php

- Downloads
- Documentation
- Get Involved
- Help

Search

PHPKonf: Istanbul PHP Conference 2017

Keyboard Shortcuts

?	This help
j	Next menu item
k	Previous menu item
g p	Previous man page
g n	Next man page
G	Scroll to bottom
g g	Scroll to top

g h
Goto homepage
g s
Goto search
(current page)
/
Focus search box

sin »
« rand

- PHP Manual
- Function Reference
- Mathematical Extensions
- Math
- Math Functions

Change language: English ▼

Edit Report a Bug

round

(PHP 4, PHP 5, PHP 7)

round — Rounds a float

Description ¶

float **round** (float `$val` [, int `$precision` = 0 [, int `$mode` = PHP_ROUND_HALF_UP]])

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.

Parameters ¶

`val`

The value to round

`precision`

The optional number of decimal digits to round to.

`mode`

Use one of the following constants to specify the mode in which rounding occurs.

Constant	Description
PHP_ROUND_HALF_UP	Round `val` up to `precision` decimal places away from zero, when it is half way there. Making 1.5 into 2 and -1.5 into -2.

`PHP_ROUND_HALF_DOWN`	Round `val` down to `precision` decimal places towards zero, when it is half way there. Making 1.5 into 1 and -1.5 into -1.
`PHP_ROUND_HALF_EVEN`	Round `val` to `precision` decimal places towards the next even value.
`PHP_ROUND_HALF_ODD`	Round `val` to `precision` decimal places towards the next odd value.

Return Values ¶

The rounded value

Examples ¶

Example #1 round() examples

```php
<?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
?>
```

Example #2 `mode` examples

```php
<?php
echo round(9.5, 0, PHP_ROUND_HALF_UP);   // 10
echo round(9.5, 0, PHP_ROUND_HALF_DOWN); // 9
echo round(9.5, 0, PHP_ROUND_HALF_EVEN); // 10
echo round(9.5, 0, PHP_ROUND_HALF_ODD);  // 9

echo round(8.5, 0, PHP_ROUND_HALF_UP);   // 9
echo round(8.5, 0, PHP_ROUND_HALF_DOWN); // 8
echo round(8.5, 0, PHP_ROUND_HALF_EVEN); // 8
echo round(8.5, 0, PHP_ROUND_HALF_ODD);  // 9
?>
```

Example #3 `mode` with precision examples

```php
<?php
/* Using PHP_ROUND_HALF_UP with 1 decimal digit precision */
echo round( 1.55, 1, PHP_ROUND_HALF_UP);   //  1.6
echo round( 1.54, 1, PHP_ROUND_HALF_UP);   //  1.5
echo round(-1.55, 1, PHP_ROUND_HALF_UP);   // -1.6
echo round(-1.54, 1, PHP_ROUND_HALF_UP);   // -1.5

/* Using PHP_ROUND_HALF_DOWN with 1 decimal digit precision */
echo round( 1.55, 1, PHP_ROUND_HALF_DOWN); //  1.5
echo round( 1.54, 1, PHP_ROUND_HALF_DOWN); //  1.5
echo round(-1.55, 1, PHP_ROUND_HALF_DOWN); // -1.5
echo round(-1.54, 1, PHP_ROUND_HALF_DOWN); // -1.5

/* Using PHP_ROUND_HALF_EVEN with 1 decimal digit precision */
```

```
echo round( 1.55, 1, PHP_ROUND_HALF_EVEN); //  1.6
echo round( 1.54, 1, PHP_ROUND_HALF_EVEN); //  1.5
echo round(-1.55, 1, PHP_ROUND_HALF_EVEN); // -1.6
echo round(-1.54, 1, PHP_ROUND_HALF_EVEN); // -1.5

/* Using PHP_ROUND_HALF_ODD with 1 decimal digit precision */
echo round( 1.55, 1, PHP_ROUND_HALF_ODD);  //  1.5
echo round( 1.54, 1, PHP_ROUND_HALF_ODD);  //  1.5
echo round(-1.55, 1, PHP_ROUND_HALF_ODD);  // -1.5
echo round(-1.54, 1, PHP_ROUND_HALF_ODD);  // -1.5
?>
```

Changelog ¶

Version	Description
5.3.0	The `mode` parameter was introduced.
5.2.7	The inner workings of **round()** was changed to conform to the C99 standard.

