Table of Contents

Table of Contents

Table of Contents

TWiki Spreadsheet Plugin

This plugin adds spreadsheet capabilities to TWiki topics. Functions such as **`%CALCULATE{$INT(7/3)}%`** are evaluated at page view time. They can be placed in table cells and outside of tables. In other words, this plugin provides general function evaluation capability, not just classic spreadsheet functions. The plugin currently has 126 functions.

Example: ***Raw text:***

Region:	Sales:
Americas	732
Europe	611
Asia	228
Total:	1571

```
| *Region:* | *Sales:* |
| Americas  |  732 |
| Europe    |  611 |
| Asia      |  228 |
| Total: |  %CALC{$SUM($ABOVE())}% |
```

Interactive example:

```
%CALCULATE{                                                                    }%
```

Result: TWiki Guest

Syntax Rules

This plugin handles the **`%CALC{...}%`** and **`%CALCULATE{...}%`** variables. Embedded formulas are evaluated as follows:

- Built-in function are of format **`$FUNCNAME(parameters)`**
- Functions may be nested, such as **`%CALC{$SUM( R2:C$COLUMN(0)..R$ROW(-1):C$COLUMN(0) )}%`**
- Functions are evaluated from left to right, and from inside to outside if nested
- A function parameter can be text, a mathematical formula, a cell address, or a range of cell addresses
- Multiple parameters form a list; they are separated by a comma, followed by optional space, such as **`%CALCULATE{$SUM( 3, 5, 7 )}%`**
- A parameter representing a string can optionally be enclosed in triple quotes; quotes are required if the string contains commas, parenthesis or newlines, such as **`%CALCULATE{$SUBSTITUTE('''Good, early morning''', morning, day)}%`**
- The `%CALCULATE{...}%` variable may span multiple lines, which is useful to indent nested functions. In case the variable contains newlines, all white space around functions and function parameters is removed. Sample nested formula:

```
%CALCULATE{
  $LISTJOIN(
    $n,
    $LISTEACH(
      | $index | $item |,
      one, two, three
    )
  )
}%
```

- A table cell can be addressed as **`R1:C1`**. Table address matrix:

`R1:C1`	**`R1:C2`**	**`R1:C3`**	**`R1:C4`**
`R2:C1`	**`R2:C2`**	**`R2:C3`**	**`R2:C4`**

- A table cell range is defined by two cell addresses separated by **`".."`**, e.g. "row 1 through 20, column 3" is: **`R1:C3..R20:C3`**

- Lists can refer to values and/or table cell ranges, such as **`%CALC{$SUM( 3, 5, $T(R1:C7), R1:C11..R1:C15 )}%`**
- Functions can only reference cells in the current or preceding row of the current table; they may not reference cells below the current table row
- Functions can also be placed outside of tables; they can reference cells in the preceding table
- Functions can be placed in a FormattedSearch, but the CALCULATE needs to be escaped. Learn how to use a CALCULATE in a formatted search
- Plain text can be added, such as **`%CALC{Total: $SUM($ABOVE()) kg}%`**

Use CALC or CALCULATE

Use `%CALC{...}%` *in* table cells:
The CALC variable handles all functions, but it gets handled with delay compared to other TWikiVariables: It gets executed after internal variables and plugin variables that use the register tag handler. You may get unexpected results if you nest CALC inside other variables (such as `%INCLUDE{%CALC{...}%}%`) because it does not get evaluated inside-out & left-to-right like ordinary TWiki variables.

Use `%CALCULATE{...}%` *outside* tables:
The CALCULATE variable is handled inside-out & left-to-right like ordinary TWiki variables, but it does ***not*** support functions that refer to table cells, such as `$LEFT()` or `$T()`.

Built-in Spreadsheet Plugin Functions

The plugin currently has 126 functions. Convention for parameters:

- Required parameters are indicated in **`( bold )`**
- Optional parameters are indicated in **`( bold italic )`**

ABOVE( ) -- address range of cells above the current cell

- Syntax: **`$ABOVE( )`**
- Example: **`%CALC{$SUM($ABOVE())}%`** returns the sum of cells above the current cell
- Related: `$LEFT()`, `$RIGHT()`

ABS(num) -- absolute value of a number

- Syntax: **`$ABS( num )`**
- Example: **`%CALCULATE{$ABS(-12.5)}%`** returns **`12.5`**
- Related: `$SIGN()`, `$EVEN()`, `$ISDIGIT()`, `$ODD()`

ADDLIST(name, list) -- append a list to a list variable

- Specify the variable name (alphanumeric characters and underscores) and the list to add. If the named list does not exist it will be created. Useful in a FormattedSearch to add items to a list. This function returns no output. Use `$GETLIST()` to retrieve a list variable.
- Syntax: **`$ADDLIST( name, list )`**
- Example:
  `%CALCULATE{$SETLIST(nums, 0, 1, 2, 3)}%` sets the `nums` variable to `0, 1, 2, 3`
  `%CALCULATE{$ADDLIST(nums, 4, 5, 6, 7)}%` adds `4, 5, 6, 7` to the `nums` variable
  `%CALCULATE{$GETLIST(nums)}%` returns `0, 1, 2, 3, 4, 5, 6, 7`

- Example formatted search: **`format="$percntCALCULATE{$ADDLIST(topics, $topic)}$percnt"`** in each search hit, adds the topic name to the `topics` list variable
- Related: `$GETLIST()`, `$SET()`, `$SETHASH()`, `$SETLIST()`

AND(list) -- logical AND of a list

- Syntax: **`$AND( list )`**
- Example: **`%CALCULATE{$AND(1, 0, 1)}%`** returns **`0`**
- Related: `$NOT()`, `$IF()`, `$OR()`, `$WHILE()`, `$XOR()`

AVERAGE(list) -- average of a list or a range of cells

- Syntax: **`$AVERAGE( list )`**
- Example: **`%CALC{$AVERAGE(R2:C5..R$ROW(-1):C5)}%`** returns the average of column 5, excluding the title row
- Related: `$LIST()`, `$MAX()`, `$MEDIAN()`, `$MIN()`, `$STDEV()`, `$STDEVP()`, `$VAR()`, `$VARP()`

BIN2DEC(num) -- convert a binary number to decimal

- Syntax: **`$BIN2DEC( num )`**
- Example: **`%CALCULATE{$BIN2DEC(1100100)}%`** returns **`100`**
- Related: `$DEC2BIN()`, `$DEC2HEX()`, `$DEC2OCT()`, `$HEX2DEC()`, `$HEXDECODE()`, `$HEXENCODE()`, `$OCT2DEC()`

BITXOR(text) -- bit-wise XOR of text

- Each bit of each character of `text` is flipped, useful to obfuscate text. Bit-XORing text twice will restore the original text.
- Syntax: **`$BITXOR( text )`**
- Example: **`%CALCULATE{$BITXOR(A123)}%`** returns **`¾ÎÍÌ`**
- Example: **`%CALCULATE{$BITXOR($BITXOR(anything))}%`** returns **`anything`**
- Related: `$HEXDECODE()`, `$HEXENCODE()`, `$LEFTSTRING()`, `$RANDSTRING()`, `$REPLACE()`, `$RIGHTSTRING()`, `$SUBSTITUTE()`, `$TRANSLATE()`, `$XOR()`

CEILING(num) -- return the smallest integer following a number

- The ceiling(x) is the smallest integer not less than x
- Use `$INT()` to truncate a number towards zero; use `$FLOOR()` to get the largest previous integer
- Syntax: **`$CEILING( num )`**
- Example: **`%CALCULATE{$CEILING(5.4)}%`** returns **`6`**
- Example: **`%CALCULATE{$CEILING(-5.4)}%`** returns **`-5`**
- Related: `$EVAL()`, `$FLOOR()`, `$INT()`, `$ROUND()`, `$VALUE()`

CHAR(number) -- ASCII character represented by number

- Syntax: **$CHAR(number)**
- Example: Example: **%CALCULATE{$CHAR(97)}%** returns **a**
- Related: $CODE()

CODE(text) -- ASCII numeric value of character

- The ASCII numeric value of the first character in text
- Syntax: **$CODE(text)**
- Example: **%CALCULATE{$CODE(abc)}%** returns **97**
- Related: $CHAR()

COLUMN(offset) -- current column number

- The current table column number with an optional offset. When used outside of a table it returns the number of columns of the preceding table.
- Syntax: **$COLUMN(*offset*)**
- Example: **%CALC{$COLUMN()}%** returns **2** for the second column
- Related: $ROW(), $T()

COUNTITEMS(list) -- count individual items in a list

- Syntax: **$COUNTITEMS(list)**
- Example: **%CALC{$COUNTITEMS($ABOVE())}%** returns **Closed: 1, Open: 2** assuming one cell above the current cell contains **Closed** and two cells contain **Open**
- Related: $COUNTSTR(), $LIST(), $SETMHASH(), $WHILE()

COUNTSTR(list, str) -- count the number of cells in a list equal to a given string

- Count the number of cells in a list equal to a given string (if str is specified), or counts the number of non empty cells in a list
- Syntax: **$COUNTSTR(list, *str*)**
- Example: **%CALC{$COUNTSTR($ABOVE())}%** counts the number of non empty cells above the current cell
- Example: **%CALC{$COUNTSTR($ABOVE(), DONE)}%** counts the number of cells equal to **DONE**
- Related: $COUNTITEMS(), $LIST(), $SETMHASH(), $WHILE()

DEC2BIN(num, width) -- convert a decimal number to binary

- Syntax: **$DEC2BIN(num, *width*)**
- Example: **%CALCULATE{$DEC2BIN(100)}%** returns **1100100**
- Example: **%CALCULATE{$DEC2BIN(9, 6)}%** returns **001001**
- Related: $BIN2DEC(), $DEC2HEX(), $DEC2OCT(), $HEX2DEC(), $HEXDECODE(), $HEXENCODE(), $OCT2DEC()

DEC2HEX(num, width) -- convert a decimal number to hexadecimal

- Syntax: **$DEC2HEX(num, *width*)**
- Example: **%CALCULATE{$DEC2HEX(165)}%** returns **A5**
- Example: **%CALCULATE{$DEC2HEX(100, 4)}%** returns **0064**
- Related: $BIN2DEC(), $DEC2BIN(), $DEC2OCT(), $HEX2DEC(), $HEXDECODE(), $HEXENCODE(), $OCT2DEC()

DEC2OCT(num, width) -- convert a decimal number to octal

- Syntax: **$DEC2OCT(num, *width*)**
- Example: **%CALCULATE{$DEC2OCT(58)}%** returns **72**
- Example: **%CALCULATE{$DEC2OCT(58, 4)}%** returns **0072**
- Related: $BIN2DEC(), $DEC2BIN(), $DEC2HEX(), $HEX2DEC(), $HEXDECODE(), $HEXENCODE(), $OCT2DEC()

DEF(list) -- find first non-empty list item or cell

- Returns the first list item or cell reference that is not empty
- Syntax: **$DEF(list)**
- Example: **%CALC{$DEF(R1:C1..R1:C3)}%**
- Related: $COUNTSTR(), $LISTIF(), $LIST()

EMPTY(text) -- test for empty text

- Returns **1** if text is empty, or **0** if not
- Syntax: **$EMPTY(*text*)**
- Example: **%CALCULATE{$EMPTY(foo)}%** returns **0**
- Example: **%CALCULATE{$EMPTY()}%** returns **1**
- Example: **%CALCULATE{$EMPTY($TRIM( ))}%** returns **1**
- Related: $EQUAL(), $EXACT(), $IF(), $ISDIGIT(), $ISLOWER(), $ISUPPER(), $ISWIKIWORD(), $TRIM(), $WHILE()

EQUAL(text1, text2) -- compare two text strings, case-insensitive

- Compares two text strings and returns **1** if they are the same ignoring the case, or **0** if not; use $EXACT() for case-sensitive compare
- Syntax: **$EQUAL(text1, *text2*)**
- Example: **%CALCULATE{$EQUAL(foo, Foo)}%** returns **1**
- Related: $EMPTY(), $EXACT(), $IF(), $ISDIGIT(), $ISLOWER(), $ISUPPER(), $ISWIKIWORD(), $TRIM()

EVAL(formula) -- evaluate a simple mathematical formula

- Addition, substraction, multiplication, division and modulus of numbers are supported. Any nesting is permitted
- Numbers may be decimal integers (1234), binary integers (0b1110011), octal integers (01234), hexadecimal integers (0x1234) or of exponential notation (12.34e-56)

- Syntax: **$EVAL(formula)**
- Example: **%CALCULATE{$EVAL((5 * 3) / 2 + 1.1)}%** returns **8.6**
- Related: $CEILING(), $EXEC(), $FLOOR(), $INT(), $MOD(), $ROUND(), $VALUE()

EVEN(num) -- test for even number

- Syntax: **$EVEN(num)**
- Example: **%CALCULATE{$EVEN(2)}%** returns **1**
- Related: $ABS(), $ISDIGIT(), $MOD(), $ODD(), $SIGN()

