Il linguaggio PHP

Docente
Mario Perna
prof.perna.mario@darzo.net

A.S.
2025/2026
Materia
TEPSIT (Laboratorio)

Come si scrive codice PHP?

Il codice PHP viene scritto attraverso i tag di apertura e chiusura:

```
<?php
        .....
        codice per logica funzionale
        .....
?>
```

*Per fare in modo che il server riconosca il codice PHP all'interno di una pagina, è necessario che il file abbia l'estensione **.php**.*

Commenti in PHP

```
<?php
        // commento singolo
        /*
                commenti multi linea
                prima riga d'istruzione commentata
                seconda riga d'istruzione commentata
        */
?>
```

Le variabili

In **PHP** tutte le variabili devono sempre iniziare con il simbolo **$**.

Ad esempio, nel codice seguente definiamo una variabile chiamata **$anni** e le assegniamo il valore 28:

```
<?php
        $anni = 28;
?>
```

È importante ricordare che i nomi delle variabili in PHP sono **case-sensitive**, cioè fanno distinzione tra maiuscole e minuscole. Quindi le variabili **$anni e $ANNI sono diverse!**

In PHP è **obbligatorio rispettare la punteggiatura**, aggiungendo a fine riga il carattere punto e virgola (;)

La scrittura delle variabili

```
<?php
        $anniUtente = 28;  // camelCase
        $anni_utente = 28; // Snake-Case
?>
```

Per convenzione, le variabili vengono scritte in **camelCase** oppure in **snake-case**.

N.B.: Una volta scelta una tipologia di scrittura è consigliato mantenerla in tutte le parti del codice.

La scrittura delle variabili

Il linguaggio è detto **Debolmente Tipizzato**, ovvero: una variabile può assumere differenti tipologie di assegnazioni (int, float, string, boolean ecc…).

```
<?php
        $anniUtente = 28;
        $anniUtente = ‘28’;
        $anniUtente = true;
?>
```

ATTENZIONE: è buona norma **non adottare questo approccio**, che potrebbe portare alla generazione di eventuali BUG.

Stampare in output una variabile (string)

In PHP possiamo **stampare il contenuto di una variabile** utilizzando il comando **echo**.
Questo ci permette di mostrare a schermo valori numerici, stringhe o altri dati contenuti nella variabile.

```php
<?php
    $nome = "Mario";
    echo $nome;  // Stampa: Mario
?>
```

N.B.: il comando echo è possibile utilizzarlo solo in presenza di variabili di tipo string

Stampare in output una variabile con **var_dump()**

La funzione var_dump() viene utilizzata per visualizzare informazioni dettagliate su una variabile, inclusi tipo e valore.

È utile per il debug, in quanto mostra anche il tipo di dato della variabile (ad esempio, intero, stringa, array, ecc.).

```php
<?php
    $variabile = "Ciao Mondo";
    var_dump($variabile); // Output: string(11) "Ciao Mondo"
?>
```

Assegnamento per valore

Assegnamento per valore: copia il valore della variabile in un'altra. La modifica della nuova variabile **non cambia** quella originale.

```php
<?php
    $a = 10; // Variabile originale

    $b = $a; // Assegnamento per valore

    $b = 20; // Modifico $b

    echo "Valore di \$a: $a"; // rimane 10
    echo "Valore di \$b: $b"; // diventa 20
?>
```

Assegnamento per riferimento

Assegnamento per riferimento: la nuova variabile punta alla stessa locazione di memoria della originale. Modificare una variabile **modifica anche l'altra**.

```
<?php
    $a = 10; // Variabile originale
    $b = &$a; // Assegnamento per riferimento
    $b = 20; // Modifico $b

    echo "Valore di \$a: $a\n"; // diventa 20
    echo "Valore di \$b: $b\n"; // diventa 20
?>
```

isset()

La funzione **isset()** viene utilizzata per verificare se una variabile è definita e se il suo valore non è NULL.

Restituisce **true** se la variabile esiste e ha un valore diverso da NULL, altrimenti **false**.

```php
<?php
    $variabile = "Test";
    $risultato = isset($variabile);
    echo $risultato; // true
?>
```

Costanti

In PHP, una **costante** è un valore che non può essere modificato durante l'esecuzione dello script. Una volta definita, una costante mantiene il suo valore fisso e non può essere alterata come una variabile.

REGOLA DI SCRITTURA

define(<nome_variabile>, <valore>);

```
<?php
        define('APP_VERSION', 1); // definizione. Creata a runtime
        echo APP_VERSION; // visualizzazione
?>
```

Altri modi per definire le costanti

In PHP è possibile scrivere le costanti anche in altro modo.

```
<?php
        const APP_VERSION = 1;
?>
```

ATTENZIONE: In PHP, le costanti definite con **const** non possono essere create all'interno di strutture di controllo come **if, while, for, ecc... o di funzioni**. Le costanti devono essere definite a livello globale o all'interno di una classe (in questo caso sono costanti di classe).

