitary array is a square array where the main diagonal elements equal 1 and all other array elements are equal to 0. Dimensions refers to the size of the array unit.
SUMPRODUCT(array 1; array 2; ...array 30)	Multiplies corresponding elements in the given arrays, and returns the sum of those products. Array 1; array 2;...array 30 are arrays whose corresponding elements are to be multiplied. At least one array must be part of the argument list. If only one array is given, all array elements are summed.

Syntax	Description
SUMX2MY2(array_X; array_Y)	Returns the sum of the difference of squares of corresponding values in two arrays. Array_X is the first array whose elements are to be squared and added. Array_Y is the second array whose elements are to be squared and subtracted.
SUMX2PY2(array_X; array_Y)	Returns the sum of the sum of squares of corresponding values in two arrays. Array_X is the first array whose arguments are to be squared and added. Array_Y is the second array, whose elements are to be added and squared.
SUMXMY2(array_X; array_Y)	Adds the squares of the variance between corresponding values in two arrays. Array_X is the first array whose elements are to be subtracted and squared. Array_Y is the second array, whose elements are to be subtracted and squared.
TRANSPOSE(array)	Transposes the rows and columns of an array. Array is the array in the spreadsheet that is to be transposed.
TREND(data_Y; data_X; new_data_X; linear_Type)	Returns values along a linear trend. Data_Y is the Y Data array. Data_X (optional) is the X Data array. New_data_X (optional) is the array of the X data, which are used for recalculating values. Linear_type is optional. If linear_type = 0, then lines will be calculated through the zero point. Otherwise, offset lines will also be calculated. The default is linear_type <> 0.

Spreadsheet functions

Use spreadsheet functions to search and address cell ranges and provide feedback regarding the contents of a cell or range of cells. You can use functions such as HYPERLINK() and DDE() to connect to other documents or data sources.

Table 10: Spreadsheet functions

Syntax	Description
ADDRESS(row; column; abs; sheet)	Returns a cell address (reference) as text, according to the specified row and column numbers. Optionally, whether the address is interpreted as an absolute address (for example, \$A\$1) or as a relative address (as A1) or in a mixed form (A\$1 or \$A1) can be determined. The name of the sheet can also be specified. Row is the row number for the cell reference. Column is the column number for the cell reference (the number, not the letter). Abs determines the type of reference. Sheet is the name of the sheet.
AREAS(reference)	Returns the number of individual ranges that belong to a multiple range. A range can consist of contiguous cells or a single cell. Reference is the reference to a cell or cell range.
CHOOSE(index; value1; ... value30)	Uses an index to return a value from a list of up to 30 values. Index is a reference or number between 1 and 30 indicating which value is to be taken from the list. Value1; ... value30 is the list of values entered as a reference to a cell or as individual values.

Syntax	Description
COLUMN(reference)	Returns the column number of a cell reference. If the reference is a cell, the column number of the cell is returned; if the parameter is a cell area, the corresponding column numbers are returned in a single-row array if the formula is entered as an array formula. If the COLUMN function with an area reference parameter is not used for an array formula, only the column number of the first cell within the area is determined. Reference is the reference to a cell or cell area whose first column number is to be found. If no reference is entered, the column number of the cell in which the formula is entered is found. Calc automatically sets the reference to the current cell.
COLUMNS(array)	Returns the number of columns in the given reference. Array is the reference to a cell range whose total number of columns is to be found. The argument can also be a single cell.
DDE(server; file; range; mode)	Returns the result of a DDE-based link. If the contents of the linked range or section changes, the returned value will also change. The spreadsheet can be reloaded, or Edit → Links selected, to see the updated links. Cross-platform links, for example from an LibreOffice installation running on a Windows machine to a document created on a Linux machine, are not supported. Server is the name of a server application. LibreOffice applications have the server name "Soffice". File is the complete file name, including path. Range is the area containing the data to be evaluated. Mode is an optional parameter that controls the method by which the DDE server converts its data into numbers.
ERRORTYPE(reference)	Returns the number corresponding to an error value occurring in a different cell. With the aid of this number, an error message text can be generated. If an error occurs, the function returns a logical or numerical value. Reference contains the address of the cell in which the error occurs.
HLOOKUP(search_criteria; array; index; sorted)	Searches for a value and reference to the cells below the selected area. This function verifies if the first row of an array contains a certain value. The function returns the value in a row of the array, named in the index , in the same column. The search supports regular expressions.
HYPERLINK(URL) or HYPERLINK(URL; cell_text)	When a cell that contains the HYPERLINK function is clicked, the hyperlink opens. URL specifies the link target. The optional cell_text argument is the text displayed in the cell. If the cell_text parameter is not specified, the URL is displayed.