EXACT(text1, text2) -- compare two text strings, case-sensitive

- Compares two text strings and returns **1** if they are exactly the same, or **0** if not; use $EQUAL() for case-insensitive compare
- Syntax: **$EXACT(text1, *text2*)**
- Example: **%CALCULATE{$EXACT(foo, Foo)}%** returns **0**
- Example: **%CALCULATE{$EXACT(foo, $LOWER(Foo))}%** returns **1**
- Related: $EMPTY(), $IF(), $ISDIGIT(), $ISLOWER(), $ISUPPER(), $ISWIKIWORD(), $TRIM()

EXEC(formula) -- execute a spreadsheet formula

- Execute a spreadsheet formula, typically retrieved from a variable. This can be used to store a formula in a variable once and execute it many times using different parameters.
- Syntax: **$EXEC(formula)**
- Example: **%CALCULATE{$SET(msg, $NOEXEC(Hi $GET(name)))}%** sets the msg variable with raw formula Hi $GET(name)
- Example: **%CALCULATE{$SET(name, Tom) $EXEC($GET(msg))}%** executes content of msg variable and returns Hi Tom
- Example: **%CALCULATE{$SET(name, Jerry) $EXEC($GET(msg))}%** returns Hi Jerry
- Related: $EVAL(), $GET(), $NOEXEC(), $SET()

EXISTS(topic) -- check if topic exists

- Topic can be TopicName or a Web.TopicName. Current web is used if web is not specified.
- Syntax: **$EXISTS(topic)**
- Example: **%CALCULATE{$EXISTS(WebHome)}%** returns **1**
- Example: **%CALCULATE{$EXISTS(ThisDoesNotExist)}%** returns **0**
- Related: $EQUAL(), $EXACT(), $IF(), $ISWIKIWORD(), $TRIM()

EXP(num) -- exponent (e) raised to the power of a number

- EXP is the inverse of the LN function
- Syntax: **$EXP(num)**
- Example: **%CALCULATE{$EXP(1)}%** returns **2.71828182845905**
- Related: $LN(), $LOG()

FILTER(expression, text) -- filter out characters from text

- Remove characters from a `text` string. The filter is applied multiple times.
- The `expression` can be a sequence of characters or a RegularExpression. Use tokens in the `expression` if needed: `$comma` for comma, `$sp` for space. The `text` may contain commas.
- Syntax: **`$FILTER( chars, text )`**
- Example: **`%CALCULATE{$FILTER(f, fluffy)}%`** returns **luy** - filter out a character multiple times
- Example: **`%CALCULATE{$FILTER(an Franc, San Francisco)}%`** returns **Sisco** - cut a string
- Example: **`%CALCULATE{$FILTER($sp, Cat and Mouse)}%`** returns **CatandMouse** - remove all spaces
- Example: **`%CALCULATE{$FILTER([^0-9], Project-ID-1234)}%`** returns **1234** - filter in digits, e.g. keep only digits
- Example: **`%CALCULATE{$FILTER([^a-zA-Z0-9 ], Stupid mistake*%@^! Fixed)}%`** returns **Stupid mistake Fixed** - keep only alphanumeric characters and spaces
- Example: **`%CALCULATE{$FILTER([^a-zA-Z0-9], $PROPER(an EXELLENT idea.))}%`** returns **AnExcellentIdea** - turn a string into a WikiWord topic name
- Related: `$FIND()`, `$INSERTSTRING()`, `$LEFTSTRING()`, `$PROPER()`, `$REPLACE()`, `$RIGHTSTRING()`, `$SEARCH()`, `$SUBSTITUTE()`, `$SUBSTRING()`, `$TRANSLATE()`, `$TRIM()`

FIND(string, text, start) -- find one string within another string

- Finds one text `string`, within another `text`, and returns the number of the starting position of `string`, from the first character of `text`. This search is case sensitive and is not a regular expression search; use `$SEARCH()` for regular expression searching. Starting position is 1; a 0 is returned if nothing is matched.
- Syntax: **`$FIND( string, text, start )`**
- Example: **`%CALCULATE{$FIND(f, fluffy)}%`** returns **1**
- Example: **`%CALCULATE{$FIND(f, fluffy, 2)}%`** returns **4**
- Example: **`%CALCULATE{$FIND(@, fluffy, 1)}%`** returns **0**
- Related: `$FILTER()`, `$INSERTSTRING()`, `$LEFTSTRING()`, `$REPLACE()`, `$RIGHTSTRING()`, `$SUBSTITUTE()`, `$SUBSTRING()`, `$SEARCH()`

FLOOR(num) -- return the largest integer preceding a number

- The floor(x) is the largest integer not greater than x
- Use `$INT()` to truncate a number towards zero; use `$CEILING()` to get the smallest following integer
- Syntax: **`$FLOOR( num )`**
- Example: **`%CALCULATE{$FLOOR(5.4)}%`** returns **5**
- Example: **`%CALCULATE{$FLOOR(-5.4)}%`** returns **-6**
- Related: `$CEILING()`, `$EVAL()`, `$INT()`, `$ROUND()`, `$VALUE()`

FORMAT(type, precision, number) -- format a number to a certain type and precision

- Supported `type`:
  - `COMMA` for comma format, such as `12,345.68`

    - ♦ CURRENCY for currency format, such as €12,345.68
        - ◊ Negative numbers are shown in parenthesis, such as (€6,789.00) for -6789
        - ◊ The currency symbol is taken from the CURRENCYSYMBOL preferences setting, such as US$ or $ for Dollar, € for Euro, ¥ for Yen, default $
    - ♦ DOLLAR for Dollar format, such as $12,345.68
        - ◊ Negative numbers are shown in parenthesis, such as ($6,789.00) for -6789
    - ♦ KB for Kilo Byte format, such as 1205.63 KB
    - ♦ MB for Mega Byte format, such as 1.18 MB
    - ♦ KBMB for Kilo/Mega/Giga/Tera Byte auto-adjust format
    - ♦ NUMBER for number, such as 12345.7
    - ♦ PERCENT for percent format, such as 12.3%
- The precision indicates the the number of digits after the dot
- Syntax: **$FORMAT(type, prec, number)**
- Example: **%CALCULATE{$FORMAT(COMMA, 2, 12345.6789)}%** returns **12,345.68**
- Example: **%CALCULATE{$FORMAT(CURRENCY, 2, 12345.6789)}%** returns **$12,345.68**
- Example: **%CALCULATE{$FORMAT(DOLLAR, 2, 12345.6789)}%** returns **$12,345.68**
- Example: **%CALCULATE{$FORMAT(KB, 2, 1234567)}%** returns **1205.63 KB**
- Example: **%CALCULATE{$FORMAT(MB, 2, 1234567)}%** returns **1.18 MB**
- Example: **%CALCULATE{$FORMAT(KBMB, 2, 1234567)}%** returns **1.18 MB**
- Example: **%CALCULATE{$FORMAT(KBMB, 2, 1234567890)}%** returns **1.15 GB**
- Example: **%CALCULATE{$FORMAT(NUMBER, 1, 12345.67)}%** returns **12345.7**
- Example: **%CALCULATE{$FORMAT(PERCENT, 1, 0.1234567)}%** returns **12.3%**
- Related: $FORMATTIME(), $FORMATTIMEDIFF(), $ROUND()

FORMATGMTIME(serial, text) -- convert a serialized date into a GMT date string

- The date string represents the time in Greenwich time zone. Same variable expansion as in $FORMATTIME().
- Syntax: **$FORMATGMTIME(serial, text)**
- Example: **%CALCULATE{$FORMATGMTIME(1041379200, $day $mon $year)}%** returns **01 Jan 2003**
- Related: $FORMATTIME(), $FORMATTIMEDIFF(), $TIME(), $TIMEADD(), $TIMEDIFF(), $TODAY()

FORMATTIME(serial, text) -- convert a serialized date into a date string

- The following variables in text are expanded:
    - ♦ $second - seconds, 00..59
    - ♦ $minute - minutes, 00..59
    - ♦ $hour - hours, 00..23
    - ♦ $day - day of month, 01..31
    - ♦ $month - month, 01..12
    - ♦ $mon - month in text format, Jan..Dec
    - ♦ $year - 4 digit year, 1999
    - ♦ $ye - 2 digit year, 99
    - ♦ $wd - day number of the week, 1 for Sunday, 2 for Monday, etc
    - ♦ $wday - day of the week, Sun..Sat
    - ♦ $weekday - day of the week, Sunday..Saturday
    - ♦ $yearday - day of the year, 1..365, or 1..366 in leap years
    - ♦ $isoweek - ISO 8601 week number, one or two digits, 1..53

    - `$isoweek(format)` - formatted ISO 8601 week number. These variables are expanded in `format`:
        - `$isoweek($year)` - year of ISO 8601 week number, such as `2009` for 2010-01-03
        - `$isoweek($wk)` - 2 digit ISO 8601 week number, such as `53` for 2010-01-03
        - `$isoweek($day)` - day of ISO 8601 week number, starting with 1 for Monday, such as `7` for 2010-01-03
        - `$isoweek($iso)` - full year-week ISO week number, such as `2009-W53` for 2010-01-03
        - `$isoweek($yearW$wk$day)` - full year-week-day ISO week number, such as `2009W537` for 2010-01-03
        - `$isoweek($year-W$wk-$day)` - full year-week-day ISO week number, such as `2009-W53-7` for 2010-01-03
        - `$isoweek($year-W$wk)` - year-week ISO 8601 week number, such as `2009-W53` for 2010-01-03
- Date is assumed to be server time; add `GMT` to `text` to indicate Greenwich time zone, or use `$FORMATGMTIME()`.
- Syntax: **`$FORMATTIME( serial, text )`**
- Example: **`%CALCULATE{$FORMATTIME(0, $year/$month/$day GMT)}%`** returns **`1970/01/01 GMT`**
- Related: `$FORMATGMTIME()`, `$TIME()`, `$FORMATTIMEDIFF()`, `$TIMEADD()`, `$TIMEDIFF()`, `$TODAY()`

FORMATTIMEDIFF(unit, precision, time, option) -- convert elapsed time to a string

- Convert elapsed `time` to a human readable format, such as: `12 hours and 3 minutes`
- The input `unit` can be `second`, `minute`, `hour`, `day`, `month`, `year`. Note: An approximation is used for month and year calculations.
- The `precision` indicates the number of output units to use
- The `option` determines the format:
    - not specified - standard format, such as: `12 hours and 3 minutes`
    - `s` - short format, such as: `12 h, 3 min`
    - `c` - compact format, such as: `12h 3m`
- Syntax: **`$FORMATTIMEDIFF( unit, precision, time, option )`**
- Example: **`%CALCULATE{$FORMATTIMEDIFF(min, 1, 200)}%`** returns **`3 hours`**
- Example: **`%CALCULATE{$FORMATTIMEDIFF(min, 2, 200)}%`** returns **`3 hours and 20 minutes`**
- Example: **`%CALCULATE{$FORMATTIMEDIFF(min, 2, 200, s)}%`** returns **`3 h, 20 min`**
- Example: **`%CALCULATE{$FORMATTIMEDIFF(min, 1, 1640)}%`** returns **`1 day`**
- Example: **`%CALCULATE{$FORMATTIMEDIFF(min, 2, 1640)}%`** returns **`1 day and 3 hours`**
- Example: **`%CALCULATE{$FORMATTIMEDIFF(min, 3, 1640)}%`** returns **`1 day, 3 hours and 20 minutes`**
- Related: `$FORMATTIME()`, `$TIME()`, `$TIMEADD()`, `$TIMEDIFF()`

GET(name) -- get the value of a variable

- Specify the variable name (alphanumeric characters and underscores). An empty string is returned if the variable does not exist. Use `$SET()` to set a variable first. Unlike table ranges, variables live for the time of the page view and persist across tables, i.e. you can use it to summarize results across several tables.