ESEMPIO DA NON SEGUIRE

```
<?php
    if (true) {
        const APP_VERSION = 1; // Errore, perché interno ad un if
    }
?>
```

Costanti di Sistema

Le **costanti di sistema** sono valori predefiniti che PHP definisce automaticamente per l'ambiente di esecuzione. **Queste costanti non possono essere modificate e sono utilizzate per ottenere informazioni vitali sul sistema, sull'ambiente di esecuzione o sulla configurazione di PHP**.

Esempi comuni:

- **PHP_VERSION**: La versione corrente di PHP in esecuzione.
- **PHP_OS**: Il sistema operativo su cui gira PHP.
- **__LINE__**: Ottengo il numero di riga nel codice
- **__FILE__**: Ottengo il percorso assoluto del file eseguito

Tipi di dati

Tipo	Esempio	ANNOTAZIONI
String	$stringa = ‘CIAO’;	
Int	$intero = 28;	
Boolean	$vero = true; $falso = false;	Se effettuo il comando echo ottengo in outuput: **echo $vero →1** **echo $falso → ” (stringa vuota)**
Array	$libri = [‘libro1’, ‘libro2’, ‘libro3’…]	**Non è possibile eseguire il comando echo su array**
Null	$valoreIniziale = null;	rappresenta **l'assenza di valore** o un valore non assegnato

Le stringhe: apici singoli e doppi

‘’ Apici singoli **NON EFFETTUANO** il parse di eventuali variabili

" " Apici doppi **EFFETTUANO** il parse di eventuali variabili.

```
<?php
        $str1 = ‘Mondo’;
        echo ‘Ciao $str1’;    Output: Ciao $str1
        echo "Ciao $str1";    Output: Ciao Mondo
?>
```

Concatenazione tra stringhe

La **concatenazione tra stringhe** in PHP permette di unire più testi o variabili in un'unica stringa. Per farlo si utilizza l'operatore . (**punto**), che mette insieme i vari elementi.

```
<?php
        $classe = '5B';

        echo 'Ciao '.$classe // output: Ciao 5B
?>
```

Escaping sulle stringhe

Escape della stringa si usa il carattere **\ (Backslash)**.

Si usa quando la stringa contiene gli apici singoli oppure quando ci sono doppi apici all'interno della stringa.

Esempio 1:

```
<?php
        echo "L'altro \"Giorno\""; // L' output L'altro "Giorno"
?>
```

Esempio 2:

```
<?php
        echo 'L\'amico di Sofia'; // output: L'amico di Sofia
?>
```

Escaping sulle stringhe

Esistono altri caratteri di escape, tra cui:

- **\n** è usato per mandare a capo una stringa lunga
- **\r** è usata in ambiente windows, è analoga al **\n**

```
<?php
        echo "Lorem ipsum dolor sit amet consectetur adipisicing elit. \n Similique, consequuntur
        quod aliquam suscipit illum qui, officia modi cupiditate, eum sequi in! Ipsum vel [...] ";
?>
```

RISULTATO

Riga 1: Lorem ipsum dolor sit amet consectetur adipisicing elit.

Riga 2: Similique, consequuntur quod aliquam suscipit illum qui, officia modi cupiditate, eum sequi in! Ipsum vel [...]

Stringhe Multilinea (heredoc)

È un modo di definire **stringhe multilinea** senza dover usare escape per le virgolette o i ritorni a capo. È molto utile per scrivere testi lunghi, HTML o SQL direttamente nel codice.

```
<?php
    $testo = <<<EOD
    Ciao, questo è un esempio
    di stringa con heredoc.
    Posso anche usare variabili come $nome.
    E scrivere su più righe senza problemi.
    EOD;

    echo $testo;
?>
```

1. **Non lasciare spazi prima / dopo l'identificatore**
2. **Dopo l'identificatore si va sempre a capo**

Il delimitatore – EOD - è il nome scelto per delimitare l'inizio e la fine di una stringa heredoc, composto da lettere, numeri o underscore e non può iniziare con un numero.

Stringhe Multilinea (heredoc) - Precisazioni

In un heredoc l'identificatore dopo <<< serve a delimitare la stringa e non deve essere per forza **EOD**: è possibile scegliere qualsiasi nome valido, rispettando le regole di lettere, numeri e underscore.

ESEMPIO 1	ESEMPIO 2	ESEMPIO 3
$testo1 = <<<**EOD** Test con EOD **EOD**;	$testo2 = <<<**FINE** Test con FINE **FINE**;	$testo3 = <<<**HTML_CONTENT** Test con HTML_CONTENT **HTML_CONTENT**;

Uso di costanti con heredoc

In un testo heredoc **non è possibile** stampare il contenuto di una costante!.
Tuttavia, esiste un **workaround** (escamotage) per ottenere il risultato desiderato.