See Also ¶

- ceil() - Round fractions up
- floor() - Round fractions down
- number_format() - Format a number with grouped thousands

add a note

User Contributed Notes 29 notes

up
down
143
takingsides at gmail dot com ¶
2 years ago

```
In my opinion this function lacks two flags:

- PHP_ROUND_UP - Always round up.
- PHP_ROUND_DOWN - Always round down.

In accounting, it's often necessary to always round up, or down to a precision of thousandths.

<?php
function round_up($number, $precision = 2)
{
    $fig = (int) str_pad('1', $precision, '0');
    return (ceil($number * $fig) / $fig);
}

function round_down($number, $precision = 2)
{
    $fig = (int) str_pad('1', $precision, '0');
    return (floor($number * $fig) / $fig);
}
?>
```

up
down
15

djcox99 at googlemail dot com ¶

2 years ago

```
I discovered that under some conditions you can get rounding errors with round when converting the
number to a string afterwards.

To fix this I swapped round() for number_format().

Unfortunately i cant give an example (because the number cant be represented as a string !)

essentially I had round(0.688888889,2);

which would stay as 0.68888889 when printed as a string.

But using number_format it correctly became 0.69.
```

up
down
9

slimusgm at gmail dot com ¶

2 years ago

```
If you have negative zero and you need return positive number simple add +0:

$number = -2.38419e-07;
var_dump(round($number,1));//float(-0)
var_dump(round($number,1) + 0);//float(0)
```

up
down
5

jongbumi at gmail dot com ¶

7 months ago

```
PHP 5.3, 5.4, 5.5
<?php
$fInfinty = pow(1000, 1000); // float(INF)
$fResult = round(123.456, $fInfinty); // double(123)
?>

PHP 5.6
<?php
$fInfinty = pow(1000, 1000); // float(INF)
$fResult = round(123.456, $fInfinty); // float(0)
?>

PHP 7
<?php
$fInfinty = pow(1000, 1000); // float(INF)
$fResult = round(123.456, $fInfinty); // null
?>
```

up
down
14

twan at ecreation dot nl ¶

16 years ago

```
If you'd only want to round for displaying variables (not for calculating on the rounded result) then
you should use printf with the float:

<?php printf ("%6.2f",3.39532); ?>

This returns: 3.40 .
```

up

down

10

Anonymous ¶

6 years ago

```
Here is function that rounds to a specified increment, but always up. I had to use it for price
adjustment that always went up to $5 increments.

<?php
function roundUpTo($number, $increments) {
    $increments = 1 / $increments;
    return (ceil($number * $increments) / $increments);
}
?>
```

up

down

5

martinr at maarja dot net ¶

8 years ago

```
Please note that the format of this functions output also depends on your locale settings. For
example, if you have set your locale to some country that uses commas to separate decimal places, the
output of this function also uses commas instead of dots.

This might be a problem when you are feeding the rounded float number into a database, which requires
you to separate decimal places with dots.

See it in action:
<?php
    echo round('3.5558', 2);
    setlocale(constant('LC_ALL'), 'et_EE.UTF-8');
    echo '<br />'. round('3.5558', 2);
?>

The output will be:
3.56
3,56
```

up

down

6

php at silisoftware dot com ¶

14 years ago

```
Here's a function to round to an arbitary number of significant digits. Don't confuse it with
rounding to a negative precision - that counts back from the decimal point, this function counts
forward from the Most Significant Digit.

ex:
```

```
<?php
round(1241757, -3); // 1242000
RoundSigDigs(1241757, 3); // 1240000
?>

Works on negative numbers too. $sigdigs should be >= 0

<?php
function RoundSigDigs($number, $sigdigs) {
    $multiplier = 1;
    while ($number < 0.1) {
        $number *= 10;
        $multiplier /= 10;
    }
    while ($number >= 1) {
        $number /= 10;
        $multiplier *= 10;
    }
    return round($number, $sigdigs) * $multiplier;
}
?>
```

up
down
2
christian at deligant dot net ¶
5 years ago

this function (as all mathematical operators) takes care of the setlocale setting, resulting in some weirdness when using the result where the english math notation is expected, as the printout of the result in a width: style attribute!

```
<?php
$a=3/4;
echo round($a, 2); // 0.75

setlocale(LC_ALL, 'it_IT@euro', 'it_IT', 'it');
$b=3/4;
echo round($b,2); // 0,75
?>
```

up
down
1
dastra ¶
4 years ago

round() will sometimes return E notation when rounding a float when the amount is small enough - see https://bugs.php.net/bug.php?id=44223 . Apparently it's a feature.

