# PHP

Elementi del linguaggio.

***Sensibilità alle maiuscole.***
PHP è case sensitive solo per i nomi delle variabili.

Esempio:

$Totale_Deposito_Cliente
$totale_deposito_cliente
$TOTALE_deposito_Cliente

sono variabili PHP tutte diverse tra loro.

***Punti e virgola.***
Vengono usati per la separazione delle istruzioni semplici. Per un'istruzione composta che utilizzi parentesi graffe per contrassegnare un blocco di codice, come un test condizionale o un ciclo, non è necessario il ; dopo la parentesi graffa di chiusura. E' invece obbligatorio il ; prima della parentesi graffa di chiusura.

***Spazi e interruzioni di riga.***
L'istruzione può essere espressa su un qualsiasi numero di righe. Si può sfruttare questa caratteristica del linguaggio utilizzando la formattazione per migliorare la leggibilità del codice, allineando assegnamenti, rientri, eccetera.

***Commenti.***
Utilizzeremo i commenti facendoli iniziare con /* e terminandoli con */. Tutto quello che è inserito al loro interno è visto come un unico commento. Si possono utilizzare anche per disattivare interi blocchi di codice.

***Letterali.***
Un letterale è il valore di un dato che appare direttamente in un programma.

Esempi:

| | |
|---|---|
| INTERI DECIMALI | 2006 |
| INTERI OTTALI | 076543210 |
| INTERI ESADECIMALI | 0xF1 |
| REALI DECIMALI | 3.14159 |
| BOOLEANI | true |
| STRINGA (virgoletta singola) | 'Mario Bianchi' |
| STRINGA (virgoletta doppia) | "inserisci un dato" |

***Identificatori.***
Un identificatore è semplicemente un nome. In PHP gli identificatori vengono utilizzati per attribuire nomi a variabili, funzioni, costanti e classi. Il primo carattere deve essere una lettera ASCII o il carattere underscore. Dopo il carattere iniziale sono validi i sopracitati caratteri e le cifre

da 0 a 9.

Esempio:

| | |
|---|---|
| Variabili | $_NumeroCartaDiCredito |
| Funzioni | calcola_numeri_di_Fibonacci |
| Classi | ClasseFelini |
| Costanti | UnMegaByte |

Nomi di variabili: iniziano sempre col carattere $.
Nomi di funzioni e di classi: sono insensibili alle maiuscole.
Costanti: una costante è l'identificatore di un valore semplice. Possono essere costanti solo valori scalari: booleani, integer, double e stringhe. Una volta impostato il valore di una costante non può essere modificato. Le costanti sono referenziate dai loro identificatori e vengono impostate mediante la funzione define().

Esempio:

```
define ('VIA',"Via Delle Industrie");
define ('NumeroCivico',125);
echo VIA, ', ', NumeroCivico;
```

Verrà visualizzata la scritta Via Delle Industrie, 125

***Parole chiave.***
Sono parole riservate a funzionalità cardine del linguaggio e sono insensibili alle maiuscole. Non è possibile avere come identificatore di una funzione, classe, metodo o costante lo stesso nome di una parola chiave. Per l'elenco completo delle parole chiave si deve fare riferimento alla versione di PHP che si utilizza. Nella seguente tabella vengono riportate le parole riservate utilizzate in questa dispensa.

| | | | |
|---|---|---|---|
| and | array | define | do |
| echo | else | for | foreach |
| function | global | if | new |
| or | static | while | xor |

***Tipi di dati.***

***Integer.*** L'intervallo dei valori interi varia a seconda della piattaforma, ma in genere va da -2.147.483.648 a +2.147.483.647. I letterali integer possono essere scritti in formato decimale, ottale o esadecimale. I valori decimali sono rappresentati da una sequenza di cifre senza zeri iniziali. La sequenza può iniziare con il segno + o -. In assenza del segno il valore è da considerarsi positivo. I numeri ottali sono composti da uno zero iniziale e una sequenza di cifre da 0 a 7. I numeri ottali possiedono le stesse regole del segno dei decimali. I valori esadecimali iniziano con '0x' seguiti da una sequenza di cifre esadecimali. Per il segno valgono le stesse proprietà dei decimali e degli ottali.