ESEMPIO NON FUNZIONANTE

```
<?php

        define("NOME", "Mario");
        $testo1 = <<<EOD
            Il mio nome è .NOME.
            Il mio nome è ${NOME}
        EOD;

?>
```

ESEMPIO FUNZIONANTE

```
<?php

        define("NOME", "Mario");
        $nome = NOME;
        $testo1 = <<<EOD
            Il mio nome è $nome
        EOD;

?>
```

Funzioni sulle stringhe

		Esempio
explode()	Da String ad Array	$users = ‘Guido Rossi Programmatore’; $arr= **explode**(‘ ’, $users); var_dump($arr); /* [0] => string (5) ‘Guido’ [1] => string (5) ‘Rossi’ [2] => string (13) ‘Programmatore’ */
implode()	Da Array a String	Riprendendo l’esempio di sopra $str = **implode**(‘,’, $arr); echo $str; // ‘Guido,Rossi,Programmatore’
addslashes()	Aggiunge il carattere slash dove è presente apice singolo o doppio apice (previene gli attacchi **SQL Injection**)	$user = " Guido D’elia »; echo **addsalshes**($user); // Guido D\’elia

Funzioni sulle stringhe

		Esempio
str_contains($pagliaio, $ago)	Cerca nella stringa "Corso PHP", "JavaScript". Restituisce true o false	$pagliaio = "Corso PHP"; $ago = "Corso PHP"; str_contains("Corso PHP", "Javascript")
str_starts_with($pagliaio, $ago)	Verifica che la stringa "Corso PHP" contenga all'inizio la parola "Corso". Restituisce true o false	$pagliaio = "Corso PHP"; $ago = "Corso"; str_starts_with("Corso PHP", "Corso") //
str_ends_with($pagliaio, $ago)	Verifica che la stringa "Corso PHP" finisca con la parola "PHP". Restituisce true o false	$str = "Corso PHP"; str_ends_with("Corso PHP", "PHP")

Funzioni sulle stringhe

trim() Rimuove tutti gli spazi di una stringa	<?php $str = " Ciao mondo! "; echo "|" . trim($str) . "|"; // Output: |Ciao mondo!| ?>

ltrim() Rimuove tutti gli spazi a sinistra di una stringa	<?php $str = " Ciao mondo! "; echo "|" . ltrim($str) . "|"; // Output: |Ciao mondo! | ?>

Funzioni sulle stringhe

rtrim() Rimuove tutti gli spazi a destra di una stringa	<?php $str = " Ciao mondo! "; echo "|" . rtrim($str) . "|"; // Output:	Ciao mondo!| ?>

Funzioni sulle stringhe

		Esempio
lcfirst($stringa)	Imposta la prima lettera in minuscolo (lowercase), lasciando invariato tutto il resto	<?php $string = "Hello World"; $result = lcfirst($string); echo $result; // Risultato: "hello World" ?>
ucfirst($stringa)	Imposta la prima lettera in MAIUSCOLO (uppercase) lasciando invariato tutto il resto	<?php $string = “hello world"; $result = ucfirst($string); echo $result; // Risultato: “Hello world" ?>

Funzioni sulle stringhe

		Esempio
ucwords($stringa)	converte in **maiuscolo la prima lettera di ogni parola** in una stringa	<?php $string = “hello world"; $result = ucwords($string); echo $result; // Risultato: “Hello World" ?>
strtolower($stringa)	Converte tutta la stringa in minuscolo	<?php $string = "Hello World"; $result = **strtolower** ($string); echo $result; // Risultato: “**hello world**" ?>

Funzioni sulle stringhe

		Esempio
strtoupper($stringa)	Converte tutta la stringa in MAIUSCOLO	<?php $string = “hello world"; $result = **strtoupper**($string); echo $result; // Risultato: “**HELLO WORLD**" ?>
strpos($stringa, $sottostringa, $offset);	cerca la posizione di una sottostringa all'interno di una stringa. **Restituisce la posizione della prima occorrenza della sottostringa trovata, o false se la sottostringa non è presente.**	<?php $stringa = "Ciao mondo!"; $sottostringa = "mondo"; $posizione = strpos($stringa, $sottostringa); ?>

Funzioni sulle stringhe

		Esempio
strlen($stringa)	Restituisce la lunghezza di una stringa, ovvero il numero di caratteri presenti al suo interno. Il conteggio include anche spazi e caratteri speciali.	<?php $string = "Hello World"; $result = **strtolower** ($string); echo $result; // Risultato: “**hello world**" ?>
htmlspecialchars($stringa)	Converti i caratteri speciali in entità HTML **PREVIENE GLI ATTACCHI Cross-Site Scripting (XSS)**	<?php $str = “<script> alert(‘ciao Mondo’); </script>”; echo htmlspecialchars($str); ?>