- Syntax: **$GET(name)**
- Example: **%CALCULATE{$GET(my_total)}%** returns the value of the my_total variable
- Related: $EXEC(), $GETHASH(), $GETLIST(), $LIST2HASH(), $HASH2LIST(), $NOEXEC(), $SET(), $SETHASH(), $SETIFEMPTY(), $SETM(), $SETMHASH(), $WHILE(), SetGetPlugin

GETHASH(name, key) -- get the value of a previously set hash key

- Specify the hash name and key; allowed characters for the hash name are alphanumeric characters, underscores and period. An empty string is returned if the hash key does not exist. Use $SETHASH() or $LIST2HASH() to set a hash. If the key is omitted, a list of all keys of the named hash is returned. If no parameters are specified, a list of all hash names is returned.
- Syntax: **$GETHASH(*name*, *key*)**
- Examples:
  %CALCULATE{$SETHASH(age, Jane, 26)}%
  %CALCULATE{$SETHASH(age, Tim, 27)}%
  %CALCULATE{$GETHASH(age, Jane)}% returns 26
  %CALCULATE{$GETHASH(age)}% returns Jane, Tim
  %CALCULATE{$GETHASH()}% returns age
- Related: $GET(), $GETLIST(), $HASH2LIST(), $HASHCOPY(), $HASHEACH(), $HASHEXISTS(), $HASHREVERSE(), $LIST2HASH(), $SET(), $SETHASH(), $SETMHASH()

GETLIST(name) -- get the list from a list variable

- Specify the variable name (alphanumeric characters and underscores). An empty list is returned if the variable does not exist. Use $SETLIST() or $ADDLIST() to set a variable.
- Syntax: **$GETLIST(name)**
- Example: **%CALCULATE{$GETLIST(nums)}%** returns the value of the nums variable
- Related: $ADDLIST(), $GET(), $GETHASH(), $SETLIST(), SetGetPlugin

HASH2LIST(name, format) -- convert a hash to a list

- Specify the hash name and optionally the format of a key/value pair. Default format is keycomma $value, e.g. key & value in list format; $key indicates the key, $comma a comma, and $value the value. Keep in mind that you will get unexpected results if keys or values contain commas. Use $SETHASH() or $LIST2HASH() to set a hash.
- Syntax: **$HASH2LIST(name, *format*)**
- Examples:
  %CALCULATE{$LIST2HASH(age, Jane, 26, Tim, 27)}%
  %CALCULATE{$HASH2LIST(age)}% returns Jane, 26, Tim, 27
  %CALCULATE{$HASH2LIST(age, $key is $value)}% returns Jane is 26, Tim is 27
  %CALCULATE{$HASH2LIST(age, $key)}% returns Jane, Tim
  %CALCULATE{$HASH2LIST(age, $value)}% returns 26, 27
- Related: $GETHASH(), $HASHCOPY(), $HASHEACH(), $HASHEXISTS(), $HASHREVERSE(), $LIST2HASH(), $SET(), $SETHASH(), $SETMHASH()

HASHCOPY(from, to) -- copy a hash

- Copy (duplicate) a hash. This function returns no output.
- Syntax: **$HASHCOPY(from, to)**
- Examples:
  %CALCULATE{$LIST2HASH(age, Jane, 26, Tim, 27, Sam, 28)}%
  %CALCULATE{$HASHCOPY(age, new)}% creates new hash `new` from hash `age`
  %CALCULATE{$HASH2LIST(new, $key: $value)}% returns `Jane: 26, Sam: 28, Tim: 27`
- Related: `$GETHASH()`, `$HASH2LIST()`, `$HASHCOPY()`, `$HASHEACH()`, `$HASHEXISTS()`, `$HASHREVERSE()`, `$LIST2HASH()`, `$SETHASH()`, `$SETMHASH()`

HASHEACH(formula, name) -- evaluate and update each hash element

- Specify a formula that is applied to each named hash element. In the formula, `$key` indicates the key, `$value` the value, and `$index` the index of the sorted key list, starting at 1. If `$value` is omitted, the item is appended to the formula. This function returns no output.
- Syntax: **$HASHEACH(formula, name)**
- Example:
  %CALCULATE{$LIST2HASH(age, Jane, 26, Tim, 27)}%
  %CALCULATE{$HASHEACH($INT($value + 2), age)}% adds 2 years to each person
  %CALCULATE{$HASH2LIST(age, $key is now $value)}% returns `Jane is now 28, Tim is now 29`
- Related: `$COUNTITEMS()`, `$COUNTSTR()`, `$GETHASH()`, `$HASH2LIST()`, `$HASHCOPY()`, `$HASHEXISTS()`, `$HASHREVERSE()`, `$LIST2HASH()`, `$LISTEACH()`, `$SETHASH()`, `$SETMHASH()`

HASHEXISTS(name, key) -- test if a hash exists

- Test if a key exists by specifying the hash name and the key. Test if a hash exists by specifying the hash name.
- Syntax: **$HASHEXISTS(name, *key*)**
- Examples:
  %CALCULATE{$LIST2HASH(age, Jane, 26, Tim, 27)}%
  %CALCULATE{$HASHEXISTS(age)}% returns `1`
  %CALCULATE{$HASHEXISTS(age, Jane)}% returns `1`
  %CALCULATE{$HASHEXISTS(age, Blake)}% returns `0`
  %CALCULATE{$HASHEXISTS(height)}% returns `0`
- Related: `$GETHASH()`, `$HASH2LIST()`, `$HASHCOPY()`, `$HASHEACH()`, `$HASHREVERSE()`, `$LIST2HASH()`, `$SETHASH()`, `$SETMHASH()`

HASHREVERSE(name) -- reverse a hash

- All keys of a hash become values, and vice versa. Hash keys are always unique, e.g. multiple identical values will be reduced to one key. This function returns no output.
- Syntax: **$HASHREVERSE(name)**
- Examples:
  %CALCULATE{$LIST2HASH(age, Jane, 26, Tim, 27, Sam, 28)}%
  %CALCULATE{$HASHREVERSE(age)}% reverses the hash and returns nothing
  %CALCULATE{$HASH2LIST(age, $key: $value)}% returns `26: Jane, 27: Tim,`

28: Sam
- Related: $GETHASH(), $HASH2LIST(), $HASHCOPY(), $HASHEACH(), $HASHEXISTS(), $LIST2HASH(), $SETHASH(), $SETMHASH()

HEX2DEC(num) -- convert a hexadecimal number to decimal

- Syntax: **$HEX2DEC(num)**
- Example: **%CALCULATE{$HEX2DEC(A5)}%** returns **165**
- Related: $BIN2DEC(), $DEC2BIN(), $DEC2HEX(), $DEC2OCT(), $HEXDECODE(), $HEXENCODE(), $OCT2DEC()

HEXDECODE(hexcode) -- convert hexadecimal code to string

- Decode a hexadecimal string, typically encoded with $HEXENCODE().
- Syntax: **$HEXDECODE(hexcode)**
- Example: **%CALCULATE{$HEXDECODE(687474703A2F2F7477696B692E6F72672F)}%** returns **http://twiki.org/**
- Related: $BITXOR(), $HEX2DEC(), $HEXENCODE(), $SUBSTITUTE(), $TRANSLATE()

HEXENCODE(text) -- encode text into hexadecimal code

- Each character of text is encoded into two hexadecimal numbers.
- Syntax: **$HEXENCODE(text)**
- Example: **%CALCULATE{$HEXENCODE(http://twiki.org/)}%** returns **687474703A2F2F7477696B692E6F72672F**
- Related: $BITXOR(), $DEC2HEX(), $HEXDECODE(), $SUBSTITUTE(), $TRANSLATE()

IF(condition, then, else) -- return a value based on a condition

- The condition can be a number (where **0** means condition not met), or two numbers with a comparison operator **<** (less than), **<=** (less than or equal), **==** (equal), **!=** (not equal), **>=** (greater than or equal), **>** (greater than).
- Syntax: **$IF(condition, value if true, value if 0)**
- Example: **%CALC{$IF($T(R1:C5) > 1000, Over Budget, OK)}%** returns **Over Budget** if value in R1:C5 is over 1000, **OK** if not
- Example: **%CALC{$IF($EXACT($T(R1:C2),), empty, $T(R1:C2))}%** returns the content of R1:C2 or **empty** if empty
- Example: **%CALC{$SET(val, $IF($T(R1:C2) == 0, zero, $T(R1:C2)))}%** sets a variable conditionally
- Related: $AND(), $EMPTY(), $EQUAL(), $EXACT(), $ISDIGIT(), $ISLOWER(), $ISUPPER(), $ISWIKIWORD(), $LISTIF(), $NOT(), $OR(), $WHILE()

INSERTSTRING(text, start, new) -- insert a string into a text string

- Insert new string into text string text to the right of start position. Position starts at 1; use a negative start to count from the end of the text
- Syntax: **$INSERTSTRING(text, start, new)**
- Example: **%CALCULATE{$INSERTSTRING(abcdefg, 2, XYZ)}%** returns **abXYZcdefg**

- Example: **%CALCULATE{$INSERTSTRING(abcdefg, -2, XYZ)}%** returns **abcdeXYZfg**
- Related: $FILTER(), $FIND(), $LEFTSTRING(), $RANDSTRING(), $REPLACE(), $RIGHTSTRING(), $SEARCH(), $SUBSTITUTE(), $SUBSTRING(), $TRANSLATE()

INT(formula) -- evaluate formula and return integer truncated towards 0

- Addition, substraction, multiplication, division and modulus of numbers are supported. Any nesting is permitted
- Numbers may be decimal integers (1234), binary integers (0b1110011), octal integers (01234), hexadecimal integers (0x1234) or of exponential notation (12.34e-56)
- If you expect a single decimal integer value with leading zeros, use **$INT($VALUE(number))**
- Syntax: **$INT(formula)**
- Example: **%CALCULATE{$INT(10 / 4)}%** returns **2**
- Example: **%CALCULATE{$INT($VALUE(09))}%** returns **9**
- Related: $CEILING(), $EVAL(), $FLOOR(), $ROUND(), $VALUE()

ISDIGIT(text) -- test for digits

- Test for one or more digits (0...9)
- Syntax: **$ISDIGIT(text)**
- Example: **%CALCULATE{$ISDIGIT(123)}%** returns **1**
- Example: **%CALCULATE{$ISDIGIT(-7)}%** returns **0**
- Related: $ABS(), $EMPTY(), $EVEN(), $EQUAL(), $EXACT(), $IF(), $ISDIGIT(), $MOD(), $ODD(), $SIGN(), $VALUE()

ISLOWER(text) -- test for lower case text

- Syntax: **$ISLOWER(text)**
- Example: **%CALCULATE{$ISLOWER(apple)}%** returns **1**
- Example: **%CALCULATE{$ISLOWER(apple tree)}%** returns **0** (text contains a space character)
- Example: **%CALCULATE{$ISLOWER(ORANGE)}%** returns **0**
- Related: $EMPTY(), $EQUAL(), $EXACT(), $LOWER(), $IF(), $ISDIGIT(), $ISUPPER(), $ISWIKIWORD(), $UPPER()

ISUPPER(text) -- test for upper case text

- Can be used to test for ACRONYMS
- Syntax: **$ISUPPER(text)**
- Example: **%CALCULATE{$ISUPPER(apple)}%** returns **0**
- Example: **%CALCULATE{$ISUPPER(ORANGE)}%** returns **1**
- Example: **%CALCULATE{$ISUPPER(ORANGE GARDEN)}%** returns **0** (text contains a space character)
- Related: $EMPTY(), $EQUAL(), $EXACT(), $LOWER(), $IF(), $ISDIGIT(), $ISLOWER(), $ISWIKIWORD(), $UPPER()

ISWIKIWORD(text) -- test for WikiWord

- A WikiWord has a sequence of UPPER, lower/digit, UPPER, optional mixed case alphanumeric characters
- Can be used together with `$ISUPPER()` to test for valid topic names
- Syntax: **`$ISWIKIWORD( text )`**
- Example: **`%CALCULATE{$ISWIKIWORD(GoldenGate)}%`** returns **`1`**
- Example: **`%CALCULATE{$ISWIKIWORD(whiteRafting)}%`** returns **`0`**
- Related: `$EMPTY()`, `$EQUAL()`, `$EXISTS()`, `$EXACT()`, `$IF()`, `$ISDIGIT()`, `$ISLOWER()`, `$ISUPPER()`, `$PROPER()`, `$PROPERSPACE()`

LEFT( ) -- address range of cells to the left of the current cell

- Syntax: **`$LEFT( )`**
- Example: **`%CALC{$SUM($LEFT())}%`** returns the sum of cells to the left of the current cell
- Related: `$ABOVE()`, `$RIGHT()`

LEFTSTRING(text, num) -- extract characters at the beginning of a text string

- Retrieve the `num` of characters from the left end of `text`; the leftmost character is returned if `num` is missing
- Syntax: **`$LEFTSTRING( text, num )`**
- Example: **`%CALCULATE{$LEFTSTRING(abcdefg)}%`** returns **`a`**
- Example: **`%CALCULATE{$LEFTSTRING(abcdefg, 5)}%`** returns **`abcde`**
- Related: `$BITXOR()`, `$FILTER()`, `$FIND()`, `$INSERTSTRING()`, `$RANDSTRING()`, `$REPLACE()`, `$RIGHTSTRING()`, `$SEARCH()`, `$SUBSTITUTE()`, `$SUBSTRING()`, `$TRANSLATE()`

LENGTH(text) -- length of text in bytes

- Syntax: **`$LENGTH( text )`**
- Example: **`%CALCULATE{$LENGTH(abcd)}%`** returns **`4`**
- Related: `$LISTSIZE()`

