money_format

(PHP 4 >= 4.3.0, PHP 5, PHP 7)

money_format — Da formato a un número como un string de moneda

Descripción ¶

money_format(string `$format`, float `$number`): string

money_format() devuelve una versión de `number` con formato. Esta función se ajusta a la función **strfmon()** de la librería de C, con la diferencia de que esta aplicación convierte sólo un número a la vez.

Parámetros ¶

`format`

La especificación del formato se compone de la siguiente secuencia:

- un caracter `%`
- indicadores opcionales
- ancho de campo opcional
- precisión a la izquierda opcional
- precisión a la derecha opcional
- un caracter de conversión obligatorio

Indicadores

Uno o más de los siguientes indicadores opcionales pueden ser utilizados:

=f

El caracter = seguido de un caracter f (byte único) que se utilizará como el caracter de relleno numérico. El carácter de relleno por defecto es el espacio.

^

Deshabilita el uso de caracteres de agrupamiento (según la definición de la configuración regional actual).

`+` o `(`

Especifica el estilo de formato para los números positivos y negativos. Si se utiliza `+`, el equivalente de la configuración regional para `+` y `-` será usado. Si se utiliza `(`, las cantidades negativas serán encerradas entre paréntesis. Si no se da ninguna especificación, el valor por defecto es `+`.

`!`

Suprime el símbolo de moneda del string producido.

`-`

Si está presente, hará todos los campos justificados a la izquierda (con relleno a la derecha), contrario al valor por defecto que es para que los campos se justifiquen a la derecha (con relleno a la izquierda).

Ancho de campo

w

Un string de dígito decimal que especifica un ancho mínimo de campo. El campo será justificado a la derecha a menos que el indicador `-` sea utilizado. El valor por defecto es 0 (cero).

Precisión a la izquierda

#n

El número máximo de dígitos (n) que se espera a la izquierda del caracter decimal (por ejemplo, el punto decimal). Es utilizado generalmente para mantener la salida con formato alineada en las mismas columnas, usando el caracter de relleno si el número de dígitos sea menor que n. Si el número de dígitos real es mayor que n, entonces esta especificación es ignorada.

Si la agrupación no ha sido suprimida mediante el indicador `^`, los separadores de agrupamiento serán insertados antes de que los caracteres de relleno (si los hay) sean agregados. Los separadores de agrupamiento no se aplicarán a los caracteres de relleno, aunque el carácter de relleno sea un dígito.

Para asegurar la alineación, cualquier caracter que aparezca antes o después del número en la salida con formato, tales como símbolos de moneda o de signo, son rellenados cuando sea necesario con caracteres de espacio para hacer sus formatos positivos y negativos de una longitud igual.

Precisión a la derecha

.p

Un punto seguido por el número de dígitos (p) después del carácter decimal. Si el valor de p es 0 (cero), el carácter decimal y los dígitos a su derecha serán omitidos. Si no se incluye la precisión al la derecha, el valor por defecto será dictado por la configuración regional actual en uso. La cantidad que se está formateando se redondea al número especificado de dígitos antes de formatearla.

Conversion characters

`i`

El número se formatea de acuerdo al formato internacional de moneda de la configuración regional (por ejemplo, para la configuración regional de EE.UU.: USD $ 1,234.56).

`n`

El número se formatea de acuerdo al formato nacional de moneda de la configuración regional (por ejemplo, para la configuración regional de_DE: EU1.234,56).

`%`

Devuelve el caracter `%`.

`number`

El número a ser formateado.

Valores devueltos ¶

Devuelve el string con formato. Los caracteres antes y después del string de formato se devolverán sin cambios. Un `number` no numérico causa que se devuelva **`null`** y se emite un **`E_WARNING`**.

Notas ¶

Nota:

La función **money_format()** sólo está definida si el sistema tiene capacidad strfmon. Por ejemplo, Windows no lo hace, así que **money_format()** no está definido en Windows.

