Focus search box

reset »
« prev

- Manual de PHP
- Referencia de funciones
- Extensiones relacionadas con variable y tipo
- Arrays
- Funciones de Arrays

Change language: Spanish

Submit a Pull Request Report a Bug

range

(PHP 4, PHP 5, PHP 7, PHP 8)

range — Crear un array que contiene un rango de elementos

Descripción ¶

range(mixed `$start`, mixed `$end`, number `$step` = 1): array

Crea un array que contiene un rango de elementos.

Parámetros ¶

`start`

Primer valor de la secuencia.

`end`

La secuencia finaliza al alcanzar el valor `end`.

`step`

Si se proporciona un valor a `step`, este será usado como el incremento entre los elementos de la secuencia. `step` debería darse como número positivo. Si no se especifica, el valor predeterminado de `step` será 1.

Valores devueltos ¶

Devuelve un array de elementos desde `start` a `end`, inclusive.

Ejemplos ¶

Ejemplo #1 Ejemplos de range()

```
<?php
// array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)
foreach (range(0, 12) as $número) {
    echo $número;
}
```

```
// El parámetro step
// array(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
foreach (range(0, 100, 10) as $número) {
    echo $número;
}

// Empleo de las secuencias de caracteres
// array('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i');
foreach (range('a', 'i') as $letra) {
    echo $letra;
}
// array('c', 'b', 'a');
foreach (range('c', 'a') as $letra) {
    echo $letra;
}
?>
```

Notas ¶

Nota:

Los valores de las secuencias de caracteres están limitados a una longitud de uno. Si se entra una longitud mayor que uno, solamente se utilizará el primer carácter.

Ver también ¶

- shuffle() - Mezcla un array
- array_fill() - Llena un array con valores
- foreach

+ add a note

User Contributed Notes 29 notes

up
down
123
Palz ¶
10 years ago

```
To create a range array like

Array
(
    [11] => 1
    [12] => 2
    [13] => 3
    [14] => 4
)

combine two range arrays using array_combine:

array_combine(range(11,14),range(1,4))
```

up
down
10
luca.favorido ATgmailDOT com ¶
6 years ago

```
The function "range" is very useful to get an array of characters as range('C','R') does.

At work, I had to extend the function range($a,$b) to work in this special case: with two
uppercase strings $a and $b, it should return all the possible strings between $a and $b.
This could be used for example to get the excel column indexes.
e.g. <?php range('A','AD') ==> array('A','B','C',...,'Z','AA','AB','AC','AD') ?>

So I wrote the function getrange($min,$max) that exactly does this.

<?php

function getcolumnrange($min,$max){
      $pointer=strtoupper($min);
      $output=array();
      while(positionalcomparison($pointer,strtoupper($max))<=0){
         array_push($output,$pointer);
         $pointer++;
      }
      return $output;
}

function positionalcomparison($a,$b){
   $a1=stringtointvalue($a); $b1=stringtointvalue($b);
   if($a1>$b1)return 1;
   else if($a1<$b1)return -1;
   else return 0;
}

/*
* e.g. A=1 - B=2 - Z=26 - AA=27 - CZ=104 - DA=105 - ZZ=702 - AAA=703
*/
function stringtointvalue($str){
   $amount=0;
   $strarra=array_reverse(str_split($str));

   for($i=0;$i<strlen($str);$i++){
      $amount+=(ord($strarra[$i])-64)*pow(26,$i);
   }
   return $amount;
}
?>
```

up
down
11
php at keith tyler dot com ¶
8 years ago

```
So with the introduction of single-character ranges to the range() function, the internal function
tries to be "smart", and (I am inferring from behavior here) apparently checks the type of the
incoming values. If one is numeric, including numeric string, then the other is treated as
numeric; if it is a non-numeric string, it is treated as zero.

But.

If you pass in a numeric string in such a way that is is forced to be recognized as type string
and not type numeric, range() will function quite differently.

Compare:
```

```
<?php
echo implode("",range(9,"Q"));
// prints 9876543210

echo implode("",range("9 ","Q"));  //space after the 9
// prints 9:;<=>?@ABCDEFGHIJKLMNOPQ

echo implode("",range("q","9 "));
// prints qponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:987654
?>

I wouldn't call this a bug, because IMO it is even more useful than the stock usage of the
function.
```