Syntax	Description
INDEX(reference; row; column; range)	Returns the content of a cell, specified by row and column number or an optional range name. Reference is a cell reference, entered either directly or by specifying a range name. If the reference consists of multiple ranges, the reference or range name must be enclosed in parentheses. Row (optional) is the row number of the reference range, for which to return a value. Column (optional) is the column number of the reference range, for which to return a value. Range (optional) is the index of the subrange if referring to a multiple range.
INDIRECT(reference)	Returns the reference specified by a text string. This function can also be used to return the area of a corresponding string. Reference is a reference to a cell or an area (in text form) for which to return the contents.
LOOKUP(search_criterion; search_vector; result_vector)	Returns the contents of a cell either from a one-row or one-column range or from an array. Optionally, the assigned value (of the same index) is returned in a different column and row. As opposed to VLOOKUP and HLOOKUP, search and result vectors may be at different positions; they do not have to be adjacent. Additionally, the search vector for the LOOKUP must be sorted, otherwise the search will not return any usable results. The search supports regular expressions. Search_criterion is the value to be searched for; entered either directly or as a reference. Search_vector is the single-row or single-column area to be searched. Result_vector is another single-row or single-column range from which the result of the function is taken. The result is the cell of the result vector with the same index as the instance found in the search vector.
MATCH(search_criterion; lookup_array; type)	Returns the relative position of an item in an array that matches a specified value. The function returns the position of the value found in the lookup_array as a number. Search_criterion is the value which is to be searched for in the single-row or single-column array. Lookup_array is the reference searched. A lookup array can be a single row or column, or part of a single row or column. Type may take the values 1, 0, or -1. This corresponds to the same function in Microsoft Excel. The search supports regular expressions
OFFSET(reference; rows; columns; height; width)	Returns the value of a cell offset by a certain number of rows and columns from a given reference point. Reference is the cell from which the function searches for the new reference. Rows is the number of cells by which the reference was corrected up (negative value) or down. Columns is the number of columns by which the reference was corrected to the left (negative value) or to the right. Height is the optional vertical height for an area that starts at the new reference position. Width is the optional horizontal width for an area that starts at the new reference position.

Syntax	Description
ROW(reference)	Returns the row number of a cell reference. If the reference is a cell, it returns the row number of the cell. If the reference is a cell range, it returns the corresponding row numbers in a one-column Array if the formula is entered as an array formula. If the ROW function with a range reference is not used in an array formula, only the row number of the first range cell will be returned. Reference is a cell, an area, or the name of an area. If a reference is not indicated, Calc automatically sets the reference to the current cell.
ROWS(array)	Returns the number of rows in a reference or array. Array is the reference or named area whose total number of rows is to be determined.
SHEET(reference)	Returns the sheet number of a reference or a string representing a sheet name. If no parameters are entered, the result is the sheet number of the spreadsheet containing the formula. Reference (optional) is the reference to a cell, an area, or a sheet name string.
SHEETS(reference)	Determines the number of sheets in a reference. If no parameters are entered, the result is the number of sheets in the current document. Reference (optional) is the reference to a sheet or an area.
STYLE(style; time; style2)	Applies a style to the cell containing the formula. After a set amount of time, another style can be applied. This function always returns the value 0, allowing it to be added to another function without changing the value. Style is the name of a cell style assigned to the cell. Time is an optional time range in seconds. Style2 is the optional name of a cell style assigned to the cell after a certain amount of time has passed.
VLOOKUP(search_criterion; array; index; sort_order)	Searches vertically with reference to adjacent cells to the right. If a specific value is contained in the first column of an array, returns the value to the same line of a specific array column named by index . The search supports regular expressions. Search_criterion is the value searched for in the first column of the array. Array is the reference, which must include at least two columns. Index is the number of the column in the array that contains the value to be returned. The first column has the number 1. Sort_order (optional) indicates whether the first column in the array is sorted in ascending order.

