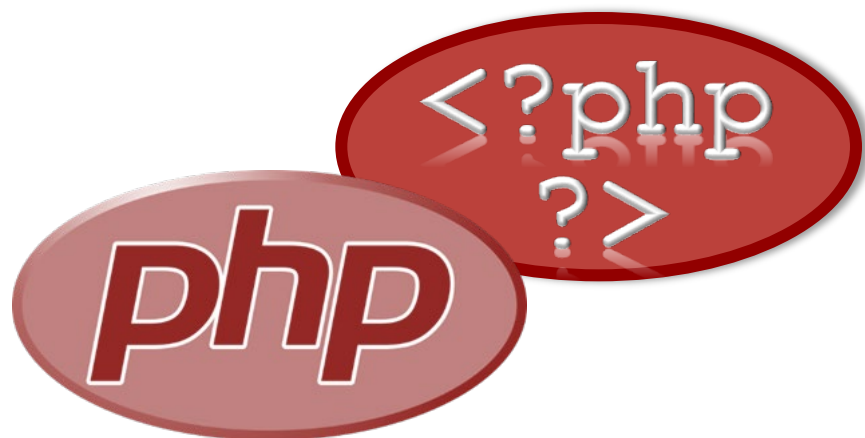
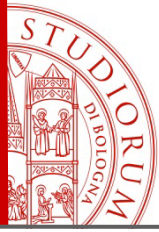


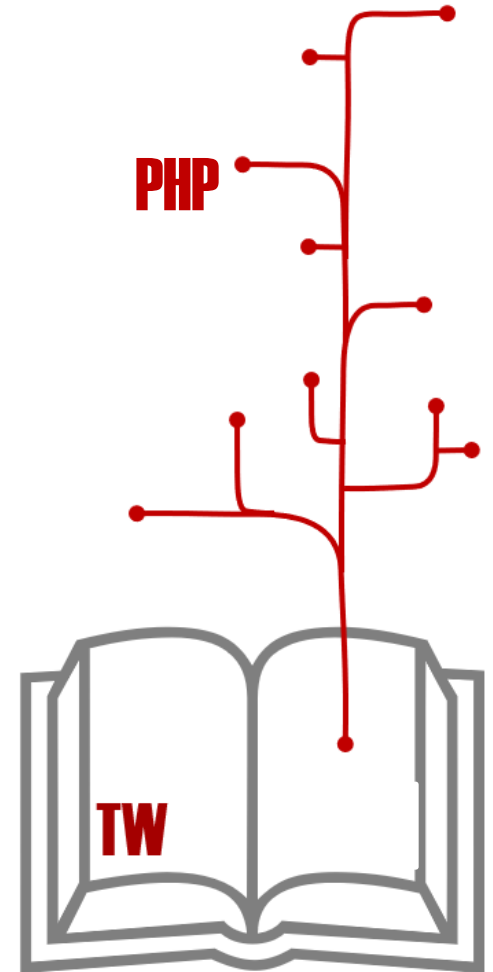
# PHP Sintassi Estesa

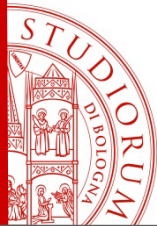




# Argomenti

- Sintassi PHP





# Variabili

- Tutte le variabili iniziano con il carattere \$

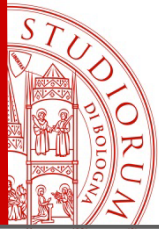
`$nome`

- È debolmente tipizzato
  - Possiamo associare alla stessa variabile più tipi di dato
    - Automaticamente PHP converte la variabile nel tipo opportuno

```
$qualcosa = 3;
```

```
$qualcosa = true;
```

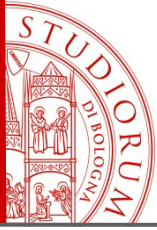
```
$qualcosa = "ciao";
```



# Regole per definire le variabili

---

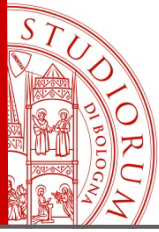
- Le variabili devono iniziare con una lettera o con il carattere \_ (underscore)
- Una variabile non può iniziare con un numero
- Una variabile può solo contenere caratteri alfanumerici e l'underscore (A-z, 0-9, and \_)
- Il nome delle variabili è case-sensitive
  - **\$nome** e **\$NOME** sono due differenti variabili!!!