Nota:

La categoría **`LC_MONETARY`** de la configuración regional, afecta el comportamiento de esta función. Utilizar setlocale() para establecer la configuración regional predeterminada apropiada antes de usar esta función.

Ejemplos ¶

Ejemplo #1 Ejemplos de money_format()

Se utilizan diferentes configuraciones regionales y especificaciones de formato para ilustrar el uso de esta función.

```
<?php

$number = 1234.56;

// muestra el formato internacional para la configuración regional en_US
setlocale(LC_MONETARY, 'en_US');
echo money_format('%i', $number) . "\n";
// USD 1,234.56

// formato nacional italiano con 2 decimales`
setlocale(LC_MONETARY, 'it_IT');
echo money_format('%.2n', $number) . "\n";
// Eu 1.234,56

// utilizando un número negativo
$number = -1234.5672;
```

```
// formato nacional US, utilizando () para los números negativos
// y 10 dígitos de precisión a la izquierda
setlocale(LC_MONETARY, 'en_US');
echo money_format('%(#10n', $number) . "\n";
// ($        1,234.57)

// formato similar que el anterior, agregando el uso de 2 dígitos de precisión
// a la derecha y '*' como caracter de relleno
echo money_format('%=*(#10.2n', $number) . "\n";
// ($********1,234.57)

// justificado a la izquierda, con 14 posiciones de ancho, 8 dígitos de
// precisión a la izquierda, 2 de precisión a la derecha, sin caracter de agrupamiento
// y utilizando el formato internacional para la configuración regional de_DE.
setlocale(LC_MONETARY, 'de_DE');
echo money_format('%=*^-14#8.2i', 1234.56) . "\n";
// Eu 1234,56****

// añadir un poco de propaganda antes y después los especificación de conversión
setlocale(LC_MONETARY, 'en_GB');
$fmt = 'The final value is %i (after a 10%% discount)';
echo money_format($fmt, 1234.56) . "\n";
// The final value is  GBP 1,234.56 (after a 10% discount)

?>
```

Ver también ¶

- setlocale() - Establecer la información del localismo
- sscanf() - Interpreta un string de entrada de acuerdo con un formato
- sprintf() - Devuelve un string formateado
- printf() - Imprimir una cadena con formato
- number_format() - Formatear un número con los millares agrupados

+ add a note

User Contributed Notes 15 notes

up
down
67
tim ¶
7 years ago

```
For most of us in the US, we don't want to see a "USD" for our currency symbol, so '%i' doesn't
cut it.  Here's what I used that worked to get what most  people expect to see for a number
format.

$number = 123.4
setlocale(LC_MONETARY, 'en_US.UTF-8');
money_format('%.2n', $number);

output:
$123.40

That gives me a dollar sign at the beginning, and 2 digits at the end.
```

up
down

18
[todoventas at xarxa-cat dot net ¶](#)
8 years ago

```
In Rafael M. Salvioni function localeconv(); returns an invalid array in my Windows XP SP3 running
PHP 5.4.13 so to prevent the Warning Message: implode(): Invalid arguments passed i just add the
$locale manually. For other languages just fill the array with the correct settings.

<?

       $locale = array(
        'decimal_point'         => '.',
        'thousands_sep'         => '',
        'int_curr_symbol'     => 'EUR',
        'currency_symbol'     => '€',
        'mon_decimal_point'     => ',',
        'mon_thousands_sep'     => '.',
        'positive_sign'         => '',
        'negative_sign'     => '-',
        'int_frac_digits'     => 2,
        'frac_digits'         => 2,
        'p_cs_precedes'         => 0,
        'p_sep_by_space'     => 1,
        'p_sign_posn'         => 1,
        'n_sign_posn'         => 1,
        'grouping'             => array(),
        'mon_grouping'         => array(0 => 3, 1 => 3)

    );
?>
```