Text functions

Use Calc's text functions to search and manipulate text strings or character codes.

Table 11: Text functions

Syntax	Description
ARABIC(text)	Calculates the value of a Roman number. The value range must be between 0 and 3999. Text is the text that represents a Roman number.
BASE(number; radix; [minimum_length])	Converts a positive integer to a specified base then into text using the characters from the base's numbering system (decimal, binary, hexadecimal, etc.). Only the digits 0-9 and the letters A-Z are used. Number is the positive integer to be converted. Radix is the base of the number system. It may be any positive integer between 2 and 36. Minimum_length (optional) is the minimum length of the character sequence that has been created. If the text is shorter than the indicated minimum length, zeros are added to the left of the string.
CHAR(number)	Converts a number into a character according to the current code table. The number can be a two-digit or three-digit integer number. Number is a number between 1 and 255 representing the code value for the character.
CLEAN(text)	Removes all non-printing characters from the string. Text refers to the text from which to remove all non-printable characters.
CODE(text)	Returns a numeric code for the first character in a text string. Text is the text for which the code of the first character is to be found.
CONCATENATE(text_1; text_2; ...; text_30)	Combines several text strings into one string. Text_1; text_2; ... text_30 are text passages that are to be combined into one string.
DECIMAL(text; radix)	Converts text with characters from a number system to a positive integer in the base radix given. The radix must be in the range 2 to 36. Spaces and tabs are ignored. The text field is not case-sensitive. Text is the text to be converted. To differentiate between a hexadecimal number, such as A1 and the reference to cell A1, place the number in quotation marks; for example, "A1" or "FACE". Radix is the base of the number system. It may be any positive integer between 2 and 36.
DOLLAR(value; decimals)	Converts a number to an amount in the currency format, rounded to a specified decimal place. Value is the number to be converted to currency; it can be a number, a reference to a cell containing a number, or a formula which returns a number. Decimals (optional) is the number of decimal places. If no decimals value is specified, all numbers in currency format will be displayed with two decimal places. The currency format is set in the system settings.
EXACT(text_1; text_2)	Compares two text strings and returns TRUE if they are identical. This function is case-sensitive. Text_1 is the first text to compare. Text_2 is the second text to compare.

Syntax	Description
FIND(find_text; text; position)	Looks for a string of text within another string. Where to begin the search can also be defined. The search term can be a number or any string of characters. The search is case-sensitive. Find_text is the text to be found. Text is the text where the search takes place. Position (optional) is the position in the text from which the search starts.
FIXED(number; decimals; no_thousands_separator)	Specifies that a number be displayed with a fixed number of decimal places and with or without a thousands separator. This function can be used to apply a uniform format to a column of numbers. Number is the number to be formatted. Decimals is the number of decimal places to be displayed. No_thousands_separator (optional) determines whether the thousands separator is used or not. If the parameter is a number not equal to 0, the thousands separator is suppressed. If the parameter is equal to 0 or if it is missing altogether, the thousands separators of the current locale setting are displayed.
LEFT(text; number)	Returns the first character or characters in a text string. Text is the text where the initial partial words are to be determined. Number (optional) is the number of characters for the start text. If this parameter is not defined, one character is returned.
LEN(text)	Returns the length of a string including spaces. Text is the text whose length is to be determined.
LOWER(text)	Converts all uppercase letters in a text string to lowercase. Text is the text to be converted.
MID(text; start; number)	Returns a text segment of a character string. The parameters specify the starting position and the number of characters. Text is the text containing the characters to extract. Start is the position of the first character in the text to extract. Number is the number of characters in the part of the text.
PROPER(text)	Capitalizes the first letter in all words of a text string. Text is the text to be converted.
REPLACE(text; position; length; new_text)	Replaces part of a text string with a different text string. This function can be used to replace both characters and numbers (which are automatically converted to text). The result of the function is always displayed as text. To perform further calculations with a number which has been replaced by text, convert it back to a number using the VALUE function. Any text containing numbers must be enclosed in quotation marks so it is not interpreted as a number and automatically converted to text. Text is text of which a part will be replaced. Position is the position within the text where the replacement will begin. Length is the number of characters in text to be replaced. New_text is the text which replaces text .
REPT(text; number)	Repeats a character string by the given number of copies. Text is the text to be repeated. Number is the number of repetitions. The result can be a maximum of 255 characters.

