RAGTIME® 5

RagTime Formulas and Functions

RagTime GmbH

This manual was written by Jan Henning and Jens F. Adam and translated into English by Stephen Lindenmayer.

The layout was designed by H. Erich Fraas and Jens F. Adam using Meta fonts (Erik Spiekermann, FontShop). Formulas, special characters and keycap symbols are set in Zapf Dingbats (Hermann Zapf, International Typeface Corporation), Prestige 12 Pitch (Bitstream Inc), Zeal (The Font Bureau) and Hilden 95 (Jens F. Adam, RagTime GmbH).

Version: 5.6.1 Solo EUS (20th December 2001)

Internet: <http://www.ragtime-online.com> (English)
Internet: <http://www.ragtime.de> (German)
E-Mail: <mailto://info@ragtime.de>

CHAPTER 1

Overview

1.1 WELCOME - FORMULAS AND FUNCTIONS

The RagTime 5 documentation consists of several parts on various media:

- **About RagTime** (printed, PDF document)
  Installation, configuration and introduction to the ideas and concepts of RagTime 5.
  Please first install RagTime 5 on your computer and then begin reading the documentation. It always makes sense to try out what you have just read.
- **Training Manual** (printed, PDF document)
  8 exercices in 3 steps for the RagTime 5 novice.
- **RagTime 5 Reference** (printed, on screen, PDF document)
  All about RagTime documents and their components, detailed descriptions of all windows, palettes, commands and dialog boxes.
- **RagTime Formulas and Functions** (on screen, PDF document)
  Descriptions and examples of calculation functions and operators.
- **RagTime 5 for RagTime 3 Users** (PDF document, on screen)
  Differences between RagTime 3 and RagTime 5

The on-screen documentation is installed with the program. It contains detailed descriptions of commands, features, functions etc. and is best used to look up information. All topics are interconnected by hyperlinks and a number of search facilities are available (index, list of commands, list of functions, full text search.)

Except for the Training Manual, which is available in two versions, the RagTime 5 documentation is suited for Microsoft Windows and Mac OS users alike. The few differences that do exist are flagged with the afore-used symbols. In this section of the RagTime 5 on-line documentation, you will find information about formulas and functions, as well as the parts they comprise.

1.2 TABLE OF CONTENTS

CHAPTER 2

Formulas

2.1 ABOUT FORMULAS

A formula is a rule for calculating a result. The result appears in the place where the formula is inserted, for example, in a spreadsheet cell a graph or a text.

You will usually use the ▸ formula palette [RagTime 5 Reference] to ▸ enter a formula [p. 12]. You can also enter formulas in some dialog boxes. Appropriate places are marked with an abacus symbol.

Formulas consist of operands, which are the values used in the calculation, and ▸ operators [p. 13], which determine how the operators are used.

There are various types of operands. In RagTime 5 ▸ numbers, ▸ dates (points in time), ▸ times, ▸ logical values and ▸ text are available. An operand can be declared in different forms:

- as a constant (Frequently used ▸ constants [p. 27] are available as functions.)
- as a ▸ reference [RagTime 5 Reference] to a spreadsheet cell with appropriate contents
- as the result of a ▸ function [p. 18]

See also ▸ Operators [p. 13]

2.2 ENTERING FORMULAS

You can use the results of formulas in many places in your documents, for example, in spreadsheet cells, texts or graphs.

☞ **To enter a formula**

1. Select the place at which the result of the formula is to appear.
2. Open, if necessary, the ▸ formula palette [RagTime 5 Reference].
3. Click in the entry field of the formula palette and enter the formula.
4. Complete the formula by pressing the ⏎ key.

The result of the formula appears.

⇢ If a spreadsheet cell is selected, you can begin entering a formula by typing "=".

⇢ If a spreadsheet cell with a formula is active, the formula is displayed in the formula palette.

⇢ A formula in text is displayed in the formula palette when exactly the result is selected. You can ▸ show formula borders in text [RagTime 5 Reference] on the screen to aid in selecting the result.

See also ▸ Operators [p. 13]

2.3 OPERATORS

Operators determine how operands are used in a formula. Operators can only be used together with one or two compatible ▸ operands. When performing a calculation, RagTime attempts to convert the value types of operands if necessary and possible. If this is not possible, the formula returns an ▸ error value.

The following operators are available:

Operators Preceding One Operand

+	as a positive sign before a number or a time span
-	as a negative sign before a number or a time span
NOT	converts a logical value into its opposite

Operators Following One Operand

%	divides a number by 100(for example, `5% = 0.05`) If the % follows another operand and a + or a -, the appropriate part of the first operand is calculated and added or subtracted from the first operand. (for example, `50 + 15% = 57.5` or `50 - 15% = 42.5`)

Arithmetic Operators between Two Operands

+	adds two numbers, two time spans, or a time span to a date
-	subtracts two numbers, two time spans, or a time span from a date
*	multiples two numbers or a number and a time span
/ or ÷	divides two numbers or a time span by a number
MOD (modulus)	calculates the whole number remainder from dividing two numbers
** or ^	raises a number by the power of the second number

Boolian Operators between Two Operands

AND	returns TRUE, if two truth values are TRUE
OR	returns TRUE, if at least one of two truth values is TRUE

Comparison Operators between Two Operands

=	tests two values for equality
<	tests if the first value is less than the second
≤ or <=	tests if the first value is less than or equal to the second
>	tests if the first value is greater than the second
≥ or >=	tests if the first value is greater than or equal to the second
≠ or <>	tests two values for inequality

Text Operators between Two Operands

&	concatinates two texts
&&	concatinates two texts, inserting a blank space between them

See also ▸ Precedence of Operators [p. 14]
▸ Comparing Values [p. 14]

2.4 PRECEDENCE OF OPERATORS

When several operators occur in a formula, they are evaluated in a predetermined sequence. The following list orders the operators from highest to lowest precedence. Operators on the same line have the same precedence and RagTime evaluates them in the order in which they appear from left to right.

+ – (Prefixes) %
NOT
**
* / MOD AND
+ – OR % used with + or –
< ≤ > ≥ ≠

To force evaluation in another order, use parentheses in the appropriate places. Expressions within parentheses are evaluated first.

2.5 COMPARING VALUES

When comparing two values with operators or when sorting, RagTime first checks the value type and evaluates them in the following order: number ‹ time span ‹ date ‹ text

Within one value type, the actual values are compared.

Texts are compared character by character, whereby a character is considered greater than those preceding it. Upper case orthography is considered greater than the lower case equivalent if there are no other differences. Characters with diacritial marks are considered greater than those without if there are no other differences.

2.6 ERROR VALUES

If an error occurs during the calculation of a formula, the result is an error value. Error values propagate themselves; an error value is the result of any formula in which error values occur as operators or arguments.

Error values are always written in upper case and are followed by an exclamation point.

ERROR! An error which does not fit in any other category.

NULL! This error is never caused by RagTime and can only occur in spreadsheets imported from Excel. The error is supported to provide compatibility with Excel.

DIV/0!	Division by zero was attempted.
VALUE!	An operator or argument has the wrong type, for example, a date instead of a number.
REF!	A formula uses a reference to a nonexistant object. Generally, the referenced object has been deleted after the formula was entered.
NAME!	This error is never caused by RagTime and can occur in spreadsheets imported from Excel. The error is supported to provide compatibility with Excel. RagTime Connect returns this error value for database fields containing no data.
NUM!	An undefined operation was attempted, or the result is outside the range of numbers possible in RagTime.
NA!	A value is not available because, for example, it could not be found in a spreadsheet, or an operand or argument has the constant value “NA”.
RANGE!	A value is outside the permitted interval.
DATE!	A date is outside the supported range, or a part of a date has an invalid value, for example 13 as month.
CIRC!	A formula contains a reference to itself or is in a chain of formulas which refers to itself at the “end”. This error value does not appear when iteration is turned on.
EVAL!	The formula uses the result of a calculation which returned the error value CIRC!.
ILLEGAL!	The formula uses a function which is not installed.
COMPLEX!	The formula is too complex; it contains too many embedded parentheses or calls to subfunctions.

CHAPTER 3

Function Overview

3.1 ABOUT FUNCTIONS

A function describes a particular group of actions which are performed with arguments which are passed to the function. A function has a name and returns its result as a value. The result may be further used as an operand or may be the result of the formula.

Some functions do not require arguments because they return a constant, for example, π, or request, for example, the computer's date or time.

When entering functions in ▸ formulas [p. 12], you must know the exact spelling as well as the number, type and meaning of the function's arguments. You may find it convenient paste a function from the ▸ Functions [RagTime 5 Reference] dialog box.

3.2 FUNCTION LIST

A

- ▸ **Abs** [p. 40] – Returns the absolute value of a number.
- ▸ **AddDay** [p. 41] – Adds days to a given date.
- ▸ **AddHour** [p. 42] – Adds hours to a given date.
- ▸ **AddMinute** [p. 43] – Adds minutes to a given date.
- ▸ **AddMonth** [p. 44] – Adds months to a given date.
- ▸ **AddSecond** [p. 45] – Adds seconds to a given date.
- ▸ **ClAddWorkDaysUSA** [p. 58] – Adds up to 1000 workdays in USA to a date.
- ▸ **AddYear** [p. 46] – Adds years to a given date.
- ▸ **And** [p. 47] – Returns true if all values in a list are true.
- ▸ **Annuity** [p. 48] – Calculates the Annuity for a given interest rate and number of periods.
- ▸ **ArcCos** [p. 49] – Calculates the arc cosine of a number.
- ▸ **ArcSin** [p. 50] – Calculates the arc sine of a number.
- ▸ **ArcTan** [p. 51] – Calculates the arc tangent of a number.
- ▸ **Average** [p. 52] – Calculates the average of numbers in a list.

B

- ▸ **Button** [p. 53] – Installs a button into a spreadsheet cell, performs an action when clicked, and returns it's title.

C

- ▸ **Ceiling** [p. 54] – Rounds a number to the next higher whole number or the next higher multiple.
- ▸ **Char** [p. 55] – Returns a character with a given character code and encoding.

- **Choose** [p. 57] – Choose any of several values or ranges.
- **ClAddWorkDaysUSA** [p. 58] – Adds up to 1000 workdays in USA to a date.
- **ClDateToNumber** [p. 59] – OBSOLETE FUNCTION: Please use ▸ Number (Date) [p. 154].
- **ClDayInfoUSA** [p. 60] – Returns, if available, information about a date (holidays in USA and other special days).
- **ClDayOfYear** [p. 61] – OBSOLETE FUNCTION: Please use ▸ DayOfYear (Date) [p. 89].
- **ClDiffWorkDaysUSA** [p. 62] – Calculates the number of workdays in USA between two dates - up a maximum of 1000 workdays.
- **Clean** [p. 63] – Removes non-printable characters from a text.
- **ClEasterSunday** [p. 64] – Calculates the date of Easter Sunday for a year.
- **ClFirstAdvent** [p. 65] – Calculates the date of the first Advent Sunday in a year.
- **ClJulianDate** [p. 66] – Converts a Gregorian date to a Julian date (Number of days since noon January 1, 4713 BC).
- **ClModJulian** [p. 67] – Converts a Gregorian date to a modified Julian date (Number of days since 00:00 November 17, 1858).
- **ClNumberToDate** [p. 68] – OBSOLETE FUNCTION: Please use ▸ Date (NumberOfDays) [p. 85].
- **ClWorkDaysInMonthUSA** [p. 69] – Calculates the number of workdays in a month in USA.
- **ClWorkDaysInYearUSA** [p. 70] – Calculates the number of workdays in a year in USA.
- **ClWorkDayUSA** [p. 71] – Tests whether a date is workday throughout the USA or a legal holiday and returns the appropriate logical constant True or False.
- **Code** [p. 72] – Returns the character code of the first character in a text having a given encoding.
- **Column** [p. 73] – Get the column where the formula or the specified cell is.
- **ColumnValue** [p. 74] – Returns the value from that column, to be used in VSearch.
- **Combinations** [p. 75] – Calculates the number of combinations of m elements chosen out of n elements.
- **Compound** [p. 76] – Calculates the interest for a given interest rate and number of periods.
- **Concat** [p. 77] – Concatenates texts.
- **Container** [p. 78] – Returns the name of the innermost container containing the formula.
- **Cos** [p. 79] – Calculates the cosine of an angle given in radians.

- **Count** [p. 80] – Counts numbers in a list.
- **CurrentCell** [p. 81] – Returns the current cell in the Search function.
- **CurrentCount** [p. 82] – Returns the number of values already accepted in the Search function.
- **CurrentIndex** [p. 83] – Returns the current index in the Search function.
- **CurrentResult** [p. 84] – Returns the current result in the Search function.

D

- **Date** [p. 85] – Converts text or a specified number of days from January 1, 1904 to a date.
- **ClDateToNumber** [p. 59] – OBSOLETE FUNCTION: Please use ▸ Number (Date) [p. 154].
- **ClDayInfoUSA** [p. 60] – Returns, if available, information about a date (holidays in USA and other special days).
- **DayOf** [p. 86] – Extracts the day from a date.
- **DayOfWeekISO** [p. 88] – Returns the day of the week, monday = 1 to sunday = 7.
- **DayOfWeek** [p. 87] – Returns the day of the week, sunday = 1 to saturday = 7.
- **DayOfYear** [p. 89] – Returns the day of the year.
- **ClDayOfYear** [p. 61] – OBSOLETE FUNCTION: Please use ▸ DayOfYear (Date) [p. 89].
- **Degrees** [p. 90] – Converts an angle from radians to degrees.
- **DiffDay** [p. 91] – Calculates days between two dates.
- **DiffDays30** [p. 92] – Calculates 30-day months between two dates.
- **DiffHour** [p. 93] – Calculates hours between two dates.
- **DiffMinute** [p. 94] – Calculates minutes between two dates.
- **DiffMonth** [p. 95] – Calculates months between two dates.
- **DiffSecond** [p. 96] – Calculates seconds between two dates.
- **ClDiffWorkDaysUSA** [p. 62] – Calculates the number of workdays in USA between two dates - up a maximum of 1000 workdays.
- **DiffYear** [p. 97] – Calculates years between two dates.
- **DocumentDate** [p. 98] – Returns the document's date which was initialized when the document was created.
- **DocumentName** [p. 99] – Returns the name of the document.

E

- **ClEasterSunday** [p. 64] – Calculates the date of Easter Sunday for a year.
- **EndingPageNumber** [p. 100] – Returns the ending page number of the layout.

- **Error** [p. 101] – Returns an error according to an error code.
- **ErrorType** [p. 102] – Returns a code according to the error type of the value.
- **Exact** [p. 103] – Returns true if two texts match exactly.
- **Exp** [p. 104] – Calculates e raised to a specific power.
- **Expl** [p. 105] – Calculates (e raised to a specific power) - 1.

F

- **FV** [p. 111] – Calculates future value for a given interest rate and a list of payments.
- **Factorial** [p. 106] – Calculates the factorial of a number.
- **False** [p. 107] – Returns the logical constant False.
- **Find** [p. 108] – Returns the position of a specified text within a target text.
- **ClFirstAdvent** [p. 65] – Calculates the date of the first Advent Sunday in a year.
- **Floor** [p. 109] – Rounds a number to the next lower whole number or the next lower multiple.
- **Frac** [p. 110] – Returns the fractional part of a number.

G

H

- **HSearch** [p. 114] – Scans a range by columns for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.
- **Hour** [p. 112] – Returns the current date and time every hour.
- **HourOf** [p. 113] – Extracts hours from a date.

I

- **If** [p. 116] – Returns second or third argument, depending on whether the first argument is true.
- **Index** [p. 117] – Extracts any value from a table.
- **Int** [p. 119] – Rounds a number down to the largest integer.
- **IsBlank** [p. 120] – Returns true if the value is empty or an empty string.
- **IsErr** [p. 121] – Returns true if the value is any error value.
- **IsEven** [p. 122] – Returns True if the number is even.
- **IsNA** [p. 123] – Returns true if the value is the “NA” error value.
- **IsNumber** [p. 124] – Returns true if the value is a number.
- **IsOdd** [p. 125] – Returns True if the number is odd.

J

- `ClJulianDate` [p. 66] – Converts a Gregorian date to a Julian date (Number of days since noon January 1, 4713 BC).

K

L

- `Large` [p. 126] – Returns the k–th largest of the numbers in the range.
- `Left` [p. 127] – Extracts the leftmost n characters of a text.
- `Length` [p. 128] – Returns the length of a text in characters.
- `Ln` [p. 129] – Calculates the natural logarithm of a number.
- `Ln1` [p. 130] – Calculates the natural logarithm of a (number + 1).
- `Log` [p. 131] – Calculates the logarithm of a number to any base.
- `Log10` [p. 132] – Calculates the base 10 logarithm of a number.
- `Log2` [p. 133] – Calculates the base 2 logarithm of a number.
- `LogRegressB` [p. 134] – Calculates the y–intercept of a best-fit exponential curve.
- `LogRegressM` [p. 135] – Calculates the increase of a best-fit exponential curve.
- `LookUp` [p. 136] – Looks for a value in a table and extracts a corresponding value from another table.
- `Lower` [p. 138] – Returns text with all letters changed to lower case.

M

- `MailMerge` [p. 139] – Extracts data from tables for serial printings.
- `Max` [p. 141] – Returns the largest number in a list.
- `Median` [p. 142] – Returns the median of the numbers in the list.
- `Mid` [p. 143] – Extracts characters from any position of a text.
- `Min` [p. 144] – Returns the smallest number in a list.
- `Minute` [p. 145] – Returns the current date and time every minute.
- `MinuteOf` [p. 146] – Extracts minutes from a date.
- `Mod` [p. 147] – Calculate x modulus y.
- `ClModJulian` [p. 67] – Converts a Gregorian date to a modified Julian date (Number of days since 00:00 November 17, 1858).
- `MonthOf` [p. 148] – Extracts the month from a date.

N

- `NA` [p. 149] – Returns the error value NA!.
- `NPV` [p. 153] – Calculates the net present value for a given interest rate and a list of future payments.
- `NoOfPages` [p. 150] – Returns the number of pages.
- `Not` [p. 151] – Negates a logical value.
- `Now` [p. 152] – Returns the current date and time.

- **Number** [p. 154] – Converts text to a number or a date into the number of days from January 1, 1904.
- **ClNumberToDate** [p. 68] – OBSOLETE FUNCTION: Please use ▸ Date (NumberOfDays) [p. 85].

O

- **Or** [p. 155] – Returns true if any value in a list is true.

P

- **Page** [p. 156] – Returns the number of the page containing the formula.
- **PageIndex** [p. 157] – Returns the index of the page containing the formula.
- **Percentile** [p. 158] – Returns any percentile of the numbers in the range.
- **Permutations** [p. 159] – Calculates the number of permutations of m elements chosen out of n elements.
- **Pi** [p. 160] – Returns the numerical constant π.
- **Pi180** [p. 161] – Returns the numerical constant π/180.
- **Plane** [p. 162] – Get the plane where the formula or the specified cell is.
- **PrintCycle** [p. 163] – Returns the sequential number of serial printings.
- **PrintStop** [p. 164] – Sets a condition to stop serial printing.
- **Proper** [p. 165] – Returns text with all initials changed to upper case.

Q

- **Quartile** [p. 166] – Returns any quartile of the numbers in the range (quartile = 1 or 3).

R

- **Radians** [p. 167] – Converts an angle from degrees to radians.
- **Rand** [p. 168] – Calculates a random number with optional limits.
- **RegressionB** [p. 169] – Calculates the y–intercept of a best-fit straight line.
- **RegressionM** [p. 170] – Calculates the slope of a best-fit straight line.
- **Repeat** [p. 171] – Repeats a given text n times.
- **Replace** [p. 172] – Replaces specific characters in a text.
- **Right** [p. 173] – Extracts the rightmost n characters of a text.
- **Round** [p. 174] – Rounds a number to a specific number of places.
- **Row** [p. 175] – Get the row where the formula or the specified cell is.

- **RowValue** [p. 176] – Returns the value from that row, to be used in HSearch.

S

- **Search** [p. 177] – Scans a range for matches of a condition and returns either the count of matches or the StartValue, which is replaced by NextValue in case of matches.
- **Second** [p. 179] – Returns the current date and time every second.
- **SecondOf** [p. 180] – Extracts seconds from a date.
- **Selection** [p. 181] – Looks for the nth line in a table that matches a condition and returns the index of that line.
- **SetCell** [p. 182] – Store a value into a cell, or into some cell in a range.
- **SetDate** [p. 183] – Creates a date from numbers.
- **SetDay** [p. 184] – Sets the day in a date.
- **SetDocName** [p. 185] – Sets the default name of the document for "Save as...".
- **SetHour** [p. 186] – Sets the hours in a date.
- **SetMinute** [p. 187] – Sets the minutes in a date.
- **SetMonth** [p. 188] – Sets the month in a date.
- **SetSecond** [p. 189] – Sets the seconds in a date.
- **SetTime** [p. 190] – Creates a time from numbers. The date part will be set to Jan 1st, 1904.
- **SetTimeSpan** [p. 191] – Creates a time span from numbers.
- **SetYear** [p. 192] – Sets the year in a date.
- **Sign** [p. 193] – Returns the sign of a number.
- **Sin** [p. 194] – Calculates the sine of an angle given in radians.
- **Small** [p. 195] – Returns the k–th smallest of the numbers range.
- **SmartConcat** [p. 196] – Concatenates texts with delimiter.
- **SpecialIf** [p. 197] – Returns second argument or cell value, depending on whether the first argument is true.
- **Sqr** [p. 198] – Calculates the square of a number.
- **SqrSum** [p. 199] – Calculates the sum of squares of all numbers in a list.
- **Sqrt** [p. 200] – Calculates the square root of a number.
- **StDev** [p. 202] – Calculates the standard deviation of numbers in a list.
- **StartingPageNumber** [p. 201] – Returns the starting page number of the layout.
- **Sum** [p. 203] – Calculates the sum of a list of numbers.
- **SumProduct** [p. 204] – Calculates the sum of products of corresponding numbers in ranges.

- **SumX2MY2** [p. 205] – Calculates the sum of sqr (x) - sqr (y), where x, y are taken from two ranges.
- **SumX2PY2** [p. 206] – Calculates the sum of sqr (x) + sqr (y), where x, y are taken from two ranges.
- **SumXMY2** [p. 207] – Calculates the sum of squares of x–y, where x, y are taken from two ranges.
- **SumXPY2** [p. 208] – Calculates the sum of squares of x+y, where x, y are taken from two ranges.
- **SystemCurrency** [p. 209] – Converts a number to text using the currency format defined by the system software.

T

- **Tan** [p. 210] – Calculates the tangent of an angle given in radians.
- **Text** [p. 211] – Converts a value to text.
- **TimeSpan** [p. 213] – Converts text to a time span.
- **Today** [p. 214] – Returns the current date.
- **Trim** [p. 215] – Removes unnecessary space characters from a text.
- **True** [p. 216] – Returns the logical constant True.
- **Trunc** [p. 217] – Truncates a number to any number of digits.
- **Type** [p. 218] – Returns a code according to the type of the value.

U

- **UniChar** [p. 219] – OBSOLETE FUNCTION: Please use Char (CharacterCode; –32768) [p. 55]
- **UniCode** [p. 220] – OBSOLETE FUNCTION: Please use Code (Text; –32768) [p. 72]
- **Upper** [p. 221] – Returns text with all letters changed to upper case.

V

- **VSearch** [p. 225] – Scans a range by rows for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.
- **Value** [p. 222] – Converts text to a value, as if entered into a spreadsheet cell.
- **ValueFormat** [p. 223] – Returns the definition of a value format.
- **Var** [p. 224] – Calculates the variance of numbers in a list.

W

- **WeekOfYearISO** [p. 228] – Returns the week of the year, ISO rules.
- **WeekOfYear** [p. 227] – Returns the week of the year, US rules.
- **WinChar** [p. 229] – OBSOLETE FUNCTION: Please use Char (CharacterCode; 512) [p. 55]

- WinCode [p. 230] – OBSOLETE FUNCTION: Please use Code (Text; 512) [p. 72]
- ClWorkDaysInMonthUSA [p. 69] – Calculates the number of workdays in a month in USA.
- ClWorkDaysInYearUSA [p. 70] – Calculates the number of workdays in a year in USA.
- ClWorkDayUSA [p. 71] – Tests whether a date is workday throughout the USA or a legal holiday and returns the appropriate logical constant True or False.

X

Y

Z

- YearOf [p. 231] – Extracts the year from a date.

3.3 ARITHMETIC FUNCTIONS

Arithmetic functions are functions which have numbers for arguments and with which you can perform numerical calculations.

- Abs [p. 40] – Returns the absolute value of a number.
- Ceiling [p. 54] – Rounds a number to the next higher whole number or the next higher multiple.
- Exp [p. 104] – Calculates e raised to a specific power.
- Expl [p. 105] – Calculates (e raised to a specific power) - 1.
- Factorial [p. 106] – Calculates the factorial of a number.
- Floor [p. 109] – Rounds a number to the next lower whole number or the next lower multiple.
- Frac [p. 110] – Returns the fractional part of a number.
- Int [p. 119] – Rounds a number down to the largest integer.
- IsEven [p. 122] – Returns True if the number is even.
- IsNumber [p. 124] – Returns true if the value is a number.
- IsOdd [p. 125] – Returns True if the number is odd.
- Ln [p. 129] – Calculates the natural logarithm of a number.
- Lnl [p. 130] – Calculates the natural logarithm of a (number + 1).
- Log [p. 131] – Calculates the logarithm of a number to any base.
- Log10 [p. 132] – Calculates the base 10 logarithm of a number.
- Log2 [p. 133] – Calculates the base 2 logarithm of a number.
- Mod [p. 147] – Calculate x modulus y.
- Number [p. 154] – Converts text to a number or a date into the number of days from January 1, 1904.
- Rand [p. 168] – Calculates a random number with optional limits.
- Round [p. 174] – Rounds a number to a specific number of places.

- **Sign** [p. 193] – Returns the sign of a number.
- **Sqr** [p. 198] – Calculates the square of a number.
- **SqrSum** [p. 199] – Calculates the sum of squares of all numbers in a list.
- **Sqrt** [p. 200] – Calculates the square root of a number.
- **Sum** [p. 203] – Calculates the sum of a list of numbers.
- **SumProduct** [p. 204] – Calculates the sum of products of corresponding numbers in ranges.
- **SumX2MY2** [p. 205] – Calculates the sum of sqr (x) - sqr (y), where x, y are taken from two ranges.
- **SumX2PY2** [p. 206] – Calculates the sum of sqr (x) + sqr (y), where x, y are taken from two ranges.
- **SumXMY2** [p. 207] – Calculates the sum of squares of x–y, where x, y are taken from two ranges.
- **SumXPY2** [p. 208] – Calculates the sum of squares of x+y, where x, y are taken from two ranges.
- **Trunc** [p. 217] – Truncates a number to any number of digits.
- **Value** [p. 222] – Converts text to a value, as if entered into a spreadsheet cell.

3.4 CONSTANTS

Constants are functions which always return the same result.

- **False** [p. 107] – Returns the logical constant False.
- **NA** [p. 149] – Returns the error value NA!.
- **Pi** [p. 160] – Returns the numerical constant π.
- **Pi180** [p. 161] – Returns the numerical constant π/180.
- **True** [p. 216] – Returns the logical constant True.

3.5 CONVERSION FUNCTIONS

Conversion functions convert their arguments into other units of measure or into another code system.

- **Char** [p. 55] – Returns a character with a given character code and encoding.
- **Code** [p. 72] – Returns the character code of the first character in a text having a given encoding.
- **Date** [p. 85] – Converts text or a specified number of days from January 1, 1904 to a date.
- **ClDateToNumber** [p. 59] – OBSOLETE FUNCTION: Please use ▸ Number (Date) [p. 154].

- **Degrees** [p. 90] – Converts an angle from radians to degrees.
- **ClDiffWorkDaysUSA** [p. 62] – Calculates the number of workdays in USA between two dates - up a maximum of 1000 workdays.
- **Error** [p. 101] – Returns an error according to an error code.
- **ClJulianDate** [p. 66] – Converts a Gregorian date to a Julian date (Number of days since noon January 1, 4713 BC).
- **ClModJulian** [p. 67] – Converts a Gregorian date to a modified Julian date (Number of days since 00:00 November 17, 1858).
- **Number** [p. 154] – Converts text to a number or a date into the number of days from January 1, 1904.
- **ClNumberToDate** [p. 68] – OBSOLETE FUNCTION: Please use ▸ Date (NumberOfDays) [p. 85].
- **Radians** [p. 167] – Converts an angle from degrees to radians.
- **Text** [p. 211] – Converts a value to text.
- **TimeSpan** [p. 213] – Converts text to a time span.
- **UniChar** [p. 219] – OBSOLETE FUNCTION: Please use ▸ Char (CharacterCode; –32768) [p. 55]
- **UniCode** [p. 220] – OBSOLETE FUNCTION: Please use ▸ Code (Text; –32768) [p. 72]
- **Value** [p. 222] – Converts text to a value, as if entered into a spreadsheet cell.
- **WinChar** [p. 229] – OBSOLETE FUNCTION: Please use ▸ Char (CharacterCode; 512) [p. 55]
- **WinCode** [p. 230] – OBSOLETE FUNCTION: Please use ▸ Code (Text; 512) [p. 72]

3.6 DATE FUNCTIONS

Date functions enable you to calculate with and convert dates.

- **AddDay** [p. 41] – Adds days to a given date.
- **AddHour** [p. 42] – Adds hours to a given date.
- **AddMinute** [p. 43] – Adds minutes to a given date.
- **AddMonth** [p. 44] – Adds months to a given date.
- **AddSecond** [p. 45] – Adds seconds to a given date.
- **ClAddWorkDaysUSA** [p. 58] – Adds up to 1000 workdays in USA to a date.
- **AddYear** [p. 46] – Adds years to a given date.
- **Date** [p. 85] – Converts text or a specified number of days from January 1, 1904 to a date.
- **ClDateToNumber** [p. 59] – OBSOLETE FUNCTION: Please use ▸ Number (Date) [p. 154].

- ClDayInfoUSA [p. 60] – Returns, if available, information about a date (holidays in USA and other special days).
- DayOf [p. 86] – Extracts the day from a date.
- DayOfWeekISO [p. 88] – Returns the day of the week, monday = 1 to sunday = 7.
- DayOfWeek [p. 87] – Returns the day of the week, sunday = 1 to saturday = 7.
- DayOfYear [p. 89] – Returns the day of the year.
- ClDayOfYear [p. 61] – OBSOLETE FUNCTION: Please use ▸ DayOfYear (Date) [p. 89].
- DiffDay [p. 91] – Calculates days between two dates.
- DiffDays30 [p. 92] – Calculates 30-day months between two dates.
- DiffHour [p. 93] – Calculates hours between two dates.
- DiffMinute [p. 94] – Calculates minutes between two dates.
- DiffMonth [p. 95] – Calculates months between two dates.
- DiffSecond [p. 96] – Calculates seconds between two dates.
- ClDiffWorkDaysUSA [p. 62] – Calculates the number of workdays in USA between two dates - up a maximum of 1000 workdays.
- DiffYear [p. 97] – Calculates years between two dates.
- DocumentDate [p. 98] – Returns the document's date which was initialized when the document was created.
- ClEasterSunday [p. 64] – Calculates the date of Easter Sunday for a year.
- ClFirstAdvent [p. 65] – Calculates the date of the first Advent Sunday in a year.
- Hour [p. 112] – Returns the current date and time every hour.
- HourOf [p. 113] – Extracts hours from a date.
- ClJulianDate [p. 66] – Converts a Gregorian date to a Julian date (Number of days since noon January 1, 4713 BC).
- Minute [p. 145] – Returns the current date and time every minute.
- MinuteOf [p. 146] – Extracts minutes from a date.
- ClModJulian [p. 67] – Converts a Gregorian date to a modified Julian date (Number of days since 00:00 November 17, 1858).
- MonthOf [p. 148] – Extracts the month from a date.
- Now [p. 152] – Returns the current date and time.
- ClNumberToDate [p. 68] – OBSOLETE FUNCTION: Please use ▸ Date (NumberOfDays) [p. 85].
- Second [p. 179] – Returns the current date and time every second.
- SecondOf [p. 180] – Extracts seconds from a date.
- SetDate [p. 183] – Creates a date from numbers.
- SetDay [p. 184] – Sets the day in a date.
- SetHour [p. 186] – Sets the hours in a date.