[up](#)
[down](#)
34
[Rafael M. Salvioni ¶](#)
13 years ago

```
This is a some function posted before, however various bugs were corrected.

Thank you to Stuart Roe by reporting the bug on printing signals.

<?php
/*
That it is an implementation of the function money_format for the
platforms that do not it bear.

The function accepts to same string of format accepts for the
original function of the PHP.

(Sorry. my writing in English is very bad)

The function is tested using PHP 5.1.4 in Windows XP
and Apache WebServer.
*/
function money_format($format, $number)
{
    $regex  = '/%((?:[\^!\-]|\+|\(|\=.)*)([0-9]+)?'.
              '(?:#([0-9]+))?(?:\.([0-9]+))?([in%])/';
    if (setlocale(LC_MONETARY, 0) == 'C') {
        setlocale(LC_MONETARY, '');
    }
    $locale = localeconv();
    preg_match_all($regex, $format, $matches, PREG_SET_ORDER);
    foreach ($matches as $fmatch) {
        $value = floatval($number);
        $flags = array(
            'fillchar'  => preg_match('/\=(.)/', $fmatch[1], $match) ?
                           $match[1] : ' ',
            'nogroup'   => preg_match('/\^/', $fmatch[1]) > 0,
            'usesignal' => preg_match('/\+|\(/', $fmatch[1], $match) ?
                           $match[0] : '+',
            'nosimbol'  => preg_match('/\!/', $fmatch[1]) > 0,
            'isleft'    => preg_match('/\-/', $fmatch[1]) > 0
        );
        $width      = trim($fmatch[2]) ? (int)$fmatch[2] : 0;
        $left       = trim($fmatch[3]) ? (int)$fmatch[3] : 0;
        $right      = trim($fmatch[4]) ? (int)$fmatch[4] : $locale['int_frac_digits'];
        $conversion = $fmatch[5];

        $positive = true;
        if ($value < 0) {
            $positive = false;
            $value  *= -1;
        }
        $letter = $positive ? 'p' : 'n';

        $prefix = $suffix = $cprefix = $csuffix = $signal = '';

        $signal = $positive ? $locale['positive_sign'] : $locale['negative_sign'];
        switch (true) {
            case $locale["{$letter}_sign_posn"] == 1 && $flags['usesignal'] == '+':
                $prefix = $signal;
                break;
            case $locale["{$letter}_sign_posn"] == 2 && $flags['usesignal'] == '+':
                $suffix = $signal;
                break;
            case $locale["{$letter}_sign_posn"] == 3 && $flags['usesignal'] == '+':
                $cprefix = $signal;
                break;
            case $locale["{$letter}_sign_posn"] == 4 && $flags['usesignal'] == '+':
                $csuffix = $signal;
                break;
            case $flags['usesignal'] == '(':
            case $locale["{$letter}_sign_posn"] == 0:
                $prefix = '(';
                $suffix = ')';
                break;
        }
        if (!$flags['nosimbol']) {
            $currency = $cprefix .
                        ($conversion == 'i' ? $locale['int_curr_symbol'] :
$locale['currency_symbol']) .
                        $csuffix;
        } else {
            $currency = '';
        }
        $space  = $locale["{$letter}_sep_by_space"] ? ' ' : '';

        $value = number_format($value, $right, $locale['mon_decimal_point'],
                 $flags['nogroup'] ? '' : $locale['mon_thousands_sep']);
        $value = @explode($locale['mon_decimal_point'], $value);

        $n = strlen($prefix) + strlen($currency) + strlen($value[0]);
        if ($left > 0 && $left > $n) {
            $value[0] = str_repeat($flags['fillchar'], $left - $n) . $value[0];
        }
        $value = implode($locale['mon_decimal_point'], $value);
        if ($locale["{$letter}_cs_precedes"]) {
            $value = $prefix . $currency . $space . $value . $suffix;
        } else {
            $value = $prefix . $value . $space . $currency . $suffix;
        }
        if ($width > 0) {
            $value = str_pad($value, $width, $flags['fillchar'], $flags['isleft'] ?
                     STR_PAD_RIGHT : STR_PAD_LEFT);
        }

        $format = str_replace($fmatch[0], $value, $format);
    }
    return $format;
}

?>
```