Syntax	Description
RIGHT(text; number)	Defines the last character or characters in a text string. Text is the text of which the right part is to be determined. Number (optional) is the number of characters from the right part of the text.
ROMAN(number; mode)	Converts a number into a Roman numeral. The value range must be between 0 and 3999; the modes can be integers from 0 to 4. Number is the number that is to be converted into a Roman numeral. Mode (optional) indicates the degree of simplification. The higher the value, the greater is the simplification of the Roman numeral.
ROT13(text)	Encrypts a character string by moving the characters 13 positions in the alphabet. After the letter Z, the alphabet begins again (Rotation). Applying the encryption function again to the resulting code, decrypts the text. Text : Enter the character string to be encrypted. ROT13(ROT13(Text)) decrypts the code.
SEARCH(find_text; text; position)	Returns the position of a text segment within a character string. The start of the search can be set as an option. The search text can be a number or any sequence of characters. The search is not case-sensitive. The search supports regular expressions. Find_text is the text to be searched for. Text is the text where the search will take place. Position (optional) is the position in the text where the search is to start.
SUBSTITUTE(text; search_text; new text; occurrence)	Substitutes new text for old text in a string. Text is the text in which text segments are to be exchanged. Search_text is the text segment that is to be replaced (a number of times). New text is the text that is to replace the text segment. Occurrence (optional) indicates how many occurrences of the search text are to be replaced. If this parameter is missing, the search text is replaced throughout.
T(value)	Converts a number to a blank text string. Value is the value to be converted. Also, a reference can be used as a parameter. If the referenced cell includes a number or a formula containing a numerical result, the result will be an empty string.
TEXT(number; format)	Converts a number into text according to a given format. Number is the numerical value to be converted. Format is the text which defines the format. Use decimal and thousands separators according to the language set in the cell format.
TRIM(text)	Removes spaces that are in front of a string, or aligns cell contents to the left. Text is the text in which leading spaces are removed, or the cell in which the contents will be left-aligned.
UPPER(text)	Converts the string specified in the text parameter to uppercase. Text is the lower case letters you want to convert to upper case.
VALUE(text)	Converts a text string into a number. Text is the text to be converted to a number.

Add-in functions

Table 12: Add-in functions

Syntax	Description
BESSELI(x; n)	Calculates the modified Bessel function $I_n(x)$. x is the value on which the function will be calculated. n is the order of the Bessel function.
BESSELJ(x; n)	Calculates the Bessel function $J_n(x)$ (cylinder function). x is the value on which the function will be calculated. n is the order of the Bessel function.
BESSELK(x; n)	Calculates the modified Bessel function $K_n(x)$. x is the value on which the function will be calculated. n is the order of the Bessel function.
BESSELY(x; n)	Calculates the modified Bessel function $Y_n(x)$, also known as the Weber or Neumann function. x is the value on which the function will be calculated. n is the order of the Bessel function.
BIN2DEC(number)	Returns the decimal number for the binary number entered. Number is the binary number.
BIN2HEX(number; places)	Returns the hexadecimal number for the binary number entered. Number is the binary number. Places is the number of places to be output.
BIN2OCT(number; places)	Returns the octal number for the binary number entered. Number is the binary number. Places is the number of places to be output.
COMPLEX(real_num; i_num; suffix)	Returns a complex number from a real coefficient and an imaginary coefficient. Real_num is the real coefficient of the complex number. I_num is the imaginary coefficient of the complex number. Suffix is list of options, "i" or "j".
CONVERT_ADD(number; from_unit; to_unit)	Converts a value from one unit of measure to the corresponding value in another unit of measure. Number is the number to be converted. From_unit is the unit from which conversion is taking place. To_unit is the unit to which conversion is taking place.
DEC2BIN(number; places)	Returns the binary number for the decimal number entered between -512 and 511. Number is the decimal number. Places is the number of places to be output.
DEC2HEX(number; places)	Returns the hexadecimal number for the decimal number entered. Number is the decimal number. Places is the number of places to be output.
DEC2OCT(number; places)	Returns the octal number for the decimal number entered. Number is the decimal number. Places is the number of places to be output.
DELTA(number_1; number_2)	Returns TRUE (1) if both numbers are equal, otherwise returns FALSE (0).

