Modul:Math/Doku

Dies ist die Dokumentationsseite für Modul:Math

Vorlage:For This module provides a number of mathematical functions. These functions can be used from #invoke or from other Lua modules.

Inhaltsverzeichnis

1 Use from other Lua modules 1
2 random 1
3 order 2
4 precision 2
5 max 3
6 median 3
7 min 3
8 sum 3
9 average 3
10 round 4
11 log10 4
12 mod 4
13 gcd 4
14 precision_format 5
15 divide 5
16 cleanNumber 6
17 See also 6

Use from other Lua modules

To use the module from normal wiki pages, no special preparation is needed. If you are using the module from another Lua module, first you need to load it, like this:

```
local mm = require('Module:Math')
```

(The `mm` variable stands for **M**odule **M**ath; you can choose something more descriptive if you prefer.)

Most functions in the module have a version for Lua and a version for #invoke. It is possible to use the #invoke functions from other Lua modules, but using the Lua functions has the advantage that you do not need to access a Lua frame object. Lua functions are preceded by `_`, whereas #invoke functions are not.

random

Vorlage:See also

```
{{#invoke:math|random}}
{{#invoke:math|random|max_value}}
{{#invoke:math|random|min_value|max_value}}
```

```
mm._random()
mm._random(max_value)
mm._random(min_value, max_value)
```

Generates a random number.

- If no arguments are specified, the number produced is greater than or equal to 0 and less than 1.
- If one argument is provided, the number produced is an integer between 1 and that argument. The argument must be a positive integer.
- If two arguments are provided, the number produced is an integer between the first and second arguments. Both arguments must be integers, but can be negative.

This function will not work properly for numbers less than -2^{32} and greater than $2^{32} - 1$. If you need to use numbers outside of this range, it is recommended that you use Module:Random.

order

```
{{#invoke:math|order|n}}
```

```
mm._order(n)
```

Determines the order of magnitude of a number.

precision

```
{{#invoke:math|precision|n}}
{{#invoke:math|precision|x=n}}
```

```
mm._precision(number_string)
```

Detemines the precision of a number. For example, for "4" it will return "0", for "4.567" it will return "3", and for "100" it will return "-2".

The function attempts to parse the string representation of the number, and detects whether the number uses E notation. For this reason, when called from Lua, very large numbers or very precise numbers should be directly input as strings to get accurate results. If they are input as numbers, the Lua interpreter will change them to E notation and this function will return the precision of the E notation rather than that of the original number. This is not a problem when the number is called from #invoke, as all input from #invoke is in string format.

max

```
{{#invoke:math|max|v1|v2|v3|...}}
```

```
mm._max(v1, v2, v3, ...)
```

Returns the maximum value from the values specified. Values that cannot be converted to numbers are ignored.

median

```
{{#invoke:math|median|v1|v2|v3|...}}
```

```
mm._median(v1, v2, v3, ...)
```

Returns the median value from the values specified. Values that cannot be converted to numbers are ignored.

min

```
{{#invoke:math|min|v1|v2|v3|...}}
```

```
mm._min(v1, v2, v3, ...)
```

Returns the minimum value from the values specified. Values that cannot be converted to numbers are ignored.

sum

```
{{#invoke:math|sum|v1|v2|v3|...}}
```

```
mm._sum(v1, v2, v3, ...)
```

Returns the sum of the values specified. Values that cannot be converted to numbers are ignored.

average

```
{{#invoke:math|average|v1|v2|v3|...}}
```

```
mm._average(v1, v2, v3, ...)
```

Returns the average of the values specified. (More precisely, the value returned is the arithmetic mean.) Values that cannot be converted to numbers are ignored.

round

```
{{#invoke:math|round|value|precision}}
{{#invoke:math|round|value=value|precision=precision}}
```

```
mm._round(value, precision)
```

Rounds a number to the specified precision.

Note: As of October 2019, there is a bug in the display of some rounded numbers. When trying to round a number that rounds to "n.0", like "1.02", to the nearest tenth of a digit (i.e. Vorlage:Para), this function should display "1.0", but it unexpectedly displays "1". Use the Vorlage:Para parameter instead.

log10

```
{{#invoke:math | log10 | x}}
```

```
mm._log10(x)
```

Returns $\log_{10}(x)$, the logarithm of x using base 10.

mod

```
{{#invoke:math|mod|x|y}}
```

```
mm._mod(x, y)
```

Gets x modulo y, or the remainder after x has been divided by y. This is accurate for integers up to 2^{53}; for larger integers Lua's modulo operator may return an erroneous value. This function deals with this problem by returning `0` if the modulo given by Lua's modulo operator is less than 0 or greater than y.

gcd

```
{{#invoke:math|gcd|v1|v2|...}}
```

```
mm._gcd(v1, v2, ...)
```

Finds the greatest common divisor of the values specified. Values that cannot be converted to numbers are ignored.

precision_format

```
{{#invoke:math|precision_format|value_string|precision}}
```

```
mm._precision_format(value_string, precision)
```

Rounds a number to the specified precision and formats according to rules originally used for Vorlage:Tl. Output is a string.

Parameter *precision* should be an integer number of digits after the decimal point. Negative values are permitted. Non-integers give unexpected results. Positive values greater than the input precision add zero-padding, negative values greater than the input order can consume all digits.

Formatting 8,765.567 with Vorlage:Tlc gives:

precision	Result
2	8.765,57
-2	8.800
6	8.765,567000
-6	0
2.5	8.765,5680426633
-2.5	8.854,3774484715

divide

```
{{#invoke:Math|divide|x|y|round=|precision=}}
```

```
mm._divide(x, y, round, precision)
```

Divide x by y.

- If y if not a number, it is returned.

- Otherwise, if x is not a number, it is returned.
- If round is true ("yes" for #invoke), the result has no decimals
- Precision indicates how many digits of precision the result should have

If any of the arguments contain HTML tags, they are returned unchanged, allowing any errors in calculating the arguments to the division function to be propagated to the calling template.

cleanNumber

```
local number, number_string = mm._cleanNumber(number_string)
```

A helper function that can be called from other Lua modules, but not from #invoke. This takes a string or a number value as input, and if the value can be converted to a number, cleanNumber returns the number and the number string. If the value cannot be converted to a number, cleanNumber returns `nil, nil`.

See also

Vorlage:Math templates