

## FUNZIONI DI STRINGHE

### Funzioni *Print* / *Print\_r* / *Echo*

Tali funzioni servono per printare una stringa ( print o echo ) o un array ( print\_r ); come unico parametro è richiesta la stringa racchiusa tra i due apici oppure la variabile assegnata ad una stringa oppure la variabile array.

```
print("Ciao a tutti");
```

```
print_r($_POST);
```

Es.:

```
<html>
<body>
<?php
$utente=array (
  "dati anagrafici"=>array(
    "nome"=>"Mario",
    "cognome"=>"verdi",
    "eta"=>24),
  "recapiti"=>array(
    "indirizzo"=>"V.Milano",
    "civico"=>10,
    "cap"=>00179)
);
print_r($utente);
$br="<BR><BR><BR><BR>";
print($br.$utente["dati
anagrafici"]["nome"].$br.$utente["recapiti"]["cap"]);
?>
</body>
</html>
```

Di risultato:

```
Array ( [dati anagrafici] => Array ( [nome] => Mario [cognome] => verdi [eta] => 24 ) [recapiti] => Array ( [indirizzo]
=> V.Milano [civico] => 10 [cap] => 15 ) )
```

*Mario*

15

### Funzione *Substr*

Tale funzione ritorna una sotto-stringa, dati due delimitatori:

```
substr("stringa",delimitatori);
```

Es.:

```
<?php
```

```
$rest=substr("abcdef ",1);
$rest=substr("abcdef ",1,3);
$rest=substr("abcdef ",0,4);
$rest=substr("abcdef ",0,-1);
?>
```

Di risultato:

```
bcdef
bcd
abcd
abcde
```

La numerazione della stringa in questione è assegnata in questo modo:

```
a b c d e f
123456
```

Il primo numero delimitatore, quindi, indica il numero precedente a quello di inizio visualizzazione, mentre il secondo l'ultimo valore da visualizzare.

### Funzione *str\_repeat*

Tale funzione serve per ripetere una stringa per un certo numero di volte passatogli come parametro della funzione stessa:

```
str_repeat("valore",num);
```

Es.:

```
<?php
echo str_repeat("=",10);
?>
```

Di risultato:

```
= = = = = = = = = =
```

### Funzione *Explode*

Tale funzione serve per dividere una stringa in un array dato un separatore contenuto nella stringa stessa; i suoi parametri sono la stringa e il separatore:

```
explode("separatore",$stringa);
```

Es.:

```
<?php
$str="Mario;Gentili";
$c=explode(";", $str);
print_r($c);
print($c[0]);
print($c[1]);
?>
```

Di risultato:

```
Array ( [0] => Mario [1] => Gentili ) MarioGentili
```

### Funzione *implode*

Tale funzione serve per estrarre un array in una stringa, dato un separatore:

```
implode("separatore",$array);
```

Es.:

```
<?php
$array=array("lastname","email","nome");
$a=implode(",",$array);
print $a;
?>
```

Di risultato:

*lastname,email,nome*

### Funzione *strlen*

Tale funzione serve per ottenere la lunghezza in caratteri di una stringa:

```
strlen("stringa");
```

Es.:

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

Di risultato:

6

## FUNZIONE "DATE"

La funzione `date()` serve per assegnare la data o l'orario ad una variabile:

```
date( string formato [, int timestamp]);
```

I seguenti caratteri sono utilizzati nella *stringa formato*:

- a - "am" o "pm"
- A - "AM" o "PM"
- B - Swatch Internet time
- d - giorno del mese, 2 cifre senza tralasciare gli zero; i.e. "01" a "31"
- D - giorno della settimana, testuale, 3 lettere; i.e. "Fri"
- F - mese, testuale, long; i.e. "January"
- g - ora, formato a 12-ore senza eventuali zero; i.e. "1" a "12"
- G - ora, formato a 24-ore senza eventuali zero; i.e. "0" a "23"
- h - ora, formato a 12-ore; i.e. "01" a "12"
- H - ora, formato a 24-ore; i.e. "00" a "23"