| Decimale | Ottale | Esadecimale |
|---|---|---|

| | | |
|---|---|---|
| 255 | 0377 | 0xFF |
| 16 | 020 | 0x10 |
| -56017 | -0155321 | -0xDAd1 |

***Numeri in virgola mobile.***
Rappresentano valori numerici, detti reali, con cifre decimali. I loro limiti dipendono dalla piattaforma utilizzata. In genere sono validi i numeri compresi tra 1.7E-308 e 1.7E+308 positivi o negativi, con 15 cifre di precisione. PHP riconosce due formati diversi: senza esponente oppure con esponente in notazione scientifica.

| Senza esponente | Con esponente |
|---|---|
| 3.14 | 0.314E1 |
| 0.017 | 17.0E-3 |

I valori in virgola mobile sono semplicemente rappresentazioni approssimate di numeri. Per esempio in molti sistemi 3.5 viene in realtà rappresentato con 3.4999999999. Ciò significa che è sconsigliabile scrivere codice che presupponga un'approssimazione assolutamente accurata come a esempio il confronto diretto di due valori in virgola mobile utilizzando l'uguaglianza. L'approccio normale consiste nel confronto fino a un certo numero di posizioni decimali.

***Booleani.***
Possiedono due valori, true e false, che determinano il risultato di codice condizionale.

***Stringhe.***
Sono sequenze di caratteri di lunghezza arbitraria.
In PHP i letterali stringa sono delimitati da virgolette singole o doppie.

| | |
|---|---|
| 'Inserisci il dato: ' | virgolette singole |
| "Il risultato è: " | virgolette doppie |

Per visualizzare o stampare il contenuto delle variabili, queste devono essere espresse fra virgolette doppie. Esempio:

```
$cognome="Bianchi";
echo "Salve signor $cognome!";
```

Il risultato sarà la seguente visualizzazione:
Salve signor Bianchi!

```
$cognome="Bianchi";
echo 'Salve signor $cognome!';
```

Il risultato sarà la seguente visualizzazione:
Salve signor $cognome!

Le virgolette doppie supportano delle sequenze di escape relative alle stringhe. Alcune di queste sono riportare nella seguente tabella:

| | |
|---|---|
| \" | Virgolette doppie (necessario solo se la stringa è racchiusa da apici doppi) |
| \[ \] | Aperta e chiusa quadra |
| \{ \} | Aperta e chiusa graffa |
| \\ | Backslash |
| \n | Nuova riga (New line) |
| \r | Ritorno del carrello |
| \t | Tabulazione orizzontale |
| \$ | Simbolo del dollaro |
| Da \0 a \777 | Carattere ASCII rappresentato da un valore ottale |
| Da \x0 a \xFF | Carattere ASCII rappresentato da un valore esadecimale |

***Array.***
Un array contiene un gruppo di valori identificabili dalla loro posizione (un numero naturale a partire da 0).

Esempio:

```
$stagioni[0]="Inverno";
$stagioni[1]="Primavera";
$stagioni[2]="Estate";
$stagioni[3]="Autunno";
```

Il costrutto array() crea un array.

Esempio:

```
$stagioni=array ('Inverno','Primavera','Estate','Autunno');
```

***Variabili.***
Sono rappresentate da identificatori preceduti dal simbolo $. Possono contenere un qualsiasi tipo di valore; sulle variabili non vi è alcuna verifica in fase di compilazione e di esecuzione. In PHP non esiste una sintassi esplicita per la dichiarazione delle variabili. La variabile viene creata la prima volta che se ne imposta il valore. Quindi l'impostazione di una variabile svolge anche la funzione di dichiarazione di tipo.

***Campo d'azione delle variabili.***
Il campo d'azione di una variabile, controllato dalla posizione della dichiarazione della stessa, determina le parti di programma che possono accedervi. In PHP esistono quattro tipi di campi d'azione:

1. Locale
2. Globale
3. Statico
4. Parametri di funzioni

***Campo d'azione locale.***

Una variabile dichiarata in una funzione è locale rispetto a tale funzione ovvero è visibile solamente al codice contenuto in tale funzione. Per default le variabili definite all'esterno della funzione, dette globali, non sono accessibili dall'interno della funzione.

