Mathematical Functions

Introduction

These math functions will only handle values within the range of the integer and float types on your computer (this corresponds currently to the C types long resp. double). If you need to handle bigger numbers, take a look at the arbitrary precision math functions.

See also the manual page on arithmetic operators.

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are always available as part of the PHP core.

Math constants

Constant	Value	Description
M_PI	3.14159265358979323846	Pi
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2
M_LN10	2.30258509299404568402	log_e 10
M_PI_2	1.57079632679489661923	pi/2
M_PI_4	0.78539816339744830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_SQRTPI	1.77245385090551602729	sqrt(pi) [5.2.0]
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT3	1.73205080756887729352	sqrt(3) [5.2.0]
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)
M_LNPI	1.14472988584940017414	log_e(pi) [5.2.0]
M_EULER	0.57721566490153286061	Euler constant [5.2.0]

Only M_PI is available in PHP versions up to and including PHP 4.0.0. All other constants are available starting with PHP 4.0.0. Constants labeled [5.2.0] were added in PHP 5.2.0.

Math Functions

abs

abs -- Absolute value

Description

number **abs** (mixed `$number`)

Returns the absolute value of *`number`*.

Parameters

`number`
The numeric value to process

Return Values

The absoulte value of *`number`*. If the argument *`number`* is of type float, the return type is also float, otherwise it is integer (as float usually has a bigger value range than integer).

Examples

Example #1 - abs() example

```
<?php
$abs = abs(-4.2); // $abs = 4.2; (double/float)
$abs2 = abs(5);   // $abs2 = 5; (integer)
$abs3 = abs(-5);  // $abs3 = 5; (integer)
?>
```

acos

acos -- Arc cosine

Description

float **acos** (float `$arg`)

Returns the arc cosine of *`arg`* in radians. acos() is the complementary function of cos(), which means that *a==cos(acos(a))* for every value of a that is within acos() ' range.

Parameters

`arg`
: The argument to process

Return Values

The arc consine of *`arg`* in radians.

See Also

- cos()
- acosh()
- asin()
- atan()

acosh

acosh -- Inverse hyperbolic cosine

Description

float **acosh** (float `$arg`)

Returns the inverse hyperbolic cosine of *`arg`*, i.e. the value whose hyperbolic cosine is *`arg`*.

Parameters

`arg`
: The value to process

Return Values

The inverse hyperbolic cosine of *`arg`*

ChangeLog

Version	Description
5.3.0	This function is now available on all platforms

See Also

- cosh()
- acos()
- asinh()
- atanh()

asin

asin -- Arc sine

Description

float **asin** (float `$arg`)

Returns the arc sine of *`arg`* in radians. asin() is the complementary function of sin(), which means that *a==sin(asin(a))* for every value of a that is within asin() 's range.

Parameters

`arg`
: The argument to process

Return Values

The arc sine of *`arg`* in radians

See Also

- sin()
- asinh()
- acos()
- atan()

asinh

asinh -- Inverse hyperbolic sine

Description

float **asinh** (float `$arg`)

Returns the inverse hyperbolic sine of *`arg`*, i.e. the value whose hyperbolic sine is *`arg`*.

Parameters

`arg`
: The argument to process

Return Values

The inverse hyperbolic sine of *`arg`*

ChangeLog

Version	Description
5.3.0	This function is now available on all platforms

See Also

- sinh()
- asin()
- acosh()
- atanh()

atan2

atan2 -- Arc tangent of two variables

Description

float **atan2** (float $y, float $x)

This function calculates the arc tangent of the two variables *x* and *y*. It is similar to calculating the arc tangent of *y* / *x*, except that the signs of both arguments are used to determine the quadrant of the result.

The function returns the result in radians, which is between -PI and PI (inclusive).

Parameters

y
: Dividend parameter

x
: Divisor parameter

Return Values

The arc tangent of *y* / *x* in radians.

See Also

- atan()

atan

atan -- Arc tangent

Description

float **atan** (float `$arg`)

Returns the arc tangent of *`arg`* in radians. atan() is the complementary function of tan(), which means that *a==tan(atan(a))* for every value of a that is within atan() 's range.

Parameters

`arg`
: The argument to process

Return Values

The arc tangent of *`arg`* in radians.

See Also

- tan()
- atanh()
- asin()
- acos()

atanh

atanh -- Inverse hyperbolic tangent

Description

float **atanh** (float `$arg`)

Returns the inverse hyperbolic tangent of *`arg`*, i.e. the value whose hyperbolic tangent is *`arg`*.