- SetMinute [p. 187] – Sets the minutes in a date.
- SetMonth [p. 188] – Sets the month in a date.
- SetSecond [p. 189] – Sets the seconds in a date.
- SetTime [p. 190] – Creates a time from numbers. The date part will be set to Jan 1st, 1904.
- SetTimeSpan [p. 191] – Creates a time span from numbers.
- SetYear [p. 192] – Sets the year in a date.
- TimeSpan [p. 213] – Converts text to a time span.
- Today [p. 214] – Returns the current date.
- WeekOfYearISO [p. 228] – Returns the week of the year, ISO rules.
- WeekOfYear [p. 227] – Returns the week of the year, US rules.
- ClWorkDayUSA [p. 71] – Tests whether a date is workday throughout the USA or a legal holiday and returns the appropriate logical constant True or False.
- ClWorkDaysInYearUSA [p. 70] – Calculates the number of workdays in a year in USA.
- ClWorkDaysInMonthUSA [p. 69] – Calculates the number of workdays in a month in USA.
- YearOf [p. 231] – Extracts the year from a date.

3.7 CALENDAR FUNCTIONS

With the calendar functions you can calculate special dates, request information regarding a day and perform calculations with workdays and holidays.

The identification and definitions of of the holidays and workdays required in some of the calendar functions are specific to the respective country. These functions are identified by an abbreviation for the country appended to the end of the function name (e.g. “USA” for the United States of America).

These functions form a group of external functions, which are installed as ▸ Add Ons [RagTime 5 Reference]. If one of these functions is missing in your installation, you can install the “Calender (Country)” group. For more details, see the “About RagTime” manual.

- ClAddWorkDaysUSA [p. 58] – Adds up to 1000 workdays in USA to a date.
- ClJulianDate [p. 66] – Converts a Gregorian date to a Julian date (Number of days since noon January 1, 4713 BC).
- ClModJulian [p. 67] – Converts a Gregorian date to a modified Julian date (Number of days since 00:00 November 17, 1858).
- ClDateToNumber [p. 59] – OBSOLETE FUNCTION: Please use ▸ Number (Date) [p. 154].

- ClDiffWorkDaysUSA [p. 62] – Calculates the number of workdays in USA between two dates - up a maximum of 1000 workdays.
- ClFirstAdvent [p. 65] – Calculates the date of the first Advent Sunday in a year.
- ClEasterSunday [p. 64] – Calculates the date of Easter Sunday for a year.
- ClDayInfoUSA [p. 60] – Returns, if available, information about a date (holidays in USA and other special days).
- ClDayOfYear [p. 61] – OBSOLETE FUNCTION: Please use ▸ DayOfYear (Date) [p. 89].
- ClWorkDayUSA [p. 71] – Tests whether a date is workday throughout the USA or a legal holiday and returns the appropriate logical constant True or False.
- ClWorkDaysInYearUSA [p. 70] – Calculates the number of workdays in a year in USA.
- ClWorkDaysInMonthUSA [p. 69] – Calculates the number of workdays in a month in USA.
- ClNumberToDate [p. 68] – OBSOLETE FUNCTION: Please use ▸ Date (NumberOfDays) [p. 85].

3.8 FINANCIAL FUNCTIONS

Financial functions perform various calculations which are of particular interest in finance.

- Annuity [p. 48] – Calculates the Annuity for a given interest rate and number of periods.
- Compound [p. 76] – Calculates the interest for a given interest rate and number of periods.
- FV [p. 111] – Calculates future value for a given interest rate and a list of payments.
- NPV [p. 153] – Calculates the net present value for a given interest rate and a list of future payments.
- SystemCurrency [p. 209] – Converts a number to text using the currency format defined by the system software.

3.9 OPERATORS

▸ Operators [p. 13] join two arguments in a single result. In RagTime, you can use operators, for example, "+" and "-", directly in a formula. To achieve compatibility with the files of other spreadsheet programs, the following operators are also available as functions:

- **And** [p. 47] – Returns true if all values in a list are true.
- **Concat** [p. 77] – Concatenates texts.
- **Mod** [p. 147] – Calculate x modulus y.
- **Not** [p. 151] – Negates a logical value.
- **Or** [p. 155] – Returns true if any value in a list is true.
- **SmartConcat** [p. 196] – Concatenates texts with delimiter.

3.10 MISCELLANEOUS FUNCTIONS

These functions perform various tasks. With them, you can determine positions in a document, determine or suggest the name of a document, define buttons, specify formats, and format the cell contents dependent on conditions.

- **Button** [p. 53] – Installs a button into a spreadsheet cell, performs an action when clicked, and returns it's title.
- **Container** [p. 78] – Returns the name of the innermost container containing the formula.
- **DocumentDate** [p. 98] – Returns the document's date which was initialized when the document was created.
- **DocumentName** [p. 99] – Returns the name of the document.
- **EndingPageNumber** [p. 100] – Returns the ending page number of the layout.
- **If** [p. 116] – Returns second or third argument, depending on whether the first argument is true.
- **NoOfPages** [p. 150] – Returns the number of pages.
- **Page** [p. 156] – Returns the number of the page containing the formula.
- **PageIndex** [p. 157] – Returns the index of the page containing the formula.
- **SetDocName** [p. 185] – Sets the default name of the document for "Save as...".
- **SpecialIf** [p. 197] – Returns second argument or cell value, depending on whether the first argument is true.
- **StartingPageNumber** [p. 201] – Returns the starting page number of the layout.
- **ValueFormat** [p. 223] – Returns the definition of a value format.

3.11 PRINT FUNCTIONS

You can control serial printing with print functions. They initialize serial printing, set the stop conditions, and make it possible for formulas to

refer to the serial number of the current printing and individualize each printed copy.

- `MailMerge` [p. 139] – Extracts data from tables for serial printings.
- `PrintCycle` [p. 163] – Returns the sequential number of serial printings.
- `PrintStop` [p. 164] – Sets a condition to stop serial printing.

3.12 ROUNDING FUNCTIONS

Rounding functions round numbers according to various procedures and, in some cases, with definable precision.

- `Ceiling` [p. 54] – Rounds a number to the next higher whole number or the next higher multiple.
- `Floor` [p. 109] – Rounds a number to the next lower whole number or the next lower multiple.
- `Frac` [p. 110] – Returns the fractional part of a number.
- `Int` [p. 119] – Rounds a number down to the largest integer.
- `Round` [p. 174] – Rounds a number to a specific number of places.
- `Trunc` [p. 217] – Truncates a number to any number of digits.

3.13 SEARCH FUNCTIONS

With the aid of search functions, you can select values from ranges or determine the addresses of data that you are seeking.

- `Choose` [p. 57] – Choose any of several values or ranges.
- `ColumnValue` [p. 74] – Returns the value from that column, to be used in VSearch.
- `CurrentCell` [p. 81] – Returns the current cell in the Search function.
- `CurrentCount` [p. 82] – Returns the number of values already accepted in the Search function.
- `CurrentIndex` [p. 83] – Returns the current index in the Search function.
- `CurrentResult` [p. 84] – Returns the current result in the Search function.
- `HSearch` [p. 114] – Scans a range by columns for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.
- `Index` [p. 117] – Extracts any value from a table.
- `Large` [p. 126] – Returns the k–th largest of the numbers in the range.
- `LookUp` [p. 136] – Looks for a value in a table and extracts a corresponding value from another table.

- MailMerge [p. 139] – Extracts data from tables for serial printings.
- Max [p. 141] – Returns the largest number in a list.
- Min [p. 144] – Returns the smallest number in a list.
- RowValue [p. 176] – Returns the value from that row, to be used in HSearch.
- Search [p. 177] – Scans a range for matches of a condition and returns either the count of matches or the StartValue, which is replaced by NextValue in case of matches.
- Selection [p. 181] – Looks for the nth line in a table that matches a condition and returns the index of that line.
- Small [p. 195] – Returns the k–th smallest of the numbers range.
- VSearch [p. 225] – Scans a range by rows for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.

3.14 SPREADSHEET FUNCTIONS

The spreadsheet functions are useful in working with cells, rows, columns, layers, and ranges.

- Column [p. 73] – Get the column where the formula or the specified cell is.
- ColumnValue [p. 74] – Returns the value from that column, to be used in VSearch.
- CurrentCell [p. 81] – Returns the current cell in the Search function.
- CurrentCount [p. 82] – Returns the number of values already accepted in the Search function.
- CurrentIndex [p. 83] – Returns the current index in the Search function.
- CurrentResult [p. 84] – Returns the current result in the Search function.
- HSearch [p. 114] – Scans a range by columns for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.
- Index [p. 117] – Extracts any value from a table.
- LookUp [p. 136] – Looks for a value in a table and extracts a corresponding value from another table.
- MailMerge [p. 139] – Extracts data from tables for serial printings.
- Plane [p. 162] – Get the plane where the formula or the specified cell is.
- Row [p. 175] – Get the row where the formula or the specified cell is.
- RowValue [p. 176] – Returns the value from that row, to be used in HSearch.

- **Search** [p. 177] – Scans a range for matches of a condition and returns either the count of matches or the StartValue, which is replaced by NextValue in case of matches.
- **Selection** [p. 181] – Looks for the nth line in a table that matches a condition and returns the index of that line.
- **SetCell** [p. 182] – Store a value into a cell, or into some cell in a range.
- **VSearch** [p. 225] – Scans a range by rows for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.

3.15 STATISTICAL FUNCTIONS

Statistical functions perform various analyses of sets of values.

- **Average** [p. 52] – Calculates the average of numbers in a list.
- **Combinations** [p. 75] – Calculates the number of combinations of m elements chosen out of n elements.
- **Count** [p. 80] – Counts numbers in a list.
- **Factorial** [p. 106] – Calculates the factorial of a number.
- **LogRegressB** [p. 134] – Calculates the y–intercept of a best-fit exponential curve.
- **LogRegressM** [p. 135] – Calculates the increase of a best-fit exponential curve.
- **Median** [p. 142] – Returns the median of the numbers in the list.
- **Percentile** [p. 158] – Returns any percentile of the numbers in the range.
- **Permutations** [p. 159] – Calculates the number of permutations of m elements chosen out of n elements.
- **Quartile** [p. 166] – Returns any quartile of the numbers in the range (quartile = 1 or 3).
- **Rand** [p. 168] – Calculates a random number with optional limits.
- **RegressionB** [p. 169] – Calculates the y–intercept of a best-fit straight line.
- **RegressionM** [p. 170] – Calculates the slope of a best-fit straight line.
- **StDev** [p. 202] – Calculates the standard deviation of numbers in a list.
- **Var** [p. 224] – Calculates the variance of numbers in a list.

3.16 STATUS FUNCTIONS

Status functions determine the type of cell content.

- **ErrorType** [p. 102] – Returns a code according to the error type of the value.
- **IsBlank** [p. 120] – Returns true if the value is empty or an empty string.
- **IsErr** [p. 121] – Returns true if the value is any error value.
- **IsNA** [p. 123] – Returns true if the value is the "NA" error value.
- **IsNumber** [p. 124] – Returns true if the value is a number.
- **Type** [p. 218] – Returns a code according to the type of the value.

3.17 TEXT FUNCTIONS

Text functions are useful in working with text as well as converting other values to text.

- **Char** [p. 55] – Returns a character with a given character code and encoding.
- **Clean** [p. 63] – Removes non-printable characters from a text.
- **Code** [p. 72] – Returns the character code of the first character in a text having a given encoding.
- **Concat** [p. 77] – Concatenates texts.
- **Date** [p. 85] – Converts text or a specified number of days from January 1, 1904 to a date.
- **DocumentName** [p. 99] – Returns the name of the document.
- **Exact** [p. 103] – Returns true if two texts match exactly.
- **Find** [p. 108] – Returns the position of a specified text within a target text.
- **Left** [p. 127] – Extracts the leftmost n characters of a text.
- **Length** [p. 128] – Returns the length of a text in characters.
- **Lower** [p. 138] – Returns text with all letters changed to lower case.
- **Mid** [p. 143] – Extracts characters from any position of a text.
- **Number** [p. 154] – Converts text to a number or a date into the number of days from January 1, 1904.
- **Proper** [p. 165] – Returns text with all initials changed to upper case.
- **Repeat** [p. 171] – Repeats a given text n times.
- **Replace** [p. 172] – Replaces specific characters in a text.
- **Right** [p. 173] – Extracts the rightmost n characters of a text.
- **SetDocName** [p. 185] – Sets the default name of the document for "Save as...".
- **SmartConcat** [p. 196] – Concatenates texts with delimiter.
- **Text** [p. 211] – Converts a value to text.
- **TimeSpan** [p. 213] – Converts text to a time span.
- **Trim** [p. 215] – Removes unnecessary space characters from a text.

- **UniChar** [p. 219] – OBSOLETE FUNCTION: Please use ▸ Char (CharacterCode; –32768) [p. 55]
- **UniCode** [p. 220] – OBSOLETE FUNCTION: Please use ▸ Code (Text; –32768) [p. 72]
- **Upper** [p. 221] – Returns text with all letters changed to upper case.
- **WinChar** [p. 229] – OBSOLETE FUNCTION: Please use ▸ Char (CharacterCode; 512) [p. 55]
- **WinCode** [p. 230] – OBSOLETE FUNCTION: Please use ▸ Code (Text; 512) [p. 72]

3.18 TRIGONOMETRIC FUNCTIONS

Trigonometric functions are used for calculations with angles.

- **ArcCos** [p. 49] – Calculates the arc cosine of a number.
- **ArcSin** [p. 50] – Calculates the arc sine of a number.
- **ArcTan** [p. 51] – Calculates the arc tangent of a number.
- **Cos** [p. 79] – Calculates the cosine of an angle given in radians.
- **Sin** [p. 194] – Calculates the sine of an angle given in radians.
- **Tan** [p. 210] – Calculates the tangent of an angle given in radians.

⇢ All trigonometric functions work with radians as a unit of measurement. The following functions are available for convenient conversion:

- **Degrees** [p. 90] – Converts an angle from radians to degrees.
- **Radians** [p. 167] – Converts an angle from degrees to radians.

CHAPTER 4

Function Reference

In this chapter, you will find detailed descriptions of all functions in alphabetical sequence.

4.1 ABS (FUNCTION)

Abs returns the absolute value of a ▸ number. In other words, negative numbers become positive, while positive numbers are not affected by the function.

Syntax

Abs (*Number*)

Number is a ▸ number of which the absolute value is to be returned.

See also
- ▸ Arithmetic Functions [p. 26]
- ▸ **Sign** [p. 193] – Returns the sign of a number.

4.2 ADDDAY (FUNCTION)

AddDay adds days to a date.

Syntax

AddDay (*Date*; *Days*)

Date is a ▸ date.

Days is a ▸ number specifing the number of seconds to be added to Date.

Examples

- If the current time is 12:00 o'clock noon on 1/1/1980,

```
AddDay (Now; 1200) = 4/15/1983 12:00:00
```

See also

- ▸ Date Functions [p. 28]
- ▸ **DiffDay** [p. 91] – Calculates days between two dates.
- ▸ **AddSecond** [p. 45] – Adds seconds to a given date.
- ▸ **AddMinute** [p. 43] – Adds minutes to a given date.
- ▸ **AddHour** [p. 42] – Adds hours to a given date.
- ▸ **AddMonth** [p. 44] – Adds months to a given date.
- ▸ **AddYear** [p. 46] – Adds years to a given date.

4.3 ADDHOUR (FUNCTION)

AddHour adds hours to a date.

Syntax

AddHour (*Date*; *Hours*)

Date is a ▸ date.

Hours is a ▸ number specifing the number of hours to be added to Date.

Examples

- If the current time is 12:00 o'clock noon on 1/1/1980,

```
AddHour (Now; 1200) = 2/20/1980 12:00:00
```

See also

- ▸ Date Functions [p. 28]
- ▸ **DiffHour** [p. 93] – Calculates hours between two dates.
- ▸ **AddSecond** [p. 45] – Adds seconds to a given date.
- ▸ **AddMinute** [p. 43] – Adds minutes to a given date.
- ▸ **AddDay** [p. 41] – Adds days to a given date.
- ▸ **AddMonth** [p. 44] – Adds months to a given date.
- ▸ **AddYear** [p. 46] – Adds years to a given date.

4.4 ADDMINUTE (FUNCTION)

AddMinute adds minutes to a date.

Syntax

AddMinute (*Date*; *Minutes*)

Date is a ▸ date.

Minutes is a ▸ number specifing the number of minutes to be added to Date.

Examples

- If the current time is 12:00 o'clock noon on 1/1/1980,

```
AddMinute (Now; 1200) = 1/2/1980 8:00:00
```

See also

▸ Date Functions [p. 28]
▸ **DiffMinute** [p. 94] – Calculates minutes between two dates.
▸ **AddSecond** [p. 45] – Adds seconds to a given date.
▸ **AddHour** [p. 42] – Adds hours to a given date.
▸ **AddDay** [p. 41] – Adds days to a given date.
▸ **AddMonth** [p. 44] – Adds months to a given date.
▸ **AddYear** [p. 46] – Adds years to a given date.

4.5 ADDMONTH (FUNCTION)

AddMonth adds months to a date.

Syntax

AddMonth (*Date*; *Months*)

Date is a ▸ date.

Months is a ▸ number specifing the number of months to be added to Date.

Examples

- If the current time is 12:00 o'clock noon on 1/1/1980,

```
AddMonth (Now; 1200) = 1/1/2080 12:00:00
```

See also

- ▸ Date Functions [p. 28]
- ▸ **DiffMonth** [p. 95] – Calculates months between two dates.
- ▸ **AddSecond** [p. 45] – Adds seconds to a given date.
- ▸ **AddMinute** [p. 43] – Adds minutes to a given date.
- ▸ **AddHour** [p. 42] – Adds hours to a given date.
- ▸ **AddDay** [p. 41] – Adds days to a given date.
- ▸ **AddYear** [p. 46] – Adds years to a given date.

4.6 ADDSECOND (FUNCTION)

AddSecond adds seconds to a date.

Syntax

AddSecond (*Date*; *Seconds*)

Date is a ▸ date.

Seconds is a ▸ number specifing the number of seconds to be added to Date.

Examples

- If the current time is 12:00 o'clock noon on 1/1/1980,

```
AddSecond (Now; 1200) = 1/1/1980 12:20:00
```

See also

- ▸ Date Functions [p. 28]
- ▸ **DiffSecond** [p. 96] – Calculates seconds between two dates.
- ▸ **AddMinute** [p. 43] – Adds minutes to a given date.
- ▸ **AddHour** [p. 42] – Adds hours to a given date.
- ▸ **AddDay** [p. 41] – Adds days to a given date.
- ▸ **AddMonth** [p. 44] – Adds months to a given date.
- ▸ **AddYear** [p. 46] – Adds years to a given date.

4.7 ADDYEAR (FUNCTION)

AddYear adds years to a date.

Syntax

AddYear (*Date*; *Years*)

Date is a ▸ date

Years is a ▸ number specifing the number of years to be added to Date.

Examples

- If the current time is 12:00 o'clock noon on 1/1/1980,

```
AddYear (Now; 1200) = 1/1/3180 12:00:00
```

See also

- ▸ Date Functions [p. 28]
- ▸ **DiffYear** [p. 97] – Calculates years between two dates.
- ▸ **AddSecond** [p. 45] – Adds seconds to a given date.
- ▸ **AddMinute** [p. 43] – Adds minutes to a given date.
- ▸ **AddHour** [p. 42] – Adds hours to a given date.
- ▸ **AddDay** [p. 41] – Adds days to a given date.
- ▸ **AddMonth** [p. 44] – Adds months to a given date.

4.8 AND (FUNCTION)

And returns true if all values in a list are true.

Syntax

And (*List*)

List is a ▸ list of which the individual elements are evaluated. If every element is true, the result ofthe function is true.

Examples

- `And (1;2;1) = 1`
- `And (1;2;0) = 0`
- `And (0;0;0) = 0`

Remarks

⇢ The ▸ truth value False is represented by the number 0, True by all other numbers.

See also

- ▸ Operators [p. 31]
- ▸ **Or** [p. 155] – Returns true if any value in a list is true.
- ▸ **Not** [p. 151] – Negates a logical value.

4.9 ANNUITY (FUNCTION)

Annuity calculates the annuity for a given interest rate and number of periods.

Syntax

Annuity (*InterestRate*; *Periods*)

InterestRate is a ▸ number specifying the interest rate.

Periods is a ▸ number specifying the number of payment periods (for credit calculations, the term of the loan)

Examples

- `Annuity (0.075; 8) = 5.8573...` (With an 8-year loan at 7.5%, 1/5.8573 of the principle is to be repaid.)

Remarks

⇢ The annuity is calculated with the formula `(1 - (1 + interest) ^-periods) / interest`.

See also ▸ Financial Functions [p. 31]

▸ **Compound** [p. 76] – Calculates the interest for a given interest rate and number of periods.

4.10 ARCCOS (FUNCTION)

ArcCos calculates the arc cosine of a number. The result is a ▸ number in the ▸ interval [0; π].

Syntax

ArcCos (*Number*)

Number is a ▸ number in the ▸ interval [-1; +1] for which the arc cosine is to be calculated.

Remarks

If you need the angle in degrees, use `Degrees (ArcCos (Number))`.

Warnings

⚠ The ▸ error value ▸ NUM! is returned if *Number* is outside the permitted interval.

See also

- ▸ Trigonometric Functions [p. 37]
- ▸ **Cos** [p. 79] – Calculates the cosine of an angle given in radians.
- ▸ **ArcSin** [p. 50] – Calculates the arc sine of a number.
- ▸ **ArcTan** [p. 51] – Calculates the arc tangent of a number.

4.11 ARCSIN (FUNCTION)

ArcSin calculates the arc sine of a number. The result is a ▸ number in the ▸ interval [-π/2; +π/2].

Syntax

ArcSin (*Number*)

Number is a ▸ number in the ▸ interval [-1; +1] for which the arc sine is to be calculated.

Remarks

If you need the angle in degrees, use `Degrees (ArcSin (Number))`.

Warnings

⚠ The ▸ error value ▸ NUM! is returned if *Number* is outside the permitted interval.

See also

- ▸ Trigonometric Functions [p. 37]
- ▸ **Sin** [p. 194] – Calculates the sine of an angle given in radians.
- ▸ **ArcCos** [p. 49] – Calculates the arc cosine of a number.
- ▸ **ArcTan** [p. 51] – Calculates the arc tangent of a number.

4.12 ARCTAN (FUNCTION)

ArcTan calculates the arc tangent of a number. The result is a ▸ number in the ▸ interval [-π/2; +π/2].

Syntax

ArcTan (*Number*)

Number is a ▸ number, for which the arc tangent is to be calculated.

Remarks

If you need the angle in degrees, use `Degrees (ArcTan (Number))`.

See also

- ▸ Trigonometric Functions [p. 37]
- ▸ **Tan** [p. 210] – Calculates the tangent of an angle given in radians.
- ▸ **ArcSin** [p. 50] – Calculates the arc sine of a number.
- ▸ **ArcCos** [p. 49] – Calculates the arc cosine of a number.

4.13 AVERAGE (FUNCTION)

Average calculates the average of numbers in a list.

Syntax

Average (*List*)

List is a ▸ list from which the arithmetic average of its numbers is found.

Examples

- `Average (1; 2; 3) = 2`
- `Average (1; 2; 'three') = 1.5`

Remarks

The arithmetic average is found by adding the numbers in a list and dividing the sum by the count of the numbers: `Average (List) = Sum (List) / Count (List)`

See also

- ▸ Statistical Functions [p. 35]
- ▸ **Count** [p. 80] – Counts numbers in a list.
- ▸ **StDev** [p. 202] – Calculates the standard deviation of numbers in a list.
- ▸ **Var** [p. 224] – Calculates the variance of numbers in a list.

4.14 BUTTON (FUNCTION)

Button installs a single button in a spreadsheet cell. When the button is clicked, a formula is calculated.

Button returns the ▸ text displayed in the button.

Syntax

Button (*ButtonTitle*; *ActionExpression*)

ButtonTitle is a ▸ text which is written in the button.

ActionExpression is a formula which is calculated when the button is clicked. The result of this calculation is lost, so only the function **SetCell** can be used here meaningfully.

Remarks

⇢ **Button** exists to guarantee compatibility with documents from earlier versions of RagTime. You should not use this function in new documents; instead use a button component, which offers substatially more flexibility in functionality and format.

⇢ **Button** can only be used in spreadsheets.

See also

▸ About Buttons [RagTime 5 Reference]

▸ Miscellaneous Functions [p. 32]

▸ **SetCell** [p. 182] – Store a value into a cell, or into some cell in a range.

4.15 CEILING (FUNCTION)

Ceiling rounds a number to the next higher whole number or the next higher multiple of an argument.

Syntax

Ceiling (*Number*; *Multiple*)

Number is a ▸ number which is to be rounded.

Multiple is an optional argument. *Number* is rounded to the next greater multiple of *Multiple*. If *Multiple* is not given, RagTime rounds up to the next greater whole number.

Examples

- `Ceiling (4.9) = 5`
- `Ceiling (-4.5) = -4`
- `Ceiling (28; 5) = 30`
- `Ceiling (-28; 5) = -25`
- `Ceiling (28; -5) = 25`

Warnings

⚠ If *Multiple* is a negative ▸ number, **Ceiling** instead rounds down to the next smaller multiple. (See the fifth example, above.)

See also

- ▸ Arithmetic Functions [p. 26]
- ▸ Rounding Functions [p. 33]
- ▸ **Int** [p. 119] – Rounds a number down to the largest integer.
- ▸ **Round** [p. 174] – Rounds a number to a specific number of places.
- ▸ **Trunc** [p. 217] – Truncates a number to any number of digits.
- ▸ **Frac** [p. 110] – Returns the fractional part of a number.
- ▸ **Floor** [p. 109] – Rounds a number to the next lower whole number or the next lower multiple.

4.16 CHAR (FUNCTION)

Char returns a character with a given character code in the specified ▸ Encoding [p. 236].

Syntax

Char (*CharacterCode*; *Encoding*)

CharacterCode is a ▸ number from 0 to 255 or, if *Encoding* is negative, from 0 to 65534 for which the corresponding character is to be returned.

Encoding is a ▸ number in the ▸ interval [-32768; 32767], which specifies how *Text* is ▸ endcoded [p. 236]. If *Encoding* is not given, the standard encoding of the operating system of the given computer is used.

Examples

- `Char (65) = A`
- `Char (97) = a`
- `Char (49) = 1`
- `Char (65; 256)` = A (Mac OS Standard Roman)
- `Char (65; 512)` = A (Microsoft Windows Standard Roman)
- `Char (154; 256)` = ö (Mac OS Standard Roman)
- `Char (246; 512)` = ö (Microsoft Windows Standard Roman)
- `Char (148; 1024)` = ö (Microsoft DOS Code Page 437)
- `Char (246; -32768)` = ö (Unicode)
- `Char (166; 256)` = ¶ (Mac OS Standard Roman)
- `Char (20; 512)` = ¶ (Microsoft Windows Standard Roman)
- `Char (182; -32768)` = ¶ (Unicode)

Remarks

⇢ The characters with the codes from 0 to 31 usually cannnot be displayed on the screen and, therefore, are either represented with a symbol for missing characters (usually a rectangle) or not displayed at all.

⇢ If you want to be sure that your formulas return the same results on all platforms and with all localized RagTime 5 versions, you should not omit *Encoding*. The encoding -32768 (Unicode) is always present and contains the largest number of characters.

⇢ Decimal fractions, if any, in *CharacterCode* and *Encoding* are ignored.

Warnings

⚠ If *CharacterCode* is less than 0 or greater than permitted for the *Encoding*, **Char** returns the ▸ error value ▸ `NUM!`.

⚠ If *Encoding* specifies a table not present, then **Char** returns ▸ error value ▸ NA!.

⚠ If *CharacterCode* does not occur in the table specified by *Encoding*, then **Char** returns ▸ error value ▸ NA!

See also

▸ Text Functions [p. 36]

▸ Conversion Functions [p. 27]

▸ **Code** [p. 72] – Returns the character code of the first character in a text having a given encoding.

4.17 CHOOSE (FUNCTION)

Choose choose any of several values or ranges.

The first argument determines which value or range is selected from the list of possibilities.

Syntax

Choose (*Index*; *Values*)

Index is a ▸ number specifying which value or range from the list is to be the result of the function.

Values is a ▸ list of ▸ values or ranges.

Examples

- `Choose (2; 'Yesterday'; 'Today'; 'Tomorrow') = Today`
- If the cells `A1:A9` contain the numbers 1 to 9:
  `Sum (Choose (3; A1:A3; A4:A6; A7:A9)) = 24`

Remarks

If **Choose** returns one of several ranges, it can only be used as an argument of another function. It may not stand alone in a cell. (See the second example, above.)

See also

▸ Search Functions [p. 33]

▸ **Index** [p. 117] – Extracts any value from a table.

4.18 ClAddWorkDaysUSA (Function)

ClAddWorkDaysUSA adds up to 1000 workdays in the USA to a ▸ date.

Syntax

ClAddWorkDaysUSA (*Date*; *NumberOfWorkingDays*)
Date is the ▸ date, to which the workdays should be added
NumberOfWorkingDays is die Anzahl der zu addierenden Werktage.

Examples

- ClAddWorkDaysUSA (2/25/1999; 10) = 3/11/1999
  AddDay (2/25/1999; 10) = 3/7/1999

Warnings

⚠ If the year of *Date* is less than 1582 or greater than 2199, **ClAddWorkDaysUSA** returns ▸ error value ▸ DATE!.

Remarks

- Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.
- **ClAddWorkDaysUSA** is based on the definitions of holidays and other special days in the USA as of the beginning of 1999. Although **ClAddWorkDaysUSA** results are also given for the years prior to the founding of the USA, it is unlikely that the Indians celebrated these.
- Only the national holidays are considered when making the distinction between workdays and holidays.
- These functions belong to a group of external functions, which are installed as ▸ Add Ons [RagTime 5 Reference]. If this function is missing, you can install the group "Calender (Country)". For more information, please refer to the "About RagTime" manual.

See also
- ▸ **AddDay** [p. 41] – Adds days to a given date.
- ▸ Calendar Functions [p. 30]
- ▸ Date Functions [p. 28]

4.19 CLDATETONUMBER (FUNCTION)

ClDateToNumber is available to maintain compatibility with documents from earlier versions of RagTime. However, you should not use this function in new documents. Please use the function ▸ Number (Date) [p. 154].

Syntax

ClDateToNumber (*Date*)

See also
▸ Calendar Functions [p. 30]
▸ Date Functions [p. 28]

4.20 ClDayInfoUSA (Function)

ClDayInfoUSA returns, if available, information regarding a ▸ date (US holidays and other special days).

Syntax

ClDayInfoUSA (*Date*)

Date is the ▸ date for which the information should be returned.

Examples

- `ClDayInfoUSA (2/25/1999) = ""` (no information)
- `ClDayInfoUSA (7/4/1999) = 'Independence Day'`
- `ClDayInfoUSA (9/6/1999) = 'Labor Day'`

Warnings

⚠ If the year of *Date* is less than 1582 or greater than 2199, **ClDayInfoUSA** returns ▸ error value ▸ `DATE!`.

Remarks

- Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.
- **ClDayInfoUSA** is based on the definitions of holidays and other special days in the USA as of the beginning of 1999. Although **ClDayInfoUSA** results are also given for the years prior to the founding of the USA, it is unlikely that the Indians celebrated these.
- These functions belong to a group of external functions, which are installed as ▸ Add Ons [RagTime 5 Reference]. If this function is missing, you can install the group "Calender (Country)". For more information, please refer to the "About RagTime" manual.

See also ▸ Calendar Functions [p. 30]
▸ Date Functions [p. 28]

4.21 CLDAYOFYEAR (FUNCTION)

ClDayOfYear is available to maintain compatibility with documents from earlier versions of RagTime. However, you should not use this function in new documents. Please use the function ▸ DayOfYear (Date) [p. 89].