Esempio:

```
function visualizza_contatore(){
  $contatore=1;
  echo $contatore;
  $contatore=$contatore+1;
}


$contatore=10;
visualizza_contatore();
echo $contatore;


/* Viene assegnato il valore 10 alla variabile globale $contatore.
Poi viene assegnato il valore 1 alla variabile locale $contatore.
Viene stampato il valore 1. Viene incrementata la variabile locale
$contatore. Viene terminata la funzione  visualizza_contatore, la
variabile locale $contatore viene deallocata e quindi l'ultima
echo visualizza il valore 10 e non 2 o 11 */
```

Il campo d'azione locale viene fornito solo dalle funzioni.

***Le variabili statiche.***
Mantengono il loro valore fra diverse chiamate della funzione dove sono state dichiarate e sono visibili solo all'interno delle stesse. Vengono dichiarate con la parola chiave STATIC. Vengono allocate solo alla prima chiamata della funzione.

Esempio:

```
function decrementa_contatore(){
  static $contatore=9;
  $contatore=$contatore-1;
  echo "Valore del contatore $contatore<BR>";
  /* Il tag HMTL <BR> viene usato per andare accapo */
}

$contatore=10;
decrementa_contatore();
```

```
decrementa_contatore();
echo "Valore variabile globale \$contatore $contatore";

/* Viene visualizzato
Valore del contatore 8
Valore del contatore 7
Valore variabile globale $contatore 10 */
```

***Campo d'azione globale.***
Le variabili dichiarate all'esterno delle funzioni sono globali. Per default esse non sono accessibili dall'interno delle funzioni. Per permettere ad una funzione di accedere a una variabile globale si utilizza la parola chiave GLOBAL nella funzione interessata al fine di rendere visibile la variabile globale.

Esempio:

```
function incrementa_contatore(){
  global $contatore;
  $contatore=$contatore+1;
}

$contatore=1;
incrementa_contatore();
echo $contatore;

/* Verrà visualizzato 2 */
```

***Parametri di funzioni.***
La definizione di una funzione può contenere parametri passati per valore (aventi campo d'azione locale) oppure per riferimento (che permettono di fornire un accesso diretto a una variabile).
In questo script il parametro $nome viene passato per valore: non sarà possibile mantenerne l'allocazione al termine della chiamata alla funzione saluti.

Esempio:

```
function saluti($nome){
  echo "Saluti a $nome";
}

saluti("Mario Rossi");

/* Viene visualizzato Saluti a Mario Rossi */
```

Invece nello script seguente la variabile `$Numero` viene passata per riferimento utilizzando il carattere & prima del nome del parametro: al termine della chiamata alla funzione trasforma_in_positivo la variabile `$Numero` eventualmente aggiornata potrà essere usata nel main.

Esempio:

```
function trasforma_in_positivo(&$Numero){
  if ($Numero<0) $Numero=-$Numero;
}

$Numero=-1;
trasforma_in_positivo($Numero);
echo $Numero;

/* Viene visualizzato 1 */
```

**Incorporazione di codice PHP nelle pagine web.**
Sebbene sia possibile scrivere ed eseguire programmi PHP autonomi, la maggior parte del codice PHP è incorporata in file HTML o XML.
L'elaborazione di tali documenti comporta la sostituzione di ogni blocco di codice PHP con l'output prodotto in seguito all'esecuzione di tali script. Poiché un singolo file può contenere codice sorgente PHP e non, è necessario un sistema per identificare le regioni relative al codice PHP da eseguire.
In seguito all'avvento di XML, la tecnica attualmente preferita per l'incorporazione di codice PHP utilizza tag XML compatibili per l'indicazione di istruzioni PHP. Sarà sufficiente inserire il codice PHP fra i seguenti marcatori:

```
<?php
   .
   .
?>
```