up
down
19
ThinkMedical at Gmail dot com ¶
14 years ago

```
foreach(range()) whilst efficiant in other languages, such as python, it is not (compared to a
for) in php*.

php is a C-inspired language and thus for is entirely in-keeping with the lanuage aethetic to use
it

<?php
//efficiant
for($i = $start; $i < $end; $i+=$step)
{
        //do something with array
}

//inefficiant
foreach(range($start, $end, $step) as $i)
{
        //do something with array
}
?>

That the officiant documentation doesnt mention the for loop is strange.

Note however, that in PHP5 foreach is faster than for when iterating without incrementing a
variable.

* My tests using microtime and 100 000 iterations consistently (~10 times) show that for is 4x
faster than foreach(range()).
```

up
down
4
ccb_bc at hotmail dot com ¶
3 years ago

```
<?php
    function natural_prime_numbers(array $range, bool $print_info = false) : array {
        $start_time = time();
        $primes_numbers = array();
        $print = '';
        $count_range  = count($range);
        foreach($range as $number){
            $values_division_number = array();
            if($number === 0 || $number === 1 || !is_int($number)){ // eliminate 0, 1 and other no
integer
                continue;
            }
            if($number != 2 && $number%2 === 0){ // eliminate 2 and pairs numbers
                continue;
            }
            for($i = 1; $i <= $number; $i++){
                $resultado_divisao = $number / $i;
                $values_division_number[$i] = $resultado_divisao;

                if($count_range <= 20){ // $count_range <= 20 (+ performance)
                    $print .= PHP_EOL;
                    $info = 'The number '.$number.' divided by the number '.$i.' is equal to: '.
($number / $i);
                    $print .= $info;
                    if($i === $number){
                        $print .= PHP_EOL;
                    }
                }

                array_walk($values_division_number, function($value, $index) use
(&$values_division_number, &$number){ // reference change values
                    // eliminate floats and others numbers not are equal 1 and own number
                    if(is_float($value) && $value != $number && $value > 1){
                        unset($values_division_number[$index]);
                    }
                });

                $values_division_number = array_values($values_division_number); // reindex array

                // here we want only array with 2 indexes with the values 1 and own number (rule
to a natural prime number)
                if(count($values_division_number) === 2 && $values_division_number[0] === $number
&& $values_division_number[1] === 1){
                    $primes_numbers[$number] = $number;
                }

            }
        }
        return array(
            'length_prime_numbers' => count($primes_numbers),
            'prime_numbers' => array_values($primes_numbers),
            'print' => $print,
            'total_time_processing' => (time() - $start_time).' seconds.',
        );
    }
    var_dump(natural_prime_numbers(range(0, 11))); // here the range() function ;-)

    // Result:
    // array (size=3)
    //   'length_prime_numbers' => int 5
    //   'prime_numbers' =>
    //     array (size=5)
    //       0 => int 2
    //       1 => int 3
    //       2 => int 5
    //       3 => int 7
    //       4 => int 11
    //   'print' => string '
    // O número 2 dividido pelo número 1 é igual a: 2
    // O número 2 dividido pelo número 2 é igual a: 1

    // O número 3 dividido pelo número 1 é igual a: 3
    // O número 3 dividido pelo número 2 é igual a: 1.5
    // O número 3 dividido pelo número 3 é igual a: 1

    // O número 5 dividido pelo número 1 é igual a: 5
    // O número 5 dividido pelo número 2 é igual a: 2.5
    // O número 5 dividido pelo número 3 é igual a: 1.6666666666667
    // O número 5 dividido pelo número 4 é igual a: 1.25
    // O número 5 dividido pelo '...

    // **************************** //
    //
    // * Remember that the function is recursive, that is: a range of 5000 takes more than 1
minute on a processor Intel® Core™ i5-8250U (3.40 GHz).
    //
    // **************************** //
?>
```