Funzioni sulle stringhe

		Esempio

strrev($str)	Data una stringa ne inverte il contenuto.	<?php
		
		$str = “Ciao Mondo”;
		
		echo strrev($str);
		// OUTPUT -> odnom oaic
		
		?>

str_replace($cerca, $sostituisci, $stringa, &$conteggio);
 	
&$conteggio è opzionale
	Sostituisce tutte le occorrenze di una stringa (o più stringhe) all’interno di un’altra stringa	<?php
		$testo = "Ciao mondo mondo!";
		$nuovoTesto = str_replace("mondo", "PHP", $testo);
		
		echo $nuovoTesto; // Output: Ciao PHP PHP!
		?>

Funzioni sulle stringhe

		Esempio

substr(string $stringa, int $inizio, ?int $lunghezza = null): string	Estrae una parte di stringa	<?php
		echo substr('abcdef', 1); // bcdef
		echo substr('abcdef', 1, 3); // bcd
		echo substr('abcdef', 0, 4); // abcd
		echo substr('abcdef', 0, 8); // abcdef
		echo substr('abcdef', -1, 1); // f
		?>

Esercizi 1

Esercizio 1
Scrivi uno script che prenda una stringa e stampi la sua lunghezza.
Esercizio 2
Trasformare la stringa "BENVENUTI AL CORSO PHP" in minuscolo.
Esercizio 3
Trasforma la stringa "benvenuti al corso php" in maiuscolo.
Esercizio 4
Sostituisci alla frase iniziale "Studio JavaScript ogni giorno" la parola "JavaScript" con "PHP"

Operatori consentiti

		Esempio
<	Minore di...	$a < $b
>	Maggiore di...	$a > $b
==	Uguaglianza	$a == $b
===	Uguaglianza stretta (uguale per valore e per tipo)	$a === $b
!=	Diverso da...	$a != $b
!==	non uguale e di tipo diverso	$a (string) !== $b (string)
<>	Diverso da... (simile a	$a <> $b

Operatori consentiti

		Esempio
<=> (Spaceship)	Compreso tra	$a < = > $b dal confronto l'operatore ritorna: • **-1 se $a è minore di $b** • **0 se $a = $b** • **1 se $a è maggiore di $b**

STRUTTURE DI CONTROLLO SELETTIVE

Istruzione if

Esegue il blocco solo se la condizione è vera.

```
<?php
      $eta = 20;
      if ($eta >= 18) {
         echo "Sei maggiorenne.\n";
      }
?>
```

Output:
Sei maggiorenne.

Selezione a cascata (if...else)

Esegue un blocco se vero, un altro se falso.

```
<?php
    $eta = 16;
    if ($eta >= 18) {
        echo "Sei maggiorenne.\n";
    } else {
        echo "Sei minorenne.\n";
    }
?>
```

Output:
Sei minorenne.

Operatore ternario

È una forma compatta di if...else.

```
<?php
	$eta = 20;
	$stato = ($eta >= 18) ? "maggiorenne" :
	"minorenne";
	echo "Sei $stato.\n";
?>
```

Output:
Sei maggiorenne.

Selezione multipla (if...elseif...else)

Permette più controlli sequenziali.

```
<?php
    $giorno = "Mercoledì";
    if ($giorno == "Lunedì") {
        echo "Oggi è Lunedì.\n";
    } elseif ($giorno == "Martedì") {
        echo "Oggi è Martedì.\n";
    } else {
        echo "Oggi non è né Lunedì né Martedì.\n";
    }
?>
```

Output:
Oggi non è né Lunedì né Martedì.

Esercizi 2

Esercizio 1

Scrivi uno script che prende un numero intero $n e stampa "Multiplo di 3" solo se $n è multiplo di 3. Se non lo è, non stampare nulla.

Esercizio 2

Dato un intero $n, stampa "Pari" se è pari, altrimenti stampa "Dispari".

Esercizio 3

Ricevi due numeri $a e $b. Usa l'operatore ternario per assegnare a $max il valore maggiore e poi stampalo.

Esercizio 4

Scrivi un programma che legge un numero da 1 a 7 e stampa il giorno della settimana corrispondente. (1 = Lunedì, 2 = Martedì, 3 = Mercoledì, 4 = Giovedì, 5 = Venerdì, 6 = Sabato e 7 = Domenica)

Esercizio 5

Scrivi un programma che legge un voto intero (0–30) e stampa la valutazione:

0–17 → "Insufficiente"

18–22 → "Sufficiente"

23–26 → "Buono"

27–29 → "Ottimo"

30 → "Eccellente"

I CICLI

Ciclo definito (for)

Il ciclo **for** si usa quando **si conosce in anticipo il numero di iterazioni**.

```
<?php
    for ($i = 1; $i <= 5; $i++) {
        echo "Numero: $i\n";
    }
?>
```