# Scope delle variabili

---

- Local
  - Una variabile dichiarata in una funzione ha visibilità locale e può essere usata solo in quella funzione
- Global
  - **NB:** Una variabile dichiarata fuori da una funzione ha scopo globale e può essere usata solo fuori dalla funzione
  - Per accederci da dentro una funzione, usare la parola chiave ***global***
- Static
  - Variabili che non vogliamo vengano cancellate o ri-inizializzate dopo l'uso

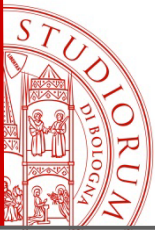


# Esempio: variabile locale

```
<?php
function test() {
    $x = 5; // visibilità locale
    echo "<p>Variabile x dentro alla
        funzione: $x</p>";
}
test();

// se si usa x fuori dalla funzione si
    genera un errore
echo "<p>Variabile x fuori dalla
    funzione: $x</p>";
?>
```

Variabile x dentro alla funzione: 5  
Variabile x fuori dalla funzione:



# Esempio: variabile globale

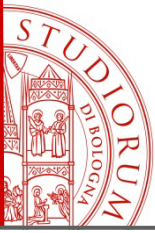
```
<?php
$x = 5; // visibilità globale

function test() {
    //usare x dentro questa funzione genera un errore
    echo "<p>Variabile x dentro la funzione: $x</p>";
}
test();

echo "<p>Variabile x fuori la funzione: $x</p>";
?>

</body>
</html>
```

Variabile x dentro la funzione:  
Variabile x fuori la funzione: 5



# Esempio: variabile globale

```
<?php
```

```
$x = 5; // visibilità globale
```

```
function test() {
```

```
    global $x;
```

```
    echo "<p>Variabile x dentro la funzione: $x</p>";
```

```
}
```

```
test();
```

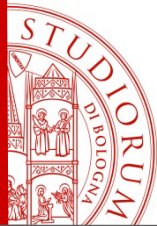
```
echo "< p>Variabile x fuori la funzione $x</p>";
```

```
?>
```

Variabile x dentro la funzione: 5

Variabile x fuori la funzione: 5

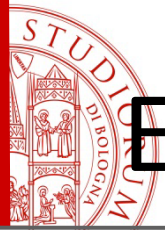




# Modificare una variabile globale

---

- Se si vuole modificare una variabile globale da dentro una funzione occorre accedere al valore attraverso l'array che contiene tutte le variabili globali
    - **\$GLOBALS['nome\_della\_variabile']**
- NB:** L'indice dell'array è il nome della variabile



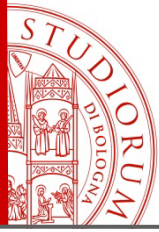
# Esempio: modifica variabile globale

15

```
<?php
$x = 5;
$y = 10;

function test() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

test();
echo $y;
?>
```

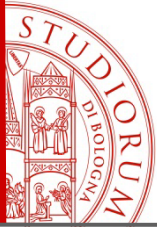


# Esempio: variabile statica

```
<?php
function test() {
    static $x = 0;
    echo $x;
    $x++;
}
```

```
test();
echo "<br>";
test();
echo "<br>";
test();
?>
```

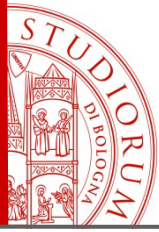
0  
1  
2



# Tipi di dato

- Boolean
- Integer
- Float (floating point numbers o double)
- String
- Array
- Object
- NULL

NB: Esiste la funzione *var\_dump()* che restituisce il tipo della variabile



# Variabili booleane

```
<?php
```

```
$vero = true;
```

```
$falso = false;
```

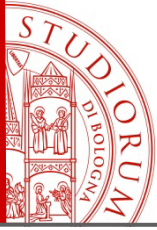
```
$vero = 1 && 1;
```

```
$falso = 1 && 0;
```

```
$vero = 1 || 0;
```

```
$falso = 0 || 0;
```

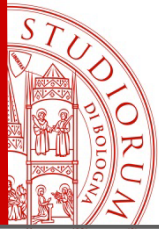
```
?>
```



# Valori interi

```
<?php
    $intero = 1;
    $intero = 1231231;
    $intero = -234224;

    $intero = 1 + 1;
    $intero = 1 - 1;
    $intero = 3 * 4;
?>
```



# Numeri in virgola mobile

```
<?php
```

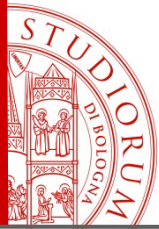
```
$float = 10.3;
```

```
$float = -33.45;
```

```
$float = 6.1e6; //  $6,1 * 10^6 \Rightarrow 6.100.000$ 
```

```
$float = 3E-7; //  $3 * 10^{-7} \Rightarrow 3/10.000.000 = 0,0000003$ 
```

```
?>
```