LIST(range) -- convert content of a cell range into a list

- Convert the content of a range of cells into a flat list, delimited by comma. Cells containing commas are merged into the list
- Syntax: **`$LIST( range )`**
- Example: **`%CALC{$LIST($LEFT())}%`** returns **`Apples, Lemons, Oranges, Kiwis`** assuming the cells to the left contain **`| Apples | Lemons, Oranges | Kiwis |`**
- Related: `$AVERAGE()`, `$COUNTITEMS()`, `$COUNTSTR()`, `$DEF()`, `$LISTEACH()`, `$LISTIF()`, `$LISTITEM()`, `$LISTJOIN()`, `$LISTNONEMPTY()`, `$LISTRAND()`, `$LISTREVERSE()`, `$LISTSHUFFLE()`, `$LISTSIZE()`, `$LISTSORT()`, `$LISTTRUNCATE()`, `$LISTUNIQUE()`, `$MAX()`, `$MEDIAN()`, `$MIN()`, `$PRODUCT()`, `$SPLIT()`, `$SUM()`, `$SUMDAYS()`, `$SUMPRODUCT()`

LIST2HASH(name, list) -- create a hash from a list

- Specify the hash name and a list. The list is expected to have a sequence of key and value pairs. If a key already exists its value is updated with the new value. The last key is ignored if the list has an uneven number of elements. Use `$GETHASH()` or `$HASH2LIST` to retrieve hash values.
- Syntax: **`$LIST2HASH( name, list )`**
- Example: **`%CALCULATE{$LIST2HASH(age, Jane, 26, Tim, 27)}%`** sets key `Jane` of hash named `age` to `26`, and key `Tim` to `27`
- Related: `$GET()`, `$GETHASH()`, `$HASH2LIST()`, `$HASHCOPY()`, `$HASHEACH()`, `$HASHEXISTS()`, `$HASHREVERSE()`, `$SET()`

LISTEACH(formula, list) -- evaluate and update each element of a list

- Specify a formula that should be applied to list element; **`$item`** indicates the element, **`$index`** the index of the list, starting at 1. If **`$item`** is omitted, the item is appended to the formula. This function returns no output.
- Syntax: **`$LISTEACH( formula, list )`**
- Deprecated syntax: **`$LISTMAP( formula, list )`**
- Example: **`%CALCULATE{$LISTEACH($index: $EVAL(2 * $item), 3, 5, 7, 11)}%`** returns **`1: 6, 2: 10, 3: 14, 4: 22`**
- Related: `$COUNTITEMS()`, `$COUNTSTR()`, `$LIST()`, `$LISTIF()`, `$LISTITEM()`, `$LISTNONEMPTY()`, `$LISTREVERSE()`, `$LISTSIZE()`, `$LISTSORT()`, `$LISTUNIQUE()`, `$SPLIT()`, `$SUM()`, `$WHILE()`

LISTIF(condition, list) -- remove elements from a list that do not meet a condition

- In addition to the condition described in `$IF()`, you can use **`$item`** to indicate the current element, and **`$index`** for the list index, starting at 1
- Syntax: **`$LISTIF( condition, list )`**
- Example: **`%CALCULATE{$LISTIF($item > 12, 14, 7, 25)}%`** returns **`14, 25`**
- Example: **`%CALCULATE{$LISTIF($NOT($EXACT($item,)), A, B, , E)}%`** returns non-empty elements **`A, B, E`**
- Example: **`%CALCULATE{$LISTIF($index > 2, A, B, C, D)}%`** returns **`C, D`**
- Related: `$EMPTY()`, `$EQUAL()`, `$EXACT()`, `$IF()`, `$LIST()`, `$LISTEACH()`, `$LISTITEM()`, `$LISTNONEMPTY()`, `$LISTREVERSE()`, `$LISTSIZE()`, `$LISTSORT()`, `$LISTUNIQUE()`, `$SUM()`, `$WHILE()`

LISTITEM(index, list) -- get one element of a list

- Index is 1 to size of list; use a negative number to count from the end of the list
- Syntax: **`$LISTITEM( index, list )`**
- Example: **`%CALCULATE{$LISTITEM(2, Apple, Orange, Apple, Kiwi)}%`** returns **`Orange`**
- Example: **`%CALCULATE{$LISTITEM(-1, Apple, Orange, Apple, Kiwi)}%`** returns **`Kiwi`**
- Related: `$COUNTITEMS()`, `$COUNTSTR()`, `$LIST()`, `$LISTEACH()`, `$LISTIF()`, `$LISTNONEMPTY()`, `$LISTRAND()`, `$LISTREVERSE()`, `$LISTSIZE()`, `$LISTSORT()`, `$LISTUNIQUE()`, `$SPLIT()`, `$SUM()`

LISTJOIN(separator, list) -- convert a list into a string

- By default, list items are separated by a comma and a space. Use this function to indicate a specific `separator` string, which may include `$comma` for comma, `$n` for newline, `$sp` for space, and `$empty` to join a list without a separator.
- Syntax: **`$LISTJOIN( separator, list )`**
- Example: **`%CALCULATE{$LISTJOIN($n, Apple, Orange, Apple, Kiwi)}%`** returns the four items separated by new lines
- Example: **`%CALCULATE{$LISTJOIN($empty, Apple, Orange, Apple, Kiwi)}%`** returns **`AppleOrangeAppleKiwi`**
- Related: `$LIST()`, `$LISTNONEMPTY()`, `$LISTSIZE()`, `$SPLIT()`

LISTNONEMPTY(list) -- remove all empty elements from a list

- Syntax: **`$LISTNONEMPTY( list )`**
- Example: **`%CALCULATE{$LISTNONEMPTY(, Apple, Orange, , Kiwi)}%`** returns **`Apple, Orange, Kiwi`**
- Related: `$LISTEACH()`, `$LISTIF()`, `$LISTITEM()`, `$LISTSIZE()`, `$LISTUNIQUE()`

LISTRAND(list) -- get one random element of a list

- Syntax: **`$LISTRAND( list )`**
- Example: **`%CALCULATE{$LISTRAND(Apple, Orange, Apple, Kiwi)}%`** returns one of the four elements
- Related: `$COUNTITEMS()`, `$COUNTSTR()`, `$LIST()`, `$LISTEACH()`, `$LISTIF()`, `$LISTITEM()`, `$LISTSHUFFLE()`, `$LISTSIZE()`, `$LISTSORT()`, `$LISTUNIQUE()`, `$RAND()`, `$RANDSTRING()`, `$SUM()`

LISTREVERSE(list) -- opposite order of a list

- Syntax: **`$LISTREVERSE( list )`**
- Example: **`%CALCULATE{$LISTREVERSE(Apple, Orange, Apple, Kiwi)}%`** returns **`Kiwi, Apple, Orange, Apple`**
- Related: `$COUNTITEMS()`, `$COUNTSTR()`, `$LIST()`, `$LISTEACH()`, `$LISTIF()`, `$LISTITEM()`, `$LISTSIZE()`, `$LISTSORT()`, `$LISTUNIQUE()`, `$SUM()`

LISTSHUFFLE(list) -- shuffle element of a list in random order

- Syntax: **`$LISTSHUFFLE( list )`**
- Example: **`%CALCULATE{$LISTSHUFFLE(Apple, Orange, Apple, Kiwi)}%`** returns the four elements in random order
- Related: `$COUNTITEMS()`, `$COUNTSTR()`, `$LIST()`, `$LISTEACH()`, `$LISTIF()`, `$LISTITEM()`, `$LISTRAND()`, `$LISTSIZE()`, `$LISTSORT()`, `$LISTUNIQUE()`, `$RAND()`, `$RANDSTRING()`, `$SUM()`

LISTSIZE(list) -- number of elements in a list

- Syntax: **$LISTSIZE(list)**
- Example: **%CALCULATE{$LISTSIZE(Apple, Orange, Apple, Kiwi)}%** returns **4**
- Related: $COUNTITEMS(), $COUNTSTR(), $LIST(), $LISTEACH(), $LISTIF(), $LISTITEM(), $LISTJOIN(), $LISTREVERSE(), $LISTSORT(), $LISTTRUNCATE(), $LISTUNIQUE(), $SPLIT(), $SUM()

LISTSORT(list) -- sort a list

- Sorts a list in ASCII order, or numerically if all elements are numeric
- Syntax: **$LISTSORT(list)**
- Example: **%CALCULATE{$LISTSORT(Apple, Orange, Apple, Kiwi)}%** returns **Apple, Apple, Kiwi, Orange**
- Related: $COUNTITEMS(), $COUNTSTR(), $LIST(), $LISTEACH(), $LISTIF(), $LISTITEM(), $LISTREVERSE(), $LISTSHUFFLE(), $LISTSIZE(), $LISTUNIQUE(), $SUM()

LISTTRUNCATE(size, list) -- truncate list to size

- Specify the desired size of the list; use a negative number to count from the end of the list
- Syntax: **$LISTTRUNCATE(size, list)**
- Example: **%CALCULATE{$LISTTRUNCATE(2, Apple, Orange, Kiwi)}%** returns **Apple, Orange**
- Related: $COUNTITEMS(), $COUNTSTR(), $LIST(), $LISTEACH(), $LISTIF(), $LISTITEM(), $LISTSIZE(), $LISTSORT(), $LISTUNIQUE(), $SUM()

LISTUNIQUE(list) -- remove all duplicates from a list

- Syntax: **$LISTUNIQUE(list)**
- Example: **%CALCULATE{$LISTUNIQUE(Apple, Orange, Apple, Kiwi)}%** returns **Apple, Orange, Kiwi**
- Related: $COUNTITEMS(), $COUNTSTR(), $LIST(), $LISTEACH(), $LISTIF(), $LISTITEM(), $LISTNONEMPTY(), $LISTREVERSE(), $LISTSIZE(), $LISTSORT(), $SUM()

LN(num) -- natural logarithm of a number

- LN is the inverse of the EXP function
- Syntax: **$LN(num)**
- Example: **%CALCULATE{$LN(10)}%** returns **2.30258509299405**
- Related: $EXP(), $LOG()

LOG(num, base) -- logarithm of a number to a given base

- base-10 logarithm of a number (if base is 0 or not specified), else logarithm of a number to the given base
- Syntax: **$LOG(num, *base*)**

- Example: **%CALCULATE{$LOG(1000)}%** returns **3**
- Example: **%CALCULATE{$LOG(16, 2)}%** returns **4**
- Related: $EXP(), $LN()

LOWER(text) -- lower case string of a text

- Syntax: **$LOWER(text)**
- Example: **%CALC{$LOWER($T(R1:C5))}%** returns the lower case string of the text in cell **R1:C5**
- Related: $ISLOWER(), $PROPER(), $PROPERSPACE(), $TRIM(), $UPPER()

MAX(list) - biggest value of a list or range of cells

- Syntax: **$MAX(list)**
- Example: To find the biggest number to the left of the current cell, write: **%CALC{$MAX($LEFT())}%**
- Related: $LIST(), $MEDIAN(), $MIN(), $PERCENTILE(), $STDEV(), $STDEVP(), $VAR(), $VARP()

MEDIAN(list) -- median of a list or range of cells

- Syntax: **$MEDIAN(list)**
- Example: **%CALCULATE{$MEDIAN(3, 9, 4, 5)}%** returns **4.5**
- Related: $LIST(), $MAX(), $MIN(), $PERCENTILE(), $STDEV(), $STDEVP(), $VAR(), $VARP()

MIN(list) -- smallest value of a list or range of cells

- Syntax: **$MIN(list)**
- Example: **%CALCULATE{$MIN(15, 3, 28)}%** returns **3**
- Related: $LIST(), $MAX(), $MEDIAN(), $PERCENTILE(), $STDEV(), $STDEVP(), $VAR(), $VARP()

MOD(num, divisor) -- reminder after dividing num by divisor

- Syntax: **$MOD(num, divisor)**
- Example: **%CALCULATE{$MOD(7, 3)}%** returns **1**
- Related: $EVAL(), $EVEN(), $ISDIGIT(), $ODD(), $SIGN()

NOEXEC(formula) -- do not execute a spreadsheet formula

- Prevent a formula from getting executed. This is typically used to store a raw formula in a variable for later use as described in $EXEC().
- Syntax: **$NOEXEC(formula)**
- Example: **%CALCULATE{$SET(msg, $NOEXEC(Hi $GET(name)))}%** sets the msg variable with the formula Hi $GET(name) without executing it
- Related: $EVAL(), $EXEC(), $GET(), $SET()

NOP(text) -- no-operation

- Text is returned verbatim, except for escape characters:
  - `$percnt` - changed to percent character '%'
  - `$quot` - changed to double quote character '"'
- Useful to delay / change the order of plugin execution. For example, it allows preprocessing to be done before `%SEARCH{...}%`.
- Syntax: **`$NOP( text )`**
- Example: **`%CALCULATE{$NOP(%)SEARCH{...}$NOP(%)}%`** returns `%SEARCH{...}%`
- Example: **`%CALCULATE{$NOP($percntSEARCH{...}$percnt)}%`** returns `%SEARCH{...}%`
- Related: `$NOTE()`

NOT(num) -- reverse logic of a number

- Returns 0 if **`num`** is not zero, 1 if zero
- Syntax: **`$NOT( num )`**
- Example: **`%CALCULATE{$NOT(0)}%`** returns **`1`**
- Related: `$AND()`, `$EMPTY()`, `$IF()`, `$OR()`, `$WHILE()`, `$XOR()`