Output:

```
Numero: 1
Numero: 2
Numero: 3
Numero: 4
Numero: 5
```

Ciclo definito (foreach)

L'istruzione **foreach** serve per scorrere **array o oggetti** senza preoccuparsi dell'indice.

```
<?php
    $frutti = ["Mela", "Banana", "Arancia"];
    foreach ($frutti as $frutto) {
        echo "Frutto: $frutto\n";
    }
?>
```

Output:
Frutto: Mela
Frutto: Banana
Frutto: Arancia

Ciclo pre-condizionato (while)

Il ciclo **while** verifica **prima** la condizione: se è vera, esegue il blocco; altrimenti non entra mai nel ciclo.

```
<?php
    $numero = 1;
    while ($numero <= 5) { // controllo prima di eseguire il ciclo
        echo "Numero: $numero\n";
        $numero++;
    }
?>
```

Output:

```
Numero: 1
Numero: 2
Numero: 3
Numero: 4
Numero: 5
```

Ciclo post-condizionato (do...while)

Il ciclo **do...while** esegue **almeno una volta** il blocco e poi verifica la condizione.

```
<?php
      $numero = 6;
      do {
         echo "Numero: $numero\n";
         $numero++;
      } while ($numero <= 5); // controllo dopo
?>
```

Output:
Numero: 6

Esercizi 3

Esercizio 1

Dato un intero positivo N, usa un ciclo for per calcolare la somma 1 + 2 + ... + N e stampare il risultato.

Esercizio 2

Dato un numero intero N, stampa i primi 10 multipli usando un ciclo for.

Esercizio 3

Scrivi un programma che prenda in input una stringa e stampi ogni carattere su una nuova riga usando foreach.

GLI ARRAY

Array: definizione

```
<?php
        $a = array('michele', 'luisa', 'miriam');
        // oppure
        $a = ['michele', 'luisa', 'miriam'];
?>
```

Array: aggiunta elemento in coda

Aggiungiamo all'array iniziale $a = ('michele', 'luisa', 'miriam'); il nome 'Luigi'.

```
<?php
        $a = array('michele', 'luisa', 'miriam');
        $a[] = "Luigi"; //michele, luisa, miriam, Luigi
        OPPURE
        array_push($a, "Luigi"); // aggiunge Luigi
?>
```

Array: aggiunta elemento in testa

Aggiungiamo all'array iniziale $a = ('michele', 'luisa', 'miriam'); il nome 'Marco Aurelio'.

```
<?php
        $a = array('michele', 'luisa', 'miriam');
        array_unshift($a, "Luigi"); // Luigi, michele, luisa, miriam
?>
```

Array: output

```
<?php
        $a = [‘michele’, ‘luisa’, ‘miriam’];
        // output dell’array
        var_dump($a);
?>
```

Il risultato del comando **var_dump()** sarà:

array(3) {[0]=>string(7) "michele" [1]=> string(5) "luisa" [2]=> string(6) "miriam"}

Array: conteggio elementi

```
<?php
        $a = [‘michele’, ‘luisa’, ‘miriam’];
        // contare gli elementi
        echo count($a); // Risultato: 3
?>
```

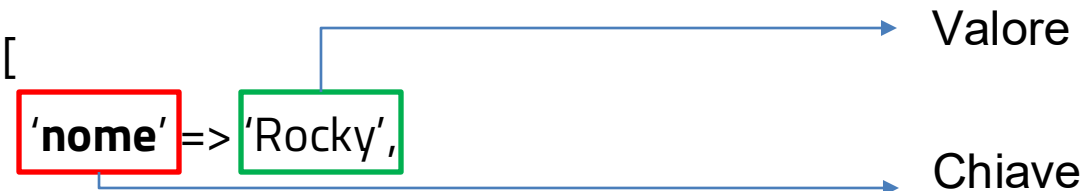
Array: recupero del valore in posizione X

```
<?php
        $a = [‘michele’, ‘luisa’, ‘miriam’];
        // visualizzare l’elemento di indice 2 (miriam)
        echo $a[2]; // miriam
?>
```

Array associativi

Un array associativo in PHP è una struttura dati che permette di memorizzare coppie **chiave-valore:** ogni elemento è identificato da una chiave univoca anziché da un indice numerico.
Si può creare assegnando valori a chiavi specifiche. I singoli valori si recuperano tramite la loro chiave.

```
<?php
        $user = [
                'nome' => 'Rocky',
                'cognome' => 'Balboa'
                'professione' => 'Pugile'
        ];
?>
```



Array associativi: recupero del valore di una specifica chiave

Come definiamo l'accesso al valore contenuto nella chiave 'nome' dell'array $user ?

```
<?php
        $user = [
                'nome' => 'Rocky',
                'cognome' => 'Balboa'
                'professione' => 'Pugile'
        ];
        $user["nome"];    // Rocky
?>
```