# Stringhe

```
<?php
```

```
$stringa = "ciao a tutto il mondo";
```

```
$stringa = 'ciao a tutto il mondo';
```

```
$stringa = "lo studente dice 'ciao a tutto il mondo'";
```

```
$stringa = 'lo studente dice "ciao a tutto il mondo"';
```

Stringhe errate!!!!

```
// stringhe non valide perché contengono lo stesso carattere di apertura
```

```
// all'interno della stringa
```

```
$stringa_non_valida = "lo studente dice "ciao a tutto il mondo"";
```

```
$stringa_non_valida = 'lo studente dice 'ciao a tutto il mondo'";
```

```
// utilizzare il backslash, \, per impiegare il carattere di apertura all'interno della stringa
```

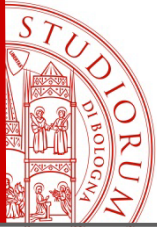
```
// tale operazione viene definita escaping delle stringhe
```

```
$stringa_valida = "lo studente dice \"ciao a tutto il mondo\"";
```

```
$stringa_valida = 'lo studente dice \'ciao a tutto il mondo\'";
```

```
?>
```

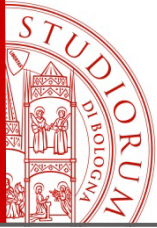




# Stringhe e variabili

---

```
<?php
    $nome = "Batman";
    $stringa = "ciao $nome, come stai?";
    //ciao Batman, come stai?
    $stringa = 'ciao $nome, come stai?';
    //ciao $nome, come stai?
?>
```



# Stringhe e variabili

Se si usano le doppie virgolette allora l'interprete convertirà le variabili nel loro valore invece di trattarle come stringhe

```
<?php
```

```
$nome = "Batman";
```

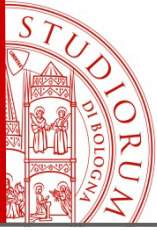
```
$stringa = "ciao $nome, come stai?";
```

```
//ciao Batman, come stai?
```

```
$stringa = 'ciao $nome, come stai?';
```

```
//ciao $nome, come stai?
```

```
?>
```



# echo

- Permette di avere in output del testo
  - Stampa del testo a video
  - Un'alternativa è *print*

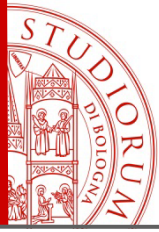
```
<?php
```

```
$x = 5;
```

```
$y = 4;
```

```
echo $x + $y;
```

```
?>
```

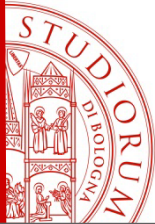


# Esempio echo

```
<?php
$txt = "PHP";
echo "Mi piace programmare in $txt!<br>";
//il . indica concatenazione di stringhe
echo "Mi piace programmare in " . $txt . "!<br>";
//la , indica che stiamo passando più parametri a echo
echo "Mi piace programmare in " , $txt , "!<br>";

?>
```

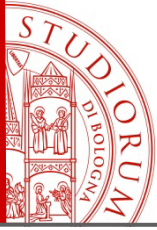
Mi piace programmare in PHP  
Mi piace programmare in PHP  
Mi piace programmare in PHP



# Manipolazione stringhe

---

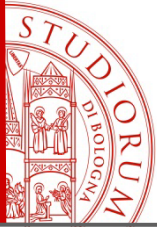
- Lunghezza di una stringa  
`echo strlen("Hello world!"); // outputs 12`
- Numero di parole in una stringa  
`echo str_word_count("Hello world!"); // outputs 2`
- Reverse una stringa  
`echo strrev("Hello world!"); // outputs !dlrow olleH`
- Cercare una specifica stringa (restituisce l'indice del primo match)  
`echo strpos("Hello world!", "world"); //outputs 6`  
NB: Il primo carattere della stringa è in posizione 0



# Manipolazione stringhe

---

- Rimpiazzare una stringa con un'altra  
`echo str_replace("world", "Batman", "Hello world!");` //outputs Hello Batman!
- E tanto altro:  
[http://www.w3schools.com/php/php\\_ref\\_string.asp](http://www.w3schools.com/php/php_ref_string.asp)