Syntax	Description
ERF(lower_limit; upper_limit)	Returns values of the Gaussian error integral. Lower_limit is the lower limit of integral. Upper_limit (optional) is the upper limit of the integral. If this value is missing, the calculation takes places between 0 and the lower limit.
ERFC(lower_limit)	Returns complementary values of the Gaussian error integral between x and infinity. Lower limit is the lower limit of integral (x).
FACTDOUBLE(number)	Returns the factorial of the number with increments of 2. If the number is even, the following factorial is calculated: $n*(N-2)*(n-4)*...*4*2$. If the number is uneven, the following factorial is calculated: $n*(N-2)*(n-4)*...*3*1$.
GESTEP(number; step)	Returns 1 if number is greater than or equal to step .
HEX2BIN(number; places)	Returns the binary number for the hexadecimal number entered. Number is the hexadecimal number. Places is the number of places to be output.
HEX2DEC(number)	Returns the decimal number for the hexadecimal number entered. Number is the hexadecimal number.
HEX2OCT(number; places)	Returns the octal number for the hexadecimal number entered. Number is the hexadecimal number. Places is the number of places to be output.
IMABS(complex_number)	Returns the absolute value (modulus) of a complex_number . The complex number is entered in the form "x + yi" or "x + yj"
IMAGINARY(complex_number)	Returns the imaginary coefficient of a complex_number . The complex number is entered in the form "x + yi" or "x + yj"
IMARGUMENT(complex_number)	Returns the argument (the phi angle) of a complex_number . The complex number is entered in the form "x + yi" or "x + yj"
IMCONJUGATE(complex_number)	Returns the conjugated complex complement to a complex_number . The complex number is entered in the form "x + yi" or "x + yj"
IMCOS(complex_number)	Returns the cosine of a complex_number . The complex number is entered in the form "x + yi" or "x + yj"
IMDIV(numerator; denominator)	Returns the division of two complex numbers. Numerator , Denominator are entered in the form "x + yi" or "x + yj"
IMEXP(complex_number)	Returns the power of e (the Eulerian number) and the complex number. The complex_number is entered in the form "x + yi" or "x + yj"
IMLN(complex_number)	Returns the natural logarithm of a complex_number . The complex_number is entered in the form "x + yi" or "x + yj"
IMLOG10(complex_number)	Returns the common logarithm of a complex_number . The complex number is entered in the form "x + yi" or "x + yj"

Syntax	Description
IMLOG2(complex_number)	Returns the binary logarithm of a complex_number . The complex number is entered in the form "x + yi" or "x + yj"
IMPOWER(complex_number; number)	Returns the integer power of a complex_number . The complex number is entered in the form "x + yi" or "x + yj". Number is the exponent.
IMPRODUCT(complex_number; complex_number_1; ...)	Returns the product of up to 29 complex_numbers . The complex numbers are entered in the form "x + yi" or "x + yj"
IMREAL(complex_number)	Returns the real coefficient of a complex_number . The complex number is entered in the form "x + yi" or "x + yj"
IMSIN(complex_number)	Returns the sine of a complex_number . The complex number is entered in the form "x + yi" or "x + yj"
IMSQRT(complex_number)	Returns the square root of a complex_number . The complex numbers are entered in the form "x + yi" or "x + yj"
IMSUB(complex_number_1; complex_number_2)	Returns the subtraction of two complex_numbers . The complex_numbers are entered in the form "x + yi" or "x + yj"
IMSUM(complex_number; complex_number_1; ...)	Returns the sum of up to 29 complex numbers. The complex_numbers are entered in the form "x + yi" or "x + yj"
OCT2BIN(number; places)	Returns the binary number for the octal number entered. Number is the octal number. Places is the number of places to be output.
OCT2DEC(number)	Returns the decimal number for the octal number entered. Number is the octal number.
OCT2HEX(number; places)	Returns the hexadecimal number for the octal number entered. Number is the octal number. Places is the number of places to be output.