- i - minuti; i.e. "00" a "59"
- l (i grande) - "1" se c'è l'ora legale, "0" altrimenti.
- j - giorno del mese senza eventuali zero; i.e. "1" a "31"
- l ('L' piccola) - giorno della settimana, testuale, long; i.e. "Friday"
- L - valore booleano per stabilire se è un anno bisestile; i.e. "0" o "1"
- m - mese; i.e. "01" a "12"
- M - mese, testuale, 3 lettere; i.e. "Jan"
- n - mese senza eventuali zero; i.e. "1" a "12"
- O - Differenza in ore dal fuso orario Greenwich; i.e. "+0200"
- r - Data formattata RFC 822; i.e. "Thu, 21 Dec 2000 16:01:07 +0200" (aggiunto nel PHP 4.0.4)
- s - secondi; i.e. "00" a "59"
- S - Suffisso ordinale Inglese per i giorni del mese, 2 caratteri; i.e. "th", "nd"
- t - numero di giorni del mese dato; i.e. "28" a "31"
- T - Fuso orario di questo computer; i.e. "MDT"
- U - secondi dall'epoca since the epoch
- w - giorno della settimana, numerico, i.e. "0" (Domenica) a "6" (Sabato)
- W - ISO-8601 Numero della settimana dell'anno, le settimane iniziano il lunedì (aggiunto in PHP 4.1.0) (Sabato)
- Y - anno, 4 cifre; i.e. "1999"
- y - anno, 2 cifre; i.e. "99"
- z - giorno dell'anno; i.e. "0" a "365"
- Z - Fuso orario in secondi (i.e. "-43200" a "43200"). Il fuso orario ad ovest dell'UTC è sempre negativo, e per quelli ad est è sempre positivo.

Es.:

```
/* Today is March 10th, 2001, 5:16:18 pm */
```

```
$today = date ("F j, Y, g:i a");
$today = date ("m.d.y");
$today = date ("j, n, Y");
$today = date ("Ymd");
$today = date ('h-i-s, j-m-y, it is w Day z ');
$today = date ('!lt !ls !th!e JS !d!ay.!');
$today = date ("D M j G:i:s T Y");
$today = date ('H:m:s !m !ls! !ml!l!th');
$today = date ("H:i:s");
```

Di risultato, rispettivamente:

```
March 10, 2001, 5:16 pm
03.10.01
10, 3, 2001
20010310
05-16-17, 10-03-01, 1631 1618 6 Fripm01
It is the 10th day.
Sat Mar 10 15:16:08 MST 2001
17:03:17 m is month
17:16:17
```

## FUNZIONE DI VARIABILE Array

Funzione - Esempio

Spiegazione

```
isset($array);
!isset($array);
```

Mi dà *True* se nell'array c'è qualcosa. Il **!** sta per NOT; quindi con il punto esclamativo davanti, mi restituisce *True* se l'array non è

settato e cioè se è vuoto. Quest'ultima funzione è equivalente alla dichiarazione: `$array = = NULL`

```
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$risultato=count($a);
```

Restituisce il numero di elementi dell'array ( 3 nell'esempio ).

```
$ar1=array("verde","blu","rosso");
$ar2=array("verde","giallo","rosso");
$risultato=array_diff($ar1,$ar2);
```

Restituisce un array contenente tutti i valori del primo array che non sono contenuti nel secondo ( o negli altri, se sono più di due ) ( `$risultato` sarà quindi il seguente array: `array("blu")` ).

```
$input_array=array("PriMo"=>1,"SecOndO"=>4);
print_r(array_change_key_case($input_array,
CASE_UPPER));
```

Trasforma tutte le chiavi in minuscolo ( `CASE_LOWER` ) o in maiuscolo ( `CASE_UPPER` ). In questo caso il risultato sarà: `Array ([PRIMO]=>1 [SECONDO]=>4)`

```
$a=array_fill(5, 6,'banana');
print_r($a);
```

Riempie l'array con `num` ( II parametro ) elementi inizializzati con il valore del parametro `valore` ( III parametro ), e con le chiavi che partono dal valore del parametro `start_index` ( I parametro ). Il risultato sarà:

```
Array(
  [5] => banana
  [6] => banana
  [7] => banana
  [8] => banana
  [9] => banana
  [10] => banana
)
```

```
array_key_exists("valore_chiave",$array);
```

Restituisce `true` se il parametro chiave esiste nell'array; è in genere usato nei cicli `IF`.

```
$ar1=array("colore"=>"rosso",2,4);
$ar2=array("a","b","colore"=>"verde","forma"=>"trapezio",4);
$risultato=array_merge($ar1,$ar2);
```