Syntax

ClDayOfYear (***Date***)

See also

▸ Calendar Functions [p. 30]
▸ Date Functions [p. 28]

4.22 CLDIFFWORKDAYSUSA (FUNCTION)

ClDiffWorkDaysUSA calculates the number of workdays in the USA between two dates - up to 1000 workdays.

Syntax

ClDiffWorkDaysUSA (*Date1*; *Date2*)

Date1 is the first ▸ date.

Date2 is the second ▸ date

Examples

- `ClDiffWorkDaysUSA (1/1/2000; 4/1/2000) = 63`
  `DiffDay (1/1/2000; 4/1/2000) = 91`
  `DiffDays30 (1/1/1999; 4/1/1999) = 90`

Warnings

⚠ If the year of *Date* is less than 1582 or greater than 2199, **ClDiffWorkDaysUSA** returns ▸ error value ▸ `DATE!`.

Remarks

⇢ Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

⇢ **ClDiffWorkDaysUSA** is based on the definitions of holidays and other special days in the USA as of the beginning of 1999. Although **ClDiffWorkDaysUSA** results are also given for the years prior to the founding of the USA, it is unlikely that the Indians celebrated these.

⇢ Only the national holidays are considered when making the distinction between workdays and holidays.

⇢ These functions belong to a group of external functions, which are installed as ▸ Add Ons [RagTime 5 Reference]. If this function is missing, you can install the group "Calender (Country)". For more information, please refer to the "About RagTime" manual.

See also

- ▸ **DiffDay** [p. 91] – Calculates days between two dates.
- ▸ **DiffDays30** [p. 92] – Calculates 30-day months between two dates.
- ▸ Calendar Functions [p. 30]
- ▸ Date Functions [p. 28]

4.23 CLEAN (FUNCTION)

Clean removes nonprintable characters from a text.

Syntax

Clean (*Text*)

Text is a ▸ text which is to be cleaned.

Remarks

Cleaning removes all characters with codes from 0 to 8, 11, 12, 14 to 31 and 127.

See also

- ▸ Text Functions [p. 36]
- ▸ **Trim** [p. 215] – Removes unnecessary space characters from a text.
- ▸ **Lower** [p. 138] – Returns text with all letters changed to lower case.
- ▸ **Upper** [p. 221] – Returns text with all letters changed to upper case.
- ▸ **Proper** [p. 165] – Returns text with all initials changed to upper case.
- ▸ **Exact** [p. 103] – Returns true if two texts match exactly.

4.24 ClEasterSunday (Function)

ClEasterSunday calculates the ▸ date of Easter Sunday for a year.

Easter Sunday is the first Sunday after the first full moon after the vernal (spring) equinox. This Sunday is calculated using the algorithm developed by C. F. Gauß (1777-1855) for calculating the phases of the moon.

Syntax

ClEasterSunday (*Year*)

Year is the ▸ number of the year for which Easter Sunday should be calculated.

Examples

- `ClEasterSunday (1999)` returns April 4, 1999.

Warnings

⚠ If *Year* is less than 1582 or greater than 2199, **ClEasterSunday** returns ▸ error value ▸ `DATE!`.

Remarks

Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

These functions belong to a group of external functions, which are installed as ▸ Add Ons [RagTime 5 Reference]. If this function is missing, you can install the group "Calender (Country)". For more information, please refer to the "About RagTime" manual.

See also ▸ Calendar Functions [p. 30]
▸ Date Functions [p. 28]

4.25 ClFirstAdvent (Function)

ClFirstAdvent calculates the ▸ date of the first Advent Sunday in a year. This is the fourth Sunday before December 24th.

Syntax

ClFirstAdvent (*Year*)

Year is the ▸ number of the year for which the first Advent Sunday should be calculated.

Examples

- `ClFirstAdvent (1988)` returns November 27, 1988.

Warnings

⚠ If *Year* is less than 1582 or greater than 2199, **ClFirstAdvent** returns ▸ error value ▸ `DATE!`.

Remarks

⇢ Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

⇢ These functions belong to a group of external functions, which are installed as ▸ Add Ons [RagTime 5 Reference]. If this function is missing, you can install the group "Calender (Country)". For more information, please refer to the "About RagTime" manual.

See also ▸ Calendar Functions [p. 30]
▸ Date Functions [p. 28]

4.26 CLJULIANDATE (FUNCTION)

ClJulianDate converts a Gregorian ▸ date into a Julian date (J.D.).

In accordance with the suggestion of J.J. Scaliger (1581), the Julian date is the number of mean solar days since the mean noon on January 1, 4713 BC. The time is expressed as a decimal fraction of the day.

Syntax

ClJulianDate (*Date*)

Date is the to be calculated ▸ date.

Examples

- `ClJulianDate (1/1/1980) = 2444239.5`
  `ClJulianDate (6/17/1953 9:38:53) = 2434545.9020023`

⇢ **ClJulianDate** interprets *Date* as GMT (Greenwich Mean Time). If you wish to calculate the date for a different time zone, you must correct accordingly, e.g. for a date in the Central European Time zone (CET):

- `ClJulianDate (AddHour(6/17/1953 9:38:53;-1)) =2434545.8603357`

Remarks

⇢ Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

⇢ These functions belong to a group of external functions, which are installed as ▸ Add Ons [RagTime 5 Reference]. If this function is missing, you can install the group "Calendar (Land)". For more information, please refer to the "About RagTime" manual.

See also ▸ Calendar Functions [p. 30]
▸ Date Functions [p. 28]

4.27 ClModJulian (Function)

ClModJulian converts a Gregorian ▸ date into a modified Julian date (M.J.D.).

In accordance with the recommendation made by the Smithsonian Institution during the International Geophysical Year 1957/58, the modified Julian date is the number of mean solar days since November 17, 1858, 00:00 (midnight) world time. The time is expressed as a decimal fraction of the day.

Syntax

ClModJulian (*Date*)

Date is the ▸ date to be calculated

Examples

- `ClModJulian (11/17/1858) = 0`
  `ClModJulian (1/1/1980) = 44239`
  `ClModJulian (6/17/1953 9:38:53) = 34545.402002315`

⇢ **ClModJulian** interprets the *Date* as in GMT (Greenwich Mean Time). If you need to convert the date to another time zone, you must correct this accordingly, e.g. for a date in the Central European Time (CET) zone:

- `ClModJulian (AddHour(6/17/1953 9:38:53;-1)) = 34545.360335648`

Remarks

⇢ Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

⇢ These functions belong to a group of external functions, which are installed as ▸ Add Ons [RagTime 5 Reference]. If this function is missing, you can install the group "Calender (Country)". For more information, please refer to the "About RagTime" manual.

See also ▸ Calendar Functions [p. 30]
▸ Date Functions [p. 28]

4.28 CLNUMBERTODATE (FUNCTION)

ClNumberToDate is available to maintain compatibility with documents from earlier versions of RagTime. However, you should not use this function in new documents. Please use the function ▸ Date (Number of Days) [p. 85].

Syntax

ClNumberToDate (***Number of Days***)

See also ▸ Calendar Functions [p. 30]
▸ Date Functions [p. 28]

4.29 ClWorkDaysInMonthUSA (Function)

ClWorkDaysInMonthUSA calculates the number of workdays in the USA in a given month.

Syntax

ClWorkDaysInMonthUSA (*Year*; *Month*)

Year is a ▸ number, which gives the year and

Month is a ▸ number, which gives the month for which the number of workdays should be calculated.

Examples

- `ClWorkDaysInMonthUSA (1999; 3) = 23`
- `ClWorkDaysInMonthUSA (1997; 3) = 21`

Warnings

⚠ If the year of *Year* is less than 1582 or greater than 2199, **ClWorkDaysInMonthUSA** returns ▸ error value ▸ DATE!.

Remarks

⇢ Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

⇢ **ClWorkDaysInMonthUSA** is based on the definitions of holidays and other special days in the USA as of the beginning of 1999. Although **ClWorkDaysInMonthUSA** results are also given for the years prior to the founding of the USA, it is unlikely that the Indians celebrated these.

⇢ Only the national holidays are considered when making the distinction between workdays and holidays.

⇢ These functions belong to a group of external functions, which are installed as ▸ Add Ons [RagTime 5 Reference]. If this function is missing, you can install the group "Calender (Country)". For more information, please refer to the "About RagTime" manual.

See also ▸ Calendar Functions [p. 30]
▸ Date Functions [p. 28]

4.30 CLWORKDAYSINYEARUSA (FUNCTION)

ClWorkDaysInYearUSA calculates the number of workdays in the USA for a given year.

Syntax

ClWorkDaysInYearUSA (*Year*)

Year is a ▸ number, which gives the year for which the number of workdays should be calculated.

Examples

- `ClWorkDaysInYearUSA (1999) = 255`
- `ClWorkDaysInYearUSA (2000) = 253`

Warnings

⚠ If the year of *Year* is less than 1582 or greater than 2199, **ClWorkDaysInYearUSA** returns ▸ error value ▸ `DATE!`.

Remarks

- Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.
- **ClWorkDaysInYearUSA** is based on the definitions of holidays and other special days in the USA as of the beginning of 1999. Although **ClWorkDaysInYearUSA** results are also given for the years prior to the founding of the USA, it is unlikely that the Indians celebrated these.
- Only the national holidays are considered when making the distinction between workdays and holidays.
- These functions belong to a group of external functions, which are installed as ▸ Add Ons [RagTime 5 Reference]. If this function is missing, you can install the group "Calender (Country)". For more information, please refer to the "About RagTime" manual.

See also ▸ Calendar Functions [p. 30]
▸ Date Functions [p. 28]

4.31 ClWorkDayUSA (Function)

ClWorkDayUSA checks, whether a ▸ date is a workday throughout the USA or legal holiday and returns a▸ truth value.

Syntax

ClWorkDayUSA (*Date*)

Date is the ▸ date, which should be checked to determine whether it is a workday or holiday.

Examples

- ClWorkDayUSA (12/25/1998) = 0 (Christmas)
- ClWorkDayUSA (11/25/1998) = 1 (Wednesday)
- ClWorkDayUSA (11/25/1999) = 0 (Thanksgiving Day)

Warnings

⚠ If the year of *Date* is less than 1582 or greater than 2199, **ClWorkDayUSA** returns ▸ error value ▸ DATE!.

Remarks

⇢ Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

⇢ **ClWorkDayUSA** is based on the definitions of holidays and other special days in the USA as of the beginning of 1999. Although **ClWorkDayUSA** results are also given for the years prior to the founding of the USA, it is unlikely that the Indians celebrated these.

⇢ Only the national holidays are considered when making the distinction between workdays and holidays.

⇢ These functions belong to a group of external functions, which are installed as ▸ Add Ons [RagTime 5 Reference]. If this function is missing, you can install the group "Calendar (Country)". For more information, please refer to the "About RagTime" manual.

See also ▸ Calendar Functions [p. 30]
▸ Date Functions [p. 28]

4.32 CODE (FUNCTION)

Code returns the character code of the first character in a text with the given ▸ encoding [p. 236].

Syntax

Code (*Text*; *Encoding*)

Text is a letter or other character of which the code is to be determined.

Encoding is a ▸ number in the ▸ interval [-32768; 32767], which specifies how *Text* is encoded. If *Encoding* is not given, the standard encoding of the operating system of the given computer is used.

Examples

- `Code ('A') = 65`
- `Code ('a') = 97`
- `Code (1) = 49`
- `Code ('A';256) = 65` (Mac OS Standard Roman)
- `Code ('A';512) = 65` (Microsoft Windows Standard Roman)
- `Code ('ö';256) = 154` (Mac OS Standard Roman)
- `Code ('ö';512) = 246` (Microsoft Windows Standard Roman)
- `Code ('ö';1024) = 148` (Microsoft DOS Code Page 437)
- `Code ('ö';-32768) = 246` (Unicode)
- `Code ('¶';256) = 166` (Mac OS Standard Roman)
- `Code ('¶';512) = 20` (Microsoft Windows Standard Roman)
- `Code ('¶';-32768) = 182` (Unicode)

Remarks

⇢ If the *Text* consists of several characters, **Code** returns the code of the first character.

⇢ Decimal fractions, if any, in *Encoding* are ignored.

Warnings

⚠ If *Text* is not in the table specified by *Encoding*, then **Code** returns ▸ error value ▸ `NA!`.

⚠ If *Encoding* specifies a table not present, then **Code** returns ▸ error value ▸ `NA!`.

See also

▸ Text Functions [p. 36]

▸ Conversion Functions [p. 27]

▸ **`Char`** [p. 55] – Returns a character with a given character code and encoding.

4.33 COLUMN (FUNCTION)

Column returns the column in which the formula is or the column of the specified cell.

Syntax

Column ()

This function requires no arguments.

Syntax

Column (*Cell*)

Cell is a ▸ range. **Column** returns the column coordinate of the upper left cell of the range. If *Cell* is not given, **Column** returns the column of the cell containing the formula.

Examples

- `Column () = 3`
- `Column (E2) = 5`

See also

▸ Spreadsheet Functions [p. 34]

▸ **Row** [p. 175] – Get the row where the formula or the specified cell is.

▸ **Plane** [p. 162] – Get the plane where the formula or the specified cell is.

4.34 COLUMNVALUE (FUNCTION)

ColumnValue returns a value from a particular column to be used in the functions **MailMerge**, **Choose** or **VSearch**.

The value of a column is the contents of the top cell within the range of the search.

Syntax

ColumnValue (*Index*; *NestingLevel*)

Index is a ▸ number specifying in which column the value is to be determined. The index is relative to the range which is searched by a search function.

NestingLevel is an optional ▸ number specifying the number of levels up in the hierarchy of nested search functions **ColumnValue** should be calculated. If *NestingLevel* is not given or equals 0, **ColumnValue** is calculated for the innermost of the nested functions.

Examples

It is assumed for the example that cells A1:A4 contain the numbers 1 to 4 and cells B1:B4 contain the numbers 5 to 8 and that the range A1:B4 is searched by VSearch.

- `ColumnValue (2) = 5`

Remarks

ColumnValue can only be used in an argument of the functions **MailMerge**, **Choose** and **VSearch**.

See also

▸ Spreadsheet Functions [p. 34]
▸ Search Functions [p. 33]
▸ **VSearch** [p. 225] – Scans a range by rows for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.
▸ **Selection** [p. 181] – Looks for the nth line in a table that matches a condition and returns the index of that line.
▸ **MailMerge** [p. 139] – Extracts data from tables for serial printings.

4.35 COMBINATIONS (FUNCTION)

Combinations calculates the number of possible combinations of m elements chosen from n elements.

Syntax

Combinations (*Elements*; *NumberChosen*)

Elements is a ▸ number specifing how many elements are available.

NumberChosen is a ▸ number specifing how many elements are to be chosen.

Examples

- `Combinations (5; 2) = 10`
- `Combinations (49; 6) = 13.983.816` (Nearly 14 millionen different results are possible in a lottery in which 6 numbers are drawn from a pool of 49.)

Remarks

Combinations is defined by the formula `Factorial (Elements) / (Factorial (NumberChosen) * Factorial (Elements-NumberChosen))`.

See also

▸ Statistical Functions [p. 35]

▸ **Factorial** [p. 106] – Calculates the factorial of a number.

▸ **Permutations** [p. 159] – Calculates the number of permutations of m elements chosen out of n elements.

4.36 COMPOUND (FUNCTION)

Compound calculates the accumulated interest for a given interest rate and number of periods.

Syntax

Compound (*InterestRate*; *Periods*)

InterestRate is a ▸ number which specifies the interest rate.

Periods is a ▸ number which specifies the number of payment periods.

Examples

- `Compound (0.05; 12) = 1.77958...` (If the principle is invested for 12 years at 5% interest, it grows to nearly 1.78 times the initial sum.)

Remarks

⇢ Compound interest is calculated with the formula `(1 + interest) ^periods`.

See also ▸ Financial Functions [p. 31]

▸ **Annuity** [p. 48] – Calculates the Annuity for a given interest rate and number of periods.

4.37 CONCAT (FUNCTION)

Concat concatenates a list of texts.

Syntax

Concat (*ListOfText*)

ListOfText is a ▸ list of which the elements are to be concatenated in a ▸ text.

Examples

- `Concat ('Hello ';'World';'!') = Hello World!`
- `Concat (47;11) = 4711`

Remarks

⇢ Elements in the list which are not text are converted to text if possible and concatenated with the other elements. (See the second example, above.)

See also

- ▸ Text Functions [p. 36]
- ▸ Operators [p. 31]
- ▸ **Replace** [p. 172] – Replaces specific characters in a text.
- ▸ **Repeat** [p. 171] – Repeats a given text n times.
- ▸ **SmartConcat** [p. 196] – Concatenates texts with delimiter.

4.38 CONTAINER (FUNCTION)

Container returns the name of the innermost container containing the formula.

Syntax

Container ()

This function requires no arguments.

Examples

- `Container () = Evaluation`

Remarks

⇢ If the container is unnamed, **Container** returns an empty text.

See also

▸ Miscellaneous Functions [p. 32]
▸ **Page** [p. 156] – Returns the number of the page containing the formula.
▸ **PageIndex** [p. 157] – Returns the index of the page containing the formula.
▸ **NoOfPages** [p. 150] – Returns the number of pages.

4.39 COS (FUNCTION)

Cos calculates the cosine of an angle.

Cos returns a ▸ number in the ▸ interval [-1; +1].

Syntax

Cos (*Angle*)

Angle is a ▸ number which expresses in radians the angle of which the cosine is to be calculated.

Remarks

⇢ If *Angle* is expressed in degrees, use `Cos (Radians (Angle))`.

See also

- ▸ Trigonometric Functions [p. 37]
- ▸ **ArcCos** [p. 49] – Calculates the arc cosine of a number.
- ▸ **Sin** [p. 194] – Calculates the sine of an angle given in radians.
- ▸ **Tan** [p. 210] – Calculates the tangent of an angle given in radians.
- ▸ **Radians** [p. 167] – Converts an angle from degrees to radians.

4.40 COUNT (FUNCTION)

Count counts the numbers in a list.

Syntax

Count (*List*)

List is a ▸ list. The function counts how many numbers occur in it.

Examples

- `Count (1; 2; 3) = 3`
- `Count (1; 2; 'three') = 2`

See also

▸ Statistical Functions [p. 35]

▸ **Average** [p. 52] – Calculates the average of numbers in a list.

▸ **StDev** [p. 202] – Calculates the standard deviation of numbers in a list.

▸ **Var** [p. 224] – Calculates the variance of numbers in a list.

4.41 CURRENTCELL (FUNCTION)

CurrentCell returns the current cell in a search function.

The current cell is the one which is being examined by a search function at any given moment.

Syntax

CurrentCell (*NestingLevel*)

NestingLevel is an optional ▸ number specifying the number of levels up in the hierarchy of nested search functions **CurrentCell** should be calculated. If *NestingLevel* is not given or equals 0, **CurrentCell** is calculated for the innermost of the nested functions.

Examples

- `Row (CurrentCell ()) = 4`

Remarks

CurrentCell can only be used within the arguments of the search functions. (Also in **Choose** and **MailMerge**.)

See also

▸ Spreadsheet Functions [p. 34]

▸ Search Functions [p. 33]

▸ **Search** [p. 177] – Scans a range for matches of a condition and returns either the count of matches or the StartValue, which is replaced by NextValue in case of matches.

▸ **VSearch** [p. 225] – Scans a range by rows for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.

▸ **HSearch** [p. 114] – Scans a range by columns for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.

▸ **Selection** [p. 181] – Looks for the nth line in a table that matches a condition and returns the index of that line.

▸ **MailMerge** [p. 139] – Extracts data from tables for serial printings.

4.42 CURRENTCOUNT (FUNCTION)

CurrentCount returns the count of matches in a search function.

The count of matches expresses how many cells fulfilling the condition have been found by the search function.

Syntax

CurrentCount (*NestingLevel*)

NestingLevel is an optional ▸ number specifying the number of levels up in the hierarchy of nested search functions **CurrentCount** should be calculated. If *NestingLevel* is not given or equals 0, **CurrentCount** is calculated for the innermost of the nested functions.

Examples

- `CurrentCount () = 4`

Remarks

⇢ **CurrentCounts** can only be used within the arguments of the search functions.

See also

▸ Spreadsheet Functions [p. 34]

▸ Search Functions [p. 33]

▸ **Search** [p. 177] – Scans a range for matches of a condition and returns either the count of matches or the StartValue, which is replaced by NextValue in case of matches.

▸ **VSearch** [p. 225] – Scans a range by rows for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.

▸ **HSearch** [p. 114] – Scans a range by columns for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.

4.43 CURRENTINDEX (FUNCTION)

CurrentIndex returns the current index in a search function.

The index expresses which nth cell of a range is currently being examined by a search function.

Syntax

CurrentIndex (*NestingLevel*)

NestingLevel is an optional ▸ number specifying the number of levels up in the hierarchy of nested search functions **CurrentIndex** should be calculated. If *NestingLevel* is not given or equals 0, **CurrentIndex** is calculated for the innermost of the nested functions.

Examples

- `CurrentIndex () = 7`

Remarks

CurrentIndex can only be used within the arguments of the search functions.

See also

▸ Spreadsheet Functions [p. 34]

▸ Search Functions [p. 33]

▸ **Search** [p. 177] – Scans a range for matches of a condition and returns either the count of matches or the StartValue, which is replaced by NextValue in case of matches.

▸ **VSearch** [p. 225] – Scans a range by rows for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.

▸ **HSearch** [p. 114] – Scans a range by columns for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.

4.44 CURRENTRESULT (FUNCTION)

CurrentResult returns the current result of a search function.

Syntax

CurrentResult (*NestingLevel*)

NestingLevel is an optional ▸ number specifying the number of levels up in the hierarchy of nested search functions **CurrentResult** should be calculated. If *NestingLevel* is not given or equals 0, **CurrentResult** is calculated for the innermost of the nested functions.

Examples

- `CurrentResult () = 23`

Remarks

⇢ **CurrentResult** can only be used within the arguments of the search functions.

See also

▸ Spreadsheet Functions [p. 34]

▸ Search Functions [p. 33]

▸ **Search** [p. 177] – Scans a range for matches of a condition and returns either the count of matches or the StartValue, which is replaced by NextValue in case of matches.

▸ **VSearch** [p. 225] – Scans a range by rows for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.

▸ **HSearch** [p. 114] – Scans a range by columns for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.

4.45 DATE (FUNCTION)

Date converts text or a numberto a date.

Syntax

Date (*Text*)

Text is a ▸ text which is to be converted to a date.

Syntax

Date (*NumberOfDays*)

NumberOfDays is the ▸ number, number of days since January 1, 1904.

If the *NumberOfDays* is negative, **Date** returns a corresponding date from BEFORE January 1, 1904.

⇢ Decimal fractions, if any, in *NumberOfDays* are treated as the time on the resulting date.

Examples

- `Date ('1/1/80') = 1/1/1980`
- `Date (27759) = 1/1/1980`
- `Date (18058.402002315) = 06/10/1953 9:38:53`

Warnings

⚠ If the argument cannot be converted to a date, **Date** returns the ▸ error value ▸ DATE!.

See also

▸ Date Functions [p. 28]

▸ Text Functions [p. 36]

▸ Conversion Functions [p. 27]

▸ **TimeSpan** [p. 213] – Converts text to a time span.

▸ **Number** [p. 154] – Converts text to a number or a date into the number of days from January 1, 1904.

▸ **Value** [p. 222] – Converts text to a value, as if entered into a spreadsheet cell.

▸ **Text** [p. 211] – Converts a value to text.

4.46 DAYOF (FUNCTION)

DayOf extracts the day from a date.

Syntax

DayOf (*Date*)

Date is a ▸ date of which the hour is calculated.

Examples

- If the current time is 9:38:53 on 2/22/1732,
  `DayOf (Now) = 22`

See also

- ▸ Date Functions [p. 28]
- ▸ **SecondOf** [p. 180] – Extracts seconds from a date.
- ▸ **MinuteOf** [p. 146] – Extracts minutes from a date.
- ▸ **HourOf** [p. 113] – Extracts hours from a date.
- ▸ **MonthOf** [p. 148] – Extracts the month from a date.
- ▸ **YearOf** [p. 231] – Extracts the year from a date.

4.47 DAYOFWEEK (FUNCTION)

DayOfWeek returns the ▸ number of the day of the week according to US standards. (Sunday: 1, Monday: 2, Tuesday: 3, Wednesday: 4, Thursday: 5, Friday: 6, Saturday: 7)

Syntax

DayOfWeek (*Date*)

Date is a ▸ date of which the day of the week is to be calculated.

Examples

- `DayOfWeek (1/1/1980) = 3`
- `DayOfWeekISO (1/1/1980) = 2`

Remarks

Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

See also

▸ Date Functions [p. 28]
▸ **DayOfWeekISO** [p. 88] – Returns the day of the week, monday = 1 to sunday = 7.
▸ **DayOf** [p. 86] – Extracts the day from a date.

4.48 DAYOFWEEKISO (FUNCTION)

DayOfWeekISO returns the ▸ number of the day of the week according to DIN/ISO standards. (Monday: 1, Tuesday: 2, Wednesday: 3, Thursday: 4, Friday: 5, Saturday: 6, Sunday: 7)

Syntax

DayOfWeekISO (*Date*)
Date is a ▸ date of which the day of the week is to be calculated.

Examples

- `DayOfWeekISO (1/1/1980) = 2`
- `DayOfWeek (1/1/1980) = 3`

Remarks

Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

See also
▸ Date Functions [p. 28]
▸ **DayOfWeek** [p. 87] – Returns the day of the week, sunday = 1 to saturday = 7.
▸ **DayOf** [p. 86] – Extracts the day from a date.

4.49 DAYOFYEAR (FUNCTION)

DayOfYear returns the sequential ▸ number of the day of the year.

Syntax

DayOfYear (*Date*)

Date is a ▸ date for which the sequential number of the day in the year is to be determined.

Examples

- `DayOfYear (1/1/1995) = 1`
- `DayOfYear (12/31/1995) = 365`
- `DayOfYear (12/31/1996) = 366`

Remarks

⇢ Leap years are taken into account.

⇢ Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

See also ▸ Date Functions [p. 28]

▸ **DayOf** [p. 86] – Extracts the day from a date.

4.50 DEGREES (FUNCTION)

Degrees converts an angle from radians to degrees.

Degrees calculates a ▸ number using `degrees = radians * 180 / π`.

Syntax

Degrees (***AngleInRadians***)

AngleInRadians is a ▸ number which is to be converted.

See also

- ▸ Trigonometric Functions [p. 37]
- ▸ Conversion Functions [p. 27]
- ▸ **Radians** [p. 167] – Converts an angle from degrees to radians.

4.51 DIFFDAY (FUNCTION)

DiffDay calculates the number of days between two dates.

Syntax

DiffDay (*Date1*; *Date2*)

Date1 is the first ▸ date.

Date2 is the second ▸ date.

Examples

- `DiffDay (Now; DiffDay(Now; 1200)) = 1200`

Remarks

⇢ If the second date is earlier than the first date, **DiffSecond** returns a negative result.

See also

- ▸ Date Functions [p. 28]
- ▸ **DiffDays30** [p. 92] – Calculates 30-day months between two dates.
- ▸ **AddDay** [p. 41] – Adds days to a given date.
- ▸ **DiffSecond** [p. 96] – Calculates seconds between two dates.
- ▸ **DiffMinute** [p. 94] – Calculates minutes between two dates.
- ▸ **DiffHour** [p. 93] – Calculates hours between two dates.
- ▸ **DiffMonth** [p. 95] – Calculates months between two dates.
- ▸ **DiffYear** [p. 97] – Calculates years between two dates.

4.52 DIFFDAYS30 (FUNCTION)

DiffDays30 calculates the number of days between two dates, assuming each month has 30 days.

Syntax

DiffDays30 (*Date1*; *Date2*)

Date1 is the first ▸ date.

Date2 is the second ▸ date.

Examples

- `DiffDays30 (Now; AddYear(Now; 1)) = 360`

Remarks

⇢ If the second date is earlier than the first date, **DiffSecond** returns a negative result.

See also ▸ Date Functions [p. 28]

▸ **DiffDay** [p. 91] – Calculates days between two dates.

4.53 DIFFHOUR (FUNCTION)

DiffHour calculates the number of hours between two dates.

Syntax

DiffHour (*Date1*; *Date2*)

Date1 is the first ▸ date.

Date2 is the second ▸ date.

Examples

- `DiffHour (Now; AddSecond(Now; 1200)) = 0`
- `DiffHour (Now; AddDay(Now; 1200)) = 28800`

Remarks

If the second date is earlier than the first date, **DiffSecond** returns a negative result.

See also

▸ Date Functions [p. 28]

▸ **AddHour** [p. 42] – Adds hours to a given date.

▸ **DiffSecond** [p. 96] – Calculates seconds between two dates.

▸ **DiffMinute** [p. 94] – Calculates minutes between two dates.

▸ **DiffDay** [p. 91] – Calculates days between two dates.

▸ **DiffMonth** [p. 95] – Calculates months between two dates.

▸ **DiffYear** [p. 97] – Calculates years between two dates.

4.54 DIFFMINUTE (FUNCTION)

DiffMinute calculates the number of minutes between two dates.

Syntax

DiffMinute (*Date1*; *Date2*)

Date1 is the first ▸ date.

Date2 is the second ▸ date.

Examples

- `DiffMinute (Now; AddSecond(Now; 1200)) = 20`

Remarks

⇢ If the second date is earlier than the first date, **DiffMinute** returns a negative result.

See also

- ▸ Date Functions [p. 28]
- ▸ **AddMinute** [p. 43] – Adds minutes to a given date.
- ▸ **DiffSecond** [p. 96] – Calculates seconds between two dates.
- ▸ **DiffHour** [p. 93] – Calculates hours between two dates.
- ▸ **DiffDay** [p. 91] – Calculates days between two dates.
- ▸ **DiffMonth** [p. 95] – Calculates months between two dates.
- ▸ **DiffYear** [p. 97] – Calculates years between two dates.

4.55 DIFFMONTH (FUNCTION)

DiffMonth calculates the number of months between two dates.

Syntax

DiffMonth (*Date1*; *Date2*)

Date1 is the first ▸ date.

Date2 is the second ▸ date.

Examples

- `DiffMonth (Now; AddDay(Now; 1200)) = 40`

Remarks

⇢ If the second date is earlier than the first date, **DiffSecond** returns a negative result.

See also

- ▸ Date Functions [p. 28]
- ▸ **AddMonth** [p. 44] – Adds months to a given date.
- ▸ **DiffSecond** [p. 96] – Calculates seconds between two dates.
- ▸ **DiffMinute** [p. 94] – Calculates minutes between two dates.
- ▸ **DiffHour** [p. 93] – Calculates hours between two dates.
- ▸ **DiffDay** [p. 91] – Calculates days between two dates.
- ▸ **DiffYear** [p. 97] – Calculates years between two dates.

4.56 DIFFSECOND (FUNCTION)

DiffSecond calculates the number of seconds between two dates.

Syntax

DiffSecond (*Date1*; *Date2*)

Date1 is the first ▸ date.

Date2 is the second ▸ date.

Examples

- `DiffSecond (Now; AddSecond(Now; 1200)) = 1200`

Remarks

⇢ If the second date is earlier than the first date, **DiffSecond** returns a negative result.

See also

- ▸ Date Functions [p. 28]
- ▸ **AddSecond** [p. 45] – Adds seconds to a given date.
- ▸ **DiffMinute** [p. 94] – Calculates minutes between two dates.
- ▸ **DiffHour** [p. 93] – Calculates hours between two dates.
- ▸ **DiffDay** [p. 91] – Calculates days between two dates.
- ▸ **DiffMonth** [p. 95] – Calculates months between two dates.
- ▸ **DiffYear** [p. 97] – Calculates years between two dates.

4.57 DIFFYEAR (FUNCTION)

DiffYear calculates the number of years between two dates.

Syntax

DiffYear (*Date1*; *Date2*)

Date1 is the first ▸ date.

Date2 is the second ▸ date.

Examples

- `DiffYear (Now; AddDay(Now; 1200)) = 3`

Remarks

⇢ If the second date is earlier than the first date, **DiffSecond** returns a negative result.

