# REFERENCIA FUNCIONES PHP

## Índice



## Expresiones Regulares

preg_match ( string $pattern , string $subject [, array &$matches [, int $flags = 0 [, int $offset = 0 ]]] ) : int

```
preg_match('/^[a-zA-Z]+/',"abcd123",$coincidencias);
=> 1
>>> $coincidencias
=> ["abcd"]
>>> preg_match('/[0-9]+$/',"abcd123",$coincidencias);
=> 1
>>> $coincidencias
=> ["123"]
```

## Cadenas

sprintf ( string $format [, mixed $args [, mixed $... ]] ) : string

```
>>> sprintf("%d %s %2.5f",5,"Hola",5.5)
=> "5 Hola 5.50000"
```

strlen ( string $string ) : int

```
>>> strlen("hola")
```

=> 4
>>> strlen("hola que tal")
=> 12

str_split ( string $string [, int $split_length = 1 ] ) : array

>>> str_split(1235)
=> ["1","2","3","5"]
>>> str_split("Pepe")
=> ["P","e","p","e"]

substr ( string $string , int $start [, int $length ] ) : string

>>> substr("Pepe",1,2)
=> "ep"
>>> substr("Pepe",1)
=> "epe"

trim ( string $str [, string $character_mask = " \t\n\r\0\x0B" ] ) : string

>>> $s="   hola que tal   ";
=> "   hola que tal   "
>>> trim($s)
=> "hola que tal"

## Ficheros

fopen ( string $filename , string $mode [, bool $use_include_path = **FALSE** [, resource $context ]] ) : resource

Valores posibles para mode:
'r' Apertura para sólo lectura; coloca el puntero al fichero al principio del fichero.
'r+' Apertura para lectura y escritura; coloca el puntero al fichero al principio del fichero.
'w' Apertura para sólo escritura; coloca el puntero al fichero al principio del fichero y trunca el fichero a longitud cero. Si el fichero no existe se intenta crear.
'w+' Apertura para lectura y escritura; coloca el puntero al fichero al principio del fichero y trunca el fichero a longitud cero. Si el fichero no existe se intenta crear.
'a' Apertura para sólo escritura; coloca el puntero del fichero al final del mismo. Si el fichero no existe, se intenta crear. En este modo, fseek()

solamente afecta a la posición de lectura; las lecturas siempre son pospuestas.
'a+' Apertura para lectura y escritura; coloca el puntero del fichero al final del mismo. Si el fichero no existe, se intenta crear. En este modo, fseek() no tiene efecto, las escrituras siempre son pospuestas.

fgets ( resource $handle [, int $length ] ) : string

Obtiene una línea desde el puntero al fichero.

fputs Esta función es un alias de: fwrite().

fwrite ( resource $handle , string $string [, int $length ] ) : int

**fwrite()** escribe el contenido de **string** al flujo de archivo apuntado por **handle**

fclose ( resource $handle ) : bool

El archivo apuntado por **handle** es cerrado.

feof ( resource $handle ) : bool

Comprueba si el puntero a un archivo está al final del archivo.

file_get_contents ( string $filename [, bool $use_include_path = **FALSE** [, resource $context [, int $offset = 0 [, int $maxlen ]]]] ) : string

Esta función es similar a file(), excepto que **file_get_contents()** devuelve el fichero a un string, comenzando por el offset especificado hasta maxlen bytes. Si falla, **file_get_contents()** devolverá **FALSE**.
**file_get_contents()** es la manera preferida de transmitir el contenido de un fichero a una cadena. Para mejorar el rendimiento, utiliza técnicas de mapeado de memoria si sistema operativo lo admite..

file ( string $filename [, int $flags = 0 [, resource $context ]] ) : array

Transfiere un fichero completo a un array.

Recorrido de un fichero de texto
---
```
$handle=fopen("fichero.txt","r");
while($linea=fgets($handle)) {
   hacer lo que haya que hacer con $linea
}
fclose($handle)
```

---
Alternativa:
---
```
$handle=fopen("fichero.txt","r");
while(!feof($handle)) {
   $linea=fgets($handle);
   hacer lo que haya que hacer con $linea
}
fclose($handle)
```