Esempio:

```
<?php

  $LatoCubo=10;
  echo "Il valore del volume del cubo è: ", pow($LatoCubo,3);

/* pow (base,esponente) è una funzione PHP che restituisce base
($LatoCubo) elevato all' esponente (3) */

?>
```

***Espressioni e operatori.***
Un'espressione è una parte di codice PHP che può essere valutata per produrre un valore. Le espressioni semplici sono i valori letterali e le variabili. La valutazione di un valore letterale è se stesso mentre quello di una variabile è il valore memorizzato al suo interno. È possibile formare espressioni complesse utilizzando espressioni semplici e operatori. Un operatore individua alcuni valori detti operandi ed esegue delle operazioni su essi. Gli operatori vengono scritti come simboli e alcuni modificano i loro operandi.

| Precedenza | Associatività | Operatore | Operazione |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 19 | N | new | Creazione di un nuovo oggetto |
| 18 | R | [ | Indice dell'array |
| 17 | R | ! | NOT logico |
| | R | ~ | NOT bit a bit |
| | R | ++ | Incremento |
| | R | -- | Decremento |
| | R | (int)(integer)(float)(real)(double)(string)(bool)(boolean)(array)(object) | Conversione di tipo |
| | R | @ | Soppressione degli errori |
| 16 | L | * | Moltiplicazione |
| | L | / | Divisione |
| | L | % | Modulo |
| 15 | L | + | Addizione |
| | L | - | Sottrazione |
| | L | . | Concatenazione di stringhe |
| 14 | L | << | Spostamento a sinistra di un bit |
| | L | >> | Spostamento a destra di un bit |
| 13 | N | <, <= | Minore di, minore o uguale a |
| | N | >, >= | Maggiore di, maggiore o uguale a |
| 12 | N | == | Uguaglianza di valori |
| | N | !=,<> | Disuguaglianza |
| | N | === | Uguaglianza di tipo e valore |
| | N | !== | Disuguaglianza di tipo e valore |
| 11 | L | & | AND bit a bit |
| 10 | L | ^ | XOR bit a bit |
| 09 | L | \| | OR bit a bit |
| 08 | L | && | AND logico |
| 07 | L | \|\| | OR logico |
| 06 | L | ?: | Operatore condizionale |
| 05 | L | = | Assegnamento |
| | L | +=,-=<br>*=,/=<br>.=,%= | Assegnamento con operazione |

| | | | |
|---|---|---|---|
| | | &=, \|=<br>^=, ~=<br><<=, >>= | |
| 04 | L | and | AND logico |
| 03 | L | xor | XOR logico |
| 02 | L | or | OR logico |
| 01 | L | , | Separatore di liste |

***Precedenza degli operatori.***
L'ordine in cui gli operatori di un'espressione vengono valutati dipende dalla loro precedenza relativa. Per esempio il risultato dell'espressione 2+4*3 è 14 e non 18, come si evince dalla tabella delle precedenze degli operatori PHP. Per fare in modo che venga eseguito un particolare ordine è possibile racchiudere tra parentesi gli operandi col proprio operatore: (2+4)*3.

***Associatività degli operatori.***
L'associatività definisce l'ordine in cui operatori con la stessa precedenza vengono valutati. Per esempio il risultato dell'espressione 2/2*2 è 2 e non 0.5 perché la precedenza degli operatori è la stessa (16) e l'associatività va da sinistra a destra. L'associatività degli operatori può essere L (left-to-right), R (right-to-left) o N (non associativo).

***Operatori aritmetici.***
I tipi binari sono addizione, sottrazione, moltiplicazione, divisione e modulo. L'operatore modulo (%) converte entrambi gli operandi in integer e restituisce il resto della divisione del primo operando per il secondo.  Per esempio 10 % 6 è 4. La negazione aritmetica è di tipo unario, precede sempre l'operando e restituisce il valore dell'operando moltiplicato per -1 modificandone il segno.

***Operatore di concatenazione di stringhe.***
Aggiunge l'operando destro a quello sinistro e restituisce la stringa risultante. Se necessario gli operandi vengono prima convertiti in stringhe.