up
down
17
jeremy ¶
14 years ago

```
If money_format doesn't seem to be working properly, make sure you are defining a valid locale.
For example, on Debian, 'en_US' is not a valid locale - you need 'en_US.UTF-8' or 'en_US.ISO-8559-
1'.

This was frustrating me for a while.  Debian has a list of valid locales at
/usr/share/i18n/SUPPORTED; find yours there if it's not working properly.
```

up
down
9
jsb17NO at SPAMcornell dot edu ¶
9 years ago

```
To drop zero value decimals, use the following:
<?php
    /*
        Same as php number_format(), but if ends in .0, .00, .000, etc... , drops the decimals
altogether
        Returns string type, rounded number - same as php number_format()):
        Examples:
            number_format_drop_zero_decimals(54.378, 2) ==> '54.38'
            number_format_drop_zero_decimals(54.00, 2) ==> '54'
     */
    function number_format_drop_zero_decimals($n, $n_decimals)
    {
        return ((floor($n) == round($n, $n_decimals)) ? number_format($n) : number_format($n,
$n_decimals));
    }
?>
Results:
number_format_drop_zero_decimals(54.377, 2) ==> 54.38
number_format_drop_zero_decimals('54.377', 2) ==> 54.38
number_format_drop_zero_decimals(54.377, 3) ==> 54.377
number_format_drop_zero_decimals(54.007, 2) ==> 54.01
number_format_drop_zero_decimals(54.000, 2) ==> 54
number_format_drop_zero_decimals(54.00, 2) ==> 54
number_format_drop_zero_decimals(54.0, 2) ==> 54
number_format_drop_zero_decimals(54.1, 2) ==> 54.10
number_format_drop_zero_decimals(54., 2) ==> 54
number_format_drop_zero_decimals(54, 2) ==> 54
number_format_drop_zero_decimals(54, 3) ==> 54
number_format_drop_zero_decimals(54 + .13, 2) ==> 54.13
number_format_drop_zero_decimals(54 + .00, 2) ==> 54
number_format_drop_zero_decimals(54.0007, 4) ==> 54.0007
number_format_drop_zero_decimals(54.0007, 3) ==> 54.001
number_format_drop_zero_decimals(54.00007, 3) ==> 54  //  take notice
```

