
Lista de funciones PHP

Funciones para:

Cadenas y Matrices

INDICE

FUNCIONES PARA CADENAS **2**

• STRTOK	2
• SUBSTR	2
• STRLEN	3
• STR_WORD_COUNT	3
• STRCMP	4
• STRSTR	5
• STRPOS	5
• STR_REPLACE	6
• STR_Ireplace	7
• UCFIRST	7
• UCWORDS	7
• STRTOUPPER	7
• STRTOLOWER	7

FUNCIONES PARA MATRICES **8**

• COUNT	8
• ARRAY MERGE	8
• SORT	9
• RSORT	9
• ASORT	10
• ARSORT	10
• KSORT	11
• KRSORT	11
• SHUFFLE	11

Funciones para Cadenas



- **strtok**

Divide una cadena (str) en cadenas más pequeñas (tokens). Sólo la primera llamada a strtok utiliza el argumento cadena. Cada llamada posterior a strtok sólo necesita el token, ya que realiza un seguimiento de su ubicación en la cadena actual

Ejemplo:

```
<?php
$cadena = "Esto es un ejemplo";
/ * utilizamos el espacio como token (marca donde cortar) * /
$ tok = strtok($cadena," ");
while($tok !== false) {
    echo "Palabra = $tok <br /> " ;
    $tok = strtok(" ");
}
?>
```

- **substr**

Devuelve parte de una cadena. Devuelve una parte del string Definida Por los parámetros start y length.  

Ejemplo longitud negativo:

```
<?php

$resto = substr("abcdef",0,-1);    //devuelve "abcde"

$resto = substr("abcdef",2,-1) ;   //devuelve "cde"

$resto = substr("abcdef",4,-4);    //devuelve false

$resto = substr("abcdef",-3,-1);   //devuelve "de"

?>
```

sin comienzo negativo

```
<?php

$resto = substr("abcdef", -1);     // devuelve "f"

$resto = substr("abcdef", -2);     // devuelve "ef"

$resto = substr("abcdef", -3 , 1); // devuelve "d"
```

?>

- **strlen**

Obtiene la longitud de un string

Ejemplo

```
<?php
$str = 'abcdef';
echo strlen($str); // 6

$str = ' ab cd ';
echo strlen($str); // 7
?>
```

- **str_word_count**

Devuelve información sobre las palabras utilizadas en un string

Cuenta el número de palabras dentro de **string**. Si no se especifica el **format** opcional, entonces el valor devuelto será un integer representando el número de palabras encontradas. En el caso en que se especifique **format**, el valor devuelto será un array cuyo contenido depende de **format**. Los posibles valores para **format** y las salidas resultantes están listadas más abajo.

format

Especifica el valor devuelto de esta función. Los valores soportados actualmente son:

- 0 - devuelve el número de palabras encontradas
- 1 - devuelve un array que contiene todas las palabras encontradas dentro del **string**
- 2 - devuelve un array asociativo, donde la clave es la posición numérica de una palabra dentro del **string** y el valor es la palabra en sí.

Ejemplo

```
<?php
$str = "Hello fri3nd, you're
      looking      good today!";
echo str_word_count($str);
print_r(str_word_count($str, 1));
print_r(str_word_count($str, 2));
?>
```

El resultado será:

7

Array (te devuelve el numero de palabras sin contar el n°)

```
(  
  [0] => Hello  
  [1] => fri  
  [2] => nd  
  [3] => you're  
  [4] => looking  
  [5] => good  
  [6] => today  
)
```

Array(te devuelve el numero de la posición que ocupa)

```
(  
  [0] => Hello  
  [6] => fri  
  [10] => nd  
  [14] => you're  
  [29] => looking  
  [46] => good  
  [51] => today  
)
```

- **strcmp**

Comparación de string segura a nivel binario. Tenga en cuenta que esta comparación es sensible a mayúsculas y minúsculas.

Ejemplo

Para aquellos que están confundidos acerca de la forma en que esta función funciona:

```
<?php  
$cadena1='a';  
$cadena2='b';  
echo strcmp($cadena1 , $cadena2); // -1  
?>
```

Alfabéticamente 'a' precede a 'b'. Si vemos las cadenas como valores 'a' es menor que 'b' y por lo tanto, la función devuelve -1.

- **strstr**

Encuentra la primera aparición de un string.

Devuelve parte del string **haystack** iniciando desde e incluyendo la primera aparición de **needle** (aguja) hasta el final del **haystack** (pajar).

Parámetros:

haystack

El string en donde buscar.

needle

Si **needle** no es un string, será convertido como número entero y se aplicará el valor ordinal de carácter.

Ejemplo

```
<?php
$email = 'name@example.com';
$domain = strstr($email, '@');
echo $domain; // mostrará @example.com

$user = strstr($email, '@', true); // Desde PHP 5.3.0
echo $user; // mostrará name
?>
```

- **strpos**

Encuentra la posición de la primera ocurrencia de un substring en un string

Ejemplo usando ===

```
<?php
$string = 'abc';
$findme = 'a';
$pos = strpos($string, $findme);

// Nótese el uso de ===. Puesto que == simple no funcionará como
// se espera
// porque la posición de 'a' está en el 1º (primer) caracter.
if ($pos === false) {
    echo "La cadena '$findme' no fue encontrada en la cadena
    '$string'";
} else {
```

```

        echo "La cadena '$findme' fue encontrada en la cadena
'$mystring'";
        echo " y existe en la posición $pos";
    }
?>