up
down
7

Alien426 ¶

7 years ago

The function will generate an array of integers even if your numerical parameters are enclosed in quotes.

```
<?php
var_dump( range('1', '2') ); // outputs  array(2) { [0]=> int(1) [1]=> int(2) }
?>
```

An easy way to get an array of strings is to map strval() to the range:

```
<?php
var_dump( array_map('strval', range('1', '2')) ); // outputs  array(2) { [0]=> string(1) "1" [1]=>
string(1) "2" }
?>
```

up
down
3

moficer at host dot sk ¶

6 years ago

php 5.6.16

```
<?php
var_export(range('Z', 'a'));

/*
array (
  0 => 'Z',
  1 => '[',
  2 => '\\',
  3 => ']',
  4 => '^',
  5 => '_',
  6 => '`',
  7 => 'a',
)
*/
```
up
down
10
gtisza at gmail dot com ¶
9 years ago
```
You might expect range($n, $n-1) to be an empty array (as in e.g. Python) but actually PHP will
assume a step of -1 if start is larger than end.
```
up
down
6
chris at laflash dot org ¶
15 years ago
```
Quick HTML menus with minimum and maximum sets of years:

<?php
    /*
    ** Quick HTML menus with minimum and maximum sets of years.
    ** @author Chris Charlton <chris@laflash.org>
    ** @license FREE!
    */

    // Years range setup
    $year_built_min = 1900;
    $year_built_max = date("Y");
?>
<select id="yearBuiltMin" size="1">
    <?php // Generate minimum years

        foreach (range($year_built_min, $year_built_max) as $year) { ?>
        <option value="<?php echo($year); ?>"><?php echo($year); ?></option>
        <?php } ?>
</select>

<select id="yearBuiltMax" size="1">
      <?php // Generate max years

        foreach (range($year_built_max, $year_built_min) as $year) { ?>
        <option value="<?php echo($year); ?>"><?php echo($year); ?></option>
        <?php } ?>
</select>
```
up
down
6
m0sh3 at hotmail dot com ¶
15 years ago
```
Here's how i use it to check if array is associative or not:

<?php

if (array_keys($arr)===range(0, sizeof($arr)-1)) {
// not associative array

} else {
// associative array

}
```

```
?>
```

up
down
7

ktamas77 at gmail dot com ¶

10 years ago

```
if you need zero padding, string prefixes or any other masks, then a simple combination of
array_map, inline functions and sprintf is your friend.

<?php

$a = array_map(function($n) { return sprintf('sample_%03d', $n); }, range(50, 59) );

print_r($a);

?>

Will result:

Array
(
    [0] => sample_050
    [1] => sample_051
    [2] => sample_052
    [3] => sample_053
    [4] => sample_054
    [5] => sample_055
    [6] => sample_056
    [7] => sample_057
    [8] => sample_058
    [9] => sample_059
)
```

up
down
11

me at phpscott dot com ¶

10 years ago

```
So, I needed a quick and dirty way to create a dropdown select for hours, minutes and seconds
using 2 digit formatting, and to create those arrays of data, I combined range with array merge..

<?php
$prepend = array('00','01','02','03','04','05','06','07','08','09');
$hours     = array_merge($prepend,range(10, 23));
$minutes     = array_merge($prepend,range(10, 59));
$seconds     = $minutes;
?>

Super simple.
```

up
down
3

Ray.Paseur often uses Gmail ¶

9 years ago

```
Interestingly, these two statements produce identical 26-character alphabet arrays.

<?php
```

```
$arr = range('A',  'Z');
$arr = range('AA', 'ZZ');
```