up
down
4
Felix Duterloo ¶
6 years ago

Improvement to Rafael M. Salvioni's solution for money_format on Windows: when no currency symbol is selected, in the formatting, the minus sign was also lost when the locale puts it in position 3 or 4. Changed $currency = ''; to: $currency = $cprefix .$csuffix;

```
function money_format($format, $number) {
        $regex = '/%((?:[\^!\-]|\+|\(|\=.)*)([0-9]+)?' .
                '(?:#([0-9]+))?(?:\.([0-9]+))?([in%])/';
        if (setlocale(LC_MONETARY, 0) == 'C') {
            setlocale(LC_MONETARY, '');
        }
        $locale = localeconv();
        preg_match_all($regex, $format, $matches, PREG_SET_ORDER);
        foreach ($matches as $fmatch) {
            $value = floatval($number);
            $flags = array(
                'fillchar' => preg_match('/\=(.)/', $fmatch[1], $match) ?
                        $match[1] : ' ',
                'nogroup' => preg_match('/\^/', $fmatch[1]) > 0,
                'usesignal' => preg_match('/\+|\(/', $fmatch[1], $match) ?
                        $match[0] : '+',
                'nosimbol' => preg_match('/\!/', $fmatch[1]) > 0,
                'isleft' => preg_match('/\-/', $fmatch[1]) > 0
            );
            $width = trim($fmatch[2]) ? (int) $fmatch[2] : 0;
            $left = trim($fmatch[3]) ? (int) $fmatch[3] : 0;
            $right = trim($fmatch[4]) ? (int) $fmatch[4] : $locale['int_frac_digits'];
            $conversion = $fmatch[5];

            $positive = true;
            if ($value < 0) {
                $positive = false;
                $value *= -1;
            }
            $letter = $positive ? 'p' : 'n';

            $prefix = $suffix = $cprefix = $csuffix = $signal = '';

            $signal = $positive ? $locale['positive_sign'] : $locale['negative_sign'];
            switch (true) {
                case $locale["{$letter}_sign_posn"] == 1 && $flags['usesignal'] == '+':
                    $prefix = $signal;
                    break;
                case $locale["{$letter}_sign_posn"] == 2 && $flags['usesignal'] == '+':
                    $suffix = $signal;
                    break;
                case $locale["{$letter}_sign_posn"] == 3 && $flags['usesignal'] == '+':
                    $cprefix = $signal;
                    break;
                case $locale["{$letter}_sign_posn"] == 4 && $flags['usesignal'] == '+':
                    $csuffix = $signal;
                    break;
                case $flags['usesignal'] == '(':
                case $locale["{$letter}_sign_posn"] == 0:
                    $prefix = '(';
                    $suffix = ')';
                    break;
            }
            if (!$flags['nosimbol']) {
                $currency = $cprefix .
                        ($conversion == 'i' ? $locale['int_curr_symbol'] :
$locale['currency_symbol']) .
                        $csuffix;
            } else {
                $currency = $cprefix .$csuffix;
            }
            $space = $locale["{$letter}_sep_by_space"] ? ' ' : '';

            $value = number_format($value, $right, $locale['mon_decimal_point'], $flags['nogroup']
? '' : $locale['mon_thousands_sep']);
            $value = @explode($locale['mon_decimal_point'], $value);

            $n = strlen($prefix) + strlen($currency) + strlen($value[0]);
            if ($left > 0 && $left > $n) {
                $value[0] = str_repeat($flags['fillchar'], $left - $n) . $value[0];
            }
            $value = implode($locale['mon_decimal_point'], $value);
            if ($locale["{$letter}_cs_precedes"]) {
                $value = $prefix . $currency . $space . $value . $suffix;
            } else {
                $value = $prefix . $value . $space . $currency . $suffix;
            }
            if ($width > 0) {
                $value = str_pad($value, $width, $flags['fillchar'], $flags['isleft'] ?
                         STR_PAD_RIGHT : STR_PAD_LEFT);
            }

            $format = str_replace($fmatch[0], $value, $format);
        }
        return $format;
    }
```

up
down
8
~B ¶
10 years ago

```
We found that after switching from Ubuntu 10.04 php -v 5.3.2, to Ubuntu 12.04 php -v 5.3.10 this
no longer worked:

<?php setlocale(LC_MONETARY, 'en_US'); ?>

Found that using:

<?php setlocale(LC_MONETARY, 'en_US.UTF-8'); ?>

worked find
```

up
down
6
andrey.dobrozhanskiy [-a-t-] gmail com ¶
12 years ago

```
This function divides integer value by commas. F.e.

<?php
echo formatMoney(1050); # 1,050
echo formatMoney(1321435.4, true); # 1,321,435.40
echo formatMoney(10059240.42941, true); # 10,059,240.43
echo formatMoney(13245); # 13,245

function formatMoney($number, $fractional=false) {
    if ($fractional) {
        $number = sprintf('%.2f', $number);
    }
    while (true) {
        $replaced = preg_replace('/(-?\d+)(\d\d\d)/', '$1,$2', $number);
        if ($replaced != $number) {
            $number = $replaced;
        } else {
            break;
        }
    }
    return $number;
}
?>
```

up
down
3
richard dot selby at uk dot clara dot net ¶
16 years ago

```
Double check that money_format() is defined on any version of PHP you plan your code to run on.
You might be surprised.

For example, it worked on my Linux box where I code, but not on  servers running  BSD 4.11
variants. (This is presumably because strfmon  is not defined - see note at the top of teis page).
It's not just a windows/unix issue.
```