NOTE(text) -- note, return empty string

- An empty string is returned - useful to comment formulas
- Syntax: **`$NOTE( comment text )`**
- Example: **`%CALCULATE{$NOTE(abc)}%`** returns an empty string
- Related: `$NOP()`

OCT2DEC(num) -- convert an octal number to decimal

- Syntax: **`$OCT2DEC( num )`**
- Example: **`%CALCULATE{$OCT2DEC(54)}%`** returns **`44`**
- Related: `$BIN2DEC()`, `$DEC2BIN()`, `$DEC2HEX()`, `$DEC2OCT()`, `$HEX2DEC()`, `$HEXDECODE()`, `$HEXENCODE()`

ODD(num) -- test for odd number

- Syntax: **`$ODD( num )`**
- Example: **`%CALCULATE{$ODD(2)}%`** returns **`0`**
- Related: `$ABS()`, `$EVEN()`, `$ISDIGIT()`, `$MOD()`, `$SIGN()`

OR(list) -- logical OR of a list

- Syntax: **`$OR( list )`**
- Example: **`%CALCULATE{$OR(1, 0, 1)}%`** returns **`1`**
- Related: `$AND()`, `$IF()`, `$NOT()`, `$WHILE()`, `$XOR()`

PERCENTILE(num, list) -- percentile of a list or range of cells

- Calculates the num-th percentile, useful to establish a threshold of acceptance. num is the percentile value, range 0..100
- Syntax: **`$PERCENTILE( num, list )`**
- Example: **`%CALCULATE{$PERCENTILE(75, 400, 200, 500, 100, 300)}%`** returns **`450`**
- Related: `$LIST()`, `$MAX()`, `$MEDIAN()`, `$MIN()`, `$STDEV()`, `$STDEVP()`, `$VAR()`, `$VARP()`

PI( ) -- mathematical constant Pi, 3.14159265358979

- Syntax: **`$PI( )`**
- Example: **`%CALCULATE{$PI()}%`** returns **`3.14159265358979`**

PRODUCT(list) -- product of a list or range of cells

- Syntax: **`$PRODUCT( list )`**
- Deprecated syntax: **`$MULT( list )`**
- Example: To calculate the product of the cells to the left of the current one use **`%CALC{$PRODUCT($LEFT())}%`**
- Related: `$LIST()`, `$PRODUCT()`, `$SUM()`, `$SUMPRODUCT()`

PROPER(text) -- properly capitalize text

- Capitalize letters that follow any character other than a letter; convert all other letters to lowercase letters
- Syntax: **`$PROPER( text )`**
- Example: **`%CALCULATE{$PROPER(a small STEP)}%`** returns **`A Small Step`**
- Example: **`%CALCULATE{$PROPER(f1 (formula-1))}%`** returns **`F1 (Formula-1)`**
- Related: `$FILTER()`, `$ISWIKIWORD()`, `$LOWER()`, `$PROPERSPACE()`, `$TRIM()`, `$UPPER()`

PROPERSPACE(text) -- properly space out WikiWords

- Properly spaces out WikiWords preceeded by white space, parenthesis, or **`][`**. Words listed in the DONTSPACE TWikiPreferences variable or DONTSPACE plugins setting are excluded
- Syntax: **`$PROPERSPACE( text )`**
- Example: Assuming DONTSPACE contains MacDonald: **`%CALCULATE{$PROPERSPACE(Old MacDonald had a ServerFarm, EeEyeEeEyeOh)}%`** returns **`Old MacDonald had a Server Farm, Ee Eye Ee Eye Oh`**
- Related: `$ISWIKIWORD()`, `$LOWER()`, `$PROPER()`, `$TRIM()`, `$UPPER()`

RAND(max) -- random number

- Random number, evenly distributed between 0 and **`max`**, or 0 and 1 if max is not specified
- Syntax: **`$RAND( max )`**
- Related: `$EVAL()`, `$LISTRAND()`, `$LISTSHUFFLE()`, `$RANDSTRING()`

RANDSTRING(set, format) -- random string & password generator

- Generate a random string from a **`set`** of characters; the set may contain sequences like **`a..z`**; default is **`a..zA..Z0..9_`**. The **`format`** defines the string length or the output format; specify a number to indicate the length of the random string; default is **`8`** characters. Alternatively, specify a format string with **`x`** as placeholders for random characters, such **`xxxx-xxxx-xxxx-xxxx`**.
- Syntax: **`$RANDSTRING( set, format )`**
- Example: **`%CALCULATE{$RANDSTRING()}%`** returns a random string with 8 characters composed of alphanumeric characters and underscores
- Example: **`%CALCULATE{$RANDSTRING(A..NP..Z1..9, xxxx-xxxx-xxxx-xxxx)}%`** returns four sets of random strings, separated by dashes, where each set has four characters composed of uppercase letters and numbers, excluding letter O and number 0
- Related: `$INSERTSTRING()`, `$SUBSTRING()`, `$LISTRAND()`, `$LISTSHUFFLE()`, `$RAND()`, `$REPEAT()`

REPEAT(text, num) -- repeat text a number of times

- Syntax: **`$REPEAT( text, num )`**
- Example: **`%CALCULATE{$REPEAT(/\, 5)}%`** returns **`/\/\/\/\/\`**
- Related: `$RANDSTRING()`, `$WHILE()`

REPLACE(text, start, num, new) -- replace part of a text string

- Replace `num` number of characters of text string `text`, starting at `start`, with new text `new`. Starting position is 1; use a negative `start` to count from the end of the text
- Syntax: **`$REPLACE( text, start, num, new )`**
- Example: **`%CALCULATE{$REPLACE(abcdefghijk, 6, 5, *)}%`** returns **`abcde*k`**
- Related: `$BITXOR()`, `$FILTER()`, `$FIND()`, `$INSERTSTRING()`, `$LEFTSTRING()`, `$RIGHTSTRING()`, `$SEARCH()`, `$SUBSTITUTE()`, `$SUBSTRING()`, `$TRANSLATE()`

RIGHT( ) -- address range of cells to the right of the current cell

- Syntax: **`$RIGHT( )`**
- Example: **`%CALC{$SUM($RIGHT())}%`** returns the sum of cells to the right of the current cell
- Related: `$ABOVE()`, `$LEFT()`

RIGHTSTRING(text, num) -- extract characters at the end of a text string

- Retrieve the `num` of characters from the right end of `text`; the rightmost character is returned if `num` is missing
- Syntax: **`$RIGHTSTRING( text, num )`**
- Example: **`%CALCULATE{$RIGHTSTRING(abcdefg)}%`** returns **`g`**
- Example: **`%CALCULATE{$RIGHTSTRING(abcdefg, 5)}%`** returns **`cdefg`**
- Related: `$BITXOR()`, `$FILTER()`, `$FIND()`, `$INSERTSTRING()`, `$LEFTSTRING()`, `$RANDSTRING()`, `$REPLACE()`, `$SEARCH()`, `$SUBSTITUTE()`, `$SUBSTRING()`, `$TRANSLATE()`

ROUND(formula, digits) -- round a number

- Evaluates a simple **`formula`** and rounds the result up or down to the number of digits if **`digits`** is positive; to the nearest integer if digits is missing; or to the left of the decimal point if digits is negative
- Syntax: **`$ROUND( formula, digits )`**
- Example: **`%CALCULATE{$ROUND(3.15, 1)}%`** returns **`3.2`**
- Example: **`%CALCULATE{$ROUND(3.149, 1)}%`** returns **`3.1`**
- Example: **`%CALCULATE{$ROUND(-2.475, 2)}%`** returns **`-2.48`**
- Example: **`%CALCULATE{$ROUND(34.9, -1)}%`** returns **`30`**
- Related: `$CEILING()`, `$EVAL()`, `$FLOOR()`, `$INT()`, `$FORMAT()`

ROW(offset) -- current row number

- The current table row number with an optional offset. When used outside of a table it returns the number of rows of the preceding table.
- Syntax: **`$ROW( offset )`**
- Example: To get the number of rows excluding table heading (first row) and summary row (last row where this function is used), write: **`%CALC{$ROW(-2)}%`**
- Related: `$COLUMN()`, `$T()`

SEARCH(string, text, start) -- search a string within a text

- Finds one text `string`, within another `text`, and returns the number of the starting position of `string`, from the first character of `text`. This search is a RegularExpression search; use `$FIND()` for non-regular expression searching. Starting position is 1; a 0 is returned if nothing is matched
- Syntax: **`$SEARCH( string, text, start )`**
- Example: **`%CALCULATE{$SEARCH([uy], fluffy)}%`** returns **`3`**
- Example: **`%CALCULATE{$SEARCH([uy], fluffy, 4)}%`** returns **`6`**
- Example: **`%CALCULATE{$SEARCH([abc], fluffy,)}%`** returns **`0`**
- Related: `$FILTER()`, `$FIND()`, `$INSERTSTRING()`, `$LEFTSTRING()`, `$REPLACE()`, `$RIGHTSTRING()`, `$SUBSTRING()`

SET(name, value) -- set a variable for later use

- Specify the variable name (alphanumeric characters and underscores) and the value. The value may contain a formula; formulas are evaluated before the variable assignment; see `$NOEXEC()` if you want to prevent that. This function returns no output. Use `$GET()` to retrieve variables. Unlike table ranges, variables live for the time of the page view and persist across tables, i.e. you can use it to summarize results across several tables and also across included topics. If the value is omitted, the named variable is deleted.
- Syntax: **`$SET( name, value )`**
- Example: **`%CALC{$SET(my_total, $SUM($ABOVE()))}%`** sets the `my_total` variable to the sum of all table cells located above the current cell and returns an empty string
- Related: `$EXEC()`, `$GET()`, `$GETHASH()`, `$HASH2LIST()`, `$LIST2HASH()`, `$NOEXEC()`, `$SETHASH()`, `$SETIFEMPTY()`, `$SETLIST()`, `SETM()`, `$SETMHASH()`, `$WHILE()`, SetGetPlugin

SETHASH(name, key, value) -- set a hash value for later use, or delete a hash

- Specify the hash name, key, and value. Allowed characters for the hash name are alphanumeric characters, underscores and period. This function returns no output. Use `$GETHASH()` to retrieve hash values. If the value is omitted, the key is deleted from the hash. If the key is omitted, the named hash is deleted. If no parameters are specified, all hashes are deleted.
- Syntax: **`$SETHASH( name, key, value )`**
- Example: **`%CALCULATE{$SETHASH(age, Jane, 26)}%`** sets key `Jane` of hash named `age` to `26`
- Example: **`%CALCULATE{$SETHASH(age, Jane,)}%`** sets key `Jane` of hash named `age` to an empty value
- Example: **`%CALCULATE{$SETHASH(age, Jane)}%`** deletes key `Jane` of hash named `age`
- Example: **`%CALCULATE{$SETHASH(age)}%`** deletes hash named `age`
- Example: **`%CALCULATE{$SETHASH()}%`** deletes all hashes
- Related: `$GET()`, `$GETHASH()`, `$HASH2LIST()`, `$HASHCOPY()`, `$HASHEACH()`, `$HASHEXISTS()`, `$HASHREVERSE()`, `$LIST2HASH()`, `$SET()`, `$SETLIST()`, `$SETMHASH()`

SETIFEMPTY(name, value) -- set a variable only if empty

- Specify the variable name (alphanumeric characters and underscores) and the value.
- Syntax: **`$SETIFEMPTY( name, value )`**
- Example: **`%CALCULATE{$SETIFEMPTY(result, default)}%`** sets the `result` variable to `default` if the variable is empty or 0; in any case an empty string is returned
- Related: `$GET()`, `$SET()`, `$SETHASH()`, `$SETMHASH()`

SETLIST(name, list) -- save a list for later use

- Specify the variable name (alphanumeric characters and underscores) and the list. This function returns no output. Use `$GETLIST()` to retrieve a list variable. Use `$ADDLIST()` to add a list to an existing variable. If no list is specified, the named variable is deleted.
- Syntax: **`$SETLIST( name, list )`**
- Example: **`%CALCULATE{$SETLIST(octals, 0, 1, 2, 3, 4, 5, 6, 7)}%`** sets the `octals` variable to `0, 1, 2, 3, 4, 5, 6, 7`
- Example: **`%CALCULATE{$SETLIST(octals)}%`** deletes the `octals` variable
- Related: `$ADDLIST()`, `$GETLIST()`, `$SET()`, `$SETHASH()`, SetGetPlugin

SETM(name, formula) -- modify an existing variable based on a formula

- Specify the variable name (alphanumeric characters and underscores) and the formula. The formula must start with an operator to **+** (add), **-** (subtract), ***** (multiply), or **/** (divide) something to the variable. This function returns no output. Use `$GET()` to retrieve variables
- Syntax: **`$SETM( name, formula )`**
- Example: **`%CALC{$SETM(total, + $SUM($LEFT()))}%`** adds the sum of all table cells on the left to the `total` variable, and returns an empty string
- Related: `$GET()`, `$SET()`, `$SETHASH()`, `$SETIFEMPTY()`, `$SETMHASH()`, `$WHILE()`