up
down
6
dries at volta dot be ¶
10 years ago

```
Ever wanted to generate an array with a range of column names for use in Excel file related
parsing?
I've wrote a function that starts at the A column and adds column names up until the column you
specified.

<?php

/**
* This function creates an array with column names up until the column
* you specified.
*/
function createColumnsArray($end_column, $first_letters = '')
{
  $columns = array();
  $length = strlen($end_column);
  $letters = range('A', 'Z');

  // Iterate over 26 letters.
  foreach ($letters as $letter) {
      // Paste the $first_letters before the next.
      $column = $first_letters . $letter;

      // Add the column to the final array.
      $columns[] = $column;

      // If it was the end column that was added, return the columns.
      if ($column == $end_column)
          return $columns;
  }

  // Add the column children.
  foreach ($columns as $column) {
      // Don't itterate if the $end_column was already set in a previous itteration.
      // Stop iterating if you've reached the maximum character length.
      if (!in_array($end_column, $columns) && strlen($column) < $length) {
          $new_columns = createColumnsArray($end_column, $column);
          // Merge the new columns which were created with the final columns array.
          $columns = array_merge($columns, $new_columns);
      }
  }

  return $columns;
}

?>

Usage:

<?php

// Return an array with all column names from A until and with BI.
```

```
createColumnsArray('BI');

?>
```

up
down
5
captvanhalen at gmail dot com ¶
14 years ago

```
Here is a home rolled range() function that uses the step feature for those unfortunate souls who
cannot use PHP5:

<?php
function my_range( $start, $end, $step = 1) {

    $range = array();

    foreach (range( $start, $end ) as $index) {

        if (! (($index - $start) % $step) ) {
            $range[] = $index;
        }
    }

    return $range;
}
?>
```

up
down
3
jay at NOspam dot myd3 dot com ¶
13 years ago

```
This is a modified version of thomas' range_string() function. It's simpler, cleaner, and more
robust, but it lacks the advanced features his function had, hopefully it will be of assitance to
someone.

Examples:

    input: "1, 2, 3, 4, 5, 6" --> output: 1, 2, 3, 4, 5, 6
    input: "1-6" --> output: 1, 2, 3, 4, 5, 6
    input: "1-6" --> output: 1, 2, 3, 4, 5, 6
    input: "1 - -6" --> output: 1, 2, 3, 4, 5, 6
    input: "0 - 0" --> output: 0
    input: "1, 4-6, 2" --> output: 1, 2, 4, 5, 6
    input: "6,3-1" --> output: 1, 2, 3, 6

<?php

define('RANGE_ARRAY_SORT', 1);
define('RANGE_ARRAY', 2);
define('RANGE_STRING_SORT', 3);
define('RANGE_STRING', 4);

function range_string($range_str, $output_type = RANGE_ARRAY_SORT)
{
    // Remove spaces and nother non-essential characters
    $find[]    = "/[^\d,\-]/";
    $replace[] = "";
    // Remove duplicate hyphens
    $find[]    = "/\-+/";
    $replace[] = "-";

    // Remove duplicate commas
    $find[]    = "/\,+/";
    $replace[] = ",";

    $range_str = preg_replace($find, $replace, $range_str);

    // Remove any commas or hypens from the end of the string
    $range_str = trim($range_str,",-");

    $range_out = array();
    $ranges    = explode(",", $range_str);

    foreach($ranges as $range)
    {

        if(is_numeric($range) || strlen($range) == 1)
        {
            // Just a number; add it to the list.
            $range_out[] = (int) $range;
        }
        else if(is_string($range))
        {

            // Is probably a range of values.
           $range_exp = preg_split("/(\D)/",$range,-1,PREG_SPLIT_DELIM_CAPTURE);

            $start = $range_exp[0];
            $end   = $range_exp[2];

            if($start > $end)
            {
                for($i = $start; $i >= $end; $i -= 1)
                {
                    $range_out[] = (int) $i;
                }
            }
            else
            {
                for($i = $start; $i <= $end; $i += 1)
                {
                    $range_out[] = (int) $i;
                }
            }

        }
    }

    switch ($output_type) {
        case RANGE_ARRAY_SORT:
            $range_out = array_unique($range_out);
            sort($range_out);

        case RANGE_ARRAY:
            return $range_out;
            break;

        case RANGE_STRING_SORT:
            $range_out = array_unique($range_out);
            sort($range_out);

        case RANGE_STRING:

        default:
            return implode(", ", $range_out);
            break;
    }
}

// Sample Usage:
$range = range_string("6, 3-1");

?>
```