```

Ejemplo usando ===

```

<?php
$mystring = 'abc';
$findme = 'a';
$pos = strpos($mystring, $findme);

// El operador !== también puede ser usado. Puesto que != no funcionará
// como se espera
// porque la posición de 'a' es 0. La declaración (0 != false) se evalúa a
// false.
if ($pos !== false) {
    echo "La cadena '$findme' fue encontrada en la cadena '$mystring'";
    echo " y existe en la posición $pos";
} else {
    echo "La cadena '$findme' no fue encontrada en la cadena '$mystring'";
}
?>

```

• str_replace

Reemplaza todas las apariciones del string buscado con el string de reemplazo

```

<?php
// Produce: <body text='black'>
$bodytag = str_replace("%body%", "black", "<body text='%body%'>");

// Produce: Hll Wrld f PHP
$vowels = array("a", "e", "i", "o", "u", "A", "E", "I", "O", "U");
$onlyconsonants = str_replace($vowels, "", "Hello World of PHP");

// Produce: You should eat pizza, beer, and ice cream every day
$phrase = "You should eat fruits, vegetables, and fiber every day.";
$healthy = array("fruits", "vegetables", "fiber");
$yummy = array("pizza", "beer", "ice cream");

$newphrase = str_replace($healthy, $yummy, $phrase);

// Produce: 2
$str = str_replace("ll", "", "good golly miss molly!", $count);
echo $count;
?>

```

- **str_ireplace**

Versión insensible a mayúsculas y minúsculas de str_replace()

- **ucfirst**

Convierte el primer carácter de una cadena a mayúsculas

```
<?php
$foo = 'hello world!';
$foo = ucfirst($foo);           // Hello world!

$bar = 'HELLO WORLD!';
$bar = ucfirst($bar);           // HELLO WORLD!
$bar = ucfirst(strtolower($bar)); // Hello world!
?>
```

- **ucwords**

Convierte a mayúsculas el primer carácter de cada palabra en una cadena

```
<?php
$foo = 'hello world!';
$foo = ucwords($foo);           // Hello World!

$bar = 'HELLO WORLD!';
$bar = ucwords($bar);           // HELLO WORLD!
$bar = ucwords(strtolower($bar)); // Hello World!
?>
```

- **strtoupper**

Convierte un string a mayúsculas

```
<?php
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtoupper($str);
echo $str; // muestra: MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
?>
```

- **strtolower**

Convierte una cadena a minúsculas

```
<?php
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtolower($str);
echo $str; // Prints mary had a little lamb and she loved it so
?>
```


FUNCIONES para MATRICES

- **count**

Cuenta todos los elementos de un array o en un objeto

```
<?php
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$result = count($a);
// $result == 3

$b[0] = 7;
$b[5] = 9;
$b[10] = 11;
$result = count($b);
// $result == 3
?>
```

- **array merge**

Combina dos o más arrays

```
<?php
$array1 = array("color" => "red", 2, 4);
$array2 = array("a", "b", "color" => "green", "shape" => "trapezoid", 4);
$result = array_merge($array1, $array2);
print_r($result);
?>
```

El resultado del ejemplo sería:

```
Array
(
    [color] => green
    [0] => 2
    [1] => 4
    [2] => a
    [3] => b
    [shape] => trapezoid
    [4] => 4
)
```

- **sort**

Ordena un array

```
<?php

$frutas = array("limón", "naranja", "banana", "albaricoque");
sort($frutas);
foreach ($frutas as $clave => $valor) {
    echo "frutas[" . $clave . "] = " . $valor . "\n";
}
?>
```

El resultado del ejemplo sería:

```
frutas[0] = albaricoque
frutas[1] = banana
frutas[2] = limón
frutas[3] = naranja
```

Las frutas han sido ordenadas en orden alfabético.

- **rsort**

Ordena un array en orden inverso

```
<?php
$fruits = array("limón", "naranja", "plátano", "manzana");
rsort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val\n";
}
?>
```

El resultado del ejemplo sería:

```
0 = plátano
1 = naranja
2 = manzana
3 = limón
```

Las frutas han sido ordenadas alfabéticamente pero en orden inverso.

- **asort**

Ordena un array y mantiene la asociación de índices

```
<?php
$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana"
, "c" => "apple");
asort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val\n";
}
?>
```

El resultado del ejemplo sería:

```
c = apple
b = banana
d = lemon
a = orange
```

Las frutas han sido ordenadas alfabéticamente, y se ha mantenido el índice asociado con cada elemento.

- **arsort**

Ordena un array en orden inverso y mantiene la asociación de índices

```
<?php
$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana"
, "c" => "apple");
arsort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val\n";
}
?>
```

El resultado del ejemplo sería:

```
a = orange
d = lemon
b = banana
c = apple
```

Las frutas han sido ordenadas en orden inverso alfabético, se ha mantenido el índice asociado con cada elemento.

- **ksort**

Ordena un array por clave

```
<?php
$fruits = array("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=
>"apple");
ksort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val\n";
}
?>
```

El resultado del ejemplo sería:

```
a = orange
b = banana
c = apple
d = lemon
```

- **krsort**

Ordena un array por clave en orden inverso

```
<?php
$fruits = array("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=
>"apple");
krsort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val\n";
}
?>
```

El resultado del ejemplo sería:

```
d = lemon
c = apple
b = banana
a = orange
```

- **shuffle**

Mezcla un array

```
<?php
$numbers = range(1, 20);
shuffle($numbers);
foreach ($numbers as $number) {
    echo "$number ";
}
?>
```