# Array

- Creazione

```
$gelati = array("cornetto", "ghiacciolo", "ricoperto");
```

- Oppure

```
$gelati[0] = "cornetto";
```

```
$gelati[1] = "ghiacciolo";
```

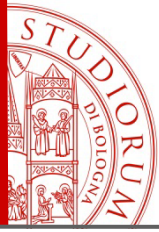
```
$gelati[2] = "ricoperto";
```

- Accedere ad un elemento di un array

```
echo $gelati[0] ;
```

- Lunghezza array (numero elementi nell'array)

```
echo count($gelati);
```



# Array

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<?php
```

```
$gelati = array("cornetto", "ghiacciolo",  
"ricoperto");
```

```
var_dump($gelati);
```

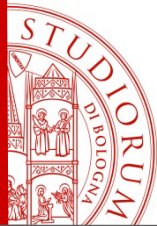
```
?>
```

```
</body>
```

```
</html>
```

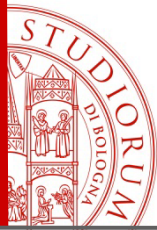
```
array(3) {  
[0]=> string(8) "cornetto"  
[1]=> string(10) "ghiacciolo"  
[2]=> string(9) "ricoperto" }
```





# Array associativi

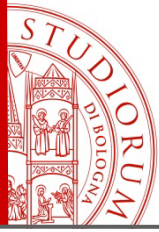
- Gli array associativi sono array i cui indici possono essere numerici o stringhe.
- Ci sono due modi per creare un array associativo:
  - `$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");`
  - Oppure
    - `$age['Peter'] = "35";`
    - `$age['Ben'] = "37";`
    - `$age['Joe'] = "43";`
- Accedere ad un elemento di un array
  - `echo $age['Peter'];`



# Ordinamento elementi array

---

- Gli elementi in un array possono essere ordinati in vari modi
  - `sort()` – ordine ascendente/crescente
  - `rsort()` – ordine discendente/decrescente
  - `asort()` – ordina gli array associativi in ordine crescente in base al valore
  - `ksort()` – ordina gli array associativi in ordine crescente in base alla chiave
  - `arsort()` - ordina gli array associativi in ordine decrescente in base al valore
  - `krsort()` - ordina gli array associativi in ordine decrescente in base alla chiave

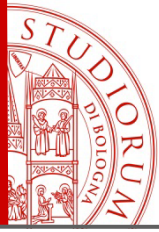


# Esempio ordine crescente

```
<?php
$gelati = array("cornetto", "ricoperto",
"ghiacciolo");
sort($gelati);

$clength = count($gelati);
for($x = 0; $x < $clength; $x++) {
    echo $gelati[$x];
    echo "<br>";
}
```

cornetto  
ghiacciolo  
ricoperto



# Esempio ordine decrescente

```
<?php
```

```
$numeri = array(4, 6, 2, 22, 11);
```

```
rsort($numeri);
```

```
?>
```

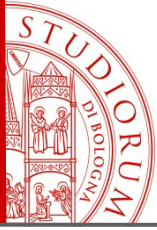
22

11

6

4

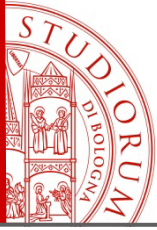
2



# Funzioni sugli array

---

- Guardate qui per avere qualsiasi dettaglio sulle funzioni su array
  - [http://www.w3schools.com/php/php\\_ref\\_array.asp](http://www.w3schools.com/php/php_ref_array.asp)



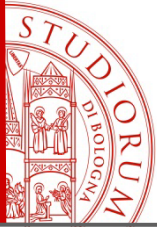
# Oggetti in PHP

```
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}
```

```
// create an object
$herbie = new Car();
```

```
// show object properties
echo $herbie->model;
?>
```

VW

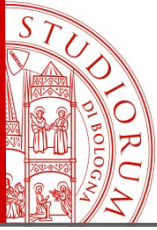


# Valore NULL

- Una variabile con valore NULL è una variabile con nessun valore associato

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

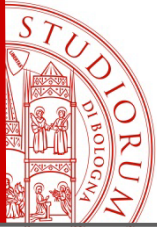
NULL



# Constanti

- Una costante è un nome per un valore che non cambierà (per definizione di costante)
- Per creare una costante bisogna usare **define(name, value, case-insensitive)**
- Dove
  - name: nome specifico della costante
  - value: specifica il valore della costante
  - case-insensitive (opzionale): specifica se il nome della costante dovrà essere case-insensitive. Di default è falso
- Sono automaticamente **globali** e possono essere usate ovunque





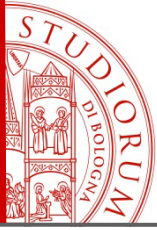
# Esempio: costanti

```
<!DOCTYPE html>
<html>
<body>

<?php
// case-sensitive constant name
define("GREETING", "Benvenuti alla lezione di PHP!");
echo GREETING;
?>