Parameters

`arg`
: The argument to process

Return Values

Inverse hyperbolic tangent of *`arg`*

ChangeLog

Version	Description
5.3.0	This function is now available on all platforms

See Also

- tanh()
- atan()
- asinh()
- acosh()

base_convert

base_convert -- Convert a number between arbitrary bases

Description

string **base_convert** (string `$number`, int `$frombase`, int `$tobase`)

Returns a string containing *`number`* represented in base *`tobase`*. The base in which *`number`* is given is specified in *`frombase`*. Both *`frombase`* and *`tobase`* have to be between 2 and 36, inclusive. Digits in numbers with a base higher than 10 will be represented with the letters a-z, with a meaning 10, b meaning 11 and z meaning 35.

Warning

base_convert() may lose precision on large numbers due to properties related to the internal "double" or "float" type used. Please see the Floating point numbers section in the manual for more specific information and limitations.

Parameters

`number`
: The number to convert

`frombase`
: The base *`number`* is in

`tobase`
: The base to convert *`number`* to

Return Values

`number` converted to base *`tobase`*

Examples

Example #2 - base_convert() example

```
<?php
$hexadecimal = 'A37334';
echo base_convert($hexadecimal, 16, 2);
?>
```

The above example will output:

```
10100011011100110011010 0
```

See Also

- intval()

bindec

bindec -- Binary to decimal

Description

number **bindec** (string `$binary_string`)

Returns the decimal equivalent of the binary number represented by the *`binary_string`* argument.

bindec() converts a binary number to an integer or, if needed for size reasons, float.

Parameters

`binary_string`
The binary string to convert

Return Values

The decimal value of *`binary_string`*

ChangeLog

Version	Description
Since 4.1.0	The function can now convert numbers that are too large to fit into the platforms integer type, larger values are returned as float in that case.

Examples

Example #3 - bindec() example

```
<?php
echo bindec('110011') . "\n";
echo bindec('000110011') . "\n";

echo bindec('111');
?>
```

The above example will output:

```
51
51
7
```

See Also

- decbin()
- octdec()
- hexdec()
- base_convert()

ceil

ceil -- Round fractions up

Description

float **ceil** (float `$value`)

Returns the next highest integer value by rounding up *`value`* if necessary.

Parameters

`value`
The value to round

Return Values

`value` rounded up to the next highest integer. The return value of ceil() is still of type float as the value range of float is usually bigger than that of integer.

Examples

Example #4 - ceil() example

```
<?php
echo ceil(4.3);    // 5
echo ceil(9.999);  // 10
echo ceil(-3.14);  // -3
?>
```

See Also

- floor()
- round()

cos

cos -- Cosine

Description

float **cos** (float `$arg`)

cos() returns the cosine of the *`arg`* parameter. The *`arg`* parameter is in radians.

Parameters

`arg`
An angle in radians

Return Values

The cosine of *`arg`*

Examples

Example #5 - cos() example

```
<?php

echo cos(M_PI); // -1

?>
```

See Also

- acos()
- sin()
- tan()
- deg2rad()

cosh

cosh -- Hyperbolic cosine

Description

float **cosh** (float `$arg`)

Returns the hyperbolic cosine of *`arg`*, defined as *(exp(arg) + exp(-arg))/2*.

Parameters

`arg`
: The argument to process

Return Values

The hyperbolic cosine of *`arg`*

See Also

- cos()
- acosh()
- sinh()
- cosh()

decbin

decbin -- Decimal to binary

Description

string **decbin** (int `$number`)

Returns a string containing a binary representation of the given *`number`* argument. The largest number that can be converted is 4294967295 in decimal resulting to a string of 32 1's.

Parameters

`number`
Decimal value to convert

Return Values

Binary string representation of *`number`*

Examples

Example #6 - decbin() example

```
<?php
echo decbin(12) . "\n";
echo decbin(26);
?>
```

The above example will output:

```
1100
11010
```

See Also

- bindec()
- decoct()
- dechex()
- base_convert()

dechex

dechex -- Decimal to hexadecimal

Description

string **dechex** (int `$number`)

Returns a string containing a hexadecimal representation of the given *`number`* argument. The largest number that can be converted is 4294967295 in decimal resulting to "ffffffff".