Esempio:

```
$numero=2;
$frase="Concatenazione di ".$numero." stringhe";
/* $frase varrà "Concatenazione di 2 stringhe" */
```

***Operatori di confronto.***
Confrontano operandi che possono essere numerici o stringa. Due stringhe numeriche vengono confrontate come se si trattasse di numeri. I confronti possono essere strettamente numerici o lessicografici (testuali).

Esempio:

```
<?php

  $PrimaStringa='100';
```

```
  $SecondaStringa='20';
  if ($PrimaStringa > $SecondaStringa) echo $PrimaStringa;

  /* Lo script visualizzerà 100 */

?>
```

***Operatori logici.***
Consentono la costruzione di espressioni logiche complesse. Gli operatori logici gestiscono i loro operandi come valori booleani e restituiscono un valore booleano.

| | |
|---|---|
| AND logico | Se il valore del primo operando è false, l'operatore sa che anche il valore risultante deve essere false e pertanto l'operatore di destra non viene valutato. Questo processo viene definito short-circuiting. |
| OR logico | Anche l'operatore logico OR è cortocircuitante. Nel caso l'operando di sinistra sia true il risultato dell'operazione deve essere true, quindi l' operando di destra non viene mai valutato. |
| XOR logico | Il risultato dell'operazione logica XOR è true se uno solo degli operandi è true. In caso contrario è false. |
| NOT logico | L'operatore di negazione logica restituisce il valore booleano true se la valutazione dell' operando è false e viceversa. |

***Operatori di assegnamento.***
Memorizzano o aggiornano i valori nelle variabili. L'operatore di base di assegnamento (=) assegna un valore a una variabile. L'operando di sinistra è sempre una variabile mentre quello di destra può essere una qualsiasi espressione. Il valore dell'operando di destra viene memorizzato nella variabile nominata dall'operatore di sinistra.

**Istruzioni per il controllo di flusso.**

**If.**
Il costrutto if si compone nel seguente modo:

```
if (espressione)
 istruzioni
else
 istruzioni
```

L'uso delle graffe è indispensabile solo in presenza di blocchi di istruzioni.

Esempio:

```
if ($codice_trovato){
  echo $codice;
  $codice=$codice+1;}
```

```
else{
  echo "codice non trovato";
  $codice=0;
}
```

**While.**
Si presenta con questo costrutto:

```
while (espressione)
  istruzioni
```

Esempio:

```
$valore=0;
while ($valore < 10){
  echo $valore;
  $valore++;
}

/* Visualizzazione in ordine crescente degli interi da 0 a 9. */
```

**Do-While.**
Il costrutto si presenta come:

```
do
  istruzioni
while (espressione)
```

Utilizzare un ciclo do-while per essere certi che il corpo del ciclo venga eseguito almeno una volta.

Esempio:

```
$dispari=1;
do{
  echo $dispari;
  $dispari=$dispari+2;
}while ($dispari < 10);

/* Visualizza 13579 */
```

**For.**
L'istruzione for è simile all'istruzione while a eccezione che aggiunge l'inizializzazione di un contatore di cicli ed espressioni per la sua manipolazione. Si presenta come:

```
for (valore iniziale; condizione; incremento)
  istruzioni
```

Esempio:

```
for ($naturale=0; $naturale<100; $naturale++) echo "$naturale ";
```

```
/* Visualizzazione dei primi 100 numeri naturali. */
```

**Foreach.**
L'istruzione foreach permette di iterare sugli elementi di un array. Per eseguire un ciclo su un array, accedendo a ogni elemento, si utilizza:

```
foreach ($array as $valorecorrente)
  istruzioni
```

Esempio:

```
$Cognomi=array('Cognomi: Bianchi, ', 'Rossi, ', 'Verdi');
foreach ($Cognomi as $CognomeCorrente) echo $CognomeCorrente;

/* Visualizzerà Cognomi: Bianchi, Rossi, Verdi */
```

PHP esegue il corpo del ciclo foreach (l'istruzione echo) una volta per ciascun elemento dell'array $Cognomi, con $CognomeCorrente impostato all'elemento corrente. Gli elementi del vettore vengono elaborati secondo il loro ordine interno.