Fonde gli elementi di due o più array in modo che i valori di un array siano accodati a quelli dell'array precedente. Restituisce l'array risultante: `Array (`

```
  [colore] => verde
  [0] => 2
  [1] => 4
  [2] => a
  [3] => b
  [forma] => trapezio
  [4] => 4
)
```

Se gli array in input hanno le stesse chiavi stringa, allora l'array "mergiato" coinciderà con il secondo; visto che se ci sono due chiavi uguali nei due array di input, il valore corrispondente del secondo array sovrascrive quello del primo relativo sempre alla stessa chiave.

```
$ar1=array("colore"=>array("preferito"=>"rosso"), 5);
$ar2=array(10,"colore"=>array("preferito"=>"verde", "blu"));
$risultato=array_merge_recursive($ar1, $ar2);
```

Fonde gli elementi di due o più array in modo tale che i valori di un array siano accodati all'array precedente. Restituisce l'array risultante. La variabile `$risultato` sarà:

```

Array (
  [colore] => Array (
    [preferito] => Array(
      [0] => rosso
      [1] => verde
    )
    [0] => blu
  )
  [0] => 5
  [1] => 10 )

```

**\$ris=array\_search("valore",\$array);**

Cerca il valore nell'array e ritorna *True* se lo trova.

**\$srr=array\_sum(\$array\_int,\$array\_int2);**

Restituisce la somma dei valori dei due array di interi o float in un altro array di interi o float.

**\$sar=array\_unique(\$array);**

Mi restituisce lo stesso array senza i valori duplicati.

**\$foo=array ("bob","fred","jussi","jouni","egon","marliese");  
\$bar = each(\$foo);  
print\_r(\$bar);**

Restituisce la corrente coppia chiave/valore dell'array ed incrementa il puntatore interno dell'array. Questa coppia è restituita in un array di quattro elementi, con le chiavi 0, 1, key, and value. Gli elementi 0 e key contengono il nome della chiave dell'elemento dell'array, mentre 1 e value contengono i dati. Se il puntatore interno dell'array punta oltre la fine dei contenuti dell'array, each() restituisce *FALSE*.  
Il risultato sarà:

```

Array
(
  [1] => bob
  [value] => bob
  [0] => 0
  [key] => 0
)

```

**\$val = current(\$array);**

Restituisce l'elemento che è attualmente puntato dal puntatore interno. Se il puntatore è alla fine dell'array, restituisce *False*.

**\$sar=end(\$array);**

Avanza il puntatore all'ultimo valore dell'array.

**\$sar=in\_array("valore",\$array);**

Cerca il valore nell'array e restituisce *true* se lo trova. Se viene passato come terzo parametro il Tipo di valore, esso confronterà anche esso.

**\$sar=next(\$array);**

Restituisce l'elemento che è attualmente "puntato" dal puntatore interno ed avanza il puntatore alla chiave successiva.

**\$sar=prev(\$array);**

Restituisce l'elemento che è attualmente "puntato" dal puntatore interno ed arretra il puntatore alla chiave precedente.

```
$ar=reset($array);
```

Porta il puntatore al primo valore dell'array.