Parameters

`number`
Decimal value to convert

Return Values

Hexadecimal string representation of *`number`*

Examples

Example #7 - dechex() example

```
<?php
echo dechex(10) . "\n";
echo dechex(47);
?>
```

The above example will output:

```
a
2f
```

See Also

- hexdec()
- decbin()
- decoct()
- base_convert()

decoct

decoct -- Decimal to octal

Description

string **decoct** (int `$number`)

Returns a string containing an octal representation of the given *`number`* argument. The largest number that can be converted is 4294967295 in decimal resulting to "37777777777".

Parameters

`number`
Decimal value to convert

Return Values

Octal string representation of *`number`*

Examples

Example #8 - decoct() **example**

```
<?php
echo decoct(15) . "\n";
echo decoct(264);
?>
```

The above example will output:

```
17
410
```

See Also

- octdec()
- decbin()
- dechex()
- base_convert()

deg2rad

deg2rad -- Converts the number in degrees to the radian equivalent

Description

float **deg2rad** (float `$number`)

This function converts *`number`* from degrees to the radian equivalent.

Parameters

`number`
: Angular value in degrees

Return Values

The radian equivalent of *`number`*

Examples

Example #9 - deg2rad() example

```
<?php

echo deg2rad(45); // 0.785398163397
var_dump(deg2rad(45) === M_PI_4); // bool(true)

?>
```

See Also

- rad2deg()

exp

exp -- Calculates the exponent of **e**

Description

float **exp** (float `$arg`)

Returns **e** raised to the power of *`arg`*.

Note
'e' is the base of the natural system of logarithms, or approximately 2.718282.

Parameters

`arg`
The argument to process

Return Values

'e' raised to the power of *`arg`*

Examples

Example #10 - exp() example

```
<?php
echo exp(12) . "\n";
echo exp(5.7);
?>
```

The above example will output:

```
1.6275E+005
298.87
```

See Also

- log()
- pow()

expm1

expm1 -- Returns exp(number) - 1, computed in a way that is accurate even when the value of number is close to zero

Description

float **expm1** (float `$arg`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

expm1() returns the equivalent to 'exp(*`arg`*) - 1' computed in a way that is accurate even if the value of *`arg`* is near zero, a case where 'exp (*`arg`*) - 1' would be inaccurate due to subtraction of two numbers that are nearly equal.

Parameters

`arg`
: The argument to process

Return Values

'e' to the power of *`arg`* minus one

ChangeLog

Version	Description
5.3.0	This function is now available on all platforms

See Also

- log1p()

- exp()

floor

floor -- Round fractions down

Description

float **floor** (float `$value`)

Returns the next lowest integer value by rounding down *`value`* if necessary.

Parameters

`number`
The numeric value to round

Return Values

`value` rounded to the next lowest integer. The return value of floor() is still of type float because the value range of float is usually bigger than that of integer.

Examples

Example #11 - floor() example

```
<?php
echo floor(4.3);   // 4
echo floor(9.999); // 9
echo floor(-3.14); // -4
?>
```

See Also

- ceil()
- round()

fmod

fmod -- Returns the floating point remainder (modulo) of the division of the arguments

Description

float **fmod** (float $x, float $y)

Returns the floating point remainder of dividing the dividend (*x*) by the divisor (*y*). The reminder (*r*) is defined as: x = i * y + r, for some integer *i*. If *y* is non-zero, *r* has the same sign as *x* and a magnitude less than the magnitude of *y*.

Parameters

x
The dividend

y
The divisor

Return Values

The floating point remainder of *x* / *y*

Examples

Example #12 - Using fmod()

```
<?php
$x = 5.7;
$y = 1.3;
$r = fmod($x, $y);
// $r equals 0.5, because 4 * 1.3 + 0.5 = 5.7
?>
```

getrandmax

getrandmax -- Show largest possible random value

Description

int **getrandmax** (void)

Returns the maximum value that can be returned by a call to rand().

Return Values

The largest possible random value returned by rand()

See Also

- rand()
- srand()
- mt_getrandmax()

hexdec

hexdec -- Hexadecimal to decimal

Description

number **hexdec** (string `$hex_string`)

Returns the decimal equivalent of the hexadecimal number represented by the *`hex_string`* argument. hexdec() converts a hexadecimal string to a decimal number.

hexdec() will ignore any non-hexadecimal characters it encounters.