SETMHASH(name, key, formula) -- modify an existing hash based on a formula

- Specify the hash name, key, and formula. Allowed characters for the hash name are alphanumeric characters, underscores and period. The formula must start with an operator to **+** (add), **-** (subtract), ***** (multiply), or **/** (divide) something to the hash variable. This function returns no output. Use `$GETHASH()` or `$HASH2LIST` to retrieve the values
- Syntax: **`$SETMHASH( name, key, formula )`**
- Examples:
  `%CALCULATE{$SETHASH(count)}%` deletes the count hash
  `%CALCULATE{$SET(people, Anna, Jane, Berta, Charlie, Jane, Tom, Anna, Jane))}%` sets the people list
  `%CALCULATE{$LISTJOIN(, $LISTEACH($SETMHASH(count, $item, +1), $GET(people)))}%` populates the count hash and returns nothing
  `%CALCULATE{$HASH2LIST(count, $key: $value)}%` returns `Anna: 2, Berta: 1, Charlie: 1, Jane: 3, Tom: 1`
- Related: `$GET()`, `$GETHASH()`, `$HASH2LIST()`, `$HASHCOPY()`, `$HASHEACH()`, `$HASHEXISTS()`, `$HASHREVERSE()`, `$LIST2HASH()`, `$SET()`, `$SETHASH()`, `$WHILE()`

SIGN(num) -- sign of a number

- Returns -1 if **num** is negative, 0 if zero, or 1 if positive
- Syntax: **`$SIGN( num )`**
- Example: **`%CALCULATE{$SIGN(-12.5)}%`** returns **-1**
- Related: `$ABS()`, `$EVAL()`, `$EVEN()`, `$INT()`, `$ISDIGIT()`, `$NOT()`, `$ODD()`

SPLIT(separator, text) -- split a string into a list

- Split `text` into a list using `separator` as a delimiter. The `separator` may be a regular expression and may include `$comma` for comma, `$sp` for space and `$empty` to split at each character. Default separator is one or more spaces (`$sp$sp*`).
- Syntax: **`$SPLIT( separator, text )`**
- Example: **`%CALCULATE{$SPLIT(, Apple Orange Kiwi)}%`** returns **`Apple, Orange, Kiwi`**
- Example: **`%CALCULATE{$SPLIT(-, Apple-Orange-Kiwi)}%`** returns **`Apple, Orange, Kiwi`**
- Example: **`%CALCULATE{$SPLIT([-:]$sp*, Apple-Orange: Kiwi)}%`** returns **`Apple, Orange, Kiwi`** (the separator means: Dash or colon, followed by optional spaces
- Example: **`%CALCULATE{$SPLIT($empty, Apple)}%`** returns **`A, p, p, l, e`**
- Related: `$LIST()`, `$LISTJOIN()`, `$LISTSIZE()`

SQRT(num) -- square root of a number

- Syntax: **`$SQRT( num )`**
- Example: **`%CALCULATE{$SQRT(16)}%`** returns **4**

STDEV(list) -- standard deviation based on a sample

- Calculates the standard deviation, assuming that the `list` is a sample of the population. Use `$STDEVP()` if your data represents the entire population. The standard deviation is a measure of how widely values are dispersed from the average (mean) value.
- Syntax: **`$STDEV( list )`**
- Example: **`%CALC{$STDEV(R2:C5..R$ROW(-1):C5)}%`** returns the standard deviation of column 5, excluding the title row
- Example: **`%CALCULATE{$STDEV(3.50, 5.00, 7.23, 2.99)}%`** returns **`1.90205152401295`**
- Related: `$AVERAGE()`, `$LIST()`, `$MAX()`, `$MEDIAN()`, `$MIN()`, `$PERCENTILE()`, `$STDEVP()`, `$VAR()`, `$VARP()`

STDEVP(list) -- standard deviation based on the entire population

- Calculates the standard deviation, assuming that the `list` is the entire population. Use `$STDEV()` if your data represents a sample of the population. The standard deviation is a measure of how widely values are dispersed from the average (mean) value.
- Syntax: **`$STDEVP( list )`**
- Example: **`%CALC{$STDEVP(R2:C5..R$ROW(-1):C5)}%`** returns the standard deviation of column 5, excluding the title row
- Example: **`%CALCULATE{$STDEVP(3.50, 5.00, 7.23, 2.99)}%`** returns **`1.64722493910213`**
- Related: `$AVERAGE()`, `$LIST()`, `$MAX()`, `$MEDIAN()`, `$MIN()`, `$MAX()`, `$MIN()`, `$PERCENTILE()`, `$STDEV()`, `$VAR()`, `$VARP()`

SUBSTITUTE(text, old, new, instance, option) -- substitute text

- Substitutes `new` text for `old` text in a `text` string. `instance` specifies which occurance of `old` you want to replace. If you specify `instance`, only that instance is replaced. Otherwise, every occurance is changed to the new text. A literal search is performed by default; a RegularExpression search if the `option` is set to **`r`**
- Syntax: **`$SUBSTITUTE( text, old, new, instance, option )`**
- Example: **`%CALCULATE{$SUBSTITUTE(Good morning, morning, day)}%`** returns **`Good day`**
- Example: **`%CALCULATE{$SUBSTITUTE(Q2-2012, 2, 3)}%`** returns **`Q3-3013`**
- Example: **`%CALCULATE{$SUBSTITUTE(Q2-2012, 2, 3, 3)}%`** returns **`Q2-2013`**
- Example: **`%CALCULATE{$SUBSTITUTE(abc123def, [0-9], 9, , r)}%`** returns **`abc999def`**
- Related: `$BITXOR()`, `$FILTER()`, `$HEXDECODE()`, `$HEXENCODE()`, `$INSERTSTRING()`, `$LEFTSTRING()`, `$RANDSTRING()`, `$REPLACE()`, `$RIGHTSTRING()`, `$SUBSTRING()`, `$TRANSLATE()`

SUBSTRING(text, start, num) -- extract a substring out of a text string

- Extract `num` number of characters of text string `text`, starting at `start`. Starting position is 1; use a negative `start` to count from the end of the text. All parameters are required - the text may contain commas.
- Syntax: **`$SUBSTRING( text, start, num )`**
- Example: **`%CALCULATE{$SUBSTRING(abcdefghijk, 3, 5)}%`** returns **`cdefg`**

- Related: `$FILTER()`, `$FIND()`, `$INSERTSTRING()`, `$LEFTSTRING()`, `$RANDSTRING()`, `$REPLACE()`, `$RIGHTSTRING()`, `$SEARCH()`, `$SUBSTITUTE()`, `$TRANSLATE()`

SUM(list) -- sum of a list or range of cells

- Syntax: **`$SUM( list )`**
- Example: To sum up column 5 excluding the title row, write **`%CALC{$SUM(R2:C5..R$ROW(-1):C5)}%`** in the last row; or simply **`%CALC{$SUM($ABOVE())}%`**
- Related: `$LIST()`, `$PRODUCT()`, `$SUMPRODUCT()`, `$WORKINGDAYS()`

SUMDAYS(list) -- sum the days in a list or range of cells

- The total number of days in a list or range of cells containing numbers of hours, days or weeks. The default unit is days; units are indicated by a **`h`**, **`hours`**, **`d`**, **`days`**, **`w`**, **`weeks`** suffix. One week is assumed to have 5 working days, one day 8 hours
- Syntax: **`$SUMDAYS( list )`**
- Example: **`%CALCULATE{$SUMDAYS(2w, 1, 2d, 4h)}%`** returns **`13.5`**, the evaluation of `(2*5 + 1 + 2 + 4/8)`
- Related: `$SUM()`, `$TIME()`, `$FORMATTIME()`

SUMPRODUCT(list, list) -- scalar product on ranges of cells

- Syntax: **`$SUMPRODUCT( list, list, `*`list...`*` )`**
- Example: **`%CALC{$SUMPRODUCT(R2:C1..R4:C1, R2:C5..R4:C5)}%`** evaluates and returns the result of **`($T(R2:C1) * $T(R2:C5) + $T(R3:C1) * $T(R3:C5) + $T(R4:C1) * $T(R4:C5))`**
- Related: `$LIST()`, `$PRODUCT()`, `$SUM()`

T(address) -- content of a cell

- Syntax: **`$T( address )`**
- Example: **`%CALC{$T(R1:C5)}%`** returns the text in cell **`R1:C5`**
- Related: `$COLUMN()`, `$ROW()`

TIME(text) -- convert a date string into a serialized date number

- Serialized date is seconds since the Epoch, e.g. midnight, 01 Jan 1970. Current time is taken if the date string is empty. Supported date formats: `31 Dec 2009`; `31 Dec 2009 GMT`; `31 Dec 09`; `31-Dec-2009`; `31/Dec/2009`; `31 Dec 2003 - 23:59`; `31 Dec 2003 - 23:59:59`; `2009/12/31`; `2009-12-31`; `2009/12/31`; `2009/12/31 23:59`; `2009/12/31 - 23:59`; `2009-12-31-23-59`; `2009/12/31 - 23:59:59`; `2009.12.31.23.59.59`. DOY (Day of Year) formats: `DOY2003.365`, `DOY2003.365.23.59`, `DOY2003.365.23.59.59`. Date is assumed to be server time; add `GMT` to indicate Greenwich time zone
- Syntax: **`$TIME( `*`text`*` )`**
- Example: **`%CALCULATE{$TIME(2003/10/14 GMT)}%`** returns **`1066089600`**
- Related: `$FORMATGMTIME()`, `$FORMATTIME()`, `$FORMATTIMEDIFF()`, `$TIMEADD()`, `$TIMEDIFF()`, `$TODAY()`, `$WORKINGDAYS()`

TIMEADD(serial, value, unit) -- add a value to a serialized date

- The `unit` is seconds if not specified; unit can be `second`, `minute`, `hour`, `day`, `week`, `month`, `year`. Note: An approximation is used for month and year calculations
- Syntax: **`$TIMEADD( serial, value, unit )`**
- Example: **`%CALCULATE{$TIMEADD($TIME(), 2, week)}%`** returns the serialized date two weeks from now
- Related: `$FORMATTIME()`, `$FORMATGMTIME()`, `$TIME()`, `$TIMEDIFF()`, `$TODAY()`

TIMEDIFF(serial_1, serial_2, unit) -- time difference between two serialized dates

- The `unit` is seconds if not specified; unit can be specified as in `$TIMEADD()`.
- Notes: An approximation is used for month and year calculations. Use `$ROUND()` to round `day` unit to account for daylight savings time change. Use `$FORMAT()`, `$FORMATTIMEDIFF()` or `$INT()` to format real numbers
- Syntax: **`$TIMEDIFF( serial_1, serial_2, unit )`**
- Example: **`%CALCULATE{$TIMEDIFF($TIME(), $EVAL($TIME()+90), minute)}%`** returns **`1.5`**
- Example: **`%CALCULATE{$ROUND($TIMEDIFF($TIME(2012-12-06),$TIME(2012-12-13), day))}%`** returns **`7`** (or **`6.95833333333333`** without the `$ROUND()`)
- Related: `$FORMAT()`, `$FORMATGMTIME()`, `$FORMATTIME()`, `$FORMATTIMEDIFF()`, `$INT()`, `$TIME()`, `$TIMEADD()`, `$TODAY()`, `$WORKINGDAYS()`

TODAY( ) -- serialized date of today at midnight GMT

- In contrast, the related `$TIME()` returns the serialized date of today at the current time, e.g. it includes the number of seconds since midnight GMT
- Syntax: **`$TODAY( )`**
- Example: **`%CALCULATE{$TODAY()}%`** returns the number of seconds since Epoch
- Related: `$FORMATTIME()`, `$FORMATGMTIME()`, `$TIME()`, `$TIMEADD()`, `$TIMEDIFF()`

TRANSLATE(text, from, to) -- translate text from one set of characters to another

- The translation is done `from` a set `to` a set, one character by one. The `text` may contain commas; all three parameters are required. The `from` and `to` parameters support tokens `$comma` for comma, `$sp` for space, `$quot` for double quote, `$aquot` for apostrophe quote, and `$n` for newline
- Syntax: **`$TRANSLATE( text, from, to )`**
- Example: **`%CALCULATE{$TRANSLATE(boom,bm,cl)}%`** returns **`cool`**
- Example: **`%CALCULATE{$TRANSLATE(one, two,$comma,;)}%`** returns **`one; two`**
- Related: `$BITXOR()`, `$FILTER()`, `$HEXDECODE()`, `$HEXENCODE()`, `$INSERTSTRING()`, `$LEFTSTRING()`, `$REPLACE()`, `$RIGHTSTRING()`, `$SUBSTRING()`, `$SUBSTITUTE()`

TRIM(text) -- trim spaces from text

- Removes all spaces from text except for single spaces between words
- Syntax: **`$TRIM( text )`**