## Arrays

in_array ( mixed $needle , array $haystack [, bool $strict = **FALSE** ] ) : bool

```
>>> $a=[1,2,3,4,5]
=> [ 1, 2, 3, 4, 5 ]
>>> in_array(2,$a)
=> true
>>> in_array(6,$a)
=> false
```

array_push ( array &$array , mixed $value1 [, mixed $... ] ) : int

```
>>> $a=[1,2,3,4,5]
=> [ 1, 2, 3, 4, 5 ]
>>> array_push($a,6)
=> 6
>>> $a
=> [1, 2, 3, 4, 5, 6]
```

array_pop ( array &$array ) : mixed

```
>>> $a=[1,2,3,4,5]
=> [ 1, 2, 3, 4, 5 ]
>>> array_pop($a)
```

```
=> 6
>>> $a
=> [1,2,3,4,5]
```

array_count_values ( array $array ) : array

```
>>> $a=[1,2,3,4,5]
=> [1,2,3,4,5]
>>> array_push($a,1)
=> 6
>>> array_push($a,1)
=> 7
>>> array_push($a,2)
=> 8
>>> array_push($a,2)
=> 9
>>> $a
=> [1,2,3,4,5,1,1,2,2]

>>> array_count_values($a)
=> [1 => 3,2 => 3,3 => 1,4 => 1,5 => 1]
```

array_shift ( array &$array ) : mixed

```
>>> $a=[1,2,3,4,5,1,1,2,2]
=> [1,2,3,4,5,1,1,2,2]
>>> array_shift($a)
=> 1
>>> $a
=> [2,3,4,5,1,1,2,2]
```

array_unshift ( array &$array [, mixed $... ] ) : int

```
>>> $a=[1,2,3,4,5,1,1,2,2]
=> [1,2,3,4,5,1,1,2,2]
>>> array_unshift($a,1)
=> 9
```

```
>>> $a
=> [1,2,3,4,5,1,1,2,2]
```

```
array_walk ( array &$array , callable $callback [, mixed $userdata = NULL ] ) : bool
```

```
>>> function cuadrado(&$x,$clave,[$userdata]) {
   $x=$x*$x;
}
>>> $a=[1,2,3,4,5,6]
=> [1,2,3,4,5,6]
>>> array_walk($a,'cuadrado')
=> true
>>> $a
=> [1,4,9,16,25,36]
```

```
array_pad ( array $array , int $size , mixed $value ) : array
```

```
>>> $a=[1,2,3]
=> [1,2,3]
>>> array_pad($a,6,5)
=> [1,2,3,5,5,5]
>>> array_pad($a,-6,5)
=> [5,5,5,1,2,3]
```

```
array_slice ( array $array , int $offset [, int $length = NULL [, bool $preserve_keys = false ]] ) : array
```

```
>>> $a=[1,2,3,4,5,6,7,8,9,10]
=> [1,2,3,4,5,6,7,8,9,10]
>>> array_slice($a,3,5)
=> [4,5,6,7,8]
>>> $b=[1,2,3,4,5]
=> [1,2,3,4,5]
>>> array_slice($b,2,2,true)
=> [2 => 3,3 => 4]
>>> array_slice($b,2)
=> [3,4,5]
```

```
implode ( string $glue , array $pieces ) : string
```

implode ( array $pieces ) : string

```
>>> $dias
=> ["lunes","martes","miércoles","jueves","viernes","sábado","domingo"]
>>> implode(", ",$dias)
=> "lunes, martes, miércoles, jueves, viernes, sábado, domingo"
>>> implode($dias)
=> "lunesmartesmiércolesjuevesviernessábadodomingo"
```

array_intersect ( array $array1 , array $array2 [, array $... ] ) : array

```
- Utiliza el operador ===
>>> array_intersect($a,$b)
=> [3 => 4,4 => 5]
>>> array_intersect($b,$a)
=> [4,5]
```