## ALTRE FUNZIONI

Funzione - Esempio	Spiegazione
<pre>include("nome_file.php"); include("nome_file.html");</pre>	<p>Mi consente di copiare ed incollare ( e quindi eseguire ) il contenuto del file "nome_file.php" o "nome_file.html" nel file originario contenente l'include. Posso decidere di includere sia file tutti in php o file scritti tutti in html o misti; rispettivamente dovrò usare i comandi mostrati affianco. Ovviamente il file incluso, se è scritto in HTML, non deve contenere Tag quali HTML o BODY , ma solo quelli riguardanti ad esempio una tabella, una semplice frase o altro... <u><a href="#">Più avanti potrete vedere qualche esempio in dettaglio.</a></u></p>
<pre>exit();</pre>	<p>Anche questa è una funzione nativa del php, che vi permette di non eseguire tutto il suo successivo contenuto. E' molto usata per verificare il punto in cui si verifica un errore nello script.</p>
<pre>header("Location: <a href="http://www.php.net/">http://www.php.net/</a>");</pre>	<p>Serve per inviare header http e cioè ridirezionare il browser al sito di PHP, in questo caso; può essere usata insieme alla funzione exit, per non far eseguire il codice php scritto dopo l'header. "Location" non trasmette solo un header al browser, ma anche un REDIRECT con codice di stato (302).</p>
<pre>unset(\$var); unset(\$_POST['key']);</pre>	<p>Serve per cancellare una variabile oppure un campo di un array.</p>
<pre>realpath(".");</pre>	<p>Serve per ottenere il percorso locale sul pc fino al punto in cui si trova il file che contiene tale istruzione; è usata per stabilire dei percorsi universali fino alla index.php insieme alle define dei percorsi delle cartelle contenenti le immagini, files php, html ( vedi esempio successivo e la figura di sotto per l'alberatura spesso usata per realizzare un sito con php )...</p>
<pre>define("Default","nome_home_page"); include("HTML/.Default.".html");</pre>	<p>Serve per definire una costante; nell'esempio affianco si definisce la pagina html di default. <b>Se voleste, invece, definire come costanti tutti i percorsi a partire dalla index,</b> allora all'inizio della pagina index.php potreste definire le seguenti costanti ( sempre se la struttura interna del vostro sito è come quella mostrata nella figura seguente ):</p> <pre>define("ROOT_DIR", realpath(". ")); define("TPL_DIR", ROOT_DIR."/tpl/"); define("INS_DIR", ROOT_DIR."/ins"); define("IMG_DIR", "img/"); define("PHP_DIR", ROOT_DIR."/php/"); define("DWNL_DIR", ROOT_DIR."/download/");  ... include(HTML_DIR."nome_file.html");</pre>

E' ovvio che ogni volta che richiamate un file dovete usare nel percorso la relativa costante; questo metodo vi consente di facilitare eventuali cambiamenti nell'alberatura interna del vostro sito. Ad esempio se voleste spostare la cartella "Download" dentro la "Ins", vi basterebbe cambiare solo il percorso della define nella index e non in tutti i percorsi delle vostre pagine che vengono incluse nella index.  
N.B.: tutto ciò funziona solo se impostate il vostro sito su una struttura che include ogni volta i vari contenuti ( più avanti capirete meglio il perchè ).



## DEFINIRE PROPRIE FUNZIONI

Il PHP consente al programmatore di scrivere delle funzioni proprie, da lui definite; ad esempio una funzione può essere definita usando la seguente sintassi:

```
function nome_func()
{
    istruzione;
}
```

Così abbiamo definito una funzione, *nome\_func*, che farà un certo numero di istruzioni da noi specificate; per poterla usare dobbiamo richiamarla così:

```
nome_func(eventuali parametri);
```

Per poter ottenere un valore di ritorno dalla funzione, devo definire una variabile di ritorno ed usare il comando **return** nella definizione della funzione stessa ed associare una variabile all'istanza della funzione.

```
$valore_ritornato = nome_func(eventuali parametri);
```

Ecco un esempio:

```
<?php
$resultato=somma(1,3); // ISTANZA DELLA FUNZIONE somma
print("questa è la somma: ".$resultato."<br>");

$resultato=differenza(1,3);
print("questa è la sottrazione: ".$resultato."<br>");

$resultato=moltiplicazione(1,3);
print("questa è la moltiplicazione: ".$resultato."<br>");

$resultato=divisione(1,3);
print("questa è la divisione: ".$resultato."<br>");

function somma($a,$b)
{
    $ret=$a+$b;
    return $ret;
}
function differenza($a,$b)
```

```
{
  $ret=$a-$b;
  return $ret;
}
function moltiplicazione($a,$b)
{
  $ret=$a*$b;
  return $ret;
}
function divisione($a,$b)
{
  $ret=$a/$b;
  return $ret;
}
?>
```

Ad esempio la funzione "somma" prende come parametri due numeri al momento dell'istanza, mentre nella sua definizione sotto, ha come parametri due variabili ad ognuna delle quali viene assegnato un numero: \$a sarà quindi = 1, mentre \$b = 3. Il valore ritornato \$ret sarà automaticamente assegnato alla variabile \$risultato al momento dell'istanza; se non si mettesse "\$risultato =" sarebbe sbagliato e non potrei quindi gestire il risultato della funzione operazione.

Di output:

```
questa è la somma: 4
questa è la sottrazione: -2
questa è la moltiplicazione: 3
questa è la divisione: 0.33333333333333
```