See also

- ▸ Date Functions [p. 28]
- ▸ **AddYear** [p. 46] – Adds years to a given date.
- ▸ **DiffSecond** [p. 96] – Calculates seconds between two dates.
- ▸ **DiffMinute** [p. 94] – Calculates minutes between two dates.
- ▸ **DiffHour** [p. 93] – Calculates hours between two dates.
- ▸ **DiffDay** [p. 91] – Calculates days between two dates.
- ▸ **DiffMonth** [p. 95] – Calculates months between two dates.

4.58 DOCUMENTDATE (FUNCTION)

DocumentDate returns the document's date.

Syntax

DocumentDate ()

This function requires no arguments.

Examples

- `DocumentDate () = 6/15/1996`

Remarks

The document date can be seen and changed in the "Document Settings" dialog box. If it has not been changed by the user, it is the date on which the document was created.

See also

- Miscellaneous Functions [p. 32]
- Date Functions [p. 28]
- **DocumentName** [p. 99] – Returns the name of the document.
- **SetDocName** [p. 185] – Sets the default name of the document for "Save as...".

4.59 DOCUMENTNAME (FUNCTION)

DocumentName returns the name of the document as ▸ text.

Syntax

DocumentName ()

This function requires no arguments.

See also

▸ Miscellaneous Functions [p. 32]

▸ Text Functions [p. 36]

▸ **SetDocName** [p. 185] – Sets the default name of the document for "Save as...".

▸ **DocumentDate** [p. 98] – Returns the document's date which was initialized when the document was created.

4.60 ENDINGPAGENUMBER (FUNCTION)

EndingPageNumber returns the page number of the last page.

Syntax

EndingPageNumber (*LayoutName*)

LayoutName is a ▸ text, giving the name of a layout. The function returns the number of the last page of this layout. If no layout is given, the number of the last page of the layout in which the formula appears is returned.

Examples

- `EndingPageNumber () = 12`
- `EndingPageNumber ('Layout 2') = 25`

Remarks

⇢ The formula with the function need not be in the layout of which the page number is returned.

See also

▸ Miscellaneous Functions [p. 32]

▸ **Page** [p. 156] – Returns the number of the page containing the formula.

▸ **PageIndex** [p. 157] – Returns the index of the page containing the formula.

▸ **NoOfPages** [p. 150] – Returns the number of pages.

▸ **StartingPageNumber** [p. 201] – Returns the starting page number of the layout.

4.61 ERROR (FUNCTION)

Error returns the error for a given error code.

Syntax

Error (*ErrorCode*)

ErrorCode is a whole number between 0 and 13.

Examples

- `Error (2) = DIV/0!`
- `Error (8) = RANGE!`

Remarks

For values below 0 or greater than 126, **Error** returns the error value **RANGE!!**; for 0, no value; and for values greater than 13, the general error **ERROR!**.

See also

- Conversion Functions [p. 27]
- **NA** [p. 149] – Returns the error value NA!.
- **ErrorType** [p. 102] – Returns a code according to the error type of the value.
- **IsErr** [p. 121] – Returns true if the value is any error value.

4.62 ERRORTYPE (FUNCTION)

ErrorType returns a code for the type of the error.

Syntax

ErrorType (*ErrorValue*)

ErrorValue is a ▸ value. **Type** returns a code which identifies what sort of error occurred. The following codes are possible:

- 0: No error
- 1: **NULL!**
- 2: **DIV/0!**
- 3: **VALUE!**
- 4: **REF!**
- 5: **NAME!**
- 6: **NUM!**
- 7: **NA!**
- 8: **RANGE!**
- 9: **DATE!**
- 10: **CIRC!**
- 11: **EVAL!**
- 12: **ILLEGAL!**
- 13: **COMPLEX!**

Examples

- `ErrorType (1/0) = 2`
- `ErrorType (23) = 0`

See also ▸ Status Functions [p. 35]

▸ **Error** [p. 101] – Returns an error according to an error code.

4.63 EXACT (FUNCTION)

Exact returns true if two texts match exactly.

Syntax

Exact (*Text1*; *Text2*)

Text1 is the first ▸ text which is to be compared.

Text2 is the second ▸ text which is to be compared.

Examples

- `Exact ('Hello';'He' & 'llo') = 1`
- `Exact ('Hello';'hello') = 0`
- `Exact ('Hello';'Héllo') = 0`

Remarks

⇢ Texts are equal only if they are identical in the use of upper and lower case orthography and accent marks. (See the second and third examples, above.)

See also

- ▸ Text Functions [p. 36]
- ▸ **Clean** [p. 63] – Removes non-printable characters from a text.
- ▸ **Trim** [p. 215] – Removes unnecessary space characters from a text.
- ▸ **Lower** [p. 138] – Returns text with all letters changed to lower case.
- ▸ **Upper** [p. 221] – Returns text with all letters changed to upper case.
- ▸ **Proper** [p. 165] – Returns text with all initials changed to upper case.

4.64 EXP (FUNCTION)

Exp calculates the exponential function of a ▸ number.

The exponential function of a number is derived by applying the number as a power of the base of the natural logarithm, `e = 2.71828182845904524...`.

Syntax

Exp (*Number*)

Number is a ▸ number by which e is raised.

Examples

- `Exp (0) = 1`
- `Exp (1) = 2.71828...`
- `Exp (2) = 7.38906...`

See also

▸ Arithmetic Functions [p. 26]

▸ **Ln** [p. 129] – Calculates the natural logarithm of a number.

▸ **Expm1** [p. 105] – Calculates (e raised to a specific power) - 1.

4.65 Exp1 (Function)

Exp1 calculates (e raised to a specific power) - 1.

Syntax

Exp1 (*Number*)

Number is a ▸ number by which e is raised. 1 is then subtracted from the result.

Examples

- Exp1 (0) = 0
- Exp1 (1) = 1.71828…
- Exp1 (2) = 6.38906…

See also

▸ Arithmetic Functions [p. 26]

▸ **Ln1** [p. 130] – Calculates the natural logarithm of a (number + 1).

▸ **Exp** [p. 104] – Calculates e raised to a specific power.

4.66 FACTORIAL (FUNCTION)

Factorial calculates n! = 1 * 2 * ... * n.

The factorial of a number is the product of all whole numbers from 1 to the number itself.

Syntax

Factorial (*Number*)

Number is a ▸ number of which the factorial is to be calculated.

Examples

- Factorial (6) = 720

Remarks

The factorial of 0 is defined as 1.

Warnings

⚠ The factorial for negative numbers is not defined. The function returns in this case the ▸ error value ▸ NUM!.

See also

▸ Statistical Functions [p. 35]
▸ Arithmetic Functions [p. 26]
▸ **Combinations** [p. 75] – Calculates the number of combinations of m elements chosen out of n elements.
▸ **Permutations** [p. 159] – Calculates the number of permutations of m elements chosen out of n elements.

4.67 FALSE (FUNCTION)

False returns the constant ▸ truth value "False."

Syntax

False ()

This function requires no arguments.

See also ▸ Constants [p. 27]

▸ **True** [p. 216] – Returns the logical constant True.

4.68 FIND (FUNCTION)

Find returns the position of a specified text within a target text.

Syntax

Find (***Target***; ***Text***; *FirstCharacter*)

Target is a ▸ text which is sought.

Text is a ▸ text which is searched.

FirstCharacter is the position of the character in *Text* from which the search is to begin. If *FirstCharacter* is not given, the search is conducted from the beginning of *Text*.

Examples

- `Find ('Die'; 'Howdie') = 4`
- `Find ('Die'; 'HowdieDoodie';5) = 10`

Remarks

Upper and lower case orthography are not regarded in searches.

See also

- ▸ Text Functions [p. 36]
- ▸ **Left** [p. 127] – Extracts the leftmost n characters of a text.
- ▸ **Right** [p. 173] – Extracts the rightmost n characters of a text.
- ▸ **Mid** [p. 143] – Extracts characters from any position of a text.
- ▸ **Replace** [p. 172] – Replaces specific characters in a text.

4.69 FLOOR (FUNCTION)

Floor rounds down a number to the next lower whole number or the next lower multiple.

Syntax

Floor (*Number*; *Multiple*)

Number is a ▸ number which is to be rounded

Multiple is an optional argument. *Number* is rounded to the next smaller multiple of the argument *Multiple*. If *Multiple* is not given, RagTime rounds down to the next smaller whole number.

Examples

- `Floor (4.9) = 4`
- `Floor (-4.5) = -5`
- `Floor (28; 5) = 25`
- `Floor (-28; 5) = -30`
- `Floor (28; -5) = 30`

Remarks

If *Multiple* is not given, **Floor** behaves like **Int**.

Warnings

If *Multiple* is a negative ▸ number, **Floor** instead rounds up to the next greater Multiple. (See the fifth example above.)

See also

- ▸ Arithmetic Functions [p. 26]
- ▸ Rounding Functions [p. 33]
- ▸ **Int** [p. 119] – Rounds a number down to the largest integer.
- ▸ **Round** [p. 174] – Rounds a number to a specific number of places.
- ▸ **Trunc** [p. 217] – Truncates a number to any number of digits.
- ▸ **Frac** [p. 110] – Returns the fractional part of a number.
- ▸ **Ceiling** [p. 54] – Rounds a number to the next higher whole number or the next higher multiple.

4.70 FRAC (FUNCTION)

Frac returns the fractional part of a ▸ number.

Syntax

Frac (*Number*)

Number is a ▸ number of which the fraction is to be returned.

Examples

- `Frac(3.14) = 0.14`
- `Frac(-3.14) = -0.14`

Remarks

⇢ Every number is the sum of its whole number part and its fractional part: `number = int (number) + frac (number)`.

See also

▸ Arithmetic Functions [p. 26]
▸ Rounding Functions [p. 33]
▸ **Int** [p. 119] – Rounds a number down to the largest integer.
▸ **Round** [p. 174] – Rounds a number to a specific number of places.
▸ **Trunc** [p. 217] – Truncates a number to any number of digits.
▸ **Floor** [p. 109] – Rounds a number to the next lower whole number or the next lower multiple.
▸ **Ceiling** [p. 54] – Rounds a number to the next higher whole number or the next higher multiple.

4.71 FV (FUNCTION)

FV calculates future value of a list of payments at a given interest rate.

Syntax

FV (*InterestRate*; *PaymentList*)

InterestRate is a ▸ number specifying the interest rate.

PaymentList is a ▸ list of numbers specifying the payment in each payment period.

Examples

- `FV (0.05; 100; 100; 100) = 315.25` (Three annual payments of $100 each grow to an end sum of $315.25.)
- `FV (0.065; A3:A8) = 5239.14`

See also ▸ Financial Functions [p. 31]

▸ **NPV** [p. 153] – Calculates the net present value for a given interest rate and a list of future payments.

4.72 HOUR (FUNCTION)

Hour returns the current date and time. Zero seconds and minutes are returned.

Syntax

Hour ()

This function requires no arguments.

Remarks

Hour is calculated when the document is opened and once per hour thereafter.

See also

- Date Functions [p. 28]
- **HourOf** [p. 113] – Extracts hours from a date.
- **Second** [p. 179] – Returns the current date and time every second.
- **Minute** [p. 145] – Returns the current date and time every minute.
- **Now** [p. 152] – Returns the current date and time.
- **Today** [p. 214] – Returns the current date.

4.73 HOUROF (FUNCTION)

HourOf extracts the hours expression from a date.

Syntax

HourOf (*Date*)

Date is a ▸ date of which the hour is calculated.

Examples

- If the current time is 9:38:53 on 2/22/1732,
  `HourOf (Now) = 9`

See also

▸ Date Functions [p. 28]
▸ **SecondOf** [p. 180] – Extracts seconds from a date.
▸ **MinuteOf** [p. 146] – Extracts minutes from a date.
▸ **DayOf** [p. 86] – Extracts the day from a date.
▸ **MonthOf** [p. 148] – Extracts the month from a date.
▸ **YearOf** [p. 231] – Extracts the year from a date.

4.74 HSEARCH (FUNCTION)

HSearch scans a range by columns for matches of a condition and returns either the count of matches or the StartValue, which is overwritten with NextValue in case of matches.

HSearch searches the top row of the given range.

If **HSearch** is called with two arguments, it returns the count of the cells for which the given condition is true. If it is called with four arguments, it uses the formula given as the fourth argument each time a cell fulfills the given condition, and returns the result in the last of these calculations. The same is true when **HSearch** is called with five arguments. However, each cell is additionally tested if the stop condition, which is given as the fifth argument, is true. If it is true, the current result of the given formula is returned.

Syntax

HSearch (*Range*; *Condition*)

Range is a ▸ range which the function is to search.

Condition is a condition for which each cell in *Range* is tested.

Syntax

HSearch (*Range*; *Condition*; *StartValue*; *NextValue*; *StopCondition*)

Range is a ▸ range which the function is to search.

Condition is a condition for which each cell in *Range* is tested.

StartValue is the initial value for calculating *NextValue*. *StartValue* is the value of **CurrentResult** (refer to that function), before **Search** begins searching *Range*.

NextValue is a formula which is used each time a cell in *Range* fulfills the condition *Condition*.

StopCondition is a condition for which each cell is tested. If it is true, *NextValue* is calculated, **Search** is interrupted and the result of *NextValue* is returned. If *StopCondition* is not given, **Search** always searches all cells in *Range*.

Examples

- A spreadsheet contains the following values:

	A	B
1	1	5
2	2	6
3	3	7
4	4	8

- `HSearch (A1:B4; IsOdd(CurrentCell)) = 2`

- `HSearch (A1:B4; IsOdd(CurrentCell); 0; CurrentResult + CurrentCell) = 6`
- `HSearch (A1:B4; IsOdd(CurrentCell); 0; CurrentResult + CurrentCell; CurrentCell>5) = 6`
- `HSearch (A1:B4; IsOdd(CurrentCell); 0; CurrentResult + RowValue(3); CurrentCell>5) = 10`

Remarks

For a meaningful use of **HSearch**, the functions **CurrentIndex**, **CurrentResult**, **CurrentCount** and **CurrentCell** are necessary.

See also

- Spreadsheet Functions [p. 34]
- Search Functions [p. 33]
- **Search** [p. 177] – Scans a range for matches of a condition and returns either the count of matches or the StartValue, which is replaced by NextValue in case of matches.
- **VSearch** [p. 225] – Scans a range by rows for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.
- **RowValue** [p. 176] – Returns the value from that row, to be used in HSearch.
- **CurrentCell** [p. 81] – Returns the current cell in the Search function.
- **CurrentIndex** [p. 83] – Returns the current index in the Search function.
- **CurrentResult** [p. 84] – Returns the current result in the Search function.
- **CurrentCount** [p. 82] – Returns the number of values already accepted in the Search function.

4.75 IF (FUNCTION)

`If` returns the second argument when the condition is true; otherwise, If returns the third argument.

Syntax

`If` (*`Condition`*; ***`ValueIfTrue`***; *`ValueIfFalse`*)

`Condition` is a ▸ truth value which the function evaluates.

`ValueIfTrue` is a ▸ value which is the result of the function when *`Condition`* is true.

`ValueIfFalse` is a ▸ value which is the result of the function when *`Condition`* is false.

Examples

- `If (1=0;'One equals zero';'One is not zero') = One is not zero`

See also ▸ Miscellaneous Functions [p. 32]

▸ **`SpecialIf`** [p. 197] – Returns second argument or cell value, depending on whether the first argument is true.

4.76 INDEX (FUNCTION)

Index returns the value of a particular cell in a range.

Syntax

Index (***Range***; ***Position***)

Range is a ▸ range within which the ▸ range numbering [p. 235] is valid.

Position is a ▸ number specifying that the nth cell of the range to be returned.

Syntax

Index (***Range***; ***RowInRange***; ***ColumnInRange***; *PlaneInRange*)

Range is a ▸ range within which the range numbering is valid.

RowInRange is a ▸ number specifying the index of a row within the range.

ColumnInRange is a ▸ number specifying the index of a column within the range.

PlaneInRange is a ▸ number specifying the index of a plane within the range. If *PlaneInRange* is not given, **Index** searches in the topmost plane.

Examples

- A spreadsheet with two planes has the following contents:

Plane 1:

	A	B
1	1	5
2	2	6
3	3	7
4	4	8

Plane 2:

	A	B
1	9	13
2	10	14
3	11	15
4	12	16

- `Index (A1:B4;2) = 2`
- `Index (A1:B4;3;1) = 3`
- `Index ([1]A1:[2]B4;3;1;2) = 11`

See also

▸ Spreadsheet Functions [p. 34]

▸ Search Functions [p. 33]

▸ **Choose** [p. 57] – Choose any of several values or ranges.

▸ **LookUp** [p. 136] – Looks for a value in a table and extracts a corresponding value from another table.

▸ **Search** [p. 177] – Scans a range for matches of a condition and returns either the count of matches or the StartValue, which is replaced by NextValue in case of matches.

▸ **`Selection`** [p. 181] – Looks for the nth line in a table that matches a condition and returns the index of that line.

4.77 INT (FUNCTION)

Int rounds a number down to the largest integer.

Syntax

Int (*Number*)

Number is a ▸ number which is to be rounded down.

Examples

- `Int (4.9) = 4`
- `Int (-4.9) = -5`

Remarks

⇢ Every number is the sum of its integer part and its fractional part: `number = int (number) + frac (number)`.

See also

▸ Arithmetic Functions [p. 26]
▸ Rounding Functions [p. 33]
▸ **Round** [p. 174] – Rounds a number to a specific number of places.
▸ **Trunc** [p. 217] – Truncates a number to any number of digits.
▸ **Frac** [p. 110] – Returns the fractional part of a number.
▸ **Floor** [p. 109] – Rounds a number to the next lower whole number or the next lower multiple.
▸ **Ceiling** [p. 54] – Rounds a number to the next higher whole number or the next higher multiple.

4.78 ISBLANK (FUNCTION)

IsBlank tests if a value is empty or an empty text and returns the result as a ▸ truth value.

Syntax

IsBlank (*Value*)

Value is a ▸ value which is tested, if it is empty.

Examples

- `IsBlank (0) = 0`
- `IsBlank ('') = 1`
- `IsBlank (' ') = 0`
- `IsBlank (A2) = 1` (if the cell A2 contains nothing)

Remarks

⇢ A text which consists only one blank space is not empty (third example, above).

See also

▸ Status Functions [p. 35]
▸ **IsNumber** [p. 124] – Returns true if the value is a number.
▸ **IsErr** [p. 121] – Returns true if the value is any error value.
▸ **IsNA** [p. 123] – Returns true if the value is the “NA” error value.
▸ **Type** [p. 218] – Returns a code according to the type of the value.

4.79 ISERR (FUNCTION)

IsErr tests if a value is an ▸ error value and returns the result as a ▸ truth value.

Syntax

IsErr (*Value*)

Value is a ▸ value which is tested if it is an ▸ error value.

Examples

- `IsErr (1/0) = 1`
- `IsErr (23) = 0`

See also

▸ Status Functions [p. 35]

▸ **Error** [p. 101] – Returns an error according to an error code.

▸ **IsNumber** [p. 124] – Returns true if the value is a number.

▸ **IsBlank** [p. 120] – Returns true if the value is empty or an empty string.

▸ **IsNA** [p. 123] – Returns true if the value is the “NA” error value.

▸ **Type** [p. 218] – Returns a code according to the type of the value.

4.80 ISEVEN (FUNCTION)

IsEven tests if a number is even and returns the result as a ▸ truth value.

Syntax

IsEven (*Number*)

Number is a ▸ number which is to be tested.

Examples

- `IsEven (1) = 1`
- `IsEven (2) = 0`

See also

▸ Arithmetic Functions [p. 26]
▸ **IsOdd** [p. 125] – Returns True if the number is odd.

4.81 IsNA (Function)

IsNA tests if the value is the error value the ▸ error value ▸ NA! and returns the result as a ▸ truth value.

Syntax

IsNA (*Value*)

Value is a ▸ value which is tested if it is **NV!**.

Examples

- IsNA (NA) = 1
- IsNA (1/0) = 0

See also

▸ Status Functions [p. 35]

▸ **NA** [p. 149] – Returns the error value NA!.

▸ **IsNumber** [p. 124] – Returns true if the value is a number.

▸ **IsBlank** [p. 120] – Returns true if the value is empty or an empty string.

▸ **IsErr** [p. 121] – Returns true if the value is any error value.

▸ **Type** [p. 218] – Returns a code according to the type of the value.

4.82 ISNUMBER (FUNCTION)

IsNumber tests if a value is a ▸ number and returns the result as a ▸ truth value.

Syntax

IsNumber (*Value*)

Value is a ▸ value which is tested if it is a ▸ number.

Examples

- `IsNumber (3.14) = 1`
- `IsNumber ('A') = 0`
- `IsNumber (1/0) = 0`

See also

▸ Arithmetic Functions [p. 26]

▸ Status Functions [p. 35]

▸ **IsBlank** [p. 120] – Returns true if the value is empty or an empty string.

▸ **IsErr** [p. 121] – Returns true if the value is any error value.

▸ **IsNA** [p. 123] – Returns true if the value is the “NA” error value.

▸ **Type** [p. 218] – Returns a code according to the type of the value.

4.83 IsOdd (Function)

IsOdd tests if a number is odd and returns the result as a ▸ truth value.

Syntax

IsOdd (*Number*)

Number is a ▸ number which is to be tested.

Examples

- `IsOdd (1) = 1`
- `IsOdd (2) = 0`

See also ▸ Arithmetic Functions [p. 26]

▸ **IsEven** [p. 122] – Returns True if the number is even.

4.84 LARGE (FUNCTION)

Large returns the k-th largest of the numbers in the range.

Syntax

Large (***Range***; ***k***)

Range is a ▸ range specifying the range containing the numbers to be checked.

k is a ▸ number specifying the k-th largest value to be returned. (If *k* is 4, for example, the fourth-largest value is returned.)

Examples

- If the range A1:A4 contains the values 8, 1, 4 and 6 :

```
Large (A1:A4;3) = 6
Large (A1:A4;2) = 4
```

Remarks

Values in *Range* which are not numbers are ignored.

See also

▸ Search Functions [p. 33]
▸ **Min** [p. 144] – Returns the smallest number in a list.
▸ **Max** [p. 141] – Returns the largest number in a list.
▸ **Small** [p. 195] – Returns the k–th smallest of the numbers range.

4.85 LEFT (FUNCTION)

Left extracts the leftmost n characters of a text.

Syntax

Left (*Text*; *CharacterCount*)

Text is a ▸ text from which characters are to be extracted.

CharacterCount is a ▸ number specifying the number of characters which are to be extracted.

Examples

- `Left ('Hello';3) = Hel`
- `Left ('Hello';6) = Hello`

Remarks

⇢ Decimal fractions, if any, in *CharacterCount* are ignored.

See also

▸ Text Functions [p. 36]

▸ **Length** [p. 128] – Returns the length of a text in characters.

▸ **Right** [p. 173] – Extracts the rightmost n characters of a text.

▸ **Mid** [p. 143] – Extracts characters from any position of a text.

▸ **Find** [p. 108] – Returns the position of a specified text within a target text.

4.86 LENGTH (FUNCTION)

Length returns the ▸ number of characters in a text.

Syntax

Length (***Text***)

Text is a ▸ text of which the characters are to be counted.

Examples

- `Length ('Hello') = 5`

See also

▸ Text Functions [p. 36]

▸ **Left** [p. 127] – Extracts the leftmost n characters of a text.

▸ **Right** [p. 173] – Extracts the rightmost n characters of a text.

▸ **Mid** [p. 143] – Extracts characters from any position of a text.

▸ **Find** [p. 108] – Returns the position of a specified text within a target text.

4.87 Ln (Function)

Ln calculates the natural logarithm of a number. The base of natural logarithms is `e = 2.71828182845904524...`.

When e is raised by the natural logarithm of a number, the result is that number.

Syntax

Ln (*Number*)

Number is a ▸ number › 0 of which the natural logarithm is to be calculated.

Examples

- `Ln (1) = 0`
- `Ln (5) = 1.6094...`

Warnings

⚠ The ▸ error value ▸ NUM! is returned if *Number* <= 0. (The logarithm is not defined in these cases.)

See also

- ▸ Arithmetic Functions [p. 26]
- ▸ **Exp** [p. 104] – Calculates e raised to a specific power.
- ▸ **Log2** [p. 133] – Calculates the base 2 logarithm of a number.
- ▸ **Log10** [p. 132] – Calculates the base 10 logarithm of a number.
- ▸ **Log** [p. 131] – Calculates the logarithm of a number to any base.
- ▸ **Ln1** [p. 130] – Calculates the natural logarithm of a (number + 1).

4.88 LN1 (FUNCTION)

Ln1 calculates the natural logarithm of (number + 1). The base of the natural logarithm is e = 2.71828182845904524….

Syntax

Ln1 (*Number*)

Number is a ▸ number › -1, raised by 1, of which the natural logarithm is to be raised.

Examples

- Ln1 (1) = 0.69314…
- Ln1 (5) = 1.79175…

Warnings

⚠ The ▸ error value ▸ NUM! is returned if *Number* <= -1. (The logarithm is not defined for these cases.)

See also

- ▸ Arithmetic Functions [p. 26]
- ▸ **Exp1** [p. 105] – Calculates (e raised to a specific power) - 1.
- ▸ **Ln** [p. 129] – Calculates the natural logarithm of a number.
- ▸ **Log2** [p. 133] – Calculates the base 2 logarithm of a number.
- ▸ **Log10** [p. 132] – Calculates the base 10 logarithm of a number.
- ▸ **Log** [p. 131] – Calculates the logarithm of a number to any base.

4.89 LOG (FUNCTION)

Log calculates the logarithm of a number to any base.

Syntax

Log (*Number*; *Base*)

Number is a ▸ number of which the base 10 logarithm is to be calculated.

Base is a ▸ number which specifies the base with which the logarithm is to be calculated.

Examples

- `Log (1; 3) = 0`
- `Log (5; 3) = 1.4649735`

Warnings

⚠ The ▸ error value ▸ `NUM!` is returned if *Number* <= 0. (The logarithm is not defined in these cases.)

See also

- ▸ Arithmetic Functions [p. 26]
- ▸ **Ln** [p. 129] – Calculates the natural logarithm of a number.
- ▸ **Log2** [p. 133] – Calculates the base 2 logarithm of a number.
- ▸ **Log10** [p. 132] – Calculates the base 10 logarithm of a number.
- ▸ **Ln1** [p. 130] – Calculates the natural logarithm of a (number + 1).

4.90 LOG10 (FUNCTION)

Log10 calculates the base 10 logarithm of a number.

When 10 is raised to the base 10 logarithm of a number, the result is that number.

Syntax

Log10 (*Number*)

Number is a ▸ number of which the base 10 logarithm is to be calculated.

Examples

- `Log10 (1) = 0`
- `Log10 (5) = 0.69897`

Warnings

⚠ The ▸ error value ▸ NUM! is returned if *Number* <= 0. (The logarithm is not defined in these cases.)

See also

- ▸ Arithmetic Functions [p. 26]
- ▸ **Ln** [p. 129] – Calculates the natural logarithm of a number.
- ▸ **Log2** [p. 133] – Calculates the base 2 logarithm of a number.
- ▸ **Log** [p. 131] – Calculates the logarithm of a number to any base.
- ▸ **Ln1** [p. 130] – Calculates the natural logarithm of a (number + 1).

4.91 LOG2 (FUNCTION)

Log2 calculates the base 2 logarithm of a number.

When 2 is raised to the base 2 logarithm of a number, the result is that number.

Syntax

Log2 (*Number*)

Number is a ▸ number › 0 of which the base 2 logarithm is to be calculated.

Examples

- `Log2 (1) = 0`
- `Log2 (5) = 2.3219…`

Warnings

⚠ The ▸ error value ▸ `NUM!` is returned if *Number* <= 0. (The logarithm is not defined in these cases.)

See also

▸ Arithmetic Functions [p. 26]
▸ **Ln** [p. 129] – Calculates the natural logarithm of a number.
▸ **Log10** [p. 132] – Calculates the base 10 logarithm of a number.
▸ **Log** [p. 131] – Calculates the logarithm of a number to any base.
▸ **Ln1** [p. 130] – Calculates the natural logarithm of a (number + 1).

4.92 LOGREGRESSB (FUNCTION)

LogRegressB calculates the y intercept of a best-fit exponential curve. For details, please refer to ▸ Regression Analysis [p. 234].

Syntax

LogRegressB (*RangeOfDependents*; *RangeOfIndependents*)

RangeOfDependents is a ▸ range containing the y values of coordinate pairs.

RangeOfIndependents is a ▸ range containing the x values of coordinate pairs.

Examples

- The x values of the coordinate pairs are in cells A1:A8 and the y values are in cells B1:B8:

```
LogRegressB (A1:A8; B1:B8) = 5.88
```

Warnings

⚠ All ranges must have the same number of cells. If this is not the case, the ▸ error value ▸ REF! is returned. (However, the ranges may be formed differently. For example, one range may be 2 cells by 4 and a second 4 cells by 2.)

See also ▸ Statistical Functions [p. 35]

▸ **RegressionB** [p. 169] – Calculates the y–intercept of a best-fit straight line.

▸ **RegressionM** [p. 170] – Calculates the slope of a best-fit straight line.

▸ **LogRegressM** [p. 135] – Calculates the increase of a best-fit exponential curve.

4.93 LOGREGRESSM (FUNCTION)

LogRegressM calculates the increase of a best-fit exponential curve.
For details, please refer to ▸ Regression Analysis [p. 234].

Syntax

LogRegressM (*RangeOfDependents*; *RangeOfIndependents*)

RangeOfDependents is a ▸ range containing the y values of coordinate pairs.

RangeOfIndependents is a ▸ range containing the x values of coordinate pairs.

Examples

- The x values of the coordinate pairs are in cells A1:A8 and the y values are in cells B1:B8:

```
LogRegressM (A1:A8; B1:B8) = 0.713
```

Warnings

⚠ All ranges must have the same number of cells. If this is not the case, the ▸ error value ▸ REF! is returned. (However, the ranges may be formed differently. For example, one range may be 2 cells by 4 and a second 4 cells by 2.)

See also ▸ Statistical Functions [p. 35]
▸ **RegressionB** [p. 169] – Calculates the y–intercept of a best-fit straight line.
▸ **RegressionM** [p. 170] – Calculates the slope of a best-fit straight line.
▸ **LogRegressB** [p. 134] – Calculates the y–intercept of a best-fit exponential curve.

4.94 LOOKUP (FUNCTION)

LookUp looks for a value in a range and returns a corresponding value from another range.

Syntax

LookUp (*SearchValue*; *SearchRange*; *LookUpRange*; *ExactNumbers*))

SearchValue is a ▸ value which is looked for in *SearchRange*.

SearchRange is a ▸ range in which *SearchValue* is searched for. Cells are searched in the order of the ▸ cell numbering [p. 235]. Numbers, time spans and dates must be sorted in ascending order. If the value searched for does not occur, the next smaller value is returned. Texts may be unsorted. If the text searched for is not found, NA! is returned.

LookUpRange is a ▸ range from which the result is taken. The result is the contents of the cell of which the ▸ sequential number [p. 235] is the same as the cell in which the search value was found.
If *LookUpRange* is not given, the result is taken from the last column of the *SearchRange* if this range has more rows than columns. Otherwise, the result is taken from the last row.

ExactNumbers is a ▸ truth value. If *ExactNumbers* is TRUE, numbers, time spans and dates in *SearchRange* may be unsorted, but will be found only if they match *SearchValue* exactly.

Examples

- A spreadsheet with rebates contains the number of units in the first row and and the effective rebate for that number in the second.

	A	B	C	D	E	F
1	1	10	50	100	500	1000
2	0%	2%	5%	10%	15%	25%

`LookUp (50; A1:F1; A2:F2)` returns 5% as rebate for 50 units.
`LookUp (94; A1:F1; A2:F2)` as rebate for more than 50 but fewer than 100 units.
`LookUp (4711; A1:F1; A2:F2)` returns 25% for 1000 or more units because 1000 is the last entry in *SearchRange*.