To work around this "feature" when converting to a string, surround your round statement with an sprintf:

```
sprintf("%.10f", round( $amountToBeRounded, 10));
```

up
down
0

greghenle at gmail dot com ¶

1 month ago

```
/**
* Round to first significant digit
* +N to +infinity
* -N to -infinity
*
*/
function round1stSignificant ( $N ) {
  if ( $N === 0 ) {
    return 0;
  }

  $x = floor ( log10 ( abs( $N ) ) );

  return ( $N > 0 )
    ? ceil( $N * pow ( 10, $x * -1 ) ) * pow( 10, $x )
    : floor( $N * pow ( 10, $x * -1 ) ) * pow( 10, $x );
}

echo round1stSignificant( 39144818 ) . PHP_EOL;
echo round1stSignificant( 124818 ) . PHP_EOL;
echo round1stSignificant( 0.07468 ) . PHP_EOL;
echo round1stSignificant( 0 ) . PHP_EOL;
echo round1stSignificant( -0.07468 ) . PHP_EOL;

/**
* Output
*
* 40000000
* 200000
* 0.08
* 0
* -0.08
*
*/
```

up
down
0

Anonymous ¶

1 month ago

```
Note that PHP 5.3 didn't just introduce $mode, it rewrote the rounding implementation completely to
eliminate many kinds of rounding errors common to rounding floating point values.

That's why round() gives you the correct result even when floor/ceil don't.
For example,  floor(0.285 * 100 + 0.5) VS round(0.285*100 + 0.5). First one gives 28, second one
gives 29.

More details here: https://wiki.php.net/rfc/rounding
```

up
down
0

serg at kalachev dot ru ¶

2 years ago

```
Excel-like ROUNDUP function:

public static function round_up($value, $places)
{
    $mult = pow(10, abs($places));
     return $places < 0 ?
    ceil($value / $mult) * $mult :
        ceil($value * $mult) / $mult;
}

echo round_up(12345.23, 1); // 12345.3
echo round_up(12345.23, 0); // 12346
echo round_up(12345.23, -1); // 12350
echo round_up(12345.23, -2); // 12400
echo round_up(12345.23, -3); // 13000
echo round_up(12345.23, -4); // 20000
```

up
down
0
esion99 at gmail dot com ¶
2 years ago

```
Unexpected result or misunderstanding (php v5.5.9)

<?php

echo round(1.55, 1, PHP_ROUND_HALF_DOWN); // 1.5
echo round(1.551, 1, PHP_ROUND_HALF_DOWN); //1.6

?>
```

up
down
0
spectrumcat at gmail dot com ¶
2 years ago

```
In case someone will need a "graceful" rounding (that changes it's precision to get a non 0 value)
here's a simple function:

function gracefulRound($val, $min = 2, $max = 4) {
    $result = round($val, $min);
    if ($result == 0 && $min < $max) {
        return gracefulRound($val, ++$min, $max);
    } else {
        return $result;
    }
}

Usage:
$_ = array(0.5, 0.023, 0.008, 0.0007, 0.000079, 0.0000048);
foreach ($_ as $val) {
    echo "{$val}: ".gracefulRound($val)."\n";
}

Output:
0.5: 0.5
```

```
0.023: 0.02
0.008: 0.01
0.0007: 0.001
0.000079: 0.0001
0.0000048: 0
```

up
down
0
craft at ckdevelop dot org ¶
3 years ago

```
function mround($val, $f=2, $d=6){
    return sprintf("%".$d.".".$f."f", $val);
}

echo mround(34.89999);  //34.90
```

up
down
0
feha at vision dot to ¶
6 years ago

```
Here is a short neat function to round minutes (hour) ...

<?php

function minutes_round ($hour = '14:03:32', $minutes = '5', $format = "H:i")
{
    // by Femi Hasani [www.vision.to]
    $seconds = strtotime($hour);
    $rounded = round($seconds / ($minutes * 60)) * ($minutes * 60);
    return date($format, $rounded);
}

?>

You decide to round to nearest minute ...
example will produce : 14:05
```

up
down
0
michaeldnelson dot mdn at gmail dot com ¶
7 years ago

```
This function will let you round to an arbitrary non-zero number.  Zero of course causes a division
by zero.

<?php
function roundTo($number, $to){
    return round($number/$to, 0)* $to;
}

echo roundTo(87.23, 20); //80
echo roundTo(-87.23, 20); //-80
echo roundTo(87.23, .25); //87.25
echo roundTo(.23, .25); //.25
?>
```

up
down
-1
Anonymous ¶
7 years ago

```
This functions return ceil($nb) if the double or float value is bigger than "$nb.5" else it's return
floor($nb)

<?php
    function arounds_int($nb) {

        if(!is_numeric($nb)) {
            return false;
        }

        $sup = round($nb);
        $inf = floor($nb);
        $try = (double) $inf . '.5' ;

        if($nb > $try) {
            return $sup;
        }

        return $inf;
    }
?>
```

up
down
-2
terry at scribendi dot com ¶
12 years ago

To round any number to a given number of significant digits, use log10 to find out its magnitude:

```
<?php round($n, ceil(0 - log10($n)) + $sigdigits); ?>
```

Or when you have to display a per-unit price which may work out to be less than a few cents/pence/yen you can use:

```
<?php
// $exp = currency decimal places - 0 for Yen/Won, 2 for most others
$dp = ceil(0 - log10($n)) + $sigdigits;
$display = number_format($amount, ($exp>$dp)?$exp:$dp);
?>
```