array_intersect_assoc ( array $array1 , array $array2 [, array $... ] ) : array

```
>>> $d1=['domingo'=>0, 'lunes'=>1,'martes'=>2,'miércoles'=>3]
=> ["domingo" => 0,"lunes" => 1,"martes" => 2,"miércoles" => 3]
>>> $d2=['martes'=>2,'miércoles'=>3,'jueves'=>4,'viernes'=>5]
=> ["martes" => 2,"miércoles" => 3,"jueves" => 4,"viernes" => 5]
>>> array_intersect_assoc($d1,$d2)
=> ["martes" => 2,"miércoles" => 3]

>>> $d1=['domingo'=>0, 'lunes'=>1,'martes'=>2,'miércoles'=>3]
=> ["domingo" => 0,"lunes" => 1,"martes" => 2,"miércoles" => 3]
>>> $d2=['martes'=>2,'miércoles'=>4,'jueves'=>5,'viernes'=>5]
=> ["martes" => 2,"miércoles" => 4,"jueves" => 5,"viernes" => 5]
>>> array_intersect_assoc($d1,$d2)
=> ["martes" => 2]

>>> $d1=['domingo'=>0, 'lunes'=>1,'martes'=>2,'miércoles'=>3]
=> ["domingo" => 0,"lunes" => 1,"martes" => 2,"miércoles" => 3]
>>> $d2=['martessss'=>2,'miércolessss'=>3,'juevessss'=>4,'viernessss'=>5]
=> ["martessss" => 2,"miércolessss" => 3,"juevessss" => 4,"viernessss" => 5]
```

```
>>> array_intersect_assoc($d1,$d2)
=> []
```

```
array_unique ( array $array [, int $sort_flags = SORT_STRING ] ) : array
```

```
>>> $a=[1,1,2,3,3,3,4,4,4,4,5,5,5,5,5]
=> [1,1,2,3,3,3,4,4,4,4,5,5,5,5,5]
>>> array_unique($a)
=> [0 => 1,2 => 2,3 => 3,6 => 4,10 => 5]
```

```
array_combine ( array $keys , array $values ) : array
```

```
>>> $dias=["lunes","martes","miércoles","jueves","viernes","sábado","domingo"]
=> ["lunes","martes","miércoles","jueves","viernes","sábado","domingo"]
>>> $valores=[1,2,3,4,5,6,7]
=> [1,2,3,4,5,6,7]
>>> array_combine($dias,$valores)
=> ["lunes" => 1,"martes" => 2,"miércoles" => 3,"jueves" => 4,"viernes" => 5,"sábado" => 6,"domingo" => 7]
```

```
array_reverse ( array $array [, bool $preserve_keys = FALSE ] ) : array
```

```
>>> $dias=["lunes","martes","miércoles","jueves","viernes","sábado","domingo"]
=> ["lunes","martes","miércoles","jueves","viernes","sábado","domingo"]
>>> $valores=[1,2,3,4,5,6,7]
=> [1,2,3,4,5,6,7]
>>> array_reverse($dias)
=> ["domingo","sábado","viernes","jueves","miércoles","martes","lunes"]
>>> array_reverse($valores)
=> [7,6,5,4,3,2,1]
>>> array_reverse(array_combine($dias,$valores))
=> ["domingo" => 7,"sábado" => 6,"viernes" => 5,"jueves" => 4,"miércoles" => 3,"martes" => 2,"lunes" => 1]
```

```
array_merge ( array $array1 [, array $... ] ) : array
```

```
>>> $dias
=> ["lunes","martes","miércoles","jueves","viernes","sábado","domingo"]
>>> array_merge($dias,$dias)
=> ["lunes","martes","miércoles","jueves","viernes","sábado","domingo","lunes","martes","miércoles","jueves","viernes","sábado","domingo"]
```

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

```
>>> range(1,5)
=> [1,2,3,4,5]
>>> range(1,20,5)
=> [1,6,11,16]
>>> range(20,1,-5)
=> [20,15,10,5]
```

shuffle ( array &$array ) : bool

```
>>> $cartas=range(1,48)
=> [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48]
>>> shuffle($cartas)
=> true
>>> $cartas
=> [40,44,41,21,6,43,4,11,37,34,7,10,29,5,45,23,36,9,39,47,1,24,26,31,16,35,8,19,42,28,48,25,27,15,13,14,32,20,3,30,2,18,12,46,17,22,33,38]

>>> $parejas=range(1,20)
=> [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
>>> $parejas2=array_merge($parejas,$parejas)
=> [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
>>> shuffle($parejas2)
=> true
>>> $parejas2
=> [14,19,6,17,14,18,18,12,11,13,2,8,13,4,20,5,7,3,9,4,15,8,17,10,10,11,3,16,1,15,2,5,1,12,20,9,19,7,6,16]
```

sort ( array &$array [, int $sort_flags = SORT_REGULAR ] ) : bool

Esta función ordena un array. Los elementos estarán ordenados de menor a mayor cuando la función haya terminado.

rsort ( array &$array [, int $sort_flags = SORT_REGULAR ] ) : bool

Esta función ordena un array en orden inverso (mayor a menor).

asort ( array &$array [, int $sort_flags = SORT_REGULAR ] ) : bool

Esta función ordena un array manteniendo la correlación de los índices del array con los elementos con los que están asociados. Esta función se utiliza principalmente para ordenar arrays asociativos en los que el orden es importante.

arsort ( array &$array [, int $sort_flags = SORT_REGULAR ] ) : bool

Esta función ordena un array de manera que los índices del array mantienen su correlación con los elementos del array asociados.
Es usado principalmente cuando se ordenan arrays asociativos cuando el orden de los elementos es importante.
Si dos miembros se comparan como iguales, su orden relativo en el array oredenado será indefinido.

ksort ( array &$array [, int $sort_flags = SORT_REGULAR ] ) : bool

Ordena un array por clave, manteniendo la correlación entre la clave y los datos. Esto es útil principalmente para arrays asociativos.