- A table of 104 chemical elements contains the symbols for the elements in the first row and the corresponding atomic number in increasing order in the second.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si
2	1	2	3	4	5	6	7	8	9	10	11	12	13	14

`LookUp ('Na'; A1:CZ1; A2:CZ2)` returns 11 as the atomic number of sodium with the symbol Na.

`LookUp ('Am'; A1:CZ1; A2:CZ2)` returns 95 as the atomic number of Americium with the symbol Am.
`LookUp ('XX'; A1:CZ1; A2:CZ2)` returns the ▸ error value ▸ `NA!` because no element has the symbol XX.
`LookUp (6; A2:CZ2; A1:CZ1)` the symbol C for carbon with the atomic number 6.
`LookUp (99; A2:CZ2; A1:CZ1)` returns the symbol Es for Einsteinium atomic number 99.

See also
- ▸ Spreadsheet Functions [p. 34]
- ▸ Search Functions [p. 33]
- ▸ **`Index`** [p. 117] – Extracts any value from a table.
- ▸ **`Search`** [p. 177] – Scans a range for matches of a condition and returns either the count of matches or the StartValue, which is replaced by NextValue in case of matches.
- ▸ **`Selection`** [p. 181] – Looks for the nth line in a table that matches a condition and returns the index of that line.

4.95 LOWER (FUNCTION)

Lower converts all letters in a text to lower case.

Syntax

Lower (*Text*)

Text is a ▸ text in which the letters are to be converted.

Examples

- `Lower ('Hello') = hello`

See also

- ▸ Text Functions [p. 36]
- ▸ **Clean** [p. 63] – Removes non-printable characters from a text.
- ▸ **Trim** [p. 215] – Removes unnecessary space characters from a text.
- ▸ **Upper** [p. 221] – Returns text with all letters changed to upper case.
- ▸ **Proper** [p. 165] – Returns text with all initials changed to upper case.
- ▸ **Exact** [p. 103] – Returns true if two texts match exactly.

4.96 MAILMERGE (FUNCTION)

MailMerge extracts a value from a range for serial printings.

Syntax

MailMerge (*Range*; *ColumnInRange*; *MatchingCondition*)

Range is a ▸ range from which data are taken.

ColumnInRange is a ▸ number specifying from which column the result is to be taken. If *ColumnInRange* is not given, the value is taken from the left column.

MatchingCondition is a condition which is tested for the first cell of each row in *Range*. **MailMerge** returns a value for this row only if the condition is true. If the condition is false, the row is skipped.

Examples

It is assumed for all examples that cells A1:C4 contain the following values: Thomas, Moore, Boston; Ellen, Smith, Flagstone; Martha, Miller, Kokomo; William, Stone, Milwaukee.

- `MailMerge (A1:D3) = Thomas` (in the first letter)
  `MailMerge (A1:D3) = Ellen` (in the second letter)
  `MailMerge (A1:D3) = Martha` (in the third letter)
  `MailMerge (A1:D3) = William` (in the fourth letter)
- `MailMerge (A1:D3;2) = Moore` (in the first letter)
  `MailMerge (A1:D3;2) = Smith` (in the second letter)
  `MailMerge (A1:D3;2) = Miller` (in the third letter)
  `MailMerge (A1:D3;2) = Stone` (in the fourth letter)
- `MailMerge (A1:D3;CurrentCell < 'S') = Ellen` (in the first letter)
  `MailMerge (A1:D3;CurrentCell < 'S') = Martha` (in the second letter)
- `MailMerge (A1:D3;3;CurrentCell < 'S') = Flagstone` (in the first letter)
  `MailMerge (A1:D3;3;CurrentCell < 'S') = Kokomo` (in the second letter)

Remarks

⇢ For testing the first cell of each row of the range, the function **CurrentCell** can be used. To determine which serial letter is currently being printed, the function **PrintCycle** can be used.

⇢ because *MatchingCondition* is not evaluated until printing begins, the screen display may differ from the content of the first letter.

See also

- Spreadsheet Functions [p. 34]
- Print Functions [p. 32]
- Search Functions [p. 33]
- `PrintCycle` [p. 163] – Returns the sequential number of serial printings.
- `PrintStop` [p. 164] – Sets a condition to stop serial printing.
- `ColumnValue` [p. 74] – Returns the value from that column, to be used in VSearch.
- `CurrentCell` [p. 81] – Returns the current cell in the Search function.

4.97 MAX (FUNCTION)

Max returns the largest value in a list.

Syntax

Max (*List*)

List is a ▸ list from which the greatest value is returned.

Examples

- `Max (1;2;4) = 4`
- `Max (1;-2;'A') = 1`

See also

▸ Search Functions [p. 33]
▸ **Min** [p. 144] – Returns the smallest number in a list.
▸ **Small** [p. 195] – Returns the k–th smallest of the numbers range.
▸ **Large** [p. 126] – Returns the k–th largest of the numbers in the range.

4.98 MEDIAN (FUNCTION)

Median returns the median of the numbers in the list.

The median is the number that is smaller than half of the remaining numbers and larger than the other half. When numbers are arranged by size, the median lies exactly in the middle.

When the median of a list with an even number of elements is to be determined, **Median** returns the arithmetic mean of the two values lying in the middle.

Syntax

Median (*List*)

List is a ▸ list containing the numbers of which the median is to be determined.

Examples

- If the range A1:A5 contains the values 8, 1, 4, 7 and 6:
  `Median (A1:A5) = 6`
- If the range A1:A4 contains the values 8, 1, 4, and 7:
  `Median (A1:A5) = 5.5`

Remarks

⇢ Values in *Range* which are not numbers are ignored.

See also

▸ Statistical Functions [p. 35]

▸ **Percentile** [p. 158] – Returns any percentile of the numbers in the range.

▸ **Quartile** [p. 166] – Returns any quartile of the numbers in the range (quartile = 1 or 3).

4.99 MID (FUNCTION)

Mid extracts ▸ text of any length from any position in a text.

Syntax

Mid (*Text*; *StartPos*; *CharacterCount*)

Text is a ▸ text from which characters are to be extracted.

StartPos is a ▸ number specifying the position from which to begin extracting characters from *Text*.

CharacterCount is a ▸ number specifying the number of characters which are to be extracted.

Examples

- `Mid ('Hello';3;2) = ll`
- `Mid ('Hello';1;5) = Hello`
- `Mid ('Hello';3:5) = llo`

Remarks

⇢ Decimal fractions, if any, in *StartPos* or*CharacterCount* are ignored.

Warnings

⚠ **Mid** returns an empty Text ("") if *StartPos* › Length(*Text*).

⚠ **Mid** returns as much text as possible if *StartPos* + *CharacterCount* › Length(*Text*).

⚠ The ▸ error value ▸ RANGE! is returned if *StartPos* ‹ 1.

⚠ The ▸ error value ▸ RANGE! is returned if *CharacterCount* ‹ 0.

See also

▸ Text Functions [p. 36]

▸ **Length** [p. 128] – Returns the length of a text in characters.

▸ **Left** [p. 127] – Extracts the leftmost n characters of a text.

▸ **Right** [p. 173] – Extracts the rightmost n characters of a text.

▸ **Find** [p. 108] – Returns the position of a specified text within a target text.

4.100 MIN (FUNCTION)

Min returns the smallest value in a list.

Syntax

Min (*List*)

List is a ▸ list from which the smallest value is returned.

Examples

- `Min (1;2;4) = 1`
- `Min (1;-2;'A') = -2`

See also

▸ Search Functions [p. 33]
▸ **Max** [p. 141] – Returns the largest number in a list.
▸ **Small** [p. 195] – Returns the k–th smallest of the numbers range.
▸ **Large** [p. 126] – Returns the k–th largest of the numbers in the range.

4.101 MINUTE (FUNCTION)

Minute returns the current date and time. Zero seconds are returned.

Syntax

Minute ()

This function requires no arguments.

Remarks

Minute is calculated when the document is opened and once per minute thereafter.

See also

- Date Functions [p. 28]
- **MinuteOf** [p. 146] – Extracts minutes from a date.
- **Second** [p. 179] – Returns the current date and time every second.
- **Hour** [p. 112] – Returns the current date and time every hour.
- **Now** [p. 152] – Returns the current date and time.
- **Today** [p. 214] – Returns the current date.

4.102 MINUTEOF (FUNCTION)

MinuteOf extracts the minutes expression from a date.

Syntax

MinuteOf (*Date*)

Date is a ▸ date of which the minute is calculated.

Examples

- If the current time is 9:38:53 on 2/22/1732,
  `MinuteOf (Now) = 38`

See also

- ▸ Date Functions [p. 28]
- ▸ **SecondOf** [p. 180] – Extracts seconds from a date.
- ▸ **HourOf** [p. 113] – Extracts hours from a date.
- ▸ **DayOf** [p. 86] – Extracts the day from a date.
- ▸ **MonthOf** [p. 148] – Extracts the month from a date.
- ▸ **YearOf** [p. 231] – Extracts the year from a date.

4.103 MOD (FUNCTION)

Mod returns the remainder from division of one ▸ number by another.

Syntax

Mod (*X*; *Y*)

X is a ▸ number which is divided.

Y is a ▸ number by which is divided.

Examples

- `Mod (13;3) = 1`
- `Mod (12;3) = 0`
- `Mod (13;5) = 3`

See also ▸ Arithmetic Functions [p. 26]
▸ Operators [p. 31]

4.104 MONTHOF (FUNCTION)

MonthOf extracts the month from a date.

Syntax

MonthOf (*Date*)

Date is a ▸ date of which the month is calculated.

Examples

- If the current time is 9:38:53 on 2/22/1732,
  `MonthOf (Now) = 2`

See also

▸ Date Functions [p. 28]
▸ **SecondOf** [p. 180] – Extracts seconds from a date.
▸ **MinuteOf** [p. 146] – Extracts minutes from a date.
▸ **HourOf** [p. 113] – Extracts hours from a date.
▸ **DayOf** [p. 86] – Extracts the day from a date.
▸ **YearOf** [p. 231] – Extracts the year from a date.

4.105 NA (FUNCTION)

NA returns the constant the ▸ error value ▸ `NA!`.

Syntax

NA ()

This function requires no arguments.

See also

▸ Constants [p. 27]

▸ **`Error`** [p. 101] – Returns an error according to an error code.

▸ **`ErrorType`** [p. 102] – Returns a code according to the error type of the value.

▸ **`IsNA`** [p. 123] – Returns true if the value is the “NA” error value.

4.106 NOOFPAGES (FUNCTION)

NoOfPages returns the number of pages of a layout.

Syntax

NoOfPages (*LayoutName*)

LayoutName is a ▸ text specifying the name of a layout. The function returns the number of pages in that layout. If no layout is given, the number of pages in the layout in which the formula occurs is returned.

Examples

- `NoOfPages () = 12`
- `NoOfPages ('Layout 2') = 25`

Remarks

⇢ The component containing the function need not appear in the layout of which the pages are to be counted.

Warnings

⚠ If the formula is in a component which is not installed in any layout, **NoOfPages** returns the ▸ error value ▸ `NA!`.

See also

▸ Miscellaneous Functions [p. 32]

▸ **Page** [p. 156] – Returns the number of the page containing the formula.

▸ **PageIndex** [p. 157] – Returns the index of the page containing the formula.

▸ **StartingPageNumber** [p. 201] – Returns the starting page number of the layout.

▸ **EndingPageNumber** [p. 100] – Returns the ending page number of the layout.

▸ **Container** [p. 78] – Returns the name of the innermost container containing the formula.

4.107 NOT (FUNCTION)

Not converts True to False and vice versa.

Syntax

Not (*Condition*)

Condition is converted to its logical opposite.

Examples

- `Not (True) = 0`
- `Not (False) = 1`

Remarks

The ▸ truth value True is represented with the number 1; False, with the number 0.

See also

▸ Operators [p. 31]

▸ **And** [p. 47] – Returns true if all values in a list are true.

▸ **Or** [p. 155] – Returns true if any value in a list is true.

4.108 NOW (FUNCTION)

Now returns the current ▸ date and time of your computer.

Syntax

Now ()

This function requires no arguments.

Examples

- `Now = 4/9/1 1999 23:45:10`

Remarks

Now is recalculated each time anything is calculated anywhere.

See also

- ▸ Date Functions [p. 28]
- ▸ **Second** [p. 179] – Returns the current date and time every second.
- ▸ **Minute** [p. 145] – Returns the current date and time every minute.
- ▸ **Hour** [p. 112] – Returns the current date and time every hour.
- ▸ **Today** [p. 214] – Returns the current date.

4.109 NPV (FUNCTION)

NPV calculates the net present value of a list of payments with interest accumulating from the first payment date.

Syntax

NPV (*InterestRate*; *PaymentList*)

InterestRate is a ▸ number specifying the interest rate.

PaymentList is a ▸ list providing the payments in individual payment periods.

Examples

- `NPV (0.05; 100; 100; 100) = 272.32` (Three annual payments of $100 each at 5% interest result in a present value of $272.32)
- `NPV (0.065; A3:A8) = 4310.87`

See also ▸ Financial Functions [p. 31]

▸ **FV** [p. 111] – Calculates future value for a given interest rate and a list of payments.

4.110 NUMBER (FUNCTION)

Number converts text to a number or a date in the number of days since January 1, 1904.

Syntax

Number (*Text*)

Text is a ▸ text which is to be converted.

Syntax

Number (*Date*)

Date is the ▸ date, of which the difference, in days, from January 1, 1904 should be calculated. If the *Date* is before January 1, 1904, the number returned is negative.

⇢ Times in *Date*, if any, will be considered in the fractional portion of the result.

Examples

- `Number ('12') = 12`
  `Number (Mid (Text (12345); 3; 2)) = 34`
- `Number (1/1/1980) = 27759`
  `Number (06/10/1953 9:38:53) = 18058.402002315`

Warnings

⚠ If the argument cannot be converted to a number, **Number** returns the ▸ error value ▸ `VALUE!`.

See also

▸ Arithmetic Functions [p. 26]
▸ Text Functions [p. 36]
▸ Conversion Functions [p. 27]
▸ **Value** [p. 222] – Converts text to a value, as if entered into a spreadsheet cell.
▸ **Text** [p. 211] – Converts a value to text.
▸ **Date** [p. 85] – Converts text or a specified number of days from January 1, 1904 to a date.
▸ **TimeSpan** [p. 213] – Converts text to a time span.

4.111 OR (FUNCTION)

Or returns true if at least one value in a list is true.

Syntax

Or (*List*)

List is a ▸ list of which the individual elements are evaluated. If at least one element is true, then the result of the function is true.

Examples

- `Or (1;2;1) = 1`
- `Or (1;2;0) = 1`
- `Or (0;0;0) = 0`

Remarks

⇢ The ▸ truth value False is represented by the number 0, True by all other numbers.

See also

▸ Operators [p. 31]

▸ **And** [p. 47] – Returns true if all values in a list are true.

▸ **Not** [p. 151] – Negates a logical value.

4.112 PAGE (FUNCTION)

Page returns the number of the page containing the formula.

If the component in which the formula is contained is installed in several layouts, the page numbers may be different. Include a layout name to request a page index in that layout.

Syntax

Page ()

This function requires no arguments.

Syntax

Page (*LayoutName*)

LayoutName is a ▸ text specifying the name of a layout. The function returns the number of the page on which the formula occurs.

Examples

- `Page () = 5`
- `Page ('Layout 2') = 3`

Remarks

Page returns the page's number, which does not necessarily correspond to its position. If the page numbering of a layout begins with 5, the result of **Page** on the first page is also 5. Compare **PageIndex**.

Warnings

If the formula occurs on several pages of the same layout (because the relevant component is installed repeatedly), **Page** returns the ▸ error value ▸ `NA!`.

If the formula is in a component which is not installed in any layout, **Page** returns the ▸ error value ▸ `NA!`.

See also

▸ Miscellaneous Functions [p. 32]

▸ **PageIndex** [p. 157] – Returns the index of the page containing the formula.

▸ **NoOfPages** [p. 150] – Returns the number of pages.

▸ **StartingPageNumber** [p. 201] – Returns the starting page number of the layout.

▸ **EndingPageNumber** [p. 100] – Returns the ending page number of the layout.

▸ **Container** [p. 78] – Returns the name of the innermost container containing the formula.

4.113 PAGEINDEX (FUNCTION)

PageIndex returns the index of the page containing the formula.

If the component in which the formula is contained is installed in several layouts, the page index numbers may be different. Include a layout name to request a page index in that layout.

Syntax

PageIndex ()

This function requires no arguments.

Syntax

PageIndex (*LayoutName*)

LayoutName is a ▸ text specifying the name of a layout. The function returns the sequential number of the page on which the formula occurs in the layout.

Examples

- `PageIndex () = 3`
- `PageIndex ('Layout 2') = 1`

Remarks

In contrast to **Page**, **PageIndex** always returns the position of the page in a layout. If, for example, the formula appears on the first page, **PageIndex** returns the result 1, even if the page numbering of thelayout begins with 5.

Warnings

If the formula occurs on several pages of the same layout (because the relevant component is installed repeatedly), **PageIndex** returns the ▸ error value ▸ NA!.

If the formula is in a component which is not installed in any layout, **PageIndex** returns the ▸ error value ▸ NA!.

See also

▸ Miscellaneous Functions [p. 32]

▸ **Page** [p. 156] – Returns the number of the page containing the formula.

▸ **NoOfPages** [p. 150] – Returns the number of pages.

▸ **StartingPageNumber** [p. 201] – Returns the starting page number of the layout.

▸ **EndingPageNumber** [p. 100] – Returns the ending page number of the layout.

▸ **Container** [p. 78] – Returns the name of the innermost container containing the formula.

4.114 PERCENTILE (FUNCTION)

Percentile returns a percentile of the numbers in the range.

The function returns the ▸ number from the given range which is greater than *Percentile* hundredth and smaller than 100 - *Percentile* hundredth of the numbers in the range. If no number fulfills these conditions, a value between the next largest and the next smallest numbers is returned.

Syntax

Percentile (***Range***; ***Percentile***)

Range is a ▸ range containing the numbers which are to be evaluated.

Percentile is a ▸ number from 0 to 1 which specifies the desired percentile.

Examples

- If the range A1:A5 contains the values 8, 1, 4, 7 and 6:

```
Percentile (A1:A5;0.1) = 2.8
Percentile (A1:A5;0.8) = 7.2
```

Remarks

Percentile returns the smallest number in the range if *Percentile* is 0; the median, if *Percentile* is 0.5; and the greatest number, if *Percentile* is 1.

See also

▸ Statistical Functions [p. 35]

▸ **Median** [p. 142] – Returns the median of the numbers in the list.

▸ **Quartile** [p. 166] – Returns any quartile of the numbers in the range (quartile = 1 or 3).

4.115 PERMUTATIONS (FUNCTION)

Permutations calculates the number of possibilities of m choosing elements from n elements. (The order is relevant.)

Syntax

Permutations (*Elements*; *NumberChosen*)

Elements is a ▸ number specifing how many elements are available.

NumberChosen is a ▸ number specifing how many element are to be chosen.

Examples

- `Permutations (5; 2) = 20`

Remarks

⇢ **Permutations** is defined by the formula `Factorial ((Elements) / Factorial (Elements-NumberChosen)`.

See also

▸ Statistical Functions [p. 35]

▸ **Factorial** [p. 106] – Calculates the factorial of a number.

▸ **Combinations** [p. 75] – Calculates the number of combinations of m elements chosen out of n elements.

4.116 PI (FUNCTION)

Pi returns the ▸ number approximated by the constant π, 3.14159265358979324…

Syntax

Pi ()

This function requires no arguments.

See also

▸ Constants [p. 27]

▸ **Pi180** [p. 161] – Returns the numerical constant π/180.

4.117 PI180 (FUNCTION)

Pi180 returns the ▸ number approximated by the constant π/180, 0.0174532925199432958...

Syntax

Pi180 ()

This function requires no arguments.

Remarks

Unlike other functions without arguments, **Pi180()** must always be written with parentheses. "Pi180" without paraentheses is interpreted as a reference to the cell in column PI and row 180.

See also

▸ Constants [p. 27]

▸ **Pi** [p. 160] – Returns the numerical constant π.

4.118 PLANE (FUNCTION)

Plane returns the plane where the formula is or the plane of the specified cell.

Syntax

Plane ()

This function requires no arguments.

Syntax

Plane (*Cell*)

Cell is a ▸ range. **Plane** returns the plane coordinate of the upper left cell of the range. If *Cell* is not given, **Plane** returns the plane of the cell containing the formula.

Examples

- `Plane () = 1`
- `Plane ([4]E2) = 4`

See also

▸ Spreadsheet Functions [p. 34]

▸ **Row** [p. 175] – Get the row where the formula or the specified cell is.

▸ **Column** [p. 73] – Get the column where the formula or the specified cell is.

4.119 PRINTCYCLE (FUNCTION)

PrintCycle returns the sequential ▸ number of a serial letter while it is being printed.

Syntax

PrintCycle ()

This function requires no arguments.

Remarks

- **PrintCycle** is recalculated every time before printing and the number raised. You can use this function, together with other functions, for example, **LookUp** to have a different result for each printed copy, individualizing the copies.
- If you use **PrintCycle**, a dialog box informs you of this before the actual printing.
- You can limit the number of printed copies by entering a number in the dialog box or by using the funtion **PrintStop**.

See also

- ▸ Print Functions [p. 32]
- ▸ **MailMerge** [p. 139] – Extracts data from tables for serial printings.
- ▸ **PrintStop** [p. 164] – Sets a condition to stop serial printing.

4.120 PRINTSTOP (FUNCTION)

PrintStop sets a condition to stop serial printing which is initiated by **PrintCycle**.

PrintStop returns the ▸ truth value given as an argument.

Syntax

PrintStop (*StopCondition*)

StopCondition is a ▸ truth value. If it is true, the serial printing initiated by **PrintCycle** is stopped.

Examples

- `PrintStop(PrintCycle> 5)` stops serial printing after the fifth copy.
- `PrintStop(True)` stops serial printing immediately.
- `PrintStop(HourOf (Now) ≥ If (DayOfWeek (Today) = 6;14;17))` stops serial printing at 5 p.m. except Fridays, when it stops at 2 p.m.

Remarks

- If you do not use **PrintCycle** in the same document, **PrintStop** has no effect.
- You should use **PrintStop** only once in a document. If the function occurs more than once, serial printing is stopped when the *StopCondition* is fulfilled for any one of the functions.
- If you use **PrintStop** and **MailMerge** in the same document, serial printing is stopped immediately when the stop condition for only one of the functions is fulfilled.

See also

▸ Print Functions [p. 32]
▸ **PrintCycle** [p. 163] – Returns the sequential number of serial printings.
▸ **MailMerge** [p. 139] – Extracts data from tables for serial printings.

4.121 PROPER (FUNCTION)

Proper converts the first letter of each word to upper case and the remaining letters to lower case.

Syntax

Proper (*Text*)

Text is a ▸ text in which the letters are to be converted.

Examples

- `Proper ('to BE or nOt TO bE.') = To be or not to be.`

See also

- ▸ Text Functions [p. 36]
- ▸ **Clean** [p. 63] – Removes non-printable characters from a text.
- ▸ **Trim** [p. 215] – Removes unnecessary space characters from a text.
- ▸ **Lower** [p. 138] – Returns text with all letters changed to lower case.
- ▸ **Upper** [p. 221] – Returns text with all letters changed to upper case.
- ▸ **Exact** [p. 103] – Returns true if two texts match exactly.

4.122 QUARTILE (FUNCTION)

Quartile returns any quartile of the numbers in the range (quartile = 1 or 3).

The function returns the ▸ number from the given range which is greater than *quartile* quarter and smaller than the 4 - *quartile* quarter of the numbers in the range. If no number fulfills these conditions, a value between the next largest and the next smallest numbers is returned.

Syntax

Quartile (*Range*; *quartile*)

Range is a ▸ range containing the numbers which are to be evaluated.

quartile is a ▸ number from 0 to 4 which specifies the desired quartile.

Examples

- If the range A1:A5 contains the values 8, 1, 4, 7 and 6:

```
Quartile (A1:A5;1) = 4
Quartile (A1:A5;3) = 7
```

Remarks

Quartile returns the smallest number in the range if *quartile* is 0; the median, if *quartile* is 2 and the greatest number, if *quartile* is 4.

See also

▸ Statistical Functions [p. 35]

▸ **Median** [p. 142] – Returns the median of the numbers in the list.

▸ **Percentile** [p. 158] – Returns any percentile of the numbers in the range.

4.123 RADIANS (FUNCTION)

Radians converts an angle from degrees to radians.

Radians calculates a ▸ number using `radians = degrees * π / 180`.

Syntax

Radians (***AngleInDegrees***)

AngleInDegrees is a ▸ number which is to be converted.

See also

- ▸ Trigonometric Functions [p. 37]
- ▸ Conversion Functions [p. 27]
- ▸ **Degrees** [p. 90] – Converts an angle from radians to degrees.

4.124 RAND (FUNCTION)

Rand calculates a random number.

The function can be called with no, one or two arguments. With no argument, it returns a ▸ number from the ▸ interval [0; 1[; with one argument, a number from the interval 0 (including 0) to the number, excluding it; and with two arguments, a number from the interval between them, including them.

Syntax

Rand ()

This function requires no arguments.

Syntax

Rand (*Limit*)

Limit is a ▸ number which specifies the upper limit of the interval from which the random number is chosen.

Syntax

Rand (*LowerInt*; *UpperInt*)

LowerInt is a ▸ number which specifies the lower limit of the interval from which the random number is chosen.

UpperInt is a ▸ number which specifies the upper limit of the interval from which the random number is chosen.

Examples

- `Rand () = 0.1256…`
- `Rand () = 0.8302…`
- `Rand (12) = 1.0422…`
- `Rand (12) = 11.7732…`
- `Rand (100; 200) = 193.3022…`
- `Rand (100; 200) = 100`

Remarks

Rand is recalculated each time anything is calculated anywhere.

If **Rand** is called with one or no argument, the result is greater than or equal to zero and less than the upper limit. If the function is called with two arguments, the result is greater than or equal to the lower limit and less than or equal to the upper limit.

See also ▸ Arithmetic Functions [p. 26]
▸ Statistical Functions [p. 35]

4.125 REGRESSIONB (FUNCTION)

RegressionB calculates the y intercept of a best-fit straight line.
For details, please refer to ▸ Regression Analysis [p. 234].

Syntax

RegressionB (*RangeOfDependents*; *RangeOfIndependents*)

RangeOfDependents is a ▸ range containing the y values of coordinate pairs.

RangeOfIndependents is a ▸ range containing the x values of coordinate pairs.

Examples

- The x values of the coordinate pairs are in cells A1:A8 and the y values are in cells B1:B8:

```
RegressionB (A1:A8; B1:B8) = 3.29
```

Warnings

⚠ All ranges must have the same number of cells. If this is not the case, the ▸ error value ▸ REF! is returned. (However, the ranges may be formed differently. For example, one range may be 2 cells by 4 and a second 4 cells by 2.)

See also

▸ Statistical Functions [p. 35]

▸ **RegressionM** [p. 170] – Calculates the slope of a best-fit straight line.

▸ **LogRegressB** [p. 134] – Calculates the y–intercept of a best-fit exponential curve.

▸ **LogRegressM** [p. 135] – Calculates the increase of a best-fit exponential curve.

4.126 REGRESSIONM (FUNCTION)

RegressionM calculates the slope of a best-fit straight line.
For details, please refer to ▸ Regression Analysis [p. 234].

Syntax

RegressionM (*RangeOfDependents*; *RangeOfIndependents*)

RangeOfDependents is a ▸ range containing the y values of coordinate pairs.

RangeOfIndependents is a ▸ range containing the x values of coordinate pairs.

Examples

- The x values of the coordinate pairs are in cells A1:A8 and the y values are in cells B1:B8:

```
RegressionM (A1:A8; B1:B8) = 2.1
```

Warnings

⚠ All ranges must have the same number of cells. If this is not the case, the ▸ error value ▸ REF! is returned. (However, the ranges may be formed differently. For example, one range may be 2 cells by 4 and a second 4 cells by 2.)

See also

▸ Statistical Functions [p. 35]

▸ **RegressionB** [p. 169] – Calculates the y–intercept of a best-fit straight line.

▸ **LogRegressB** [p. 134] – Calculates the y–intercept of a best-fit exponential curve.

▸ **LogRegressM** [p. 135] – Calculates the increase of a best-fit exponential curve.

4.127 REPEAT (FUNCTION)

Repeat returns a ▸ text consisting of repetions of a given text.

Syntax

Repeat (*Text*; *RepeatCount*)

Text is a ▸ text which is to be repeated.

RepeatCount is a ▸ number specifying how often to repeat *Text*.

Examples

- `Repeat ('Hello';3) = HelloHelloHello`
- `Repeat ('-';10) = -------`

Remarks

Decimal fractions in *RepeatCount*, if any, are ignored.

Warnings

⚠ The ▸ error value ▸ RANGE! is returned if *RepeatCount* < 0.

See also

- ▸ Text Functions [p. 36]
- ▸ **Replace** [p. 172] – Replaces specific characters in a text.
- ▸ **Concat** [p. 77] – Concatenates texts.
- ▸ **SmartConcat** [p. 196] – Concatenates texts with delimiter.

4.128 REPLACE (FUNCTION)

Replace replaces part of a text with new text.

Syntax

Replace (*Text*; *StartPos*; *CharacterCount*; *ReplaceText*)

Text is a ▸ text in which characters are to be replaced.

StartPos is a ▸ number specifying the position from which to begin replacing characters from *Text*.

CharacterCount is a ▸ number specifying the number of characters which are to be replaced.

ReplaceText is a ▸ text which replaces the giving range in *Text*.

Examples

- `Replace ('Hello';2;1;'u') = Hullo`
- `Replace ('Hello';10;1;'there') = Hellothere`
- `Replace ('Hello';3;1;'') = Helo`
- `Replace ('Hello';3;2;'there') = Hethereo`

Remarks

⇢ If *CharacterCount* is zero, *ReplaceText* is inserted before the character *StartPos* in *Text*.

⇢ If *StartPos* is greater than the length of the text, *ReplaceText* is appended to the end of *Text*.

See also

▸ Text Functions [p. 36]

▸ **Find** [p. 108] – Returns the position of a specified text within a target text.

▸ **Repeat** [p. 171] – Repeats a given text n times.

▸ **Concat** [p. 77] – Concatenates texts.

▸ **SmartConcat** [p. 196] – Concatenates texts with delimiter.

4.129 RIGHT (FUNCTION)

Right extracts the rightmost n characters of a text.

Syntax

Right (*Text*; *CharacterCount*)

Text is a ▸ text from which characters are to be extracted.

CharacterCount is a ▸ number specifying the number of characters which are to be extracted.

Examples

- `Right ('Hello';3) = llo`
- `Right ('Hello';6) = Hello`

Remarks

⇢ Decimal fractions, if any, in *CharacterCount* are ignored.

See also

▸ Text Functions [p. 36]
▸ **Length** [p. 128] – Returns the length of a text in characters.
▸ **Left** [p. 127] – Extracts the leftmost n characters of a text.
▸ **Mid** [p. 143] – Extracts characters from any position of a text.
▸ **Find** [p. 108] – Returns the position of a specified text within a target text.

4.130 ROUND (FUNCTION)

Round rounds a number the the closest integer or to a specific number of places.

Syntax

Round (*Number*; *DecimalPlaces*)

Number is a ▸ number which is to be rounded.

DecimalPlaces is an optional argument which specifies the desired number of places after the decimal.

Examples

- `Round (4.49) = 4`
- `Round (4.5) = 5`
- `Round (3.14; 1) = 3.1`
- `Round (256; -1) = 260`

Remarks

If *DecimalPlaces* is not given, RagTime rounds to a whole number. If *DecimalPlaces* is a negative value, RagTime rounds to the appropriate place before the decimal. (See the fourth example above.)

See also

▸ Arithmetic Functions [p. 26]

▸ Rounding Functions [p. 33]

▸ **Int** [p. 119] – Rounds a number down to the largest integer.

▸ **Trunc** [p. 217] – Truncates a number to any number of digits.

▸ **Frac** [p. 110] – Returns the fractional part of a number.