- Example: **%CALCULATE{$TRIM(eat spaces)}%** returns **eat spaces**
- Related: $EMPTY(), $EQUAL(), $EXACT(), $FILTER(), $ISWIKIWORD(), $PROPER(), $PROPERSPACE()

UPPER(text) -- upper case string of a text

- Syntax: **$UPPER(text)**
- Example: **%CALC{$UPPER($T(R1:C5))}%** returns the upper case string of the text in cell **R1:C5**
- Related: $ISLOWER(), $ISUPPER(), $LOWER(), $PROPER(), $PROPERSPACE(), $TRIM()

VALUE(text) -- convert text to number

- Extracts a number from **text**. Returns **0** if not found
- Syntax: **$VALUE(text)**
- Example: **%CALCULATE{$VALUE(US$1,200)}%** returns **1200**
- Example: **%CALCULATE{$VALUE(PrjNotebook1234)}%** returns **1234**
- Example: **%CALCULATE{$VALUE(Total: -12.5)}%** returns **-12.5**
- Related: $CEILING(), $EVAL(), $FLOOR(), $INT(), $ISDIGIT(), $ROUND()

VAR(list) -- variance based on a sample

- This assumes that the list is a sample of the population. Use $VARP() if your data represents the entire population.
- Syntax: **$VARP(list)**
- Example: **%CALC{$VAR(R2:C5..R$ROW(-1):C5)}%** returns the variance of column 5, excluding the title row
- Example: **%CALCULATE{$VAR(3.50, 5.00, 7.23, 2.99)}%** returns **3.6178**
- Related: $AVERAGE(), $LIST(), $MAX(), $MEDIAN(), $MIN(), $MAX(), $MIN(), $PERCENTILE(), $STDEV(), $STDEVP(), $VARP()

VARP(list) -- variance based on the entire population

- This assumes that the list is the entire population. Use $VAR() if your data represents a sample of the population.
- Syntax: **$VARP(list)**
- Example: **%CALC{$VARP(R2:C5..R$ROW(-1):C5)}%** returns the variance of column 5, excluding the title row
- Example: **%CALCULATE{$VARP(3.50, 5.00, 7.23, 2.99)}%** returns **2.71335**
- Related: $AVERAGE(), $LIST(), $MAX(), $MEDIAN(), $MIN(), $MAX(), $MIN(), $PERCENTILE(), $STDEV(), $STDEVP(), $VAR()

WHILE(condition, do) -- do something while a condition is true

- The condition can be a number (where **0** means condition not met), or two numbers with a comparison operator **<** (less than), **<=** (less than or equal), **==** (equal), **!=** (not equal), **>=** (greater than or equal), **>** (greater than).
- The condition and do are evaluated in each cycle; a $counter starting at 1 can be used in condition and do.

- Syntax: **`$WHILE( condition, do something )`**
- Example: **`%CALCULATE{$WHILE($counter<=10, $counter )}%`** returns **`1 2 3 4 5 6 7 8 9 10`**
- Example: **`%CALCULATE{$SET(i, 0) $WHILE($GET(i)<10, $SETM(i, +1) $EVAL($GET(i) * $GET(i)), )}%`** returns **`1, 4, 9, 16, 25, 36, 49, 64, 81, 100,`**
- Related: `$AND()`, `$EMPTY()`, `$EQUAL()`, `$EXACT()`, `$GET()`, `$IF()`, `$LISTIF()`, `$NOT()`, `$OR()`, `$SET()`, `$SETHASH()`, `$SETM()`, `$SETMHASH()`, `$REPEAT()`

WORKINGDAYS(serial_1, serial_2) -- working days between two serialized dates

- Working days are Monday through Friday (sorry, Israel!)
- Syntax: **`$WORKINGDAYS( serial_1, serial_2 )`**
- Example: **`%CALCULATE{$WORKINGDAYS($TIME(2012-07-15 GMT), $TIME(2012-08-03 GMT))}%`** returns **`14`**
- Related: `$SUMDAYS()`, `$TIME()`, `$TIMEDIFF()`

XOR(list) -- logical XOR of a list

- Syntax: **`$XOR( list )`**
- Example: **`%CALCULATE{$XOR(0, 0)}%`** returns **`0`**
- Example: **`%CALCULATE{$XOR(0, 1)}%`** returns **`1`**
- Example: **`%CALCULATE{$XOR(1, 0)}%`** returns **`1`**
- Example: **`%CALCULATE{$XOR(1, 1)}%`** returns **`0`**
- Example: **`%CALCULATE{$XOR(1, 0, 1)}%`** returns **`0`**
- Related: `$AND()`, `$BITXOR()`, `$IF()`, `$NOT()`, `$OR()`, `$WHILE()`

FAQ

Can I use CALCULATE in a formatted search?

Specifically, how can I output some conditional text in a FormattedSearch?

You need to escape the CALCULATE so that it executes once per search hit. This can be done by escaping the `%` signs of `%CALCULATE{...}%` with `$percnt`. For example, to execute `$IF($EXACT($formfield(Tested), Yes), %ICONURL{choice-yes}%, %ICONURL{choice-no}%)` in the `format=""` parameter, write this:

```
%SEARCH{ .... format="| $topic |
$percntCALCULATE{$IF($EXACT($formfield(Tested), Yes),
%ICONURL{choice-yes}%, %ICONURL{choice-no}%)}$percnt |" }%
```

What are hashes and what are they used for?

Hashes are named sets of key & value pairs. For example, a hash called `age` may have first names as keys and ages as values. In TWiki's case you might do a FormattedSearch and store the result in hashes, then display the result in various formats. For example, you want to show feature requests in multiple tables, organized by status. Adding a search for each status is possible, but it can be slow. The page loads faster if

you search only once, and store the status, summary, date and other fields in hashes. Then you use those hashes to render each table by status.

Example:

```
%CALCULATE{$SETHASH()}% <!-- clear all hashes -->
%SEARCH{
 "form.name='FeatureForm'"
 type="query"
 nonoise="on"
 format="$percntCALCULATE{$SETHASH(status, $topic, $formfield(Status))$SETHASH(summary, $t
}%
---++ Proposed Features
| *Feature* | *Updated* |
%CALCULATE{
  $LISTJOIN(
    $n,
    $LISTEACH(
      | [[$item]]: $GETHASH(summary, $item) | $GETHASH(date, $item) |,
      $LISTIF(
        $EXACT(
          $GETHASH(status, $item),
          Proposed
        ),
        $GETHASH(status)
      )
    )
  )
}%
---++ Accepted Features
| *Feature* | *Updated* |
%CALCULATE{
  $LISTJOIN(
    $n,
    $LISTEACH(
      | [[$item]]: $GETHASH(summary, $item) | $GETHASH(date, $item) |,
      $LISTIF(
        $EXACT(
          $GETHASH(status, $item),
          Accepted
        ),
        $GETHASH(status)
      )
    )
  )
}%
```

First we search all features and store the status, summary and date in hashes, using the topic name as the key. Then we build a table that shows all proposed features, followed by a table showing all accepted features. Reading the CALCULATE formula from inside out:

- `$GETHASH(status)` - returns the list of all keys of the status hash, e.g. all topics found
- `$LISTIF($EXACT($GETHASH(status, $item), Proposed), ...)` - filter the topic list and keeps only those of status Proposed
- `$LISTEACH(| [[$item]]: $GETHASH(summary, $item) | ... |, ...)` - format each topic as a table row
- `$LISTJOIN($n, ...)` - convert the comma-space list into lines separated by newlines

How can I easily repeat a formula in a table?

To repeat the same formula in all cells of a table row define the formula once in a preferences setting and use that in the CALC. The preferences setting can be defined at the site level, web level or topic level, and may be hidden in HTML comments. Example:

```
<!--
   * Set MYFORMULA = $EVAL($SUBSTITUTE(...etc...))
-->
| A | 1 | %CALC{%MYFORMULA%}% |
| B | 2 | %CALC{%MYFORMULA%}% |
| C | 3 | %CALC{%MYFORMULA%}% |
```

CALC in Included Topics

By default, CALCs in an included topic are evaluated with delay. The SKIPINCLUDE setting tells the plugin to evaluate the CALCs once all INCLUDEs are processed. This default behavior is chosen so that it is possible to compose a bigger table from several includes and do some spreadsheet calculation over the whole table.

Attention: You can get unexpected results if you INCLUDE a topic that has other variables taking action on CALCs. For example, a CHART in an included topic sees unprocessed CALCs, which may result in a chart with incorrect values. To get he desired result you need to set the following preference setting in the topic that *includes* the topic containing the CHART:

- Set SPREADSHEETPLUGIN_SKIPINCLUDE = 0

This setting tells the SpreadSheetPlugin to process the CALCs in the included page, e.g. it will not delay the evaluation of the functions.

Bug Tracking Example

Bug#:	Priority:	Subject:	Status:	Days to fix
Bug:1231	Low	File Open ...	Open	3
Bug:1232	High	Memory Window ...	Fixed	2
Bug:1233	Medium	Usability issue ...	Assigned	5
Bug:1234	High	No arrange ...	Fixed	1
Total: 4	High: 2 Low: 1 Medium: 1	.	Assigned: 1 Fixed: 2 Open: 1	Total: 11

The last row is defined as:

```
| Total: %CALC{$ROW(-2)}% \
  | %CALC{$COUNTITEMS( R2:C$COLUMN()..R$ROW(-1):C$COLUMN() )}% | . \
  | %CALC{$COUNTITEMS( R2:C$COLUMN()..R$ROW(-1):C$COLUMN() )}% \
  |  Total: %CALC{$SUM( R2:C$COLUMN()..R$ROW(-1):C$COLUMN() )}% |
```

Above table is created manually. The table can be build dynamically with a formatted search, or by a plugin that pulls data from an external source, such as a bug tracking system.

Plugin Settings

Show details Hide details
Plugin settings are stored as preferences variables. To reference a plugin setting write **%<plugin>_<setting>%**, i.e. **%SPREADSHEETPLUGIN_SHORTDESCRIPTION%**

- One line description, is shown in the TextFormattingRules topic:
  - Set SHORTDESCRIPTION = Add spreadsheet calculation like "`$SUM( $ABOVE() )`" to TWiki tables or anywhere in topic text

- Debug plugin: (See output in `data/debug.txt`)
  - Set DEBUG = 0

- Do not handle `%CALC{}%` variable in included topic while including topic: (default: 1) (See note CALC in Included Topics)
  - Set SKIPINCLUDE = 1

- Currency symbol. Specify `US$` or `$` for Dollar, `&#8364;` for Euro, `&#165;` for Yen:
  - Set CURRENCYSYMBOL = $

- WikiWords to exclude from being spaced out by the **$PROPERSPACE(text)** function. This comma delimited list can be overloaded by a DONTSPACE preferences variable:
  - Set DONTSPACE = CodeWarrior, MacDonald, McIntosh, RedHat, SuSE

Plugin Installation Instructions

This plugin is pre-installed. TWiki administrators can upgrade the plugin as needed on the TWiki server.

Show details Hide details

- For an ***automated installation***, run the configure script and follow "Find More Extensions" in the in the ***Extensions*** section.

- Or, follow these ***manual installation*** steps:
  - Download the ZIP file from the Plugins home (see below).
  - Unzip **SpreadSheetPlugin.zip** in your twiki installation directory. Content:

File:	Description:
data/TWiki/SpreadSheetPlugin.txt	Plugin topic
data/TWiki/SpreadSheetPluginTestCases.txt	Test cases
data/TWiki/VarCALC.txt	Documentation of the CALC variable
data/TWiki/VarCALCULATE.txt	Documentation of the CALCULATE variable
lib/TWiki/Plugins/SpreadSheetPlugin.pm	Plugin Perl module
lib/TWiki/Plugins/SpreadSheetPlugin/Calc.pm	Plugin core module

  - Set the ownership of the extracted directories and files to the webserver user.

- Plugin ***configuration and testing***:
  - Run the configure script and enable the plugin in the ***Plugins*** section.
  - Test if the installation was successful: See example above and SpreadSheetPluginTestCases.