Array associativi: inserimento di una nuova coppia chiave / valore

Come definiamo l'accesso al valore contenuto nella chiave 'nome' dell'array $user ?

```
<?php
        $user = [
                'nome' => 'Rocky',
                'cognome' => 'Balboa'
                'professione' => 'Pugile'
        ];
        // vogliamo aggiungere nazionalità
        $user['nazionalità'] = 'Italiana';
?>
```

Array associativi: uso di string interpolation

In PHP, la **string interpolation** è quando inseriamo variabili direttamente dentro una stringa, senza doverle concatenare con (**.**). Si usa **solo con le stringhe delimitate da doppi apici (") o con l'heredoc**, **non con gli apici singoli (')**.

Come posso visualizzare la scritta **Rocky Balboa – Pugile** usando i doppi apici?

Utilizziamo la sintassi di **interpolazione di stringhe** (string template) attraverso l'uso delle parentesi graffe **{}**

```
<?php
        echo "{$user["nome"]} {$user["cognome"]} - {$user["professione"]}";
?>
```

Array associativi: Uso di extract()

```
<?php
        $user = [
                'nome' => 'Rocky',
                'cognome' => 'Balboa'
                'professione' => 'Pugile'
        ];
        extract($user)
        echo $nome, $cognome, $professione;
?>
```

La funzione extract() in PHP prende un array associativo e crea variabili con i nomi delle chiavi dell'array, assegnandogli i rispettivi valori.

Metodi applicabili su Array

Considerando l'array **$alunni = ['Simona', 'Marcello', 'Luca'];**

		Esempio (partendo sempre dalla base di partenza dell'array sopracitato)
array_push()	Inserimento in coda	array_push($alunni, 'Ludovica') Risultato: ['Simona', 'Marcello', 'Luca', '**Ludovica**'];
array_pop()	Rimozione dell'ultimo elemento in coda	array_pop($alunni) Risultato: ['Simona', 'Marcello'];
array_unshift()	Inserimento in testa	array_unshift($alunni, 'Ludovica') Risultato: ['**Ludovica**', 'Simona', 'Marcello', 'Luca'];

Metodi applicabili su Array

Considerando l'array **$alunni = ['Simona', 'Marcello', 'Luca'];**

		Esempio (partendo sempre dalla base di partenza dell'array sopracitato)
array_shift()	Rimozione del primo elemento in testa	array_shift($alunni) Risultato: ['Marcello', 'Luca'];

array_rand($array, $num);	viene utilizzata per estrarre una o più chiavi casuali da un array. **Può essere usata sia con array numerici che con array associativi.**	<?php
		// Estrai una chiave casuale
		$chiaveCasuale = array_rand($alunni);
		echo "La chiave casuale è: $chiaveCasuale\n"; // Restituisce un indice (es. 2)
		echo "Il valore corrispondente è: " . $array[$chiaveCasuale]; // Restituisce **Luca**
		?>

Metodi applicabili su Array

Considerando l'array **$alunni = ['Simona', 'Marcello', 'Luca'];**

		Esempio (partendo sempre dalla base di partenza dell'array sopracitato)
array_splice($arr,n,m)	Rimuove gli elementi partendo dall'indice n e ne rimuove m	array_splice($alunni, 1, 1) Risultato: ['Simona', 'Luca'];

Array: unpacking / spreading

UNPACKING / SPREADING
Unione di due array appartenenti alla stessa tipologia

```
<?php
        $arr1 = [1,2,3,4];
        $arr2 = [5,6,7,8];
        $arrFinale = [...$arr1,...$arr2];
        // $arrFinale    [1,2,3,4,5,6,7,8]
?>
```

Array: sort, rsort

SORT
Ordina i valori di un array in ordine crescente, riassegnando le chiavi numeriche
	$arr = [3, 1, 4, 1, 5];
	sort($arr);
	print_r($arr);
	// Output: [1, 1, 3, 4, 5]

RSORT
Ordina i valori di un array in ordine decrescente, riassegnando le chiavi numeriche
	$arr = [3, 1, 4, 1, 5];
	rsort($arr);
	print_r($arr);
	// Output: [5, 4, 3, 1, 1]

Array: asort, ksort

ASORT
Ordina i valori di un array in ordine crescente mantenendo l'associazione delle chiavi
	<?php
	$arr = ["a" => 3, "b" => 1, "c" => 4];
	asort($arr);
	print_r($arr); // oppure var_dump($arr)
	// Output: ["b" => 1, "a" => 3, "c" => 4]
	?>

KSORT
Ordina un array in base alle chiavi in ordine crescente
	<?php
	$arr = ["c" => 3, "a" => 1, "b" => 4];
	ksort($arr);
	print_r($arr); // oppure var_dump($arr)
	// Output: ["a" => 1, "b" => 4, "c" => 3]
	?>