up
down
1
swapnet ¶
14 years ago

```
Consider formatting currency for some South Asian countries that use ##,##,###.## money format.
The following code generates something like Rs. 4,54,234.00 and so on.

<?php
```

```
function convertcash($num, $currency){
    if(strlen($num)>3){
            $lastthree = substr($num, strlen($num)-3, strlen($num));
            $restunits = substr($num, 0, strlen($num)-3); // extracts the last three digits
            $restunits = (strlen($restunits)%2 == 1)?"0".$restunits:$restunits; // explodes the
remaining digits in 2's formats, adds a zero in the beginning to maintain the 2's grouping.

            $expunit = str_split($restunits, 2);
            for($i=0; $i<sizeof($expunit); $i++){
                $explrestunits .= (int)$expunit[$i].","; // creates each of the 2's group and adds
a comma to the end
            }

            $thecash = $explrestunits.$lastthree;
    } else {
           $thecash = $convertnum;
    }

    return $currency.$thecash.".00"; // writes the final format where $currency is the currency
symbol.
}
?>

now call the function as  convertcash($row['price'], 'Rs '); // that's the price from the database
I called using an Indian Rupees prefix where the price has to be a plain number format, say
something like 454234.
```

up
down
-1
kaigillmann at googlemail dot com ¶
8 years ago

```
If you get "EUR" instead of the euro symbol, set the locale to utf8 charset like this:

<?php
setlocale(LC_MONETARY, 'de_DE.utf8');
echo money_format('%+n', 1234.56);
?>
```

up
down
-1
phpdeveloperbalaji at gmail dot com ¶
11 years ago

```
Hi,

For South Asian Currencies, this function could be a handy one.

It will handle negative as well as float(Paise).

<?php
function my_money_format($number)
{
    if(strstr($number,"-"))
    {
        $number = str_replace("-","",$number);
        $negative = "-";
    }

    $split_number = @explode(".",$number);
    $rupee = $split_number[0];
    $paise = @$split_number[1];

    if(@strlen($rupee)>3)
    {
        $hundreds = substr($rupee,strlen($rupee)-3);
        $thousands_in_reverse = strrev(substr($rupee,0,strlen($rupee)-3));
        for($i=0; $i<(strlen($thousands_in_reverse)); $i=$i+2)
        {
            $thousands .= $thousands_in_reverse[$i].$thousands_in_reverse[$i+1].",";
        }
        $thousands = strrev(trim($thousands,","));
        $formatted_rupee = $thousands.",".$hundreds;

    }
    else
    {
        $formatted_rupee = $rupee;
    }

    if((int)$paise>0)
    {
        $formatted_paise = ".".substr($paise,0,2);
    }

    return $negative.$formatted_rupee.$formatted_paise;

}
?>
```

Thanks,

up

down

-1

scot from ezyauctionz.co.nz ¶

15 years ago

This is a handy little bit of code I just wrote, as I was not able to find anything else suitable for my situation.
This will handle monetary values that are passed to the script by a user, to reformat any comma use so that it is not broken when it passes through an input validation system that checks for a float.

It is not foolproof, but will handle the common input as most users would input it, such as 1,234,567 (outputs 1234567) or 1,234.00 (outputs 1234.00), even handles 12,34 (outputs 12.34), I expect it would work with negative numbers, but have not tested it, as it is not used for that in my situation.

This worked when other options such as money_format() were not suitable or possible.

```
<?php
//////////////
// BEGIN CODE convert all price amounts into well formatted values
function converttonum($convertnum,$fieldinput){
        $bits = explode(",",$convertnum); // split input value up to allow checking

        $first = strlen($bits[0]); // gets part before first comma (thousands/millions)
        $last = strlen($bits[1]); // gets part after first comma (thousands (or decimals if
incorrectly used by user)

        if ($last <3){ // checks for comma being used as decimal place
            $convertnum = str_replace(",",".",$convertnum);
        }
        else{ // assume comma is a thousands seperator, so remove it
            $convertnum = str_replace(",","",$convertnum);
        }

        $_POST[$fieldinput] = $convertnum; // redefine the vlaue of the variable, to be the new
corrected one
}

@converttonum($_POST[inputone],"inputone");
@converttonum($_POST[inputtwo],"inputtwo");
@converttonum($_POST[inputthree],"inputthree");
// END CODE
//////////////

?>
```