Parameters

`hex_string`
The hexadecimal string to convert

Return Values

The decimal representation of *`hex_string`*

ChangeLog

Version	Description
Since 4.1.0	The function can now convert values that are to big for the platforms integer type, it will return the value as float instead in that case.

Examples

Example #13 - hexdec() example

```
<?php
var_dump(hexdec("See"));
var_dump(hexdec("ee"));
// both print "int(238)"

var_dump(hexdec("that")); // print "int(10)"
```

```
var_dump(hexdec("a0")); // print "int(160)"
?>
```

See Also

- dechex()
- bindec()
- octdec()
- base_convert()

hypot

hypot -- Calculate the length of the hypotenuse of a right-angle triangle

Description

float **hypot** (float $x, float $y)

hypot() returns the length of the hypotenuse of a right-angle triangle with sides of length *x* and *y*, or the distance of the point (*x*, *y*) from the origin. This is equivalent to *sqrt(x*x + y*y)*.

Parameters

x
: Length of first side

y
: Length of second side

Return Values

Calculated length of the hypotenuse

is_finite

is_finite -- Finds whether a value is a legal finite number

Description

bool **is_finite** (float `$val`)

Checks whether *`val`* is a legal finite on this platform.

Parameters

`val`
: The value to check

Return Values

TRUE if *`val`* is a legal finite number within the allowed range for a PHP float on this platform, else **FALSE**.

See Also

- is_infinite()
- is_nan()

is_infinite

is_infinite -- Finds whether a value is infinite

Description

bool **is_infinite** (float `$val`)

Returns **TRUE** if *`val`* is infinite (positive or negative), like the result of *log(0)* or any value too big to fit into a float on this platform.

Parameters

`val`
: The value to check

Return Values

TRUE if *`val`* is infinite, else **FALSE**.

See Also

- is_finite()
- is_nan()

is_nan

is_nan -- Finds whether a value is not a number

Description

bool **is_nan** (float `$val`)

Checks whether *`val`* is 'not a number', like the result of *acos(1.01)*.

Parameters

`val`
: The value to check

Return Values

Returns **TRUE** if *`val`* is 'not a number', else **FALSE**.

See Also

- is_finite()
- is_infinite()

lcg_value

lcg_value -- Combined linear congruential generator

Description

float **lcg_value** (void)

lcg_value() returns a pseudo random number in the range of (0, 1). The function combines two CGs with periods of 2^31 - 85 and 2^31 - 249. The period of this function is equal to the product of both primes.

Return Values

A pseudo random float value in the range of (0, 1)

See Also

- rand()
- mt_rand()

log10

log10 -- Base-10 logarithm

Description

float **log10** (float `$arg`)

Returns the base-10 logarithm of *`arg`*.

Parameters

`arg`
: The argument to process

Return Values

The base-10 logaritm of *`arg`*

See Also

- log()

log1p

log1p -- Returns log(1 + number), computed in a way that is accurate even when the value of number is close to zero

Description

float **log1p** (float `$number`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

log1p() returns log(1 + *`number`*) computed in a way that is accurante even when the value of *`number`* is close to zero. log() might only return log(1) in this case due to lack of precision.

Parameters

`number`
The argument to process

Return Values

log(1 + *`number`*)

ChangeLog

Version	Description
5.3.0	This function is now available on all platforms

See Also

- expm1()

- log()
- log10()

log

log -- Natural logarithm

Description

float **log** (float `$arg` [, float `$base`])

If the optional *`base`* parameter is specified, log() returns log $_{base}$ *`arg`*, otherwise log() returns the natural logarithm of *`arg`*.

Parameters

`arg`
: The value to calculate the logarithm for

`base`
: The optional logarithmic base to use (defaults to 'e' and so to the natural logarithm).

Return Values

The logarithm of *`arg`* to *`base`*, if given, or the natural logarithm.

ChangeLog

Version	Description
Since 4.3.0	The optinal parameter *`base`* became available. For older versions you can calculate the logarithm in base *b* of a number *n*, but using the mathematical identity: $\log_b(n) = \log(n)/\log(b)$, where log is the neperian (or natural) logarithm.

See Also

- log10()
- exp()
- pow()

max

max -- Find highest value

Description

mixed **max** (array `$values`)

mixed **max** (mixed `$value1`, mixed `$value2` [, mixed `$value3...`])

If the first and only parameter is an array, max() returns the highest value in that array. If at least two parameters are provided, max() returns the biggest of these values.