Array: arsort, krsort

ARSORT
Ordina i valori di un array in ordine decrescente mantenendo l'associazione delle chiavi.
	$arr = ["a" => 3, "b" => 1, "c" => 4];
	arsort($arr);
	print_r($arr);
	// Output: ["c" => 4, "a" => 3, "b" => 1]

KRSORT
Ordina un array in base alle chiavi in ordine decrescente.
	$arr = ["c" => 3, "a" => 1, "b" => 4];
	krsort($arr);
	print_r($arr);
	// Output: ["c" => 3, "b" => 4, "a" => 1]

LE FUNZIONI

Funzioni personalizzate in PHP

Le funzioni sono un ottimo strumento messo a disposizione del programmatore, perché permettono di "scomporre" il programma in sottoprogrammi. La sintassi per costruire una funzione è:

```php
<?php
    function nomeFunzione(/*Argomenti funzione*/){
        //Istruizioni
        return /*valore da restituire;*/;
    }
?>
```

Funzioni personalizzate in PHP

In PHP puoi dare a un parametro di funzione un valore predefinito (default) così:

```
function saluta($nome = "ospite") {
    echo "Ciao, $nome!";
}

saluta();      // Stampa: Ciao, ospite!
saluta("Luca"); // Stampa: Ciao, Luca!
```

I parametri con default devono andare dopo quelli obbligatori
PHP non permette di inserire un parametro con valore default **prima** di uno senza default:

```
function esempio($a = 1, $b) { } //  ERRORE
function esempio($a, $b = 1) { } // CORRETTO
```

Funzioni con passaggio di parametri per valore

Vediamo la funzione per calcolare il fattoriale di un numero passato come parametro:

```php
<?php
    //Funzione con passaggio di parametri per valore
    function calcoloFattoriale($n){
        $fat=1;
        for($i=1; $i<=$n; $i++){
                $fat*=$i;
        }
        return $fat;
    }
    $numero=5;
    echo "il Fattoriale di $numero! è: ".calcoloFattoriale($numero);
?>
```

Funzioni con passaggio di parametri per indirizzo

Vediamo una funzione per lo scambio di due variabili con parametri passati per indirizzo di riferimento:

```php
<?php
    //Funzione con passaggio di parametri per indirizzo
    function scambia(&$n1, &$n2){
        $temp; //Variabile locale
        $temp=$n1;
        $n1=$n2;
        $n2=$temp;
    }
    $num1=90;
    $num2=56;
    echo "<b>Prima dello scambio</b>";

    echo "Il contenuto di num1 è: $num1<br/>
          Il contenuto di num2 è: $num2";
    //Invocazione della funzione
    scambia($num1,$num2);
    echo "<b>Dopo lo scambio</b>";
    echo "Il contenuto di num1 è: $num1<br/>
          Il contenuto di num2 è: $num2";
?>
```

Funzioni ricorsive

Vediamo un esempio di funzione ricorsiva per il calcolo dell'MCD:

```php
<?php
    //Funzione ricorsiva
    function calcoloMCD($n1, $n2){
        $resto=$n1%$n2;
        return ($resto==0)?$n2:calcoloMCD($n2,$resto);
    }
    $num1=98;
    $num2=32;
    echo "<p>L'MCD tra $num1 e $num2 è: ".calcoloMCD($num1,$num2)."</p>";
?>
```

REQUIRE() & INCLUDE()

require()

Include un file esterno e, se il file non esiste o genera errori, l'esecuzione dello script viene interrotta (errore fatale).

Uso tipico: Quando il file è indispensabile per l'applicazione.

```php
// main.php
require 'config.php'; // Se config.php non esiste, lo script si ferma.
echo "Ciao, mondo!";
```

require_once()

Come require, ma **evita di includere lo stesso file più di una volta**, prevenendo errori di dichiarazioni duplicate

Uso tipico: Quando c'è il rischio di includere più volte un file.

```php
// main.php
require_once 'config.php';
require_once 'config.php'; // Questo non viene eseguito una seconda volta.
```

include()

Include un file esterno, ma se il file non esiste o genera errori, lo script continua l'esecuzione (genera solo un "warning").

Uso tipico: Quando il file non è critico per il funzionamento dell'applicazione.

```php
// main.php
include 'header.php'; // Se header.php non esiste, lo script continua.
echo "Benvenuto!";
```

include_once()

Come include, ma evita di includere lo stesso file più di una volta.

Uso tipico: **Quando un file non è critico, ma potrebbe essere accidentalmente incluso più volte.**

```php
// main.php
include_once 'header.php';
include_once 'header.php'; // Questo non viene eseguito una seconda volta.
```

ARRAY SUPERGLOBALI

Array Superglobali: $_GET

$_GET è utilizzato per accedere ai dati inviati tramite il metodo **HTTP GET**.