## Crypto

crypt ( string $str [, string $salt ] ) : string

```
>>>  crypt ( "123456","$6$1234567890123456")
=> "$6$1234567890123456$0KoX401sF24k4S/HUAS0ye1lBDIjEgY.e2aEZA8Skr.ew.R1xvvnOsLgj08bK.MdKEw9kxzRn4ipOUFh9RSUE/"
```

password_hash ( string $password , integer $algo [, array $options ] ) : string

**password_hash()** crea un nuevo hash de contraseña usando un algoritmo de hash fuerte de único sentido.
**password_hash()** es compatible con crypt(). Por lo tanto, los hash de contraseñas creados con crypt() se pueden usar con **password_hash()**.

```
>>> password_hash("123456", PASSWORD_DEFAULT)
=> "$2y$10$FrCCmcmIKsU1E23v2vOzbeZqXKD06cZDfwD2.3DONVT9n1HcVzgUG"
>>> password_hash("123456",  PASSWORD_BCRYPT)
=> "$2y$10$SMl5TteQch1QdvGTV5r0WeKYQSfIqe9BIjopwNrJnqLSebFD4kK46"
>>> password_hash("123456",  PASSWORD_ARGON2I)
=> "$argon2i$v=19$m=65536,t=4,p=1$Y29RdXRCQ3FqeE5ocmZkZw$PaoSa0/DcD85Ak6yLn0VOQ412TLVxYeQ/
Nvolhew/AM"
>>> password_hash("123456",  PASSWORD_ARGON2ID)
=>
"$argon2id$v=19$m=65536,t=4,p=1$M3RYSFlwUzdWZHM5TmV4SA$W6noyNJn01GCLPxqI07SAuzq9dcnRq1fMIbzstcJ8
lk"
```

password_verify ( string $password , string $hash ) : bool

Comprueba que el hash proporcionado coincida con la contraseña facilitada.
Observe que password_hash() devuelve el algoritmo, el coste y el salt como parte del hash devuelto. Por lo tanto, toda la información que es necesaria para verificar el hash está incluida. Esto permite a la función de verificación comprobar el hash sin la necesidad de almacenar por separado la información del salt o del algoritmo.
Esta función es segura contra ataques basado en tiempo.
>>>
password_verify("123456","$6$1234567890123456$0KoX401sF24k4S/HUAS0ye1lBDIjEgY.e2aEZA8Skr.ew.R1xvvnOsLgj08bK.MdKEw9kxzRn4ipOUFh9RSUE/")
=> true
Mejor usar comillas simples para el hash ya que contiene el símbolo del dólar y en el siguiente ejemplo se pensaría que hay una variable que sustituir y daría problemas:
>>> password_verify("123456",'$2y$10$FrCCmcmIKsU1E23v2vOzbeZqXKD06cZDfwD2.3DONVT9n1HcVzgUG')
=> true
>>> password_verify("123456",'$2y$10$SMl5TteQch1QdvGTV5r0WeKYQSfIqe9BIjopwNrJnqLSebFD4kK46')
=> true
>>> password_verify("123456",'$argon2i$v=19$m=65536,t=4,p=1$Y29RdXRCQ3FqeE5ocmZkZw$PaoSa0/DcD85Ak6yLn0VOQ412TLVxYeQ/Nvolhew/AM')
=> true
>>>
password_verify("123456",'$argon2id$v=19$m=65536,t=4,p=1$M3RYSFlwUzdWZHM5TmV4SA$W6noyNJn01GCLPxqI07SAuzq9dcnRq1fMIbzstcJ8lk')
=> true

## Misceláneas

move_uploaded_file ( string $filename , string $destination ) : bool

Esta función intenta asegurarse de que el archivo designado por filename es un archivo subido válido (lo que significa que fue subido mediante el mecanismo de subida HTTP POST de PHP). Si el archivo es válido, será movido al nombre de archivo dado por destination.