Note
PHP will evaluate a non-numeric string as *0* if compared to integer, but still return the string if it's seen as the numerically highest value. If multiple arguments evaluate to *0*, max() will return a numeric *0* if given, else the alphabetical highest string value will be returned.

Parameters

`values`
An array containing the values.

Return Values

max() returns the numerically highest of the parameter values.

Examples

Example #14 - Example uses of max()

```
<?php
echo max(1, 3, 5, 6, 7);  // 7
echo max(array(2, 4, 5)); // 5

echo max(0, 'hello');     // 0
echo max('hello', 0);     // hello
echo max(-1, 'hello');    // hello

// With multiple arrays, max compares from left to right
```

```
// so in our example: 2 == 2, but 4 < 5
$val = max(array(2, 4, 8), array(2, 5, 7)); // array(2, 5, 7)

// If both an array and non-array are given, the array
// is always returned as it's seen as the largest
$val = max('string', array(2, 5, 7), 42);   // array(2, 5, 7)
?>
```

See Also

- min()
- count()

min

min -- Find lowest value

Description

mixed **min** (array `$values`)

mixed **min** (mixed `$value1`, mixed `$value2` [, mixed `$value3...`])

If the first and only parameter is an array, min() returns the lowest value in that array. If at least two parameters are provided, min() returns the smallest of these values.

Note
PHP will evaluate a non-numeric string as *0* if compared to integer, but still return the string if it's seen as the numerically lowest value. If multiple arguments evaluate to *0*, min() will return the lowest alphanumerical string value if any strings are given, else a numeric *0* is returned.

Parameters

`values`
An array containing the values.

Return Values

min() returns the numerically lowest of the parameter values.

Examples

Example #15 - Example uses of min()

```
<?php
echo min(2, 3, 1, 6, 7);  // 1
echo min(array(2, 4, 5)); // 2

echo min(0, 'hello');     // 0
echo min('hello', 0);     // hello
echo min('hello', -1);    // -1

// With multiple arrays, min compares from left to right
// so in our example: 2 == 2, but 4 < 5
$val = min(array(2, 4, 8), array(2, 5, 1)); // array(2, 4, 8)
```

```
// If both an array and non-array are given, the array
// is never returned as it's considered the largest
$val = min('string', array(2, 5, 7), 42);   // string
?>
```

See Also

- max()
- count()

mt_getrandmax

mt_getrandmax -- Show largest possible random value

Description

int **mt_getrandmax** (void)

Returns the maximum value that can be returned by a call to mt_rand().

Return Values

Returns the maximum random value returned by mt_rand()

See Also

- mt_rand()
- mt_srand()
- getrandmax()

mt_rand

mt_rand -- Generate a better random value

Description

int **mt_rand** (void)

int **mt_rand** (int `$min`, int `$max`)

Many random number generators of older libcs have dubious or unknown characteristics and are slow. By default, PHP uses the libc random number generator with the rand() function. The mt_rand() function is a drop-in replacement for this. It uses a random number generator with known characteristics using the » Mersenne Twister, which will produce random numbers four times faster than what the average libc rand() provides.

If called without the optional *`min`*, *`max`* arguments mt_rand() returns a pseudo-random value between 0 and **RAND_MAX**. If you want a random number between 5 and 15 (inclusive), for example, use *mt_rand (5, 15)*.

Note
As of PHP 4.2.0, there is no need to seed the random number generator with srand() or mt_srand() as this is now done automatically.

Parameters

`min`
Optional lowest value to be returned (default: 0)

`max`
Optional highest value to be returned (default: RAND_MAX)

Return Values

A random integer value between *`min`* (or 0) and *`max`* (or RAND_MAX, inclusive)

ChangeLog

Version	Description
Since 3.0.7	In versions before 3.0.7 the meaning of *`max`*

	was *range*. To get the same results in these versions the short example should be *rand (5, 11)* to get a random number between 5 and 15.

Examples

Example #16 - mt_rand() example

```
<?php
echo mt_rand() . "\n";
echo mt_rand() . "\n";

echo mt_rand(5, 15);
?>
```

The above example will output something similar to:

```
1604716014
1478613278
6
```

See Also

- mt_srand()
- mt_getrandmax()
- rand()

mt_srand

mt_srand -- Seed the better random number generator

Description

void **mt_srand** ([int `$seed`])

Seeds the random number generator with *seed* or with a random value if no *seed* is given.

Note
As of PHP 4.2.0, there is no need to seed the random number generator with srand() or mt_srand() as this is now done automatically.