▸ **Floor** [p. 109] – Rounds a number to the next lower whole number or the next lower multiple.

▸ **Ceiling** [p. 54] – Rounds a number to the next higher whole number or the next higher multiple.

4.131 ROW (FUNCTION)

Row returns the row in which the formula is or the row of the specified cell.

Syntax

Row (*Cell*)

Cell is a ▸ range. **Row** returns the row coordinate of the upper left cell of the range. If *Cell* is not given, **Row** returns the row of the cell containing the formula.

Examples

- `Row () = 4`
- `Row (E2) = 2`

See also

▸ Spreadsheet Functions [p. 34]

▸ **Column** [p. 73] – Get the column where the formula or the specified cell is.

▸ **Plane** [p. 162] – Get the plane where the formula or the specified cell is.

4.132 ROWVALUE (FUNCTION)

RowValue returns a value from a particular row to be used in HSearch.

The value of a row is the contents of the leftmost cell within the range of the search.

Syntax

RowValue (*Index*; *NestingLevel*)

Index is a ▸ number specifying in which row the value is to be determined. The index is relative to the range which is searched by a search function.

NestingLevel is an optional ▸ number specifying the number of levels up in the hierarchy of nested search functions **RowValue** should be calculated. If *NestingLevel* is not given or equals 0, **RowValue** is calculated for the innermost of the nested functions.

Examples

It is assumed for the example that cells A1:A4 contain the numbers 1 to 4 and cells B1:B4 contain the numbers 5 to 8 and that the range A1:B4 is searched by HSearch.

- `RowValue (2) = 2`

Remarks

RowValue can only be used in an argument in **HSearch**.

See also

▸ Spreadsheet Functions [p. 34]

▸ Search Functions [p. 33]

▸ **HSearch** [p. 114] – Scans a range by columns for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.

4.133 SEARCH (FUNCTION)

Search scans a range for matches of a condition and returns either the count of matches or the StartValue, which is overwritten with NextValue in case of matches.

If **Search** is called with two arguments, it returns the count of the cells for which the given condition is true. If it is called with four arguments, it uses the formula given as the fourth argument each time a cell fulfills the given condition, and returns the result in the last of these calculations. The same is true when **Search** is called with five arguments. However, each cell is additionally tested if the stop condition, which is given as the fifth argument, is true. If it is true, the current result of the given formula is returned.

Syntax

Search (*Range*; *Condition*)

Range is a ▸ range which the function is to search.

Condition is a condition for which each cell in *Range* is tested.

Syntax

Search (*Range*; *Condition*; *StartValue*; *NextValue*; *StopCondition*)

Range is a ▸ range which the function is to search.

Condition is a condition for which each cell in *Range* is tested.

StartValue is the initial value for calculating *NextValue*. *StartValue* is the value of **CurrentResult** (refer to that function), before **Search** begins searching *Range*.

NextValue is a formula which is used each time a cell in *Range* fulfills the condition *Condition*.

StopCondition is a condition for which each cell is tested. If it is true, *NextValue* is calculated, **Search** is interrupted and the result of *NextValue* is returned. If *StopCondition* is not given, **Search** always searches all cells in *Range*.

Examples

- A spreadsheet contains the following values:

	A	B
1	1	5
2	2	6
3	3	7
4	4	8

- `Search (A1:B4; IsEven(CurrentCell)) = 4`
- `Search (A1:B4; IsEven(CurrentCell); 0; CurrentResult + CurrentCell) = 20`

- `Search (A1:B4; IsEven(CurrentCell); 0; CurrentResult + CurrentCell; CurrentCell>5) = 8`

Remarks

⇢ For a meaningful use of **Search**, the functions **CurrentIndex**, **CurrentResult**, **CurrentCount** and **CurrentCell** are necessary.

⇢ *Range* is searched by row from top to bottom.

See also

▸ Spreadsheet Functions [p. 34]
▸ Search Functions [p. 33]
▸ **VSearch** [p. 225] – Scans a range by rows for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.
▸ **HSearch** [p. 114] – Scans a range by columns for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.
▸ **CurrentCell** [p. 81] – Returns the current cell in the Search function.
▸ **CurrentIndex** [p. 83] – Returns the current index in the Search function.
▸ **CurrentResult** [p. 84] – Returns the current result in the Search function.
▸ **CurrentCount** [p. 82] – Returns the number of values already accepted in the Search function.

4.134 SECOND (FUNCTION)

Second returns the current ▸ date and time.

Syntax

Second ()

This function requires no arguments.

Remarks

Second is calculated when the document is opened and once per second thereafter.

See also

- ▸ Date Functions [p. 28]
- ▸ **SecondOf** [p. 180] – Extracts seconds from a date.
- ▸ **Minute** [p. 145] – Returns the current date and time every minute.
- ▸ **Hour** [p. 112] – Returns the current date and time every hour.
- ▸ **Now** [p. 152] – Returns the current date and time.
- ▸ **Today** [p. 214] – Returns the current date.

4.135 SECONDOF (FUNCTION)

SecondOf extracts the seconds expression from a date.

Syntax

SecondOf (*Date*)

Date is a ▸ date of which the second is calculated.

Examples

- If the current time is 9:38:53 on 2/22/1732,

```
SecondOf (Now) = 53
```

See also

- ▸ Date Functions [p. 28]
- ▸ **MinuteOf** [p. 146] – Extracts minutes from a date.
- ▸ **HourOf** [p. 113] – Extracts hours from a date.
- ▸ **DayOf** [p. 86] – Extracts the day from a date.
- ▸ **MonthOf** [p. 148] – Extracts the month from a date.
- ▸ **YearOf** [p. 231] – Extracts the year from a date.

4.136 SELECTION (FUNCTION)

Selection looks for the nth line in a range that matches a condition and returns the index of that line.

Selection searches the range by row. With the aid of the function **CurrentCell**, the first cell of the current row can be tested.

Syntax

Selection (*SearchRange*; *MatchingCondition*; *nthMatch*)

SearchRange is a ▸ range which is to be searched.

MatchingCondition is a condition for which the first cell in each row of *SearchRange* is tested.

nthMatch is a ▸ number specifying that the nth row which fulfills the condition is to be sought.

Examples

It is assumed for all examples that the cells A1:A4 contain the numbers 11 to 14 and the cells B1:B4 , the numbers 15 to 18.

- `Selection (A1:B4; IsEven(CurrentCell); 2) = 4`
- `Selection (A1:B4; IsOdd(ColumnValue(2)); 2) = 3`

Remarks

For a meaningful use of **Selection**, the function **CurrentCell** or **ColumnValue** is necessary.

Warnings

If an insufficient number of rows in the given range fulfill the conditions, for example, when the third odd number is sought in a range that has only two, **Selection** returns the ▸ error value ▸ NA! .

See also

▸ Spreadsheet Functions [p. 34]

▸ Search Functions [p. 33]

▸ **ColumnValue** [p. 74] – Returns the value from that column, to be used in VSearch.

▸ **CurrentCell** [p. 81] – Returns the current cell in the Search function.

▸ **Index** [p. 117] – Extracts any value from a table.

▸ **LookUp** [p. 136] – Looks for a value in a table and extracts a corresponding value from another table.

4.137 SETCELL (FUNCTION)

SetCell writes a value in a given cell or in a particular cell of a range.

Syntax

SetCell (*Value*; *Cell*)
Value is a ▸ value which is to be written in the cell.
Cell is a cell in which *Value* is to be written.

Syntax

SetCell (*Value*; *Range*; *CellInRange*)
Value is a ▸ value which is to be written in the cell.
Range is a ▸ range.
CellInRange is a ▸ number specifying the ▸ number of the cell [p. 235] in *Range*, in which *Value* is to be written.

Syntax

SetCell (*Value*; *Range*; *RowInRange*; *ColumnInRange*)
Value is a ▸ value which is to be written in the cell.
Range is a ▸ range.
RowInRange is a ▸ number. *RowInRange* is the relative row coordinate of the cell within *Range* in which *Value* is to be written.
ColumnInRange is a ▸ number. *ColumnInRange* is the relative column coordinate of the cell within *Range* in which *Value* is to be written.

Examples

All the following examples write the text 'abc' in the cell C5:

- `SetCell ('abc';C5)`
- `SetCell ('abc';A5:D5;3)`
- `SetCell ('abc';A4:D5;7)`
- `SetCell ('abc';A4:D5;2;3)`

Remarks

In the second variation of **SetCell**, the cells of the range are counted by row from left to right. The upper left cell or a range is number 1, the cell to the right is number 2 and so on.

See also ▸ Spreadsheet Functions [p. 34]
▸ **Button** [p. 53] – Installs a button into a spreadsheet cell, performs an action when clicked, and returns it's title.

4.138 SetDate (Function)

SetDate creates a date from numbers.

Syntax

SetDate (*Year*; *Month*; *Day*; *Hour*; *Minute*; *Second*)
Year is a ▸ number specifying the year of the date.
Month is a ▸ number specifying the month of the date.
Day is a ▸ number specifying the day of the date.
Hour is a ▸ number specifying the hour of the date.
Minute is a ▸ number specifying the minute of the date.
Second is a ▸ number specifying the second of the date.

Examples

- `SetDate (1987; 10; 3) = 10/3/1987 0:00:00`
- `SetDate (1987; 10; 3; 7) = 10/3/1987 7:00:00`
- `SetDate (1987; 10; 3; 7; 32; 44) = 10/3/1987 7:32:44`

Remarks

⇢ If *Hour*, *Minute* or *Second* is not given, **SetDate** sets it to zero.

Warnings

⚠ If an argument lies outside the permitted range (for example, 13 for *Month*), **SetDate** returns the ▸ error value ▸ `DATE!`.

See also

▸ Date Functions [p. 28]
▸ **SetTime** [p. 190] – Creates a time from numbers. The date part will be set to Jan 1st, 1904.
▸ **SetTimeSpan** [p. 191] – Creates a time span from numbers.
▸ **SetSecond** [p. 189] – Sets the seconds in a date.
▸ **SetMinute** [p. 187] – Sets the minutes in a date.
▸ **SetHour** [p. 186] – Sets the hours in a date.
▸ **SetDay** [p. 184] – Sets the day in a date.
▸ **SetMonth** [p. 188] – Sets the month in a date.
▸ **SetYear** [p. 192] – Sets the year in a date.

4.139 SETDAY (FUNCTION)

SetDay sets the day in a date.

Syntax

SetDay (*Date*; *Day*)

Date is a ▸ date of which the day is to be set.

Day is a ▸ number which replaces the previous day of a date.

Examples

- If the current time is 9:38:53 on 2/22/1732,

```
SetDay (Now; 10) =  2/10/1732 9:38:53
SetDay (Now; 31) =  2/28/1732 9:38:53
```

Remarks

If a day in a month with fewer than 31 days is set to a value between 31 and the number of days in the month, **SetDay** returns the last day of the month. (See the second example, above.)

Warnings

⚠ If *Day* is 32 or greater, or a negative number, **SetYear** returns the ▸ error value ▸ DATE!.

See also

- ▸ Date Functions [p. 28]
- ▸ **SetDate** [p. 183] – Creates a date from numbers.
- ▸ **SetTime** [p. 190] – Creates a time from numbers. The date part will be set to Jan 1st, 1904.
- ▸ **SetSecond** [p. 189] – Sets the seconds in a date.
- ▸ **SetMinute** [p. 187] – Sets the minutes in a date.
- ▸ **SetHour** [p. 186] – Sets the hours in a date.
- ▸ **SetMonth** [p. 188] – Sets the month in a date.
- ▸ **SetYear** [p. 192] – Sets the year in a date.

4.140 SetDocName (Function)

SetDocName sets the default name of the document for the command "Save As...".

SetDocName returns the ▸ text which was passed as an argument.

Syntax

SetDocName (*DocumentPath*)

DocumentPath is a ▸ text which is to be used as the suggested name. The volume or a particular folder can also be set. Their names must precede the document's name and be separated with colons (:). If names of nonexistent volumes or folders are given, an error message is displayed when you use the command "Save As...".

Examples

- `SetDocName ('Letter to B & E')` sets "Letter to B & E" as the suggested name in the current folder.
- `SetDocName ('Archive:Letter to B & E')` sets "Letter to B & E" as the suggested name in the highest level of the volume "Archive."
- `SetDocName ('Archive:Thank You:Letter to B & E')` sets "Letter to B & E" as the suggested name in the folder "Thank You" of the volume "Archive."

Remarks

⇢ This command has no effect on the command "Save."

⇢ The name and folders can still be changed when "Save As..." is used. **SetDocName** only sets a suggestion.

⇢ Be sure not to use a colon in the name unless you want to specify a volume or a folder.

Warnings

⚠ Different computers permit longer or shorter names for documents and files. Please refer to the handbook for your computer for details.

See also

▸ Miscellaneous Functions [p. 32]
▸ Text Functions [p. 36]
▸ **DocumentName** [p. 99] – Returns the name of the document.
▸ **DocumentDate** [p. 98] – Returns the document's date which was initialized when the document was created.

4.141 SETHOUR (FUNCTION)

SetHour sets the hour in a date.

Syntax

SetHour (*Date*; *Hour*)

Date is a ▸ date of which the hour is to be set.

Hour is a ▸ number which replaces the previous hour of a date.

Examples

- If the current time is 9:38:53 on 2/22/1732,

```
SetHour (Now; 10) =  2/22/1732 10:38:53
```

Remarks

⇢ If *Hour* is 24 or greater, or a negative number, **SetHour** returns the ▸ error value ▸ DATE!.

See also

▸ Date Functions [p. 28]

▸ **SetDate** [p. 183] – Creates a date from numbers.

▸ **SetTime** [p. 190] – Creates a time from numbers. The date part will be set to Jan 1st, 1904.

▸ **SetSecond** [p. 189] – Sets the seconds in a date.

▸ **SetMinute** [p. 187] – Sets the minutes in a date.

▸ **SetDay** [p. 184] – Sets the day in a date.

▸ **SetMonth** [p. 188] – Sets the month in a date.

▸ **SetYear** [p. 192] – Sets the year in a date.

4.142 SETMINUTE (FUNCTION)

SetMinute sets the minute in a date.

Syntax

SetMinute (*Date*; *Minute*)

Date is a ▸ date of which the minute is to be set.

Minute is a ▸ number which replaces the previous minute of a date.

Examples

- If the current time is 9:38:53 on 2/22/1732,

```
SetMinute (Now; 10) =  2/22/1732 9:10:53
```

Warnings

⚠ If *Minute* is 60 or greater, or a negative number, **SetMinute** returns the ▸ error value ▸ DATE!.

See also

▸ Date Functions [p. 28]

▸ **SetDate** [p. 183] – Creates a date from numbers.

▸ **SetTime** [p. 190] – Creates a time from numbers. The date part will be set to Jan 1st, 1904.

▸ **SetSecond** [p. 189] – Sets the seconds in a date.

▸ **SetHour** [p. 186] – Sets the hours in a date.

▸ **SetDay** [p. 184] – Sets the day in a date.

▸ **SetMonth** [p. 188] – Sets the month in a date.

▸ **SetYear** [p. 192] – Sets the year in a date.

4.143 SETMONTH (FUNCTION)

SetMonth sets the month in a date.

Syntax

SetMonth (*Date*; *Month*)

Date is a ▸ date of which the month is to be set.

Month is a ▸ number which replaces the previous month of a date.

Examples

- If the current time is 9:38:53 on 2/22/1732,

```
SetMonth (Now; 10) =  10/22/1732 9:38:53
```

Warnings

⚠ If *Month* is 13 or greater, or a negative number, **SetMonth** returns the ▸ error value ▸ DATE!.

See also

- ▸ Date Functions [p. 28]
- ▸ **SetDate** [p. 183] – Creates a date from numbers.
- ▸ **SetTime** [p. 190] – Creates a time from numbers. The date part will be set to Jan 1st, 1904.
- ▸ **SetSecond** [p. 189] – Sets the seconds in a date.
- ▸ **SetMinute** [p. 187] – Sets the minutes in a date.
- ▸ **SetHour** [p. 186] – Sets the hours in a date.
- ▸ **SetDay** [p. 184] – Sets the day in a date.
- ▸ **SetYear** [p. 192] – Sets the year in a date.

4.144 SETSECOND (FUNCTION)

SetSecond sets the second in a date.

Syntax

SetSecond (*Date*; *Seconds*)

Date is a ▸ date of which the second is to be set.

Seconds is a ▸ number which replaces the previous second of a date.

Examples

- If the current time is 9:38:53 on 2/22/1732,

```
SetSecond (Now; 10) =  2/22/1732 9:38:10
SetSecond (Now; 10.8) =  2/22/1732 9:38:11
```

Warnings

⚠ If *Second* is 60 or greater, or a negative number, **SetSecond** returns the ▸ error value ▸ DATE!.

See also

▸ Date Functions [p. 28]

▸ **SetDate** [p. 183] – Creates a date from numbers.

▸ **SetTime** [p. 190] – Creates a time from numbers. The date part will be set to Jan 1st, 1904.

▸ **SetMinute** [p. 187] – Sets the minutes in a date.

▸ **SetHour** [p. 186] – Sets the hours in a date.

▸ **SetDay** [p. 184] – Sets the day in a date.

▸ **SetMonth** [p. 188] – Sets the month in a date.

▸ **SetYear** [p. 192] – Sets the year in a date.

4.145 SETTIME (FUNCTION)

SetTime creates a time from numbers. The date part is set to January 1, 1904.

Syntax

SetTime (*Hour*; *Minute*; *Second*)

Hour is a ▸ number specifying the hour of the date.

Minute is a ▸ number specifying the minute of the date.

Second is a ▸ number specifying the second of the date.

Examples

- `SetTime (7; 10; 53) = 1/1/1904 7:10:53`

Warnings

⚠ If an argument lies outside the permitted range (for example, 24 for *Hour*), **SetTime** returns the ▸ error value ▸ DATE!.

See also

- ▸ Date Functions [p. 28]
- ▸ **SetTimeSpan** [p. 191] – Creates a time span from numbers.
- ▸ **SetDate** [p. 183] – Creates a date from numbers.
- ▸ **SetSecond** [p. 189] – Sets the seconds in a date.
- ▸ **SetMinute** [p. 187] – Sets the minutes in a date.
- ▸ **SetHour** [p. 186] – Sets the hours in a date.
- ▸ **SetDay** [p. 184] – Sets the day in a date.
- ▸ **SetMonth** [p. 188] – Sets the month in a date.
- ▸ **SetYear** [p. 192] – Sets the year in a date.

4.146 SETTIMESPAN (FUNCTION)

SetTimeSpan creates a time span from numbers.

Syntax

SetTimeSpan (*Days*; *Hours*; *Minutes*; *Seconds*)

Days is a ▸ number specifying the number of days in the time span.

Hours is a ▸ number specifying the number of hours in the time span.

Minutes is a ▸ number specifying the number of minutes in the time span.

Seconds is a ▸ number specifying the number of seconds in the time span.

Examples

- `SetTimeSpan (10; 7; 10; 53) = 10d 7h 10m 53s`
- `SetTimeSpan (10; 51; 10; 53) = 12d 3h 10m 53s`
- `SetTimeSpan (10; 0; 10; 53) = 10d 10m 53s`
- `SetTimeSpan (10; 2; 10; 0) = 10d 2h 10m`

See also

▸ Date Functions [p. 28]

▸ **SetDate** [p. 183] – Creates a date from numbers.

▸ **SetTime** [p. 190] – Creates a time from numbers. The date part will be set to Jan 1st, 1904.

4.147 SETYEAR (FUNCTION)

SetYear sets the year in a date.

Syntax

SetYear (*Date*; *Year*)

Date is a ▸ date of which the year is to be set.

Year is a ▸ number which replaces the previous year of a date.

Examples

- If the current time is 9:38:53 on 2/22/1732,

```
SetYear (Now; 10) =  2/22/2010 9:38:53
SetYear (Now; 52) =  2/22/1952 9:38:53
SetYear (Now; 1877) =  2/22/1877 9:38:53
```

Warnings

⚠ If *Year* is 29001 or greater, or a negative number, **SetYear** returns the ▸ error value ▸ DATE!.

See also

- ▸ Date Functions [p. 28]
- ▸ **SetDate** [p. 183] – Creates a date from numbers.
- ▸ **SetTime** [p. 190] – Creates a time from numbers. The date part will be set to Jan 1st, 1904.
- ▸ **SetSecond** [p. 189] – Sets the seconds in a date.
- ▸ **SetMinute** [p. 187] – Sets the minutes in a date.
- ▸ **SetHour** [p. 186] – Sets the hours in a date.
- ▸ **SetDay** [p. 184] – Sets the day in a date.
- ▸ **SetMonth** [p. 188] – Sets the month in a date.

4.148 SIGN (FUNCTION)

Sign returns a ▸ number that represents the sign of a given number.

Syntax

Sign (*Number*)

Number is a ▸ number of the the sign is to be determined.

1 is returned for *number* › 0 (positive)

0 is returned for *number* = 0

-1 is returned for *number* ‹ 0 (negative)

See also ▸ Arithmetic Functions [p. 26]

▸ **Abs** [p. 40] – Returns the absolute value of a number.

4.149 SIN (FUNCTION)

Sin calculates the sine of an angle.

Sin returns a ▸ number in the ▸ interval [-1; +1].

Syntax

Sin (*Angle*)

Angle is a ▸ number which expresses in radians the angle of which the sine is to be calculated.

Remarks

⇢ If *Angle* is expressed in degrees, use `Sin (Radians (Angle))`.

See also

- ▸ Trigonometric Functions [p. 37]
- ▸ **ArcSin** [p. 50] – Calculates the arc sine of a number.
- ▸ **Cos** [p. 79] – Calculates the cosine of an angle given in radians.
- ▸ **Tan** [p. 210] – Calculates the tangent of an angle given in radians.
- ▸ **Radians** [p. 167] – Converts an angle from degrees to radians.

4.150 SMALL (FUNCTION)

Small returns the k-th smallest number in a range.

Syntax

Small (*Range*; *k*)

Range is a ▸ range specifying the range containing the numbers to be checked.

k is a ▸ number specifying the k-th smallest value to be returned. (If *k* is 4, for example, the fourth-smallest value is returned.)

Examples

- If the range A1:A4 contains the values 8, 1, 4 and 6 :

```
Small (A1:A4;2) = 4
Small (A1:A4;3) = 6
```

Remarks

⇢ Values in *Range* which are not numbers are ignored.

See also

- ▸ Search Functions [p. 33]
- ▸ **Min** [p. 144] – Returns the smallest number in a list.
- ▸ **Max** [p. 141] – Returns the largest number in a list.
- ▸ **Large** [p. 126] – Returns the k–th largest of the numbers in the range.

4.151 SMARTCONCAT (FUNCTION)

SmartConcat concatenates a list of texts, each with a connecting text.

The result consists of the elements of the text list, each separated by the connecting text.

Syntax

SmartConcat (*Delimiter*; *ListOfText*)

Delimiter is a ▸ text which is to be inserted between the elements of a list.

Examples

- `SmartConcat ('-';'Jack';'Queen';'King';'Ace') = Jack-Queen-King-Ace`
- `SmartConcat ('… ';18;20;'Pass') = 18… 20… Pass`

Remarks

Elements in the list which are not text are converted to text if possible and concatenated with the other elements. (See the second example, above.)

See also

- ▸ Text Functions [p. 36]
- ▸ Operators [p. 31]
- ▸ **Replace** [p. 172] – Replaces specific characters in a text.
- ▸ **Repeat** [p. 171] – Repeats a given text n times.
- ▸ **Concat** [p. 77] – Concatenates texts.

4.152 SPECIALIF (FUNCTION)

SpecialIf returns second argument or cell value when the condition is true; otherwise, the cell remains unchanged.

Syntax

SpecialIf (*Condition*; *ValueIfTrue*)

Condition is a ▸ truth value which the function evaluates.

ValueIfTrue is a ▸ value which is the result of the function when *Condition* is true.

Examples

- With the following formula, you can keep in a cell the greatest value that another cell assumes in the course of a series of calculations. The following formula is in cell A1 and holds the largest value that cell B1 has assumed so far. In order for the example to function, a number, for example, 0, must be entered in A1 as a start value.

```
SpecialIf (B1>A1;B1)
```

Remarks

⇢ This function changes the contents of the cell in which it located is only when *Condition* is true. When *Condition* is false, the previous contents remain in the cell.

See also ▸ Miscellaneous Functions [p. 32]

▸ **If** [p. 116] – Returns second or third argument, depending on whether the first argument is true.

4.153 SQR (FUNCTION)

Sqr calculates the square of a number.

Syntax

Sqr (*Number*)

Number is a ▸ number which is to be squared.

See also

▸ Arithmetic Functions [p. 26]

▸ **Sqrt** [p. 200] – Calculates the square root of a number.

▸ **SqrSum** [p. 199] – Calculates the sum of squares of all numbers in a list.

4.154 SQRSUM (FUNCTION)

SqrSum calculates the sum of squares of all numbers in a list.

Syntax

SqrSum (*List*)

List is a list of values. **SqrSum** returns the sum of the squares of all numbers in this list.

Examples

- `SqrSum (1; 2; 3) = 14`
- `SqrSum (1; 2; -3) = 14`
- `SqrSum ('A Song'; 2; 3) = 13`

See also

- Arithmetic Functions [p. 26]
- **Sum** [p. 203] – Calculates the sum of a list of numbers.
- **SumProduct** [p. 204] – Calculates the sum of products of corresponding numbers in ranges.
- **SumXPY2** [p. 208] – Calculates the sum of squares of x+y, where x, y are taken from two ranges.
- **SumXMY2** [p. 207] – Calculates the sum of squares of x–y, where x, y are taken from two ranges.
- **SumX2PY2** [p. 206] – Calculates the sum of sqr (x) + sqr (y), where x, y are taken from two ranges.
- **SumX2MY2** [p. 205] – Calculates the sum of sqr (x) - sqr (y), where x, y are taken from two ranges.

4.155 SQRT (FUNCTION)

Sqrt calculates the square root of a number.

Syntax

Sqrt (*Number*)

Number is a positive ▸ number of which the square root is to be found.

Warnings

⚠ The ▸ error value ▸ NUM! is returned if *number* ‹ 0. (The root is not defined for these cases.)

See also ▸ Arithmetic Functions [p. 26]

▸ **Sqr** [p. 198] – Calculates the square of a number.

4.156 StartingPageNumber (Function)

StartingPageNumber returns the page number of the first page.

Syntax

StartingPageNumber (*LayoutName*)

LayoutName is a ▸ text, giving the name of a layout. The function returns the number of the first page of this layout. If no layout is given, the number of the first page of the layout in which the formula appears is returned.

Examples

- `StartingPageNumber () = 12`
- `StartingPageNumber ('Layout 2') = 25`

Remarks

⇢ The formula with the function need not be in the layout of which the page number is returned.

See also

▸ Miscellaneous Functions [p. 32]

▸ **Page** [p. 156] – Returns the number of the page containing the formula.

▸ **PageIndex** [p. 157] – Returns the index of the page containing the formula.

▸ **NoOfPages** [p. 150] – Returns the number of pages.

▸ **EndingPageNumber** [p. 100] – Returns the ending page number of the layout.

4.157 STDEV (FUNCTION)

StDev calculates the standard deviation of numbers in a list.

Syntax

StDev (*List*)

List is a ▸ list from which the standard deviation of its numbers is calculated.

Examples

- `StDev (1; 2; 3) = 1`
- `StDev (1; 2; 'three') = 0.7071…`

Remarks

Standard deviation is calculated according to

$$s = \sqrt{\frac{\sum x_i^2 - \frac{(\sum x_i)^2}{n}}{n-1}}; i = 1, 2, 3, \ldots, n$$

where x is the numbers *List* and n is the count of these numbers.

This equation is based on the assumption that the data represent a sample of a population. You can calculate the standard deviation of a complete population with `Sqrt (StDev (List)^2 * (Count (List) - 1) /Count (List))`. The difference between the two values decreases as n increases.

See also

- ▸ Statistical Functions [p. 35]
- ▸ **Average** [p. 52] – Calculates the average of numbers in a list.
- ▸ **Var** [p. 224] – Calculates the variance of numbers in a list.

4.158 SUM (FUNCTION)

Sum calculates the sum of a list of numbers.

Syntax

Sum (*List*)

List is a ▸ list of numbers. **Sum** returns the sum of all the numbers in this list.

Examples

- `Sum (1; 2; 3) = 6`
- `Sum (1; 2; -3) = 0`
- `Sum ('A Song'; 2; 3) = 5`

See also

▸ Arithmetic Functions [p. 26]

▸ **SqrSum** [p. 199] – Calculates the sum of squares of all numbers in a list.

▸ **SumProduct** [p. 204] – Calculates the sum of products of corresponding numbers in ranges.

▸ **SumXPY2** [p. 208] – Calculates the sum of squares of x+y, where x, y are taken from two ranges.

▸ **SumXMY2** [p. 207] – Calculates the sum of squares of x–y, where x, y are taken from two ranges.

▸ **SumX2PY2** [p. 206] – Calculates the sum of sqr (x) + sqr (y), where x, y are taken from two ranges.

▸ **SumX2MY2** [p. 205] – Calculates the sum of sqr (x) - sqr (y), where x, y are taken from two ranges.

4.159 SUMPRODUCT (FUNCTION)

SumProduct calculates the sum of products of corresponding numbers in ranges.

The first numbers in all ranges are multiplied together and added to the product of the second numbers in all ranges and so on.

Syntax

SumProduct (*Range1*; *Range2*; ...)

Range1 is a ▸ range containing the first set of numbers.

Range2 is a ▸ range containing the second set of numbers.

... are additional ranges with numbers.

Examples

- `SumProduct (A1:B8;E1:F8;I1:J8) = 1271`

Warnings

⚠ All ranges must have the same number of cells. If this is not the case, the ▸ error value ▸ `REF!` is returned. (However, the ranges may be formed differently. For example, one range may be 2 cells by 4 and a second 4 cells by 2.)

See also

▸ Arithmetic Functions [p. 26]

▸ **Sum** [p. 203] – Calculates the sum of a list of numbers.

▸ **SqrSum** [p. 199] – Calculates the sum of squares of all numbers in a list.

▸ **SumXPY2** [p. 208] – Calculates the sum of squares of x+y, where x, y are taken from two ranges.

▸ **SumXMY2** [p. 207] – Calculates the sum of squares of x–y, where x, y are taken from two ranges.

▸ **SumX2PY2** [p. 206] – Calculates the sum of sqr (x) + sqr (y), where x, y are taken from two ranges.

▸ **SumX2MY2** [p. 205] – Calculates the sum of sqr (x) - sqr (y), where x, y are taken from two ranges.

4.160 SUMX2MY2 (FUNCTION)

SumX2MY2 calculates the sum of sqr (x) - sqr (y), where x and y are taken from two ranges.

The function adds the difference between the squares of the first numbers in each of two ranges to the difference between the squares of the second numbers and so on.

Syntax

SumX2MY2 (*Range1*; *Range2*)

Range1 is a ▸ range containing the first set of numbers.

Range2 is a ▸ range containing the second set of numbers.

Examples

- `SumX2MY2 (A1:B4;C1:D4) = 73`

Warnings

⚠ All ranges must have the same number of cells. If this is not the case, the ▸ error value ▸ `REF!` is returned. (However, the ranges may be formed differently. For example, one range may be 2 cells by 4 and a second 4 cells by 2.)

See also

▸ Arithmetic Functions [p. 26]

▸ **Sum** [p. 203] – Calculates the sum of a list of numbers.

▸ **SqrSum** [p. 199] – Calculates the sum of squares of all numbers in a list.

▸ **SumProduct** [p. 204] – Calculates the sum of products of corresponding numbers in ranges.

▸ **SumXPY2** [p. 208] – Calculates the sum of squares of x+y, where x, y are taken from two ranges.

▸ **SumXMY2** [p. 207] – Calculates the sum of squares of x–y, where x, y are taken from two ranges.

▸ **SumX2PY2** [p. 206] – Calculates the sum of sqr (x) + sqr (y), where x, y are taken from two ranges.

4.161 SUMX2PY2 (FUNCTION)

SumX2PY2 calculates the sum of sqr (x) + sqr (y), where x and y are taken from two ranges.