This is suitable for the English usage, it may need tweaking to work with other types.

up
down
-4
sainthyoga2003 at gmail dot com ¶
2 years ago

Be aware, since PHP 7.3 this method is deprecated and from PHP 7.4 this launch a deprecated error, so, setup your PHP web server to untrack E_DEPRECATED error reporting.

up
down
-5
justsomeone ¶
5 years ago

Using the money_format function with float values which have more than two decimal digits can result in rounding errors.
Maybe this function will help you to avoid these failures:

```
<?php
// A product with a base price of 12.95
$price = 1295;

// The quantity is also an integer but translated it would be 11.91
$quantity = 1191;

// Result: 154.2345
// It's the same like 12.95 * 11.91
$sum = ($price / 100) * ($quantity  /100);

// Wrong result: 154.23
money_format('%!i', $sum);

// Wrong result: 154.23
number_format($sum, 2);

// Wrong result: 154.23
bcmul($price / 100, $quantity / 100, 2);
```

```
// Correct result : 154.24
money_format_rounded('%!i', $sum);

/**
* Formats a number as a currency string. Rounds every decimal digit to a defined precision on its
own.
*
* @param string $format The format for the money_format function
* @param float|int|string $number The number to be formatted
* @param int $maxPrecision Round up to the $maxPrecision number of decimal digit. Default is: 2
* @param int $roundingType Rounding type for the round function. Default is: \PHP_ROUND_HALF_UP
*
* @return string
*/
function money_format_rounded($format, $number, $maxPrecision = 2, $roundingType =
\PHP_ROUND_HALF_UP)
{
    $strlen = strlen($number);
    if ($strlen === 0) {
        return money_format($format, $number);
    }

    $length = $strlen - strrpos($number, '.') - 1;
    if ($length <= 0) {
        return money_format($format, $number);
    }

    $rounded = $number;
    for ($i = --$length; $i >= $maxPrecision; $i--) {
        $rounded = round($rounded, $i, $roundingType);
    }

    return money_format($format, $rounded);
}
```

+ add a note

- Funciones de strings
    - addcslashes
    - addslashes
    - bin2hex
    - chop
    - chr
    - chunk_split
    - convert_uudecode
    - convert_uuencode
    - count_chars
    - crc32
    - crypt
    - echo
    - explode
    - fprintf
    - get_html_translation_table
    - hebrev
    - hex2bin
    - html_entity_decode
    - htmlentities
    - htmlspecialchars_decode
    - htmlspecialchars

- implode
- join
- lcfirst
- levenshtein
- localeconv
- ltrim
- md5_file
- md5
- metaphone
- money_format
- nl_langinfo
- nl2br
- number_format
- ord
- parse_str
- print
- printf
- quoted_printable_decode
- quoted_printable_encode
- quotemeta
- rtrim
- setlocale
- sha1_file
- sha1
- similar_text
- soundex
- sprintf
- sscanf
- str_contains
- str_ends_with
- str_getcsv
- str_ireplace
- str_pad
- str_repeat
- str_replace
- str_rot13
- str_shuffle
- str_split
- str_starts_with
- str_word_count
- strcasecmp
- strchr
- strcmp
- strcoll
- strcspn
- strip_tags
- stripcslashes
- stripos
- stripslashes
- stristr
- strlen
- strnatcasecmp
- strnatcmp
- strncasecmp
- strncmp
- strpbrk
- strpos
- strrchr

- strrev
- strripos
- strrpos
- strspn
- strstr
- strtok
- strtolower
- strtoupper
- strtr
- substr_compare
- substr_count
- substr_replace
- substr
- trim
- ucfirst
- ucwords
- utf8_decode
- utf8_encode
- vfprintf
- vprintf
- vsprintf
- wordwrap

- Deprecated
  - convert_cyr_string
  - hebrevc