up
down
2
jazzduck AT gmail DOT com ¶
8 years ago

Despite the line above that says that the $step value should be "given as a positive number," the range() function will in fact correctly handle reversed (decrementing) ranges. For example:

```
<?php print_r( range( 24, 20 ) ); ?>
Array
(
    [0] => 24
    [1] => 23
    [2] => 22
    [3] => 21
    [4] => 20
)

<?php print_r( range( 20, 11, -3 ) ); ?>
Array
(
    [0] => 20
    [1] => 17
    [2] => 14
    [3] => 11
)
```

It will actually ignore the sign of the $step argument, and determine whether to increment or decrement based purely on whether $start > $end or $end > $start. For example:

```
<?php print_r( range( 20, 11, 3 ) ); ?>
Array
(
    [0] => 20
    [1] => 17
    [2] => 14
    [3] => 11
)

<?php print_r( range( 11, 20, -3 ) ); ?>
```

```
Array
(
    [0] => 11
    [1] => 14
    [2] => 17
    [3] => 20
)
```

up
down
2

manuel at levante dot de ¶

16 years ago

```
<?php
function srange ($s) {
  preg_match_all("/([0-9]{1,2})-?([0-9]{0,2}) ?,?;?/", $s, $a);
  $n = array ();
  foreach ($a[1] as $k => $v) {
    $n  = array_merge ($n, range ($v, (empty($a[2][$k])?$v:$a[2][$k])));
  }
  return ($n);
}

$s = '1-4 6-7 9-10';
print_r(srange($s));
?>

Return:
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 4
    [4] => 6
    [5] => 7
    [6] => 9
    [7] => 10
)
```

up
down
1

qz ¶

6 years ago

```
If you're looking to fill an array to get a hash with 0-9 numerical values, using
range(0,9);
is a faster solution compared to
array_fill(0, 10, '');
```

up
down
1

krdr dot mft at gmail dot com ¶

9 years ago

```
I've been introduced with range() function not so long ago, and I found that examples about it is
somewhat wrong, even inefficient:

<?php
$o = "";
$time_start = microtime(true);
```

```
foreach(range(1, 10000) as $val) {
    $o .= $val;
}
$time_end = microtime(true);
$time = $time_end - $time_start;
echo 'rangein: '.$time.'<br />';

$o = "";
$time_start = microtime(true);
$a = range(1, 10000);
foreach($a as $val) {
    $o .= $val;
}
$time_end = microtime(true);
$time = $time_end - $time_start;
echo 'rangeout: '.$time.'<br />';

?>

Which gives results:

rangein: 0.0025348663330078
rangeout: 0.0019199848175049

In some cases difference is even bigger and proportional to the range generated. I suppose that
results of range() are cached/hashed.

Note: execution order does affects execution times, but difference still exists
```

up
down
1
lsblsb at gmx dot de ¶
8 years ago

```
I needed a function, that creates a letter range with arbitrary length.
You specify via the $length parameter, how many entries you need.
Logic is analog to the logic of the column-titles in a calc-sheet.

<?php

/**
  * create a letter range with arbitrary length
  * @param int $length
  * @return array
  */
function createLetterRange($length)
{
    $range = array();
    $letters = range('A', 'Z');
    for($i=0; $i<$length; $i++)
    {
        $position = $i*26;
        foreach($letters as $ii => $letter)
        {
            $position++;
            if($position <= $length)
                $range[] = ($position > 26 ? $range[$i-1] : '').$letter;
        }
    }
    return $range;
}
?>
```