</body>
</html>
```

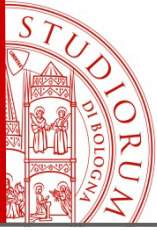
Benvenuti alla  
lezione di PHP!



# Operatori

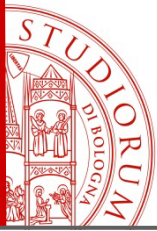
---

- Operatori aritmetici
- Operatori di assegnamento
- Operatori di confronto
- Operatori di incremento/decremento
- Operatori logici
- Operatori di stringa
- Operatori di array



# Operatori aritmetici

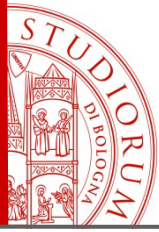
Operatore	Nome	Esempio	Risultato
+	Addizione	$\$x + \$y$	Somma di x e y
-	Sottrazione	$\$x - \$y$	Differenza tra x e y
*	Moltiplicazione	$\$x * \$y$	Prodotto di x e y
/	Divisione	$\$x / \$y$	Quoziente di x e y
%	Modulo	$\$x \% \$y$	Resto di x diviso y
**	Elevazione ad esponente	$\$x ** \$y$	Elevazione di x alla potenza y



# Operatori di assegnamento

- $L' =$  (uguale) permette di assegnare alla variabile/costante a sinistra dell'uguale il valore a destra
- Se i due valori sono numeri ( $x$  e  $y$ ) allora

Assegnamento	Stesa cosa di ...	Descrizione
$x = y$	$x = y$	$x$ prende il valore di $y$
$x += y$	$x = x + y$	Addizione
$x -= y$	$x = x - y$	Sottrazione
$x *= y$	$x = x * y$	Moltiplicazione
$x /= y$	$x = x / y$	Divisione
$x \% = y$	$x = x \% y$	Modulo



# Operatori di confronto

- Valgono sia per stringhe che numeri

Operatori	Nome	Esempio	Risultato
==	Uguale	<code>\$x == \$y</code>	True se x e y sono uguali
===	identico	<code>\$x === \$y</code>	True se sono uguali e se sono dello stesso tipo



# Operatore ===

```
<?php
```

```
$x = 100;
```

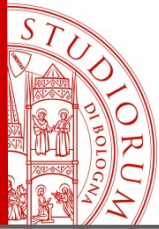
```
$y = "100";
```

```
var_dump($x === $y);
```

```
// restituisce falso perché hanno tipi diversi
```

```
?>
```

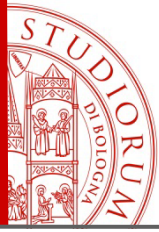
False



# Operatori di confronto

- Valgono sia per stringhe che numeri

Operatori	Nome	Esempio	Risultato
==	Uguale	<code>\$x == \$y</code>	True se x e y sono uguali
===	identico	<code>\$x === \$y</code>	True se sono uguali e se sono dello stesso tipo
!=	Diverso	<code>\$x != \$y</code>	True se i valori sono diversi
<>	Diverso	<code>\$x &lt;&gt; \$y</code>	True se i valori sono diversi
!==	Non identico	<code>\$x !== \$y</code>	True se i valori non sono uguali o non sono dello stesso tipo



# Operatore !==

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<?php
```

```
$x = 100;
```

```
$y = "100";
```

```
var_dump($x !== $y);
```

```
// returns true because types are not equal
```

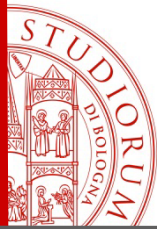
```
?>
```

```
</body>
```

```
</html>
```

True

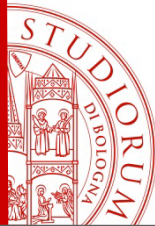




# Operatori di confronto

- Valgono sia per stringhe che numeri

Operatori	Nome	Esempio	Risultato
==	Uguale	<code>\$x == \$y</code>	True se x e y sono uguali
===	identico	<code>\$x === \$y</code>	True se sono uguali e se sono dello stesso tipo
!=	Diverso	<code>\$x != \$y</code>	True se i valori sono diversi
<>	Diverso	<code>\$x &lt;&gt; \$y</code>	True se i valori sono diversi
!==	Non identico	<code>\$x !== \$y</code>	True se i valori non sono uguali o non sono dello stesso tipo
>	Maggiore di	<code>\$x &gt; \$y</code>	True se x è maggiore di y
<	Minore di	<code>\$x &lt; \$y</code>	True se x è minore di y
>=	Maggiore o uguale di	<code>\$x &gt;= \$y</code>	True se x è uguale o maggiore di y
<=	Minore o uguale di	<code>\$x &lt;= \$y</code>	True se x è uguale o minore di y



# Operatori di incremento/decremento

Operatori	Nome	Descrizione
<code>++\$x</code>	Pre-incremento	Incrementa x di uno poi restituisce il valore di x
<code>\$x++</code>	Post-incremento	Restituisce il valore di x poi incrementa di uno il valore di x
<code>--\$x</code>	Pre-decremento	Decrementa x di uno poi restituisce il valore di x
<code>\$x--</code>	Post-decremento	Restituisce il valore di x poi decrementa di uno il valore di x