El orden de comprobación es especialmente importante si hay cualquier posibilidad de que cualquier cosa hecha con los archivos subidos pueda revelar su contenido al usuario, o incluso a otros usuarios en el mismo sistema.

## JSON

json_decode ( string $json [, bool $assoc = false [, int $depth = 512 [, int $options = 0 ]]] ) : mixed

```
$jsonCurso='{ "curso" : \
...     { \
...        "alumnos": [  \
...           { \
...              "alumno" : { \
...                 "nombre": "Luis",\
...                 "ap1": "Pérez",\
...                 "ap2": "Martínez",\
...                 "dir": "C/ La Loma, 33",\
...                 "telef":956778899\
...              }\
...           },\
...           {\
...              "alumno" : { \
...                 "nombre": "Luisa",\
...                 "ap1": "López",\
...                 "ap2": "Martínez",\
...                 "dir": "C/ La Colina, 35",\
...                 "telef":956778899\
...              }\
...           }\
...        ]\
...     }\
```

```
... }'
=> """
  { "curso" : \n
    { \n
      "alumnos": [  \n
        { \n
          "alumno" : { \n
            "nombre": "Luis",\n
            "ap1": "Pérez",\n
            "ap2": "Martínez",\n
            "dir": "C/ La Loma, 33",\n
            "telef":956778899\n
          }\n
        },\n
        {\n
          "alumno" : { \n
            "nombre": "Luisa",\n
            "ap1": "López",\n
            "ap2": "Martínez",\n
            "dir": "C/ La Colina, 35",\n
            "telef":956778899\n
          }\n
        }\n
      ]\n
    }\n
  }
  """
>>> $alu=json_decode($jsonCurso)
=> {#2541
    +"curso": {#2544
```

```
    +"alumnos": [
      {#2543
        +"alumno": {#2548
          +"nombre": "Luis",
          +"ap1": "Pérez",
          +"ap2": "Martínez",
          +"dir": "C/ La Loma, 33",
          +"telef": 956778899,
        },
      },
      {#2542
        +"alumno": {#2549
          +"nombre": "Luisa",
          +"ap1": "López",
          +"ap2": "Martínez",
          +"dir": "C/ La Colina, 35",
          +"telef": 956778899,
        },
      },
    ],
  },
}
>>> $alu->curso->alumnos[0]->alumno->nombre
=> "Luis"
>>> $alu=json_decode($jsonCurso,true)
=> [
  "curso" => [
    "alumnos" => [
      [
        "alumno" => [
```

```
            "nombre" => "Luis",
            "ap1" => "Pérez",
            "ap2" => "Martínez",
            "dir" => "C/ La Loma, 33",
            "telef" => 956778899,
          ],
        ],
        [
          "alumno" => [
            "nombre" => "Luisa",
            "ap1" => "López",
            "ap2" => "Martínez",
            "dir" => "C/ La Colina, 35",
            "telef" => 956778899,
          ],
        ],
      ],
    ],
  ]
>>> $alu['curso']['alumnos'][0]['alumno']['nombre']
=> "Luis"
```

```
json_encode ( mixed $value [, int $options = 0 [, int $depth = 512 ]] ) : string
```

Devuelve un string con la representación JSON de **value**.
Es conveniente usar como segundo parámetro JSON_UNESCAPED_UNICODE

```
>>> $datos
=> [
    "nombre" => "Luis",
    "ap1" => "Pérez",
```

```
    "ap2" => "López",
    "usuario" => "Luislo",
    "pass" => "$2y$10$buksU.9Vkgg/gR0mxA40pOq.3HnoJEF.YOpBReoTzIFywqwr15hwu",
    "email" => "luislo@gmail.com",
  ]
>>> json_encode($datos)
=> "{"nombre":"Luis","ap1":"P\u00e9rez","ap2":"L\u00f3pez","usuario":"Luislo","pass":"$2y$10$buksU.9Vkgg\/
gR0mxA40pOq.3HnoJEF.YOpBReoTzIFywqwr15hwu","email":"luislo@gmail.com"}"
>>> json_encode($datos,JSON_UNESCAPED_UNICODE)
=> "{"nombre":"Luis","ap1":"Pérez","ap2":"López","usuario":"Luislo","pass":"$2y$10$buksU.9Vkgg\/
gR0mxA40pOq.3HnoJEF.YOpBReoTzIFywqwr15hwu","email":"luislo@gmail.com"}"
```