Questo array associativo contiene i dati passati nell'URL come **query string** (parametri visibili nella barra degli indirizzi key=value).

Solitamente viene utilizzato quando vengono inviati dati provenienti da form.

Es.: pagina.php**?firstName=federico&lastName=rossi&age=29**

```
<?php
        $firstName = $_GET['firstName'];
        $lastName = $_GET['lastName'];
        $age = $_GET['age'];
        [...]
?>
```

Array Superglobali: $_POST

$_POST è utilizzato per accedere ai dati inviati tramite il metodo **HTTP POST**.

Questo array associativo contiene i valori dei campi di un modulo HTML inviato in modo **sicuro (i dati non sono visibili nell'URL).**

```html
<form action="process.php" method="POST">
    <label for="nome">Nome:</label>
    <input type="text" name="nome" id="nome">

    <label for="email">Email:</label>
    <input type="email" name="email" id="email">

    <button type="submit">Invia</button>
</form>
```

```php
<?php

    // Recupera i dati dal modulo
    $nome = $_POST['nome'];
    $email = $_POST['email'];

    echo "Nome: $nome<br>";
    echo "Email: $email<br>";


?>
```

La tecnica postback

La tecnica postback permette alla pagina di inviare i dati di un form a sé stessa. Per utilizzare questa tecnica bisogna utilizzare la variabile d'ambiente (array superglobale) **$_SERVER['PHP_SELF']**, la quale reindirizza i dati del form alla stessa pagina.

Questa tecnica può essere utilizzata sia con il metodo POST che con il metodo GET .

```php
<?php
    /*Restituisce FALSE se il modulo non è stato ancora inviato,
      altrimenti restituisce TRUE*/
    if($_GET){
        $utente=(isset($_GET["nome"]))?$_GET["nome"]:"Campo vuoto";
        echo "<h2>Ciao $utente!</h2>";
    }
    else{
        ?>
        <!--Ambiente HTML-->
        <!--Innesto $_SERVER['PHP_SELF'] all'interno del codice HTML-
        <form action="<?php $_SERVER['PHP_SELF'] ?>" method="get">
            <label>Inserisci il tuo nome:</label>
            <input type="text" name="nome"/><br/>
            <input type="submit"/>
        </form>
        <?php
    }
?>
```

Array Superglobali: $_SERVER

L’array $_SERVER è una **variabile superglobale** di PHP, cioè una variabile sempre disponibile in qualsiasi parte del codice, senza bisogno di dichiararla o passarla come parametro.

Il suo compito principale è quello di raccogliere e rendere disponibili una serie di **informazioni sull’ambiente di esecuzione**: sul server, sul browser del client, sulla connessione e sulla richiesta HTTP che ha attivato lo script.

Array Superglobali: $_SERVER

Di seguito vengono proposti alcuni delle informazioni disponibili quando si invoca l'array superglobale **$_SERVER**;

DENOMINAZIONE	DESCRIZIONE
$_SERVER['**PHP_SELF**']	percorso dello script in esecuzione
$_SERVER['**SERVER_NAME**']	nome del server (es. www.miosito.com)
$_SERVER['**HTTP_USER_AGENT**']	info sul browser del client
$_SERVER['**REMOTE_ADDR**']	IP del client
$_SERVER['**REQUEST_METHOD**']	metodo HTTP usato (GET, POST…)

Uso della funzione header()

La funzione **header()** permette al server di inviare intestazioni HTTP al client prima che venga inviato qualsiasi contenuto della risposta.

Questi headers HTTP forniscono informazioni aggiuntive sulla risposta del server, come **il formato del contenuto**, **i codici di stato della richiesta**, **il comportamento di caching**, e molto altro.

```php
<?php
// Imposta un codice di stato 200
header("HTTP/1.1 200 OK");

// Controlla il caching
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Expires: 0");
```

```php
// Specifica il tipo di contenuto
header("Content-Type: application/json");

// Risposta JSON
$data = ["status" => "success", "message" => "Funzione header spiegata"];
echo json_encode($data);
?>
```

Uso della funzione header() per la redirection

È possibile utilizzare la funzione header() per effettuare delle redirection.

```php
header("Location: URL", bool $replace = true, int $response_code = 302);
```

Analizziamo i parametri:

- "**Location: URL**": specifica l'URL della destinazione della redirezione.
- **$replace (facoltativo, default = true)**:
  - se **TRUE**, sovrascrive le intestazioni HTTP precedenti con lo stesso tipo.
  - se **FALSE**, aggiunge una nuova intestazione senza rimpiazzare quelle esistenti
- **$response_code (facoltativo, default 302)**: Codice di stato HTTP da inviare con la risposta. Per le redirezioni, i codici più usati sono:
  - 301: Redirezione permanente
  - 302: Redirezione temporanea