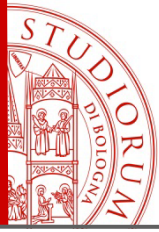
# Esempi

```
<?php  
$x = 10;
```

```
echo "x vale" . $x++ . "<br>";  
echo "x vale" . ++$x . "<br>";
```

```
?>
```





# Esempi

```
<?php  
$x = 10;
```

```
echo "x vale" . $x++ . "<br>";  
echo "x vale" . ++$x . "<br>";
```

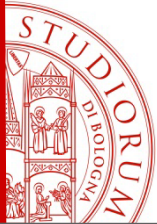
```
?>
```

```
x vale 10  
x vale 12
```



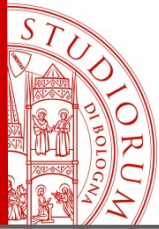
# Operatori logici

Operatore	Nome	Esempio	Risultato
and	And	$\$x \text{ and } \$y$	True se x e y sono veri
or	Or	$\$x \text{ or } \$y$	True se almeno uno tra x e y è vero
xor	Xor	$\$x \text{ xor } \$y$	True se solo uno tra x e y è vero
&&	And	$\$x \ \&\& \ \$y$	True se x e y sono veri
	Or	$\$x \    \ \$y$	True se almeno uno tra x e y è vero
!	not	$!\$x$	True se x non è vero



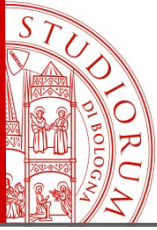
# Operatori di stringa

Operatore	Nome	Esempio	Risultato
.	Concatenazione	\$txt1 . \$txt2	Concatenazione di txt1 e txt2
.=	Concatenazione e assegnazione	\$txt1 .= \$txt2	Appende txt2 a txt1



# Operatori per Array

Operatore	Nome	Esempio	Risultato
+	Unione	$\$x + \$y$	Unione dei due array
==	Uguaglianza	$\$x == \$y$	True se x e y hanno le stesse coppie chiave/valori
===	Identità	$\$x === \$y$	True se x e y hanno le stesse coppie chiave/valori, nello stesso ordine e dello stesso tipo
!=	Disuguaglianza	$\$x != \$y$	True se x è diverso da y
<>	Disuguaglianza	$\$x <> \$y$	True se x è diverso da y
!==	Non identità	$\$x !== \$y$	True se x non è identico a y



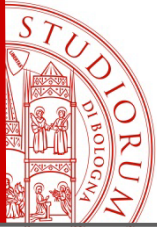
# If e if-else statement

- Sintassi

```
if (condition) {  
    // code to be executed if condition is true;  
}
```

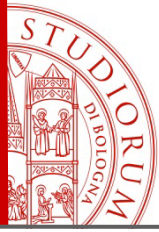
```
if (condition) {  
    // code to be executed if condition is true;  
} else {  
    // code to be executed if condition is false;  
}
```





# if-elseif-else statement

```
if (condition) {  
    // code to be executed if condition is true;  
} elseif (condition) {  
    // code to be executed if condition is true;  
} else {  
    // code to be executed if condition is false;  
}
```

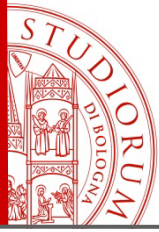


# Esempio

```
<?php
$t = date("H");
echo "<p>Secondo il server sono le" . $t;
echo ", e quindi vi dico:</p>";

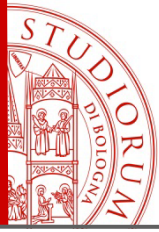
if ($t < "10") {
    echo "<p>Buon giorno!!</p>";
} elseif ($t < "14") {
    echo "<p>Buon pomeriggio!</p>";
} else {
    echo "<p>Buona serata!</p>";
}
?>
```

Secondo il server  
sono le 11, e  
quindi vi dico:  
Buon pomeriggio!