The function adds the sum of the squares of the first number of each of two ranges to the sum of the squares of the second numbers and so on.

Syntax

SumX2PY2 (*Range1*; *Range2*)

Range1 is a ▸ range containing the first set of numbers.

Range2 is a ▸ range containing the second set of numbers.

Examples

- `SumX2PY2 (A1:B4;C1:D4) = 284`

Warnings

⚠ All ranges must have the same number of cells. If this is not the case, the ▸ error value ▸ `REF!` is returned. (However, the ranges may be formed differently. For example, one range may be 2 cells by 4 and a second 4 cells by 2.)

See also

▸ Arithmetic Functions [p. 26]

▸ **Sum** [p. 203] – Calculates the sum of a list of numbers.

▸ **SqrSum** [p. 199] – Calculates the sum of squares of all numbers in a list.

▸ **SumProduct** [p. 204] – Calculates the sum of products of corresponding numbers in ranges.

▸ **SumXPY2** [p. 208] – Calculates the sum of squares of x+y, where x, y are taken from two ranges.

▸ **SumXMY2** [p. 207] – Calculates the sum of squares of x–y, where x, y are taken from two ranges.

▸ **SumX2MY2** [p. 205] – Calculates the sum of sqr (x) - sqr (y), where x, y are taken from two ranges.

4.162 SUMXMY2 (FUNCTION)

SumXMY2 calculates the sum of squares of x-y, where x and y are taken from two ranges.

The function adds the square of the difference between the first numbers in each of two ranges to the square of the difference between the second numbers of each range and so on.

Syntax

SumXMY2 (*Range1*; *Range2*)

Range1 is a ▸ range containing the first set of numbers.

Range2 is a ▸ range containing the second set of numbers.

Examples

- `SumXMY2 (A1:B4;C1:D4) = 54`

Warnings

⚠ All ranges must have the same number of cells. If this is not the case, the ▸ error value ▸ `REF!` is returned. (However, the ranges may be formed differently. For example, one range may be 2 cells by 4 and a second 4 cells by 2.)

See also

▸ Arithmetic Functions [p. 26]

▸ **Sum** [p. 203] – Calculates the sum of a list of numbers.

▸ **SqrSum** [p. 199] – Calculates the sum of squares of all numbers in a list.

▸ **SumProduct** [p. 204] – Calculates the sum of products of corresponding numbers in ranges.

▸ **SumXPY2** [p. 208] – Calculates the sum of squares of x+y, where x, y are taken from two ranges.

▸ **SumX2PY2** [p. 206] – Calculates the sum of sqr (x) + sqr (y), where x, y are taken from two ranges.

▸ **SumX2MY2** [p. 205] – Calculates the sum of sqr (x) - sqr (y), where x, y are taken from two ranges.

4.163 SUMXPY2 (FUNCTION)

SumXPY2 calculates the sum of squares of x+y, where x and y are taken from two ranges.

Syntax

SumXPY2 (*Range1*; *Range2*)

Range1 is a ▸ range containing the first set of numbers.

Range2 is a ▸ range containing the second set of numbers.

Examples

- `SumXPY2 (A1:B4;C1:D4) = 311`

Warnings

⚠ All ranges must have the same number of cells. If this is not the case, the ▸ error value ▸ `REF!` is returned. (However, the ranges may be formed differently. For example, one range may be 2 cells by 4 and a second 4 cells by 2.)

See also

▸ Arithmetic Functions [p. 26]

▸ **Sum** [p. 203] – Calculates the sum of a list of numbers.

▸ **SqrSum** [p. 199] – Calculates the sum of squares of all numbers in a list.

▸ **SumProduct** [p. 204] – Calculates the sum of products of corresponding numbers in ranges.

▸ **SumXMY2** [p. 207] – Calculates the sum of squares of x–y, where x, y are taken from two ranges.

▸ **SumX2PY2** [p. 206] – Calculates the sum of sqr (x) + sqr (y), where x, y are taken from two ranges.

▸ **SumX2MY2** [p. 205] – Calculates the sum of sqr (x) - sqr (y), where x, y are taken from two ranges.

4.164 SYSTEMCURRENCY (FUNCTION)

SystemCurrency converts a number to text using the currency format defined by the system software.

The number is rounded during conversion.

Syntax

SystemCurrency (*Number*; *DecimalPlaces*)

Number is a ▸ number which is to be converted.

DecimalPlaces is the desired ▸ number of decimal places. If no number is given, two places after the decimal are used.

Examples

- `SystemCurrency (123.456) = $123.46`
- `SystemCurrency (123.456; 0) = $123`

Warnings

⚠ If *DecimalPlaces* is a negative number, **SystemCurrency** returns the ▸ error value ▸ RANGE!.

See also ▸ Financial Functions [p. 31]

4.165 TAN (FUNCTION)

Tan calculates the tangent of an angle.

Tan returns a ▸ number in the ▸ interval]-∞; +∞[.

Syntax

Tan (*Angle*)

Angle is a ▸ number which expresses in radians the angle of which the tangent is to be calculated.

Remarks

If *Angle* is expressed in degrees, use `Tan (Radians (Angle))`.

Warnings

⚠ The ▸ error value ▸ `DIV/0!` is returned if *Angle* is an odd multiple of $\pi/2$. (A tangent cannot be represented in this case.)

See also

- ▸ Trigonometric Functions [p. 37]
- ▸ **ArcTan** [p. 51] – Calculates the arc tangent of a number.
- ▸ **Sin** [p. 194] – Calculates the sine of an angle given in radians.
- ▸ **Cos** [p. 79] – Calculates the cosine of an angle given in radians.
- ▸ **Radians** [p. 167] – Converts an angle from degrees to radians.

4.166 TEXT (FUNCTION)

Text converts a value to text.

Numbers are rounded during conversion.

Syntax

Text (***Number***; *DecimalPlaces*)

Number is a ▸ number which is to be converted to text.

DecimalPlaces is a ▸ number specifying the desired number of decimal places. If *DecimalPlaces* is not given, *Number* is converted as a whole number.

Syntax

Text (***Number***; *ValueFormatDefinition*)

Number is a ▸ number which is to be converted to text.

ValueFormatDefinition is a ▸ text specifying the desired value format.

Syntax

Text (***Date***; ***ValueFormatDefinition***)

Date is a ▸ date which is to be converted to text.

ValueFormatDefinition is a ▸ text specifying the desired value format. If no format definition is given, *date* is converted to the format “Standard.”

Syntax

Text (***TimeSpan***; ***ValueFormatDefinition***)

TimeSpan is a time span which is to be converted to text.

ValueFormatDefinition is a ▸ text specifying the desired value format.

Examples

- `Text (2) = 2`
- `Text (2.7) = 3`
- `Text (3.1415; 2) = 3.14`
- `Text (100000; '#,##0.00') = 100,000.00`
- `Text (Now) = 7/19/88 4:23:19`
- `Text (Now; 'h":"mm') = 4:23`

See also ▸ Text Functions [p. 36]

▸ Conversion Functions [p. 27]

▸ **ValueFormat** [p. 223] – Returns the definition of a value format.

▸ **Number** [p. 154] – Converts text to a number or a date into the number of days from January 1, 1904.

- `Value` [p. 222] – Converts text to a value, as if entered into a spreadsheet cell.
- `Date` [p. 85] – Converts text or a specified number of days from January 1, 1904 to a date.
- `TimeSpan` [p. 213] – Converts text to a time span.

4.167 TIMESPAN (FUNCTION)

TimeSpan converts text to a time span.

Syntax

TimeSpan (*Text*)

Text is a ▸ text which is to be converted to a time span.

Examples

- `TimeSpan ('28h 40m') = 1d 4h 40m`

Warnings

⚠ If *Text* can be converted into a ▸ number, this number will be interpreted in the unit "Day (d)" and returned as the result.

⚠ If the argument cannot be converted to a time span, **TimeSpan** returns the ▸ error value ▸ `DATE!`.

See also

▸ Date Functions [p. 28]

▸ Text Functions [p. 36]

▸ Conversion Functions [p. 27]

▸ **Date** [p. 85] – Converts text or a specified number of days from January 1, 1904 to a date.

▸ **Number** [p. 154] – Converts text to a number or a date into the number of days from January 1, 1904.

▸ **Value** [p. 222] – Converts text to a value, as if entered into a spreadsheet cell.

▸ **Text** [p. 211] – Converts a value to text.

4.168 TODAY (FUNCTION)

Today returns the current ▸ date of your computer. The time of the result is 00:00:00.

Syntax

Today ()

This function requires no arguments.

Examples

- `Today = 4/19/1999`

Remarks

⇢ **Today** is recalculated when a document is opened.

See also

- ▸ Date Functions [p. 28]
- ▸ **Second** [p. 179] – Returns the current date and time every second.
- ▸ **Minute** [p. 145] – Returns the current date and time every minute.
- ▸ **Hour** [p. 112] – Returns the current date and time every hour.
- ▸ **Now** [p. 152] – Returns the current date and time.
- ▸ **DayOf** [p. 86] – Extracts the day from a date.

4.169 TRIM (FUNCTION)

Trim removes unnecessary blank space characters from a text.

Every group of two or more blank space characters is replaced by a single blank space character.

Syntax

Trim (***Text***)

Text is a ▸ text from which unnecessary characters are to be removed.

Examples

- `Trim ('To␣be␣or␣␣␣␣␣not␣to␣be.') =`
  `To␣be␣or␣not␣to␣be.`
- `Trim ('To␣be␣or␣not␣to␣be.') =`
  `To␣be␣or␣not␣to␣be.`
- `Trim ('Hello') = Hello`

See also

- ▸ Text Functions [p. 36]
- ▸ **Clean** [p. 63] – Removes non-printable characters from a text.
- ▸ **Lower** [p. 138] – Returns text with all letters changed to lower case.
- ▸ **Upper** [p. 221] – Returns text with all letters changed to upper case.
- ▸ **Proper** [p. 165] – Returns text with all initials changed to upper case.
- ▸ **Exact** [p. 103] – Returns true if two texts match exactly.

4.170 TRUE (FUNCTION)

True returns the constant ▸ truth value "True."

Syntax

True ()

This function requires no arguments.

See also

▸ Constants [p. 27]

▸ **False** [p. 107] – Returns the logical constant False.

4.171 TRUNC (FUNCTION)

Trunc truncates a ▸ number to the given number of places.

Syntax

Trunc (*Number*; *Places*)

Number is a ▸ number of which the decimal places is to be reduced.

Places is an optional argument. **Trunc** cuts the decimal places of *Number* to the number of *Places*. If *Places* is not given, all decimal places are cut.

Examples

- `Trunc (3.14) = 3`
- `Trunc (-3.14) = -3`
- `Trunc (3.14; 1) = 3.1`
- `Trunc (256; -2) = 200`
- `Trunc (256; -4) = 0`

Remarks

⇢ If *Places* is a negative ▸ number the appropriate number of places before the decimal are replaced with zeros. (See the fourth and fifth examples, above.)

See also

▸ Arithmetic Functions [p. 26]

▸ Rounding Functions [p. 33]

▸ **Int** [p. 119] – Rounds a number down to the largest integer.

▸ **Round** [p. 174] – Rounds a number to a specific number of places.

▸ **Frac** [p. 110] – Returns the fractional part of a number.

▸ **Floor** [p. 109] – Rounds a number to the next lower whole number or the next lower multiple.

▸ **Ceiling** [p. 54] – Rounds a number to the next higher whole number or the next higher multiple.

4.172 TYPE (FUNCTION)

Type returns a code for the type of the value.

Syntax

Type (*Value*)

Value is a ▸ value of which the type is returned. If *Value* is a ▸ number, 1 is returned; if *Value* is a ▸ text, 2 is returned; if *Value* is a ▸ date, 8 is returned; and if the value is an ▸ error value, 16 is returned.

Examples

- `Type (A2) = 0` (if cell A2 is empty)
- `Type (0) = 1`
- `Type ('Hello') = 2`
- `Type (24/1/1984) = 8`
- `Type (1/0) = 16`

Remarks

If *Value* is a reference, then **Type** returns the type of the contents of the referenced cell.

Warnings

Type does not return a reliable result if *Value* is a reference to a ▸ range.

See also

▸ Status Functions [p. 35]

▸ **IsErr** [p. 121] – Returns true if the value is any error value.

▸ **IsNumber** [p. 124] – Returns true if the value is a number.

▸ **IsBlank** [p. 120] – Returns true if the value is empty or an empty string.

▸ **IsNA** [p. 123] – Returns true if the value is the “NA” error value.

4.173 UNICHAR (FUNCTION)

UniChar returns a character with a given character code, using the Unicode character set.

The character code adheres to the standard code set in Unicode.

Syntax

UniChar (*CharacterCode*)

CharacterCode is a ▸ number from 0 to 65533 for which the corresponding character is to be returned.

Examples

- `UniChar (65) = A`
- `UniChar (97) = a`
- `UniChar (49) = 1`
- `UniChar (183) = ·`

Remarks

The characters with the codes from 0 to 31 usually cannnot be displayed on the screen and, therefore, are either represented with a symbol for missing characters (usually a rectangle) or not displayed at all.

Warnings

⚠ Most characters in unicode having a code greater than 127 cannot be displayed on a Macintosh and are either represented with a symbol for missing characters or not displayed at all.

⚠ If *CharacterCode* is less than 0 or greater than 65533, **UniChar** returns the ▸ error value ▸ `NUM!`.

See also

▸ Text Functions [p. 36]

▸ Conversion Functions [p. 27]

▸ **Char** [p. 55] – Returns a character with a given character code and encoding.

▸ **Code** [p. 72] – Returns the character code of the first character in a text having a given encoding.

4.174 UNICODE (FUNCTION)

UniCode returns the character code of the first character in a text, using the Unicode character set.

The character code adheres to the standard code set in Unicode.

Syntax

UniCode (*Text*)

Text is a character or other letter of which the code is to be determined.

Examples

- `UniCode ('A') = 65`
- `UniCode ('a') = 97`
- `UniCode (1) = 49`

Remarks

If the *Text* consists of several characters, **UniCode** returns the code of the first character.

See also

- Text Functions [p. 36]
- Conversion Functions [p. 27]
- **Code** [p. 72] – Returns the character code of the first character in a text having a given encoding.
- **Char** [p. 55] – Returns a character with a given character code and encoding.

4.175 UPPER (FUNCTION)

Upper converts all letters in a text to upper case.

Syntax

Upper (*Text*)

Text is a ▸ text in which the letters are to be converted.

Examples

- `Upper ('Hello') = HELLO`

See also

- ▸ Text Functions [p. 36]
- ▸ **Clean** [p. 63] – Removes non-printable characters from a text.
- ▸ **Trim** [p. 215] – Removes unnecessary space characters from a text.
- ▸ **Lower** [p. 138] – Returns text with all letters changed to lower case.
- ▸ **Proper** [p. 165] – Returns text with all initials changed to upper case.
- ▸ **Exact** [p. 103] – Returns true if two texts match exactly.

4.176 VALUE (FUNCTION)

Value converts text to a value, as if entered into a spreadsheet cell.

Syntax

Value (*Text*)

Text is a ▸ text which is to be converted to a value.

Examples

- `Value ('12') = 12`
- `Value (Mid (Text (12345); 3; 2)) = 34`

Warnings

⚠ If *Text* cannot be converted to a value, **Value** returns the ▸ error value ▸ VALUE!.

See also

▸ Arithmetic Functions [p. 26]
▸ Conversion Functions [p. 27]
▸ **Number** [p. 154] – Converts text to a number or a date into the number of days from January 1, 1904.
▸ **Text** [p. 211] – Converts a value to text.
▸ **Date** [p. 85] – Converts text or a specified number of days from January 1, 1904 to a date.
▸ **TimeSpan** [p. 213] – Converts text to a time span.

4.177 ValueFormat (Function)

ValueFormat the definition of a value format.

Syntax

ValueFormat (*FormatName*)

FormatName is a ▸ text which is the name of an available value format.

Examples

- `ValueFormat ('2 Decimals') = 0.00`
- `ValueFormat ('General') = GENERAL`

See also

▸ Miscellaneous Functions [p. 32]
▸ **Text** [p. 211] – Converts a value to text.

4.178 VAR (FUNCTION)

Var calculates the variance of numbers in a list.

Syntax

Var (*List*)

List is a ▸ list of which the variance of the numbers is to be calculated.

Examples

- `Var (1; 2; 3) = 1`
- `Var (1; 2; 'three') = 0.5`

Remarks

⇢ The variance of a list is the square of its

▸ **StDev** [p. 202] – Calculates the standard deviation of numbers in a list.

See also

▸ Statistical Functions [p. 35]

▸ **Average** [p. 52] – Calculates the average of numbers in a list.

▸ **StDev** [p. 202] – Calculates the standard deviation of numbers in a list.

4.179 VSEARCH (FUNCTION)

VSearch scans a range by rows for matches of a condition and returns either the count of matches or the StartValue, which is overwritten with NextValue in case of matches.

VSearch searches the left column of the given range.

If **VSearch** is called with two arguments, it returns the count of the cells for which the given condition is true. If it is called with four arguments, it uses the formula given as the fourth argument each time a cell fulfills the given condition, and returns the result in the last of these calculations. The same is true when **VSearch** is called with five arguments. However, each cell is additionally tested if the stop condition, which is given as the fifth argument, is true. If it is true, the current result of the given formula is returned.

Syntax

VSearch (*Range*; *Condition*)

Range is a ▸ range which the function is to search.

Condition is a condition for which each cell in *Range* is tested.

Syntax

VSearch (*Range*; *Condition*; *StartValue*; *NextValue*; *StopCondition*)

Range is a ▸ range which the function is to search.

Condition is a condition for which each cell in *Range* is tested.

StartValue is the initial value for calculating *NextValue*. *StartValue* is the value of **CurrentResult** (refer to that function), before **VSearch** begins searching *Range*.

NextValue is a formula which is used each time a cell in *Range* fulfills the condition *Condition*.

StopCondition is a condition for which each cell is tested. If it is true, *NextValue* is calculated, **Search** is interrupted and the result of *NextValue* is returned. If *StopCondition* is not given, **Search** always searches all cells in *Range*.

Examples

- A spreadsheet contains the follwing values:

	A	B
1	1	5
2	2	6
3	3	7
4	4	8

- `VSearch (A1:B4; IsOdd(CurrentCell)) = 2`

- `VSearch (A1:B4; IsOdd(CurrentCell); 0; CurrentResult + CurrentCell) = 4`
- `VSearch (A1:B4; IsOdd(CurrentCell); 0; CurrentResult + ColumnValue(2); CurrentCell>5) = 12`

Remarks

For a meaningful use of **VSearch**, the functions **CurrentIndex**, **CurrentResult**, **CurrentCount** and **CurrentCell** are necessary.

See also

- Spreadsheet Functions [p. 34]
- Search Functions [p. 33]
- **Search** [p. 177] – Scans a range for matches of a condition and returns either the count of matches or the StartValue, which is replaced by NextValue in case of matches.
- **HSearch** [p. 114] – Scans a range by columns for matches of a condition and return seither the count of matches or the StartValue, which is replaced by NextValue in case of matches.
- **ColumnValue** [p. 74] – Returns the value from that column, to be used in VSearch.
- **CurrentCell** [p. 81] – Returns the current cell in the Search function.
- **CurrentIndex** [p. 83] – Returns the current index in the Search function.
- **CurrentResult** [p. 84] – Returns the current result in the Search function.
- **CurrentCount** [p. 82] – Returns the number of values already accepted in the Search function.

4.180 WEEKOFYEAR (FUNCTION)

WeekOfYear returns the ▸ number of the calender week, according to US standards, in which a date falls. The first week of a year is that in which January 1 falls.

Syntax

WeekOfYear (*Date*)

Date is a ▸ date for which the calender week is to be calculated.

Examples

- `WeekOfYear (1/1/1995) = 1`
- `WeekOfYear (1/1/1996) = 1`
- `WeekOfYear (1/1/1999) = 1`
- `WeekOfYearISO (1/1/1995) = 52`
- `WeekOfYearISO (1/1/1996) = 1`
- `WeekOfYearISO (1/1/1999) = 53`

Remarks

Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

See also

▸ Date Functions [p. 28]

▸ **WeekOfYearISO** [p. 228] – Returns the week of the year, ISO rules.

4.181 WEEKOFYEARISO (FUNCTION)

WeekOfYearISO returns the ▸ number of the calender week, according to DIN/ISO standards, in which a date falls. A week belongs to the year in which the majority of the days fall. In other words, the first week of a year is that in which the first Thursday of a year falls.

Syntax

WeekOfYearISO (*Date*)

Date is a ▸ date for which the calender week is to be calculated.

Examples

- `WeekOfYearISO (1/1/1995) = 52`
- `WeekOfYearISO (1/1/1996) = 1`
- `WeekOfYearISO (1/1/1999) = 53`
- `WeekOfYear (1/1/1995) = 1`
- `WeekOfYear (1/1/1996) = 1`
- `WeekOfYear (1/1/1999) = 1`

Remarks

Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

See also

▸ Date Functions [p. 28]
▸ **WeekOfYear** [p. 227] – Returns the week of the year, US rules.

4.182 WinChar (Function)

WinChar returns a character with a given character code in the Windows character set.

The character code adheres to the standard code set by Microsoft for Windows-compatible computers. Codes from 0 to 127 adhere to the ASCII standard.

Syntax

WinChar (*CharacterCode*)

CharacterCode is a ▸ number from 0 to 255 for which the corresponding character is to be returned.

Examples

- `WinChar (65) = A`
- `WinChar (97) = a`
- `WinChar (49) = 1`
- `WinChar (183) = ·`

Remarks

The characters with the codes from 0 to 31 usually cannnot be displayed on the screen and, therefore, are either represented with a symbol for missing characters (usually a rectangle) or not displayed at all.

Warnings

⚠ Some characters having a code greater than 127 on a Windows-compatible computer cannot be displayed on a Macintosh and are either represented with a symbol for missing characters or not displayed at all.

⚠ If *CharacterCode* is less than 0 or greater than 255, **WinChar** returns the ▸ error value ▸ `NUM!`.

See also

▸ Text Functions [p. 36]

▸ Conversion Functions [p. 27]

▸ **Char** [p. 55] – Returns a character with a given character code and encoding.

▸ **Code** [p. 72] – Returns the character code of the first character in a text having a given encoding.

4.183 WINCODE (FUNCTION)

WinCode returns the character code of the first character in a text, using the Windows character set.

The character code adheres to the standard code set by Microsoft for Windows-compatible computers. Codes from 0 to 127 adhere to the ASCII standard.

Syntax

WinCode (*Text*)

Text is a letter or other character of which the code is to be determined.

Examples

- `WinCode ('A') = 65`
- `WinCode ('a') = 97`
- `WinCode (1) = 49`

Remarks

If the *Text* consists of several characters, **WinCode** returns the code of the first character.

Warnings

If the first character of *Text* is not defined in the Windows character set, **WinCode** returns 127.

See also

- Text Functions [p. 36]
- Conversion Functions [p. 27]
- **Code** [p. 72] – Returns the character code of the first character in a text having a given encoding.
- **Char** [p. 55] – Returns a character with a given character code and encoding.

4.184 YEAROF (FUNCTION)

YearOf extracts the year from a date.

Syntax

YearOf (*Date*)

Date is a ▸ date of which the second is calculated.

Examples

- If the current time is 9:38:53 on 2/22/1732,
  `YearOf (Now) = 1732`

See also

- ▸ Date Functions [p. 28]
- ▸ **SecondOf** [p. 180] – Extracts seconds from a date.
- ▸ **MinuteOf** [p. 146] – Extracts minutes from a date.
- ▸ **HourOf** [p. 113] – Extracts hours from a date.
- ▸ **DayOf** [p. 86] – Extracts the day from a date.
- ▸ **MonthOf** [p. 148] – Extracts the month from a date.

CHAPTER 5

Background Information

5.1 REGRESSION ANALYSIS

Assume you have a set of points, for example measured values of two variables. You know theoretically that these points should lie on a particular curve; that is, there is a dependence between the two variables. With regression, you can find the curve which best approximates your data.

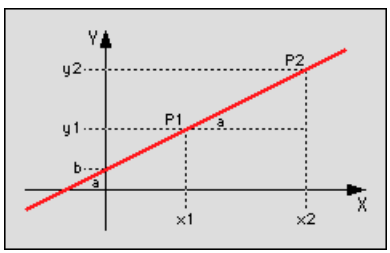
If you know the equation of this curve, that is, its type and parameters, you can estimate other, not measured, points on the curve by solving the equation for some values of the independent variable.

You should be aware that these estimates are not dependable if you use data outside the range of measured values which you used for calculating the curve's parameters. Errors also occur when the type of the cureve is based on an incorrect assumption or the dependent and independent variables are confused.

For further informtion, please consult astatistics textbook, in which you can also learn more about the methods used in RagTime for determining curve parameters.

Linear Fit

One form of the common linear equation is `y = m*x + b` where m is the **slope** and b is the **y intercept**.



The **slope m** is the tangent of the angle **a** formed by the line and the x axis. In other words, to calculate the slope, take any two points `P1 (x1, y1)` and `P2 (x2, y2)` on the slope and solve the equation `m = (y2 - y1) / (x2 - x1)`.

The **y intercept b** is the y value at the intersection of the line and the y axis.

If you know that your points fall on a straight line, you can calculate its parameters with the following functions:

- `RegressionM` [p. 170] – Calculates the slope of a best-fit straight line.
- `RegressionB` [p. 169] – Calculates the y–intercept of a best-fit straight line.

m and b are calculated according to

$$m = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}; i = 1, 2, 3, \ldots, n$$

$$b = \frac{\sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i}{n\sum x_i^2 - (\sum x_i)^2}; i = 1, 2, 3, \ldots, n$$

where n is the number of points, x is the independent variable and y is the dependent variable.

Example: You have an oven with a thermostat expressing degrees Fahrenheit and an antique cookbook with temperatures in centigrade. You can put a centigrade thermometer in the oven and take some measurements with the thermostat as the independent variable and the thermometer as the dependent variable. The linear equation in this case is `F = 9/5 C + 32`; thus, m = 9/5|b = 32. **Kids!** Don't try this without your parents.

Fitting an Exponential Curve

One form of the exponential curve is `y = m^x * b` where m is the **increase in slope** and b is the **y intercept**. The **curvature m** is a positive number. If 0 ‹ m ‹ 1, the curve falls. If m = 1, the curve degenerates to a line parallel to the x axis. If m › 1, the curve rises.

The **y intercept b** is the y value at the intersection of the curve and the y axis.

If you know that your points fall on an exponential curve, you can calculate their parameters with the following functions:

- `LogRegressM` [p. 135] – Calculates the increase of a best-fit exponential curve.
- `LogRegressB` [p. 134] – Calculates the y–intercept of a best-fit exponential curve.

m and b are calculated according to

$$\ln m = \frac{n\sum x_i \ln y_i - \sum x_i \sum \ln y_i}{n\sum x_i^2 - (\sum x_i)^2}; i = 1, 2, 3, \ldots, n$$

$$\ln b = \frac{\sum \ln y_i \sum x_i^2 - \sum x_i \sum x_i \ln y_i}{n\sum x_i^2 - (\sum x_i)^2}; i = 1, 2, 3, \ldots, n$$

where n is the number of points, x is the independent variable and y is the dependent variable.

5.2 NUMBERING IN RANGES

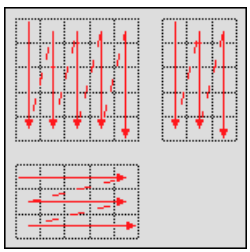
For some functions which access a ▸ range of a spreadsheet, it is necessary to number the cells within the range. The sequence of

numbers is used to search the cells in a defined sequence or to address a particular cell within the range.

The numbering of cells is determined by the proportions of the range.

If the range has **more rows than columns** or it is **square**, the cells are numbered column by column from top to bottom.

If the range has **fewer rows than columns**, the cells are numbered row by row from left to right.



5.3 ENCODING

All characters such as letters, punctuation marks, symbols and so on are stored internally in computers as numbers. The correlation between character and number is determined in so-called codes or encodings. You will find details under

- ▸ **About Encoding [p. 237]**

The following encodings are supported in RagTime 5. Encodings for which 256 characters suffice (so-called one byte codings) are indicated by positive numbers; the numbers in systems with more than 256 characters (double byte) are negative.

256 ▸ Mac OS Standard Roman [p. 238]
285 ▸ Mac OS Central European Roman [p. 239]
262 ▸ Mac OS Greek [p. 240]
263 ▸ Mac OS Cyrillic [p. 241]
512 ▸ Microsoft Windows Standard Roman [p. 242]
750 ▸ Microsoft Windows Central European Roman [p. 243]
716 ▸ Microsoft Windows Cyrillic [p. 244]
673 ▸ Microsoft Windows Greek [p. 245]
674 ▸ Microsoft Windows Turkish [p. 246]
1024 ▸ Microsoft DOS Code Page 437 [p. 247]
-23935 Microsoft Windows Korean
-24317 Mac OS Korean
-24319 ▸ Shift JIS [p. 248]

-24295 ▸ Simplified Chinese [p. 249]
-24318 ▸ Traditional Chinese [p. 250]
-32768 ▸ Unicode [p. 251] (recommended)

5.4 ABOUT ENCODING

All characters such as letters, punctuation marks, symbols and so on are stored internally in computers as numbers. The correlation between character and number is determined in so-called codes or encodings.

A well-known example of encoding is the "American Standard Code for Information Interchange" (ASCII). As the name suggests, ASCII was created from the American-English point of view and therefore contains hardly any characters needed in other languages. This includes not only Roman letters with diacritical markings (for example accented vowels in French or umlauts in German) but also all non-Roman writing systems (Greek, Indic, Japanese and so on).

Because only 128 characters (7 bit) are specified and many computers manage memory in units which permit 256 characters (8 bits = 1 byte), many computer manufacturers have included the remaining 128 characters. However, the manufacturers have each gone their own way and consequently various encodings, each with 256 characters, exist. To be sure, these encodings contain more characters than needed for English, but not all those needed for other languages using Roman characters.

In addition, other characters, for example Cyrillic, are needed for some localized versions of computer operating systems. Thus various encodings with 256 characters developed for use in various languages.

Some scripts manage with 256 characters (for example, Roman with 26 basic characters or Cyrillic with 33), while others need more characters, even substantially more (for example Chinese, Japanese and Korean with thousands of ideograms).

Recently, development has started on an encoding, known as "Unicode", that unifies encodings which are different and incompatible according to computer type and language. ▸ Unicode [p. 251] is versatile and extensive and enables the best compatibility between RagTime versions and other programs on various computers or in various localizations and settings of the operating system. RagTime 5 uses Unicode for storing its documents and converts the characters according to the fonts and type of computer used.