This always displays at least the number of decimal places required by the currency, but more if displaying the unit price with precision requires it - eg: 'English proofreading from $0.0068 per word', 'English beer from $6.80 per pint'.

up
down
-4
omnibus at omnibus dot edu dot pl ¶
6 years ago

Beware strange behaviour if number is negative and precision is bigger than the actual number of digits after comma.

```
round(-0.07, 4);

returns

-0.07000000000000001

So if you validate it against a regular expression requiring the maximum amount of digits after
comma, you'll get into trouble.
```

up
down
-3
lossantis at ig dot com dot br ¶
6 years ago

```
Since the mode parameter for options like PHP_ROUND_HALF_UP is available as of PHP 5.3, here is an
alternative for ceiling:

<?php echo 252 / 40; // 6.3 ?>

If I round this:

<?php echo round(252 / 40); // 6 ?>

You can also use a ceil (which might be useful for pagination):

<?php echo ceil(252/40); // 7 ?>

[Edited by: googleguy@php.net for clarity]
```

up
down
-4
tom at crysis-online dot com ¶
9 years ago

```
I just found out then that even if you round a double (3.7) to an integer (4), it's data type remains
as 'double'. So it's always good to use the settype() function when using the round() function to
prevent any problems with your scripts.
```

up
down
-3
ianring (at) golden.net ¶
13 years ago

```
The round() function may indeed work properly with half-values (eg. 1.5), but this little method will
give you peace of mind. Add some "fuzz" to your function with a miniscule delta value.

<?php
$delta = 0.00001;
$x = round($x+$delta);
?>

This is fine, unless $x has a value of 1.49999 ... if you worried about that, use this method
instead:

<?php
if(($x-floor($x))==0.5){
    $x+=$delta;
}
$x = round($x);
?>

you can change your "optimistic" delta into a "pessimistic" delta by subtracting instead of adding.

Cheers,
Ian Ring
```

up
down
-4
tichoux at charlevoix dot net ¶
14 years ago

```
for a poll, if you want to have 100% and not 99 or 99.99 % you can do that :

<?php
round( number_format( (($individual_result*100)/$total_result), 2), 1);
?>
```

up
down
-2
php at persignum dot com ¶
1 year ago

```
Because this function is missing round up and round down constants and the top note doesn't really
show you how to round up or down to the nearest number, here is an easy way to always round up or
always round down to the nearest number.

int is the number you want to round

n is the nearest number you want rounded to.

Round up to the nearest number

function round_up($int, $n) {
    return ceil($int / $n) * $n;
}

And to round down to the nearest number

function round_down(int, $n) {
    return floor($int / $n) * $n;
}
```

up
down
-5
Bevan ¶
7 years ago

```
Formats a number to the specified number of significant figures.

<?php
/**
* Formats numbers to the specified number of significant figures.
*
```

```
 * @author Bevan Rudge, Drupal.geek.nz
 *
 * @param number $number
 *   The number to format.
 * @param integer $sf
 *   The number of significant figures to round and format the number to.
 * @return string
 *   The rounded and formatted number.
 */
function format_number_significant_figures($number, $sf) {
  // How many decimal places do we round and format to?
  // @note May be negative.
  $dp = floor($sf - log10(abs($number)));
  // Round as a regular number.
  $number = round($number, $dp);
  // Leave the formatting to format_number(), but always format 0 to 0dp.
  return number_format($number, 0 == $number ? 0 : $dp);
}
?>
```

up
down
-8
hugues at zonereseau dot com ¶
6 years ago

```
I had problem with round() function I didn't gave me the same result in windows or on a linux server
:

<?php
round(4.725, 2); // gave me 4.72 on linux
round(4.725, 2); // gave me 4.73 on windows
?>

The expected result was 4.73

Here my function to resolve my problem

<?php
function mround($number, $precision=0) {

    $precision = ($precision == 0 ? 1 : $precision);
    $pow = pow(10, $precision);

    $ceil = ceil($number * $pow)/$pow;
    $floor = floor($number * $pow)/$pow;

    $pow = pow(10, $precision+1);

    $diffCeil     = $pow*($ceil-$number);
    $diffFloor     = $pow*($number-$floor)+($number < 0 ? -1 : 1);

    if($diffCeil >= $diffFloor) return $floor;
    else return $ceil;
}
```

```
echo mround(4.725, 2); // Yes 4.73
?>
```

up
down
-7
maxteiber at gmail dot com ¶
10 years ago

```
the result of this function always depends on the underlying C function. There have been a lot of
compiler bugs and floating-point precission problems involving this function. Right now the following
code:

<?php
echo round(141.075, 2);
?>

returns:

141.07

on my machine.
So never really trust this function when you do critical calculations like accounting stuff!
Instead: use only integers or use string comparisons.
```

add a note