# Switch statement

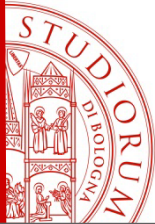
```
switch (n) {  
    case label1:  
        // code to be executed if n=label1;  
        break;  
    case label2:  
        // code to be executed if n=label2;  
        break;  
    case label3:  
        //code to be executed if n=label3;  
        break;  
    ...  
    default:  
        // code to be executed if n is different from all labels;  
}  
}
```



# Esempio

```
<?php
$colore = "viola";
switch ($colore) {
    case "red":
        echo "Il tuo colore preferito è rosso!";
        break;
    case "green":
        echo "Il tuo colore preferito è verde!";
        break;
    case "blue":
        echo "Il tuo colore preferito è blu!";
        break;
    default:
        echo "Il tuo colore preferito non è un colore primario addittivo"; } ?>
```

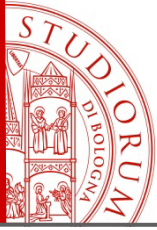
Il tuo colore  
preferito non è  
un colore  
primario  
addittivo



# Cicli (loops)

---

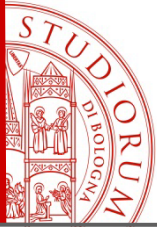
- Si può scegliere tra
  - while
    - Cicla fino a che non è soddisfatta la condizione del while
  - do ... while
    - Esegue il codice definito nel *do* fino a che non è soddisfatta la condizione specificata nel *while*. Esegue il codice del *do* almeno una volta (visto che la condizione di uscita è definita dopo)
  - For
    - Cicla per un numero definito di volte
  - Foreach
    - Cicla per ogni elemento nell'array



# Sintassi

```
while (condition is true) {  
    // code to be executed;  
}
```

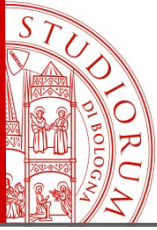
```
do {  
    //code to be executed;  
} while (condition is true);
```



# Sintassi

```
for (init counter; test counter; increment counter) {  
    // code to be executed;  
}
```

```
foreach ($array as $value) {  
    code to be executed;  
}
```



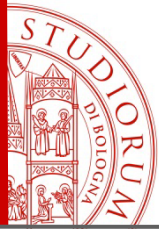
# Esempio While

```
<?php
$x = 1;

while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
}
?>
```

The number is: 1  
The number is: 2  
The number is: 3  
The number is: 4  
The number is: 5



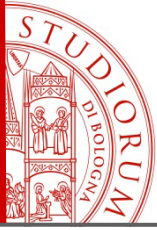


# Esempio Do While

```
<?php
$x = 1;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

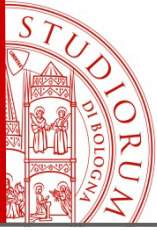
The number is: 1  
The number is: 2  
The number is: 3  
The number is: 4  
The number is: 5



# Esempio For

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

The number is: 0  
The number is: 1  
The number is: 2  
The number is: 3  
The number is: 4  
The number is: 5  
The number is: 6  
The number is: 7  
The number is: 8  
The number is: 9  
The number is: 10

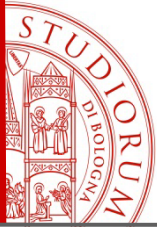


# Esempio Foreach

```
<?php
$colori = array("rosso", "verde", "blu", "giallo");

foreach ($colori as $colore) {
    echo "$colore <br>";
}
?>
```

rosso  
verde  
blu  
giallo

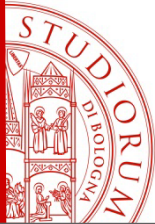


# Esempio Foreach (2)

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37",
"Joe"=>"43");

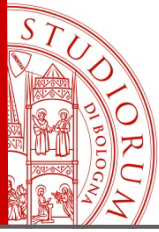
foreach($age as $key => $value) {
    echo "Key=" . $key . ", Value=" . $value;
    echo "<br>";
}
?>
```

Key=Peter, Value=35  
Key=Ben, Value=37  
Key=Joe, Value=43



# Funzioni

- Si definisce con la parola chiave **function**  
**function** nomeFunzione(var1, var2, var3,...) {  
    //tanto codice...  
}
- Una volta dichiarata la funzione, possiamo eseguirla in qualsiasi momento semplicemente scrivendone il suo nome seguito da due parentesi tonde ()
- Non bisogna né definire il tipo di parametri in ingresso ne quelli restituiti
- Può non avere parametri in ingresso



# Esempio funzione

```
<?php
```

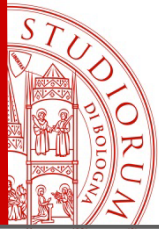
```
$nome = "Batman";
```

```
function stampaNome($nome) {  
    echo "<strong>Ciao " . $nome . "</strong>";  
}
```

```
stampaNome($nome);
```

```
?>
```

**Ciao Batman**



# Valori di default nelle variabili

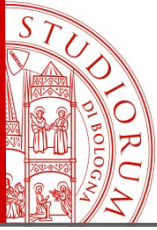
```
<?php
function nuovaColazione($numcaffè = 1, $numpaste = 1) {
    echo "Colazione composta da: $numcaffè caffè e
        $numpaste paste" .
    <br>";
}

nuovaColazione(2, 0);
nuovaColazione();// userà i valori di default
nuovaColazione(1, 3);
?>
```