Plugin Info

Plugin Author:	TWiki:Main.PeterThoeny
Copyright:	© 2001-2018 Peter Thoeny, TWiki.org © 2008-2018 TWiki:TWiki.TWikiContributor © 2015 Wave Systems Corp.
Sponsor:	Wave Systems Corp. for hash functions
License:	GPL (GNU General Public License)
Plugin Version:	2018-07-05

Show Change History Hide Change History

2018-07-05:	TWikibug:Item7841: Copyright update to 2018
2017-10-21:	TWikibug:Item7826: Test case for HASHREVERSE now works with Perl's hash key randomization; TWikibug:Item7827: Workaround for SPLIT with unthreaded Perl
2017-10-19:	TWikibug:Item7640: Fix bug for unit tests issue: DEC2HEX and DEC2BIN create longer strings with 64bit Perl
2016-07-05:	TWikibug:Item7746: Fix bug LIST2HASH fails with empty list entries
2016-05-10:	TWikibug:Item7703: Document deprecated LISTMAP() and MULT()
2016-01-08:	TWikibug:Item7708: Copyright update to 2016
2015-06-07:	TWikibug:Item7656: Rename COMMENT() to NOTE() - if someone uses COMMENT we can restore it as an undocumented function
2015-05-18:	TWikibug:Item7656: Add COMMENT(), EQUAL()
2015-02-24:	TWikibug:Item7617: Add range check in SUBSTRING()
2015-01-14:	TWikibug:Item7607: PROPERSPACE() to space also uppercase to number transition
2015-01-09:	TWikibug:Item7604: Switch to GPL v3
2015-01-07:	TWikibug:Item7583: Code cleanup for RANDSTRING()
2014-10-23:	TWikibug:Item7583: Add RANDSTRING(); better layout in interactive example
2014-09-23:	TWikibug:Item7553: Allow newlines and indent around functions and function parameters
2014-09-22:	TWikibug:Item7552: Allow newlines in triple-quoted strings
2014-03-04:	TWikibug:Item7445: Add FORMAT(CURRENY, ...) with support for currency symbol
2014-01-22:	TWikibug:Item7419: Add ADDLIST(), GETLIST(), SETLIST()
2014-01-22:	TWikibug:Item7418: Fix VALUE function bug with incorrect exponential number; ability to delete SET variable; do not strip trailing spaces in SETIFEMPTY
2013-10-10:	TWikibug:Item7154: Doc update: Put TOC on top right for easy reference
2013-09-14:	TWikibug:Item7299: Fix plural of month to months in FORMATTIMEDIFF
2013-08-21:	TWikibug:Item7322: TRANSLATE supporting double quote and apostrophe-quote (single quote) escape tokens
2013-07-18:	TWikibug:Item7299: Add short and compact format to FORMATTIMEDIFF
2013-06-20:	TWikibug:Item7154: Fix encoding of URL parameter in the interactive example so that `$IF(1>2, true, false)` works properly
2013-05-21:	TWikibug:Item7154: Small doc fixes
2013-04-09:	TWikibug:Item7221: Fix for CALC referencing preceding table returning incorrect last cell value
2013-04-07:	TWikibug:Item7218: Support and document ROW() and COLUMN() below a table
2013-03-26:	TWikibug:Item7203: Support '''triple quoted''' parameters for strings that contain comma and parenthesis
2013-03-25:	TWikibug:Item7199: Remove restriction on permitted characters for hash key
2013-03-24:	TWikibug:Item7199: Added HASCOPY(), HASHEACH(); renamed LISTMAP() to LISTEACH() while keeping LISTMAP() as an undocumented feature
2013-03-23:	

	TWikibug:Item7199: Added LIST2HASH(), HASH2LIST(), HASHEXISTS(), HASHREVERSE(), SETMHASH()
2013-03-21:	TWikibug:Item7199: Added SETHASH(), GETHASH()
2013-03-14:	TWikibug:Item7196: Added BIN2DEC(), DEC2BIN(), DEC2HEX(), DEC2OCT(), HEX2DEC(), OCT2DEC()
2013-03-14:	TWikibug:Item7190: Enumeration of function brackets should never be negative
2013-03-11:	TWikibug:Item7184: Allow commas in SUBSTRING input string
2013-01-09:	TWikibug:Item7091: Use TWISTY in variable section, installation instructions and change history
2012-11-11:	TWikibug:Item7020: Categorize TWiki Variables CALC and CALCULATE
2012-11-05:	TWikibug:Item7023: Added $STDEV(), $STDEVP(), $VAR(), $VARP()
2012-11-03:	TWikibug:Item7018: Added SpreadSheetPluginTestCases topic; refactor plugin to use function hash for better performance
2012-10-06:	TWikibug:Item6960: Fixed form action of interactive formula evaluation feature -- TWiki:Main.HideyoImazu
2012-06-30:	TWikibug:Item6898: Added $FILTER()
2012-06-29:	TWikibug:Item6897: Added $ISDIGIT(), $ISLOWER(), $ISUPPER(), $ISWIKIWORD()
2012-04-04:	TWikibug:Item6866: Added CALCULATE variable using register tag handler to support proper inside-out, left-to-right eval order like ordinary TWiki variables
2012-01-13:	TWikibug:Item6804: Added $FLOOR() and $CEILING()
2011-09-07:	TWikibug:Item6803: Fix for EVAL function bug with zeroes after decimal point
2011-07-09:	TWikibug:Item6725: Change global package variables from "use vars" to "our"
2011-04-25:	TWikibug:Item6690: Added $BITXOR(), $HEXDECODE(), $HEXENCODE(), $XOR()
2011-04-08:	TWikibug:Item6681: Added $WHILE()
2011-04-06:	TWikibug:Item6679: Fixed small issue with $LISTRAND(), where last item only got 50% of fair share to get picked
2011-03-25:	TWikibug:Item6669: Added $LISTNONEMPTY()
2011-03-24:	TWikibug:Item6668: Fixed $LIST() not flattening a list in a cell TWikibug:Item6667: Fixed $RIGHT() having wrong result due to incorrect start cell
2011-03-22:	TWikibug:Item6666: Added $SPLIT(); renamed $nop separator token of $LISTJOIN() to $empty (keeping $nop as undocumented feature)
2010-08-27:	TWikibug:Item6526: Added ISO 8601 week number to $FORMATTIME(), contributed by TWiki:Main.PeterPayne
2010-08-04:	TWikibug:Item6537: Fixed for $EVAL(2+08) returning "illegal octal digit" error instead of 10
2010-07-17:	TWikibug:Item6525: Added $n token to TRANSLATE for newline
2010-05-27:	TWikibug:Item6506: Document delayed evaluation of CALC in included topics
2010-05-26:	TWikibug:Item6504: Added empty ($nop) separator to $LISTJOIN()
2010-06-25:	TWikibug:Item6493: Fixed $PRODUCT(0,4) returning 4 instead of 0 TWikibug:Item5163: Fix for plugin causing table to misrender an empty "||" row
2010-05-22:	TWikibug:Item6472: Added support for DOY in $TIME(), contributed by TWiki:Main/EmanueleCupido
2010-05-15:	TWikibug:Item6433: Doc improvements; replacing TWIKIWEB with SYSTEMWEB
2010-02-27:	Doc improvements
2009-11-22:	Enhanced $NOP(): Added $quot replacement for quote character, changed $per replacement with $percnt, contributed by TWiki:Main/HorstEsser
09 May 2009:	Fixed bug in $WORKINGDAYS(): Incorrect number of days if daylight savings time change happens between start date and end date
26 Mar 2009:	Added $INSERTSTRING()

25 Mar 2009:	Added $EMPTY(), $LEFTSTRING(), $RIGHTSTRING(), $SUBSTRING()
24 Mar 2009:	Fixed bug in $REPLACE() if to-be-replaced string is "0"; fixed bug in $SUBSTITUTE() if replace string is empty; improved docs
13 Oct 2007:	Added $FORMATTIMEDIFF()
09 Sep 2007:	Enhanced documentation for $EVAL() and $INT()
02 Jun 2007:	Added VarCALC to have `%CALC{}%` listed in TWikiVariables
14 Apr 2007:	Fixing bug in $EXISTS() that required full `web.topic` instead of just `topic`
11 Mar 2007:	Fixing bug in $VALUE() and $INT(), introduced by version 09 Mar 2007
09 Mar 2007:	Added $EXP(), $LN(), $LOG(), $PI(), $SQRT(); fixed $ROUND() bug, contributed by TWiki:Main/SergejZnamenskij
23 Jan 2007:	Enhanced documentation
18 Dec 2006:	Added $LISTRAND(), $LISTSHUFFLE(), $LISTTRUNCATE(); fixed spurious newline at end of topic, contributed by TWiki:Main/MichaelDaum
10 Oct 2006:	Enhanced documentation
13 May 2006:	Added $SETIFEMPTY(); fixes in documentation
17 Jun 2005:	Added $NOEXEC(), $EXEC()
25 Mar 2005:	Fixed evaluation bug when using SpeedyCGI accelerator; code refactor to load module only when needed, contributed by TWiki:Main/CrawfordCurrie
24 Oct 2004:	Added $EXISTS(), contributed by TWiki:Main/RodrigoChandia; added $PERCENTILE()
18 Oct 2004:	Added $LISTJOIN()
26 Sep 2004:	Added $FORMAT(KB), $FORMAT(MB), contributed by TWiki:Main/ArthurClemens; added $FORMAT(KBMB), $EVEN(), $ODD()
17 Jul 2004:	Added $WORKINGDAYS(), contributed by TWiki:Main/CrawfordCurrie
24 May 2004:	Refactored documentation (no code changes)
03 Apr 2004:	Added $ABS(), $LISTIF(); fixed $VALUE() to remove leading zeros; changed $FIND() and $SEARCH() to return 0 instead of empty string if no match
21 Mar 2004:	Added $LISTITEM(); fixed call to unofficial function
16 Mar 2004:	Added $LISTMAP(), $LISTREVERSE(), $LISTSIZE(), $LISTSORT(), $LISTUNIQUE(), $SETM(); retired $COUNTUNIQUE() in favor of $COUNTITEMS($LISTUNIQUE()); fixed evaluation order issue of $IF(); fixed missing eval error messages suppressed since version 06 Mar 2004; redirect stderr messages to warning
08 Mar 2004:	Added $LIST()
06 Mar 2004:	Added $AND(), $MOD(), $NOT(), $OR(), $PRODUCT(), $PROPER(), $PROPERSPACE(), $RAND(), $REPEAT(), $SIGN(), $VALUE(); added digits parameter to $ROUND(); renamed $MULT() to $PRODUCT(); $MULT() is deprecated and undocumented
27 Feb 2004:	Added $COUNTUNIQUE()
24 Oct 2003:	Added $SET(), $GET(), $MEDIAN(); added $SUMPRODUCT(), inspired by TWiki:Main/RobertWithrow; added $SUMDAYS(), contributed by TWiki:Main/SvenDowideit
21 Oct 2003:	Added support for lists `(1, 2, 3)` and lists of table ranges `(R1:C1..R1:C5, R3:C1..R3:C5)` for all functions that accept a table range; added $TIMEADD(); in $TIMEDIFF() added week unit; in $FORMATTIME() changed $weekday to $wd and added $wday and $weekday
14 Oct 2003:	Added $TIME(), $TODAY(), $FORMATTIME(), $FORMATGMTIME(), $TIMEDIFF()
13 Oct 2003:	Added $MULT(), contributed by TWiki:Main/GerritJanBaarda
30 Jul 2003:	Added $TRANSLATE()
19 Jul 2003:	Added $FIND(), $NOP(), $REPLACE(), $SEARCH(), $SUBSTITUTE(), contributed by TWiki:Main/PaulineCheung

19 Apr 2003:	Added $COUNTSTR(), $EXACT(), $IF(), $ROUND(), $TRIM(); added $FORMAT(), contributed by TWiki:Main/JimStraus; support % modulus operator in $EVAL(), $INT(), and $ROUND(); fixed bug in $DEF()
07 Jun 2002:	Added $DEF(), contributed by TWiki:Main/MartinFuzzey; allow values with HTML formatting like <u>102</u>, suggested by TWiki:Main/GladeDiviney; added SKIPINCLUDE setting
12 Mar 2002:	Support for multiple functions per nesting level
15 Jan 2002:	Added $CHAR(), $CODE() and $LENGTH()
12 Nov 2001:	Added $RIGHT()
12 Aug 2001:	Fixed bug of disappearing multi-column cells
19 Jul 2001:	Fixed incorrect $SUM() calculation of cell with value 0
14 Jul 2001:	Changed to plug & play
01 Jun 2001:	Fixed insecure dependencies for $MIN() and $MAX()
16 Apr 2001:	Fixed div by 0 bug in $AVERAGE()
17 Mar 2001:	Initial version with $ABOVE(), $AVERAGE(), $COLUMN(), $COUNTITEMS(), $EVAL(), $INT(), $LEFT(), $LOWER(), $MAX(), $MIN(), $ROW(), $SUM(), $T(), $UPPER()
CPAN Dependencies:	none
Plugin Benchmarks:	GoodStyle 99%, FormattedSearch 99%, SpreadSheetPlugin 95%
Other Dependencies:	none
Perl Version:	5.000 and up
Plugin Home:	http://TWiki.org/cgi-bin/view/Plugins/SpreadSheetPlugin
Feedback:	http://TWiki.org/cgi-bin/view/Plugins/SpreadSheetPluginDev
Appraisal:	http://TWiki.org/cgi-bin/view/Plugins/SpreadSheetPluginAppraisal

Related Topics: SpreadSheetPluginTestCases, TWikiPreferences, TWikiPlugins, VarCALC, VarCALCULATE, VarIF

This topic: TWiki > SpreadSheetPlugin
Topic revision: r1 - 2018-07-06 - TWikiContributor

or Ideas, requests, problems regarding TWiki? use Discourse or Send feedback
Note: Please contribute updates to this topic on TWiki.org at TWiki:TWiki.SpreadSheetPlugin