up
down
1
mohammed dot hussein dot mahmoud at gmail dot com ¶
10 months ago

```
You could use negative numbers in place of the `step` parameter. You need to make sure that the
`start` is bigger than `end`. Note that range() function in php generates the range inclusive,
i.e. it also includes the `end` parameter and not just up to it but not including it like most
other languages.
The following snippet of code should explain what I mean about negative steps:

<?php

// 100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 0
print_r(range(100, 0, -10));

?>

What happens basically is that the range function does not really care about what is bigger or
smaller, it just adds the step to the start and appends that to the a temp result variable as long
as it did not reach the end param value. In this case, adding negative numbers is like minus
(computers do that for 2's complement under the hood.) This will cause the number to go from 100
to 90 and then the function will check if 90 reached 0 yet. Since it wouldn't have done that, it
will keep adding -step (-10 in that case) to the latest result (i.e. 90) and so on and so forth.

Since range() is said to be better and faster than array_fill() I believe it was important for me
to try it out and actually post this note on the official documentation just to make sure people
can use this.
```

up
down
0
derek at php dot net ¶
17 years ago

```
This should emulate range() a little better.
<?php
function range_wroar($low, $high, $step = 1) {
    $arr = array();
    $step = (abs($step)>0)?abs($step):1;
    $sign = ($low<=$high)?1:-1;
    if(is_numeric($low) && is_numeric($high)) {
        //numeric sequence
        for ($i = (float)$low; $i*$sign <= $high*$sign; $i += $step*$sign)
            $arr[] = $i;
    }   else    {
        //character sequence
        if (is_numeric($low))
            return $this->range($low, 0, $step);
        if (is_numeric($high))
            return $this->range(0, $high, $step);
        $low = ord($low);
        $high = ord($high);
        for ($i = $low; $i*$sign <= $high*$sign; $i += $step*$sign) {

            $arr[] = chr($i);
        }
    }
    return $arr;
}
?>
```

up
down
-1
pyetrosafe at gmail dot com ¶
9 years ago

```
To create a simple array or a multidimensional array with defined size and null values, use this
expression:

<?php

$SimpleArray = array_map(function($n) { return null; }, range(1, 3) );
$MultiArray = array_map(function($n) { return array_map(function($n) { return null; }, range(1, 2)
); }, range(1, 3) );

var_dump($SimpleArray);
var_dump($MultiArray);

// And will print:
?>
>>$SimpleArray
array(3) {
  [0]=>  NULL
  [1]=>  NULL
  [2]=>  NULL
}

>>$MultiArray
array(3) {
  [0]=>  array(2) {
    [0]=>    NULL
    [1]=>    NULL
  }
  [1]=>  array(2) {
    [0]=>    NULL
    [1]=>    NULL
  }
  [2]=>  array(2) {
    [0]=>    NULL
    [1]=>    NULL
  }
}

?>
```

up
down
-2
subscription101 at hotmail dot com ¶
16 years ago

```
A much simpler way of creating a range of even numbers is by starting with an even number:

<?php

    range(2, 10, 2);
```

```
?>
```