Colazione  
composta da:  
2 caffè e 0 paste

Colazione  
composta da:  
1 caffè e 1 paste

Colazione  
composta da:  
1 caffè e 3 paste



# Valore restituito

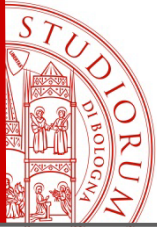
```
<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}
echo "5 + 10 = " . sum(5,10) . "<br>";
echo "7 + 13 = " . sum(7,13) . "<br>";
echo "2 + 4 = " . sum(2,4);
?>
```

5 + 10 = 15

7 + 13 = 20

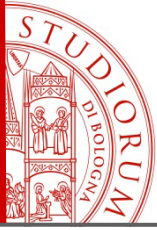
2 + 4 = 6





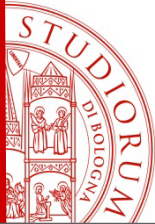
# Gestione file

- Aprire un file e leggere il suo contenuto
  - `readfile("dizionario.txt");`
- Aprire un file specificando più dettagli
  - `fopen($myfile,$permessi);`
    - `$permessi` può assumere valore
      - `r` aprirlo solo il lettura
      - `w` aprirlo in modifica. Crea un nuovo file se non esiste
      - `a` scrive nel file, appendendo il nuovo testo
      - `x` crea un nuovo file per scriverci. Restituisce false se il file esiste già
      - ..... ([http://www.w3schools.com/php/php\\_file\\_open.asp](http://www.w3schools.com/php/php_file_open.asp))



# Gestione file

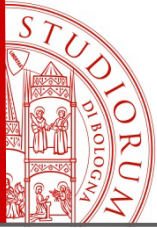
- `fread()`
  - Legge da un file aperto. Il primo parametro è il file, il secondo la grandezza massima del file
    - `fread($myfile, filesize("dizionario.txt"));`
- `fclose`
  - Chiude un file. Un parametro, il nome del file
    - `fclose($myfile);`
- `fgets()`
  - Legge una singola linea del file



# Gestione file

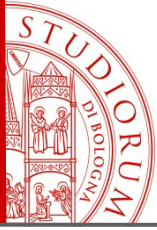
---

- `fwrite()` oppure `fputs()`
  - Scrive blocchi di dati nel file
- `feof()`
  - Restituisce true quando si arriva alla fine del file
- Dettagli sulla gestione del filesystem in generale (directory e percorsi)
  - [http://www.w3schools.com/php/php\\_ref\\_filesystem.asp](http://www.w3schools.com/php/php_ref_filesystem.asp)



# Esempio: lettura file

```
<?php
/* Aprire lo stream solo in lettura */
$fp = fopen ('dati.txt', 'r');
if (!$fp) {
    die ("Errore! Il file non può essere aperto.");
}
/* Leggere una riga del file */
$line = fgets($fp);
/* Chiudere il file handle */
fclose($fp);
?>
```



# Esempio: scrittura file

```
<?php
```

```
/* Open a file in read/write mode and binary mode, and place  
* the stream pointer at the beginning of the stream. */
```

```
$fp = fopen("/tmp/tempfile", "rb+");
```

```
/* Try to read a block of 4096 bytes from the file */
```

```
$block = fread($fp, 4096);
```

```
/* Write that same block of data to the stream again
```

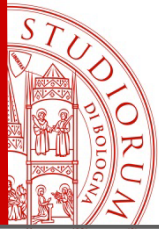
```
* just after the first one */
```

```
fwrite($fp, $block);
```

```
/* Close the stream */
```

```
fclose($fp);
```

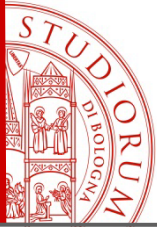
```
?>
```



# Esempio: leggere tutte le righe

```
<?php
/* Set the include path */
ini_set('include_path', '/etc:/usr/local/etc:');
/* Open handle to file */
$fp = fopen('php.ini', 'r', TRUE);
/* Read all lines and print them */
while (!feof($fp)) {
    $line = trim(fgets($fp, 256));
    echo ">$line<\n";
}
/* Close the stream handle */
fclose($fp);
?>
```

Questo pezzo di codice evidenziato permette di cercare il file php.ini prima in /etc poi in /usr/local/etc ed infine nella directory corrente



# Siti di riferimento

---

- W3Schools:  
<https://www.w3schools.com/php/>
- Sito web di PHP:  
<https://secure.php.net/>
- NB: conviene comunque impraticchirsi con il W3School perché è l'unico disponibile durante l'esame!



# Domande?

---