Parameters

seed
An optional seed value

ChangeLog

Version	Description
Since 4.2.0	The *seed* becomes optional and defaults to a random value if omitted.
Since 5.2.1	The Mersenne Twister implementation in PHP now uses a new seeding algorithm by Richard Wagner. Identical seeds no longer produce the same sequence of values they did in previous versions. This behavior is not expected to change again, but it is considered unsafe to rely upon it nonetheless.

Examples

Example #17 - mt_srand() example

```
<?php
// seed with microseconds
function make_seed()
{
 list($usec, $sec) = explode(' ', microtime());
 return (float) $sec + ((float) $usec * 100000);
}
mt_srand(make_seed());
$randval = mt_rand();
?>
```

See Also

- mt_rand()
- mt_getrandmax()
- srand()

octdec

octdec -- Octal to decimal

Description

number **octdec** (string `$octal_string`)

Returns the decimal equivalent of the octal number represented by the *`octal_string`* argument.

Parameters

`octal_string`
The octal string to convert

Return Values

The decimal representation of *`octal_string`*

ChangeLog

Version	Description
Since 4.1.0	The function can now convert numbers that are too large to fit into the platforms integer type, larger values are returned as float in that case.

Examples

Example #18 - octdec() example

```
<?php
echo octdec('77') . "\n";
echo octdec(decoct(45));
?>
```

The above example will output:

```
63
```

```
45
```

See Also

- decoct()
- bindec()
- hexdec()
- base_convert()

pi

pi -- Get value of pi

Description

float **pi** (void)

Returns an approximation of pi. The returned float has a precision based on the precision directive in *php.ini*, which defaults to *14*. Also, you can use the **M_PI** constant which yields identical results to pi().

Return Values

The value of pi as float.

Examples

Example #19 - pi() example

```
<?php
echo pi(); // 3.1415926535898
echo M_PI; // 3.1415926535898
?>
```

pow

pow -- Exponential expression

Description

number **pow** (number `$base`, number `$exp`)

Returns *`base`* raised to the power of *`exp`*.

Warning
In PHP 4.0.6 and earlier pow() always returned a float, and did not issue warnings.

Parameters

`base`
The base to use

`exp`
The exponent

Return Values

`base` raised to the power of *`exp`*. If the result can be represented as integer it will be returned as type integer, else it will be returned as type float. If the power cannot be computed **FALSE** will be returned instead.

ChangeLog

Version	Description
Since 4.0.6	The function will now return integer results if possible, before this it always returned a float result. For older versions, you may receive a bogus result for complex numbers.
Since 4.2.0	PHP stops to emit a warning if the value can't be computed, it will now silently return **FALSE** only.

Examples

Example #20 - Some examples of pow()

```
<?php

var_dump(pow(2, 8)); // int(256)
echo pow(-1, 20); // 1
echo pow(0, 0); // 1

echo pow(-1, 5.5); // PHP >4.0.6  NAN
echo pow(-1, 5.5); // PHP <=4.0.6 1.#IND
?>
```

See Also

- exp()
- sqrt()
- bcpow()
- gmp_pow()

rad2deg

rad2deg -- Converts the radian number to the equivalent number in degrees

Description

float **rad2deg** (float `$number`)

This function converts *`number`* from radian to degrees.

Parameters

`number`
A radian value

Return Values

The equivalent of *`number`* in degrees

Examples

Example #21 - rad2deg() example

```
<?php

echo rad2deg(M_PI_4); // 45

?>
```

See Also

- deg2rad()

rand

rand -- Generate a random integer

Description

int **rand** (void)

int **rand** (int `$min`, int `$max`)

If called without the optional *`min`*, *`max`* arguments rand() returns a pseudo-random integer between 0 and **RAND_MAX**. If you want a random number between 5 and 15 (inclusive), for example, use *rand (5, 15)*.

Note
On some platforms (such as Windows) **RAND_MAX** is only 32768. If you require a range larger than 32768, specifying *`min`* and *`max`* will allow you to create a range larger than RAND_MAX, or consider using mt_rand() instead.

Note
As of PHP 4.2.0, there is no need to seed the random number generator with srand() or mt_srand() as this is now done automatically.

Parameters

`min`
: The lowest value to return (default: 0)

`max`
: The highest value to return (default: RAND_MAX)

Return Values

A pseudo random value between *`min`* (or 0) and *`max`* (or RAND_MAX, inclusive).