5.5 ENCODING 256 - MAC OS STANDARD ROMAN

The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
128	Ä	Å	Ç	É	Ñ	Ö	Ü	á	à	â	ä	ã	å	ç	é	è
144	ê	ë	í	ì	î	ï	ñ	ó	ò	ô	ö	õ	ú	ù	û	ü
160	†	°	¢	£	§	•	¶	ß	®	©	™	´	¨	≠	Æ	Ø
176	∞	±	≤	≥	¥	µ	∂	Σ	Π	π	∫	ª	º	Ω	æ	ø
192	¿	¡	¬	√	ƒ	≈	Δ	«	»	…		À	Ã	Õ	Œ	œ
208	–	—	“	”	‘	’	÷	◊	ÿ	Ÿ	/	¤	‹	›	fi	fl
224	‡	·	‚	„	‰	Â	Ê	Á	Ë	È	Í	Î	Ï	Ì	Ó	Ô
240		Ò	Ú	Û	Ù	ı	ˆ	˜	¯	˘	˙	°	¸	˝	˛	ˇ

5.6 ENCODING 285 - MAC OS CENTRAL EUROPEAN ROMAN

The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
128	Ä	Ā	ā	É	Ą	Ö	Ü	á	ą	Č	ä	č	Ć	ć	é	Ź
144	ź	Ď	í	ď	Ē	ē	Ė	ó	ė	ô	ö	õ	ú	Ě	ě	ü
160	†	°	Ę	£	§	•	¶	ß	®	©	™	ę	¨	≠	ģ	į
176	Į	Ī	≤	≥	ī	Ķ	∂	Σ	ł	Ļ	ļ	Ľ	ľ	Ĺ	ĺ	Ņ
192	ņ	Ń	¬	√	ń	Ň	Δ	«	»	…		ň	Ő	Õ	ő	Ō
208	-	—	“	”	‘	’	÷	◇	ō	Ŕ	ŕ	Ř	<	>	ř	Ŗ
224	ŗ	Š	,	„	š	Ś	ś	Á	Ť	ť	Í	Ž	ž	Ū	Ó	Ô
240	ū	Ů	Ú	ů	Ű	ű	Ų	ų	Ý	ý	ķ	Ż	Ł	ż	Ģ	ˇ

5.7 ENCODING 262 - MAC OS GREEK

The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
128	Ä	¹	²	É	³	Ö	Ü	¨	à	â	ä	΄	¨	ç	é	è
144	ê	ë	£	™	î	ï	•	½	‰	ô	ö	¦	-	ù	û	ü
160	†	Γ	Δ	Θ	Λ	Ξ	Π	ß	®	©	Σ	Ϊ	§	≠	°	·
176	Α	±	≤	≥	¥	Β	Ε	Ζ	Η	Ι	Κ	Μ	Φ	Ψ	Ψ	Ω
192	ά	Ν	¬	Ο	Ρ	≈	Τ	«	»	…		Υ	Χ	Ά	Έ	œ
208	–	—	“	”	‘	’	÷	Ή	Ί	Ό	Ύ	έ	ή	ί	ό	Ώ
224	ύ	α	β	ψ	δ	ε	φ	γ	η	ι	ξ	κ	∂	μ	υ	ο
240	π	ώ	ρ	σ	τ	θ	ω	ς	χ	υ	ζ	ϊ	ϋ	ΐ	ΰ	ˇ

5.8 ENCODING 263 - MAC OS CYRILLIC

The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
128	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
144	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
160	†	°	¢	£	§	•	¶	І	®	©	™	Ђ	ђ	≠	Ѓ	ѓ
176	∞	±	≤	≥	і	µ	∂	Ј	Є	є	Ї	ї	Љ	љ	Њ	њ
192	ј	Ѕ	¬	√	ƒ	≈	Δ	«	»	…		Ћ	ћ	Ќ	ќ	ѕ
208	–	—	“	”	‘	’	÷	„	Ў	ў	Џ	џ	№	Ё	ё	я
224	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
240	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	¤

5.9 ENCODING 512 - MICROSOFT WINDOWS STANDARD ROMAN

The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
128	€	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	□	□
144	□	‘	’	“	”	•	–	—	˜	™	š	›	œ	□	□	Ÿ
160		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
176	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
192	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
208	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
224	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
240	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

5.10 Encoding 750 - Microsoft Windows Central European Roman

The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.

The highlighted Euro symbol € (Number 128 in this encoding, Unicode 8364) in the table is only available with appropriately equipped system software.

The character is supported under Windows 98 or Windows 95 or NT suitably updated with the corresponding service packs. If you have access to the World Wide Web, you will find further information under <http://officeupdate.microsoft.com>.

The character is supported by Mac OS 8.5 or earlier versions with suitably updated fonts and printer drivers. If the Euro symbol is not present the international currency symbol ¤ will appear. If you have access to the World Wide Web, you will find further information under <http://developer.apple.com/technotes/tn/tn1140.html>.

Besides the system, your font must also contain the Euro symbol. If necessary, please contact the supplier of the font.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
128	€	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Ś	Ť	□	Ź
144	□	‘	’	“	”	•	–	—	˜	™	š	›	ś	ť	□	ź
160		ˇ	˘	Ł	¤	Ą	¦	§	¨	©	Ş	«	¬	-	®	Ż
176	°	±	˛	ł	´	µ	¶	·	¸	ą	ş	»	Ľ	˝	ľ	ż
192	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
208	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
224	ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
240	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

5.11 ENCODING 716 - MICROSOFT WINDOWS CYRILLIC

The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.

The highlighted Euro symbol € (Number 136 in this encoding, Unicode 8364) in the table is only available with appropriately equipped system software.

The character is supported under Windows 98 or Windows 95 or NT suitably updated with the corresponding service packs. If you have access to the World Wide Web, you will find further information under <http://officeupdate.microsoft.com>.

The character is supported by Mac OS 8.5 or earlier versions with suitably updated fonts and printer drivers. If the Euro symbol is not present the international currency symbol ¤ will appear. If you have access to the World Wide Web, you will find further information under <http://developer.apple.com/technotes/tn/tn1140.html>.

Besides the system, your font must also contain the Euro symbol. If necessary, please contact the supplier of the font.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
128	Ђ	Ѓ	‚	ѓ	„	…	†	‡	€	‰	Љ	‹	Њ	Ќ	Ћ	Џ
144	ђ	‘	’	“	”	•	–	—	˜	™	љ	›	њ	ќ	ћ	џ
160		Ў	ў	Ј	¤	Ґ	¦	§	Ё	©	Є	«	¬	-	®	Ї
176	°	±	І	і	ґ	µ	¶	·	ё	№	є	»	ј	Ѕ	ѕ	ї
192	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
208	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
224	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
240	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

5.12 Encoding 673 - Microsoft Windows Greek

The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.

The highlighted Euro symbol € (Number 128 in this encoding, Unicode 8364) in the table is only available with appropriately equipped system software.

The character is supported under Windows 98 or Windows 95 or NT suitably updated with the corresponding service packs. If you have access to the World Wide Web, you will find further information under <http://officeupdate.microsoft.com>.

The character is supported by Mac OS 8.5 or earlier versions with suitably updated fonts and printer drivers. If the Euro symbol is not present the international currency symbol ¤ will appear. If you have access to the World Wide Web, you will find further information under <http://developer.apple.com/technotes/tn/tn1140.html>.

Besides the system, your font must also contain the Euro symbol. If necessary, please contact the supplier of the font.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
128	€	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	□	□
144	□	‘	’	“	”	•	–	—	˜	™	š	›	œ	□	□	Ÿ
160		΅	Ά	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	―
176	°	±	²	³	΄	µ	¶	·	Έ	Ή	Ί	»	Ό	½	Ύ	Ώ
192	ΐ	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
208	Π	Ρ	□	Σ	Τ	Υ	Φ	Χ	Ψ	Ω	Ϊ	Ϋ	ά	έ	ή	ί
224	ΰ	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
240	π	ρ	ς	σ	τ	υ	φ	χ	ψ	ω	ϊ	ϋ	ό	ύ	ώ	□

5.13 ENCODING 674 - MICROSOFT WINDOWS TURKISH

The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.

The highlighted Euro symbol € (Number 128 in this encoding, Unicode 8364) in the table is only available with appropriately equipped system software.

The character is supported under Windows 98 or Windows 95 or NT suitably updated with the corresponding service packs. If you have access to the World Wide Web, you will find further information under <http://officeupdate.microsoft.com>.

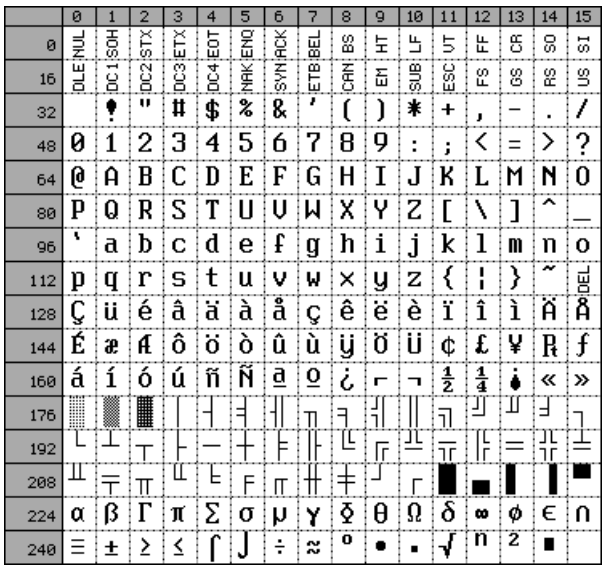
The character is supported by Mac OS 8.5 or earlier versions with suitably updated fonts and printer drivers. If the Euro symbol is not present the international currency symbol ¤ will appear. If you have access to the World Wide Web, you will find further information under <http://developer.apple.com/technotes/tn/tn1140.html>.

Besides the system, your font must also contain the Euro symbol. If necessary, please contact the supplier of the font.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
128	€	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	□	□
144	□	‘	’	“	”	•	–	—	˜	™	š	›	œ	□	□	Ÿ
160		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
176	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
192	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
208	Ğ	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	İ	Ş	ß
224	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
240	ğ	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ı	ş	ÿ

5.14 ENCODING 1024 - MICROSOFT DOS CODE PAGE 437

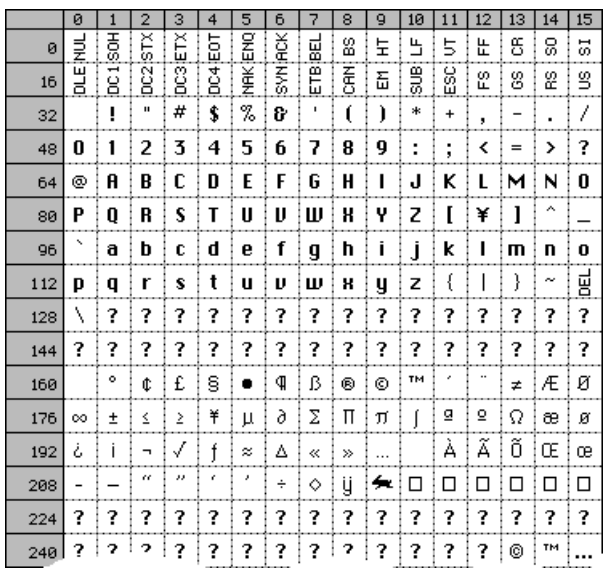
The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.



5.15 ENCODING -24319 - SHIFT JIS

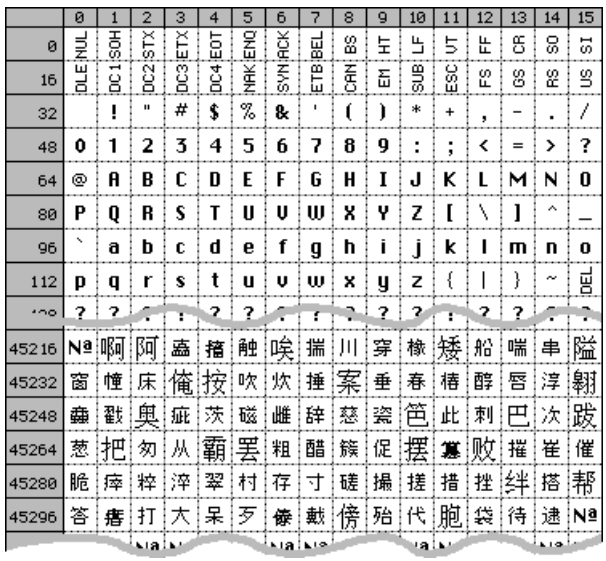
Shift JIS includes several thousand characters. A complete description excedes the limits of the RagTime documentation. You can get information about the Shift JIS encoding together with the Japanese script system for your computer and in the documentation of the keyboard layout and input methods.

The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.



5.16 Encoding -24295 - Simplified Chinese

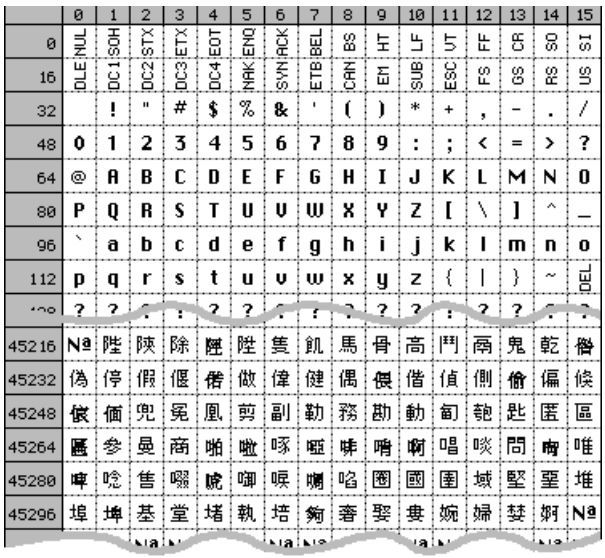
The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
45216	№	啊	阿	埃	挨	哎	唉	哀	皑	癌	蔼	矮	艾	碍	爱	隘
45232	鞍	氨	安	俺	按	暗	岸	胺	案	肮	昂	盎	凹	敖	熬	翱
45248	袄	傲	奥	懊	澳	芭	捌	扒	叭	吧	笆	八	疤	巴	拔	跋
45264	靶	把	耙	坝	霸	罢	爸	白	柏	百	摆	佰	败	拜	稗	斑
45280	班	搬	扳	般	颁	板	版	扮	拌	伴	瓣	半	办	绊	邦	帮
45296	答	帮	打	大	呆	歹	傣	戴	傥	殆	代	贷	袋	待	逮	№

5.17 ENCODING -24318 - TRADITIONAL CHINESE

The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.



5.18 ENCODING -32768 - UNICODE

Unicode includes tens of thousands characters of the languages of the world. A complete description excedes the limits of the RagTime documentation. Please refer to the following books:

The Unicode Consortium: The Unicode Standard - Worldwide Character Encoding - Version 1.0.- Reading, MA (Addison-Wesley Publishing Company, Inc.)

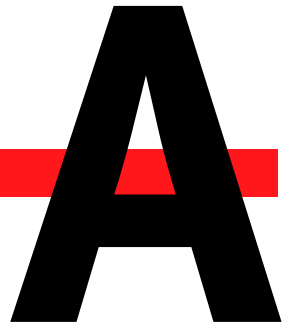
Volume 1 (1991): XX+682 pp; ISBN 0-201-56788-1. Volume 2 (1992): XII+441 pp; ISBN 0-201-60845-6.

If you have access to the World Wide Web, you will find further information under <http://www.unicode.org>

The characters are visible only if the appropriate script system is installed on your computer and you use the appropriate font. RagTime 5 uses a question mark to represent characters missing from a font.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
128	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
144	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
160		¡	¢	£	¤	¥	?	§	¨	©	ª	«	¬	?	®	¯
176	°	±	?	?	´	µ	¶	·	¸	?	º	»	?	?	?	¿
192	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
208	?	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	?	?	ß
224	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
240	?	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	?	?	ÿ
256	?															

APPENDIX



Legend

Fonts, styles, symbols and colors used in the RagTime 5 documentation are listed here.

A.1 CONVENTIONS

Fonts and Styles

COMMANDS, BUTTON NAMES AND ALL OTHER NAMES THAT APPEAR IN MENUS, DIALOG BOXES, WINDOWS, PALETTES, ETC. ARE WRITTEN IN THIS FONT.

`Formulas, value formats and similar entries are written in this font.`

Various Symbols

 **Windows**

 **Mac OS**

These symbols identify sections that refer to differences between RagTime 5 variants for Windows and Macintosh operating systems.

 **Remark**

Pointers and remarks are highlighted by this symbol.

 **Warning**

This symbol flags warnings that demand special attention.

References, Links and Hyperlinks

Whenever possible, references to other sections within the documentation are highlighted by underlined and/or colored text.

 If you read the documentation on-screen, you are taken to the reference by clicking the highlighted text. Of course, this applies to the on-screen documentation, but it works with PDF documents as well.

▸ **Reference [p. 999]**

In print material and PDF documents, this symbol appears before the reference, while the corresponding page number or document is indicated after the reference. References without a page number or a document refer to terms explained in the Glossary

Formulas and Functions

f **Function**

This symbol identifies the names of arithmetic functions.

Argument **Required Argument of a Function**

This style is used in the description of functions to identify arguments which must be supplied.

Argument **Optional Argument of a Function**

This style is used in the description of functions to identify arguments which may be omited.

" ' **Primitive quotation marks**

These quotation marks are only used to identify text within formulas.

␣ **Spaces**

Spaces that are of special importance are represented by this symbol.

A.2 KEYBOARD SYMBOLS

↩ **Return, carriage return**

⌅ **Enter**

Please note that return and enter serve different purposes in RagTime 5.

⇥ **Tabulator**

⎋ **Escape**

↑ **Up arrow**

↓ **Down arrow**

← **Left arrow**

→ **Right arrow**

⇞ **Page up or PgUp**

⇟ **Page down or PgDn**

Modifier Keys

Modifier keys only work in combination with other keys or mouse clicks.

alt **Alt**

altgr **AltGr**

⌘ **Command or Apple**

⌥ **Option**

^ **Control or ctrl**

⇧ **Shift**

A.3 USER INTERFACE ELEMENTS

Button or push button

Identifies buttons that trigger immediate action.

Radio button

Identifies a group of switches, of which only one can be turned on at a time.

Check box

Identifies a switch that can only be on or off.

Menu

Identifies a common menu (pull-down menu).

Submenu

Identifies a hierarchical menu.

Submenu title

Identifies the title of a hierarchical menu.

⌘ **Menu command**

Identifies an ordinary menu command.

✓ **Checkmark**

The checkmark identifies the menu's or palette's effective setting.

▼ **Menu**

Identifies menus that are coupled to tool buttons.

 **Pop-up menu**

Identifies a menu that pops up.

ⓘ **Display field**

Identifies display fields containing information that cannot be altered directly.

 **Entry field**

Identifies a field in which an entry can be made.

 **Entry field with a menu, combo box**

Identifies an entry field in which you can quickly enter frequently used values from a pop-up menu.

 **More/less**

Identifies a switch you can use to specify whether more rarely used elements should appear in a dialog box or window.

APPENDIX B

Glossary

The glossary provides definitions of the terms frequently used in RagTime 5 and the associated documentation.

Add On
An Add On is an external file which extends the functionality of RagTime. An Add On might provide, for example, functions, additional capability to a component or even a new component type.

Alt key
alt The Alt key.

AltGr key
alt^{gr} The AltGr key.

Argument
An argument is a value which is passed to a function and is the basis for its calculation. Arguments follow the name of a function and must be enclosed in parentheses. If a function requires two or more arguments, they must be separated by semicolons: `Sum (2; 3; 4)`. An argument can be, for example, a number, a constant, a reference or a formula.

Auxiliary
An auxiliary is a part of a RagTime document which appears in the inventory and which acts on other elements in various ways. You can create, modify and delete auxiliaries in editing dialog boxes.

Bézier Curve
A Bézier curve consists of line segments defined by end points and control points which define the tangent of a line segment on the end points.

Bit Depth
The bit depth is the number of bits used to define the color of a single pixel. A larger bit depth allows more colors but uses more memory and makes the file larger.

Border
A border is the line defining the edges of a drawing object.

Bounding Box
A bounding box is a rectangle with selecting handles enclosing curves and curved, transformed or irregular shapes. When you move or resize a bounding box, the object is moved or resized accordingly.

CIRC! (Error Value)
A formula contains a reference to itself or is in a chain of formulas which refers to itself at the “end”. This error value does not appear when iteration is turned on. (▸ error value)

Circular Reference
A circular reference in a formula is one that directly or indirectly refers to the cell in which the formula containing the reference is located.

Command key
⌘ The Macintosh Command key, which is also labled with an Apple on some keyboards.

COMPLEX! (Error Value)
The formula is too complex; it contains too many embedded parentheses or calls to subfunctions. (▸ error value)

Component
Components include layouts, master layouts, text, spreadsheets, graphs, drawings, pictures, sound, movies and buttons. The components of a document are listed in the inventory and can be displayed in windows or installed in various objects.

Container
A container is an area, for example a drawing object, in which a component, for example, a text, may be installed, viewed and edited.

Contents
The term “contents” refers to a component installed in a container.

Control key
^ The Control key. It is labled as “Ctrl” on some keyboards.

Converter
A converter (also called filter or translator) is a document containing special information needed to import documents created with other applications. Converters cannot be opened.

Data
Data is information which you enter into your computer. Data may be, for example, words, numbers, dates, pictures, graphics and so on.

Date

A date is a specific point in time. In RagTime, a date consists of a year, month day, hour, minute, and second. A date can be formatted so that only some of these parts are displayed (`3.4.1995 9:30:04; 3. April 1995; 03.04.95; 9:30; 9:30:04`).

Dates in the range from 1/1/0001 to 12/31/29999 are valid. An error value is displayed when you use dates outside this range.

Because the Gregorian calender, which is the basis for RagTime's date functions, was introduced in 1582, calculations with earlier dates have limited meaning. Please be also be aware that some countries introduced the Gregorian calender significantly later.

DATE! (Error Value)

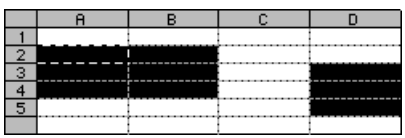
A date is outside the supported range, or a part of a date has an invalid value, for example 13 as month. (▸ error value)

Default Value

A default value is a value, action or setting that RagTime assumes until you give it a different instruction. A default value is also called a pre-set value. You can set default values in the "document settings" or "settings" dialog boxes.

Discontinuous Cell Range

A discontinuous range consists of two or more cells or ranges not sharing a common boundary.



DIV/0! (Error Value)

Division by zero was attempted. (▸ error value)

Down arrow key

↓ The key with the arrow pointing down.

Enter key

⌅ The Enter key is often shortened to simply "Enter". It is frequently confused with the Return key (↩). In RagTime however, these two keys frequently have different functions.

Error Value
An error value is the result of an erroneous, undefined or impermissible calculation. Error values propagate themselves in the results of all formulas in which an error value is used as an argument or operator. Error values are always displayed in upper case letters followed by an exclamation mark.

ERROR! (Error Value)
An error which does not fit in any other category. (▸ error value)

Escape key
⎋ The Escape key.

EVAL! (Error Value)
The formula uses the result of a calculation which returned the error value `CIRC!`. (▸ error value)

Export
To export means to save a selection in the form of an exchange format file, such as plain text, to make it available to other programs.

File
A file is any named collection of information stored on a disk. For example, documents created with RagTime are files.

Fill
The fill is the area of a drawing object, spreadsheet cell or graph label or title. A fill may have a color, a pattern or a gradient or be transparent.

Formula
A "formula" is a rule for calculating a value. Formulas, which you can enter in the formulas palette, can be used in various components. A formula may be complex or as simple as a cell reference or "1+1". Formulas can, but need not, include functions.

Function
A function is a predefined set of actions that perform a calculation using values you specify or supply.

Functions can be inserted in formulas to do mathematical, text, date, search, print and other operations.

ILLEGAL! (Error Value)
The formula uses a function which is not installed. (▸ error value)

Import
To import material means to insert the data in one document into a second document.

Insertion Marker
The insertion marker is a blinking line indicating where text will be inserted in a document. You can position the insertion marker by clicking the pointer over text.

|

Interval
An interval describes a limited set of values. The lower and upper limits are given in square brackets: `[-1; +1]`.

If the bracket opens away from the value, it is not included in the interval. For example, `[0; 1[` includes all numbers equal to or greater than 0 to and less than 1.

Inventory
The inventory is a list of components, style sheets and certain other elements, such as rulers and units, contained in a document. You can open components and style sheet editors by double-clicking the name in the inventory list or drag items onto document pages or to other documents.

Left and Right of Axes and Lines
Because lines and axes may be rotated, "left" and "right" are defined in terms of the "direction of motion." For lines, the direction is defined to be from the starting point to the end point. For value axes, the direction is defined to be in the direction of ascending values; for category axes, the sequence of the categories.

For closed drawing objects, left is the equivalent of outside and right, inside. Consequently, when you open or close a polygon or Bézier curve, the line may be drawn in a different position.

Left arrow key
← The key with the arrow pointing left.

Line
Straight lines and arcs are lines. Lines cannot contain components. Open polygons and Bézier curves are lines in the sense that they can have arrowheads, but they are not lines in the sense that they can contain components.

Line Spacing
Line spacing is the vertical space between lines of text measured from baseline to baseline.

List
A list consists of one or more values (constants or references, including references to ranges). In formulas, the elements in a list are separated with semicolons (for example, `47; 11; 8; 15` or `A1; B7; C1:D3` or `Z1S1; Z7S2; Z1S3:Z3S4`).

Magnetic Grid
The magnetic grid helps size and align objects precisely on the page. When you drag or create an object, its borders snap to the nearest grid lines. You can use the magnetic grid whether or not the grid is visible.

Mixed
If the selection includes more than one variation for a single attribute, the setting for that attribute is "mixed." For example, if a single text selection includes Chicago and Geneva fonts, the setting for fonts is "mixed."

Multiline Text
Multiline text is broken at the end of a line in a spreadsheet cell.

NA! (Error Value)
A value is not available because, for example, it could not be found in a spreadsheet, or an operand or argument has the constant value "NA". (▸ error value)

NAME! (Error Value)
This error is never caused by RagTime and can only occur in spreadsheets imported from Excel. The error is supported to provide compatibility with Excel. (▸ error value)

NULL! (Error Value)
This error is never caused by RagTime and can occur in spreadsheets imported from Excel. The error is supported to provide compatibility with Excel. RagTime Connect returns this error value for database fields containing no data. (▸ error value)

NUM! (Error Value)
An undefined operation was attempted, or the result is outside the range of numbers possible in RagTime. (▸ error value)

Number
A number is a numerical amount. RagTime can work with positive and negative integers and decimal fractions. In addition to the common forms of notations, RagTime supports exponential notation and thousands separators (`42; -4711; 123.45; -2,987.65; 1.41e8`).

Object
In the most general sense, an object is any element that you can select. Often, the term "object" is used as a short form of "drawing objects," for example, lines, rectangles and so on.

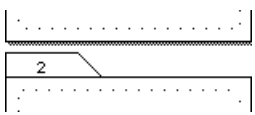
Operand
An operand is a value acted on to perform a calculation in a formula. In the equation "1+2=3" "1" and "2" are operands.

Operators
Operators are symbols used in a formula to define the action to be performed, for example "+" for addition.

Option key
⌥ The Option key.

Page Down key
⇟ The Page Down key.

Page Gap
A page gap is the space above the first page, between pages and after the last page of a document.

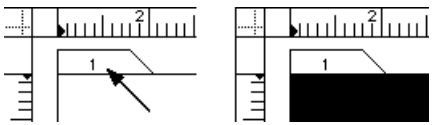


Click anywhere in this gap to select it. A new page may then be inserted in or appended at that particular place, or a copied page may be pasted.

You can also drag pages from other documents to the page gap.

Page Size
The page size is the area defined for each page of the document. Document pages larger than the available paper size can be printed on several sheets of paper.

Page Tag
A page tag is a small extension at the top of a page displaying the page number and name, if any. Click this tag to select the entire page and activate the layout component.

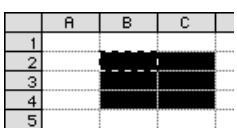


Page Up key
The Page Up key.

Palette
A palette is a window which is always in front of other windows. Tool palettes make frequently used commands quickly accessible. You can tear off some submenus to create palettes which stay open while you are working on a document. In the submenu of the command “Palettes” in the Windows menu, you can find a number of tool palettes.

Panel
Panels are subdivisions of extensive dialog boxes or windows in which related settings are grouped.
Panel names are displayed on tabs above the current panel.
Icons with the panel names appear in a list on the left side of the dialog box.

Range
A range is a contiguous, rectangular group of spreadsheet cells. In formulas, a range is defined by its upper left and lower right cells separated with a colon (for example `B2:C4`, `Z2S2:Z4S3` or `[1]A1:[3]D4`).



RANGE! (Error Value)
A value is outside the permitted range. (▸ error value)

REF! (Error Value)
A formula uses a reference to a nonexistant object. Generally, the referenced object has been deleted after the formula was entered. (▸ error value)

Return
Calculating a function or a formula results in a value. The function or formula "returns" this value, which may then appear, for example, in a spreadsheet, or be used in calculating another function or a formula.

Return key
↩ The Return key is also known as the carriage return. It is often confused with the Enter key (⌅). In RagTime however, these two keys frequently have different functions. Only use this key at the end of a paragraph not at the end of lines.

Right arrow key
⇢ The key with the arrow pointing right.

Rotation Point
The rotation point is a small circle which appears in the center of a selected drawing object. When the pointer moves over it, it changes to the rotation tool pointer.

Scope
The scope is the region in which certain operations, for example, search and replace or checking spelling, are performed. Possible scopes are components, layouts, windows, all open documents and so onl.

Screening
Screening is a process for printing pictures on a black-and-white printer in which gray tones are reproduced by a pattern of black dots or lines on white paper. If the dots are small enough, they will seem gray from a normal viewing distance.

Script
A script is a set of commands which you can activate with a single command. You can use scripts to automate repetitive tasks. Scripts allow numerous programs to work together.

Search Expression
A search expression is understood as the characters, words or special characters that should be found using the command Search. It is also possible to consider the format when searching.

Selection Handles
Selection handles are black squares appearing on the border of a drawing object, a bounding box or the ends of a line when an object is selected. Drag them to change the size or form of the object.

Shift key
⇧ The Shift key.

Special Character
Special characters are not normally visible in text, but influence the appearance of text, for example, blank spaces, tabulators and paragraph ends.

Tab key
⇥ The Tab key.

Tearing Off Menus
To create a palette by tearing off a submenu, move the pointer up, down or to the right through the submenu.

Text
A text is a sequence of letters and other characters.

In formulas, text must be enclosed in single or double quotation marks (for example, `'This is a text'` or `"Oh no!*2$%&#@"`).

Time Span
A time span is a period of time.

In formulas, a time span can be given in the units day [d], hour [h], minute [m], and second [s] (for example `3d 5h 2m 28s`).

At least two units must be given in order for a time span to be recognized. If necessary, use a 0 as the second value: `5h 0m`.

Transform
To transform a drawing object means to modify it by rotating, skewing or scaling.

Truth Value
A truth value is either `True` or `False`.

When converted to numerical values, `False` has the value 0 and `True`, 1. If numbers are converted to truth values, 0 has the value `False` and all numbers ≠ 0 are `True`.

Up arrow key
↑ The key with the arrow pointing up.

Value
A value is a number, a text, a reference, a date, a truth value, or an error value. In other words, this expression includes all possible sorts of arguments for functions that RagTime supports.

VALUE! (Error Value)
An operator or argument has the wrong type, for example, a date instead of a number. (▸ error value)

APPENDIX C

Selected Readings

In this chapter, you will find selected readings on RagTime and its accessories.

C.1 BUSINESS PUBLISHING

Business Publishing mit RagTime (in German)

Thomas Maschke (2001): Business Publishing mit RagTime 5.5 (Macintosh/Windows Version).- XIV + 298 S. 245 Abb., 1 CD-ROM; Berlin/Heidelberg (Springer-Verlag) ISBN: 3-540-66438-6.
<http://www.springer.de>

Handbuch der Printmedien (in German)

Helmut Kipphan (2000): Handbuch der Printmedien. Technologien und Produktionsverfahren. XVIII + 1246 S. 1274 Abb., 85 Tab., mit CD-ROM. Geb.; Berlin/Heidelberg (Springer-Verlag) ISBN: 3-540-66941-8.
<http://www.springer.de>

Digitales Colormanagement (in German)

Jan-Peter Homann (2000): Digitales Colormanagement. Farbe in der Publishing-Praxis. (Macintosh/Windows Version).- 2. erw. Auflage. 267 S. 200 Abb., 130 in Farbe, 1 CD-ROM; Berlin/Heidelberg (Springer-Verlag) ISBN: 3-540-66274-X.
<http://www.springer.de>

APPENDIX D

Index

General index entries usually refer to a section of text, i.e. the page number refers to a section's header (light typeface against a dark background). If a section carries over onto the subsequent page, the entry may also appear on the subsequent page.

Commands, buttons, menu entries, etc. are listed in the index in the order in which they appear in the application and documentation. The entries are highlighted using fonts, styles and symbols that are summarized in the ▸ Legend [p. 253].

Many key words are listed alphabetically as well as under the following categories: commands, definitions, document settings, entry fields, functions, basic settings, buttons, menus, palettes, panels, overviews, style sheets, tools and tool bars.

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Y