up
down
-2
j dot gizmo at aon dot at ¶
18 years ago

```
i figured i'd add some more functionality to the myRange() functions below.
now you can, besides giving a $step parameter,
1. count backwards
2. count with letters
3. give whatever parameter you want, there's nothing (i know of) that will cause an endless loop
(try a negative $step for the previous function....)

<?php
function myRange($num1, $num2, $step=1)
{
    if (is_numeric($num1) && is_numeric($num2))
    {
        //we have a numeric range
        $step = ( abs($step)>0 ? abs($step) : 1 ); //make $step positive
        $dir = ($num1<=$num2 ? 1 : -1); //get the direction
        for($i = (float)$num1; $i*$dir <= $num2*$dir; $i += $step*$dir)
        {
            $temp[] = $i;
        }
    }
    else
    {
        //we have a character range
        $num1=ord((string)$num1); //convert to ascii value
        $num2=ord((string)$num2);
        $step = ( abs($step)>0 ? abs($step) : 1 ); //make $step positive
        $dir = ($num1<=$num2 ? 1 : -1); //get direction
        for($i = $num1; $i*$dir <= $num2*$dir; $i += $step*$dir)
        {
            $temp[] = chr($i);
        }
    }
    return $temp;
}

print_r(myRange( 1, 3, 0.5 )); //you can use fractional steps
print_r(myRange( "a", "k", 3 )); //or count letters
print_r(myRange( "5", "9" )); //numbers are detected even if hidden in strtings
print_r(myRange( "!", "%", 1/pi() )); //or mess around with senseless parameters

?>
```

up
down
-3
emory underscore smith at hotmail ¶
17 years ago

```
since its not stated explicitly above, thought id point out that you arent limited to using
integers.

however, be careful when doing so, as you might not get the range you expect!
```

```
to illustrate:

<?php
$am = range(500,1600,10);
$fm = range(88.1,107.9,.2);
print_r($am);
print_r($fm);
?>

print_r($am) yields the expected result:

Array
(
    [0] => 500
    [1] => 510
    [2] => 520
    ...
    [109] => 1590
    [110] => 1600
)

print_r($fm), however, falls a bit (1%) short:

Array
(
    [0] => 88.1
    [1] => 88.3
    [2] => 88.5
    ...
    [97] => 107.5
    [98] => 107.7
)

so, if you want to use a non-integral step size params for numeric ranges, be sure to account for
fp representation accuracy and error accumulation; a step size of something like pi or 1/10 could
spell disaster for a large range. if in doubt, use integral steps and divide ... something like <?
php range(88.1,108,.2) ?> might work to recover 107.9, but would not be scalable like, say <?php
array_map(create_function('$x','return $x/10;'),range(881,1079,2)) ?>.

-emory
```

up
down
-3
unicod3 at hotmail dot com ¶
8 years ago

```
a function to get column index by letter

function getColumnNumber($char){
    $alphabet = range('a','z');
    $alphabet2 = range('a','z');
    $newAlphabet = $alphabet;
    foreach($alphabet as $k => $r)
    {
        foreach($alphabet2 as $row){
            $newAlphabet[] = $r.$row;
        }
    }
    $key = array_search($char, $newAlphabet);
    return ($key !== false) ? $key : null;
}
```

up
down
-5
Aram Kocharyan ¶
11 years ago

```
Here's a function to generate ranges from strings:

<?php

/*  Creates an array of integers based on a given range string of format "int - int"
    Eg. range_str('2 - 5'); */
function range_str($str) {
    preg_match('#(\\d+)\\s*-\\s*(\\d+)#', $str, $matches);
    if ( count($matches) == 3 ) {
        return range($matches[1], $matches[2]);
    }
    return FALSE;
}

// Test
$array = range_str(' 2 - 4 ');
print_r($array);

?>

This outputs:

Array
(
    [0] => 2
    [1] => 3
    [2] => 4
)
```

+ add a note

- Funciones de Arrays
  - array_change_key_case
  - array_chunk
  - array_column
  - array_combine
  - array_count_values
  - array_diff_assoc
  - array_diff_key
  - array_diff_uassoc
  - array_diff_ukey
  - array_diff
  - array_fill_keys
  - array_fill
  - array_filter
  - array_flip
  - array_intersect_assoc
  - array_intersect_key
  - array_intersect_uassoc
  - array_intersect_ukey
  - array_intersect
  - array_is_list

    - uasort
    - uksort
    - usort
- Deprecated
    - each