ChangeLog

Version	Description

Since 3.0.7	In versions before 3.0.7 the meaning of *max* was *range*. To get the same results in these versions the short example should be *rand (5, 11)* to get a random number between 5 and 15.

Examples

Example #22 - rand() example

```
<?php
echo rand() . "\n";
echo rand() . "\n";

echo rand(5, 15);
?>
```

The above example will output something similar to:

```
7771
22264
11
```

See Also

- srand()
- getrandmax()
- mt_rand()

round

round -- Rounds a float

Description

float **round** (float `$val` [, int `$precision`])

Returns the rounded value of *`val`* to specified *`precision`* (number of digits after the decimal point). *`precision`* can also be negative or zero (default).

Note
PHP doesn't handle strings like *"12,300.2"* correctly by default. See converting from strings.

Note
The *`precision`* parameter was introduced in PHP 4.

Parameters

`val`
: The value to round

`precision`
: The optional number of decimal digits to round to, defaults to 0

Return Values

The rounded value

Examples

Example #23 - round() examples

```
<?php
echo round(3.4);         // 3
echo round(3.5);         // 4
echo round(3.6);         // 4
echo round(3.6, 0);      // 4
echo round(1.95583, 2);  // 1.96
echo round(1241757, -3); // 1242000
echo round(5.045, 2);    // 5.05
echo round(5.055, 2);    // 5.06
?>
```

See Also

- ceil()
- floor()
- number_format()

sin

sin -- Sine

Description

float **sin** (float `$arg`)

sin() returns the sine of the *`arg`* parameter. The *`arg`* parameter is in radians.

Parameters

`arg`
A value in radians

Return Values

The sine of *`arg`*

Examples

Example #24 - sin() example

```
<?php

// Precision depends on your precision directive
echo sin(deg2rad(60));  //  0.866025403 ...
echo sin(60);           // -0.304810621 ...

?>
```

See Also

- asin()
- sinh()
- cos()
- tan()
- deg2rad()

sinh

sinh -- Hyperbolic sine

Description

float **sinh** (float `$arg`)

Returns the hyperbolic sine of *`arg`*, defined as *(exp(arg) - exp(-arg))/2*.

Parameters

`arg`
: The argument to process

Return Values

The hyperbolic sine of *`arg`*

See Also

- sin()
- asinh()
- cosh()
- tanh()

sqrt

sqrt -- Square root

Description

float **sqrt** (float `$arg`)

Returns the square root of *`arg`*.

Parameters

`arg`
: The argument to process

Return Values

The square root of *`arg`* or the special value *NAN* for negative numbers.

Examples

Example #25 - sqrt() example

```
<?php
// Precision depends on your precision directive
echo sqrt(9); // 3
echo sqrt(10); // 3.16227766 ...
?>
```

See Also

- pow()

srand

srand -- Seed the random number generator

Description

void **srand** ([int `$seed`])

Seeds the random number generator with *seed* or with a random value if no *seed* is given.

Note
As of PHP 4.2.0, there is no need to seed the random number generator with srand() or mt_srand() as this is now done automatically.

Parameters

seed
Optional seed value

ChangeLog

Version	Description
Since 4.2.0	The *seed* becomes optional and defaults to a random value if omitted.

Examples

Example #26 - srand() example

```
<?php
// seed with microseconds
function make_seed()
{
 list($usec, $sec) = explode(' ', microtime());
 return (float) $sec + ((float) $usec * 100000);
}
srand(make_seed());
$randval = rand();
```

```
?>
```

See Also

- rand()
- getrandmax()
- mt_srand()

tan

tan -- Tangent

Description

float **tan** (float `$arg`)

tan() returns the tangent of the *arg* parameter. The *arg* parameter is in radians.

Parameters

arg
The argument to process in radians

Return Values

The tangent of *arg*

Examples

Example #27 - tan() example

```
<?php

echo tan(M_PI_4); // 1

?>
```

See Also

- atan()
- atan2()
- sin()
- cos()
- tanh()
- deg2rad()

tanh

tanh -- Hyperbolic tangent

Description

float **tanh** (float `$arg`)

Returns the hyperbolic tangent of `arg`, defined as *sinh(arg)/cosh(arg)*.

Parameters

`arg`
: The argument to process

Return Values

The hyperbolic tangent of `arg`

See Also

- tan()
- atanh()
- sinh()
- cosh()