digitando $ seguito da un nome andiamo a dichiarare una variabile in ph

$eta = 30;

echo $eta;

echo "<br>";

echo "L'età di Michele è:  " . $eta;

echo "<br>";

considerazioni sulla tipologia dei dati che si possono associare alle variabili

$x = 10;

echo "il valore della x è: " . $x . "<br>";

# --- Argomento 1 ---

dimostrazione di come le assegnazioni delle variabili sono collegate ai valori. Come assegnare le assegnazioni per referenza

una variabile può prendere il valore da un altra variabile

$y = $x;

echo "il valore della y è: " . $y . "<br>";

andiamo a cambiare il valore della variabile x e stampiamo entrambi i valori

$x = 20;

echo "il valore della x è: " . $x . "<br>";

echo "il valore della y è: " . $y . "<br>";

questo perchè per default le assegnazioni in php vanno per valore,

invece se volessimo fare una assegnazione per referenza dobbiamo utilizzare il &

in quanto & indica a Php di puntare al riferimento della variabile e non al suo valore

--- fare la modifica per verificare $y = &$x; ---

# --- Argomento 2 ---

Perchè definiamo php un linguaggio tipizzato dinamicamente?

PHP è un linguaggio tipizzato dinamicamente, il che significa che per impostazione predefinita

non è necessario specificare il tipo di una variabile,

poiché questo verrà determinato in fase di esecuzione.

$z = 700;

echo $z . "<br>";

$z = "sono diventato una stringa";

echo $z . "<br>";

# -- Argomento 3 --

Possiamo utilizzare delle variabili per richiamare altre variabili

$storico = "Cesare, l'uomo che ha reso grane Roma";

$spionaggio = 'La spia che venne dal freddo';

$romanzo = "Assassinio sull'Orient Express";

$fantascienza = "Cronache marziane";

$giallo = 'La ragazza senza nome';

$avventura = 'Il giro del mondo in 80 giorni';

$azione = 'Rambo';

$fantasy = 'I grandi sogni dei piccoli uomini';

$libro = 'giallo';

echo  'oggi ti è stato assegnato questo libro: ' . $$libro . "<br>";

$$libro => il primo $ è la definizione di una variabile; il secondo blocco $lingua indica la

variabile che vogliamo utilizzare e quindi diventerà $giallo

e potrà essere modificata con altre variabili

Questo è molto utile per rendere dinamico il nostro codice

secondo esempio

$inglese = 'Hi';

$francese = 'Salut';

$spagnolo = 'Hola';

$italiano = 'Ciao';

$lingua = 'spagnolo';

echo $$lingua . ' a tutti i miei amici' . "<br>";

# --- Argomento 4 ---

Scrittura del codice php per generare codice html. Fare un esempio con un testo in elenco

$invio = "<br>";

echo "Stampa dei risultati:";

   echo $invio;

   echo "<ul>";

   echo "<li>" . "Primo: Salvatore". "</li>";

   echo $invio;

   echo "<li>" . "Secondo: Michele". "</li>";

   echo $invio;

   echo "<li>" . "Terzo: Federico". "</li>";

   echo "</ul>";

qui abbiamo dichiarato una variabile $invio che ci permette di utilizzare il ritorno a capo con br.

la funzione echo serve per utilizzare la variabile e per visualizzare le stringhe di testo

# --- Argomento 5 ---

Scrittura  del codice html utilizzando Heredoc

$str1 = <<<STRING

$invio

<ul>

<li> Primo: Salvatore </li>

$invio

<li> Secondo: Michele </li>

$invio

<li> Terzo: Federico  </li>

</ul>

STRING;

   echo $str1;

questo strumento ci permette di snellire il codice:

è necessario utilizzare <<< seguito da un identificatore come apertura dell'Heredoc.

Ci permette di: evitare l'utilizzo della funzione echo, evitare l'utilizzo degli apici,

di evitare il . per concatenare il testo e di evitare il  ; per il fine istruzione.

E' necessario ripetere l'identificatore utilizzato seguito dal ; senza ulteriori aggiunte

# --- Argomento 6 ---

Scrittura  del codice html utilizzando Nowdoc

$str2 = <<<'STR'

$invio

<ul>

<li> Primo: Salvatore </li>

$invio

<li> Secondo: Michele </li>

$invio

<li> Terzo: Federico  </li>

</ul>

STR;

   echo $str2;

è del tutto simile ad Heredoc con due differenze sostanziali:

il marker è racchiuso tra apici in fase di definizione  e non viene eseguito il parsing delle variabili.

il parsing delle variabili si riferisce al modo in cui le variabili vengono interpretate

all’interno delle stringhe.

# --- Argomento 7 ---

Ci sono quattro modi per scrivere un numero intero

echo 11, "<br>";

echo 011, "<br>";

echo 0x11, "<br>";

echo 0b00000011 ;

Apparentemente sembrano tutti rappresntare il numero 11 ma rappresentano numeri diversi:

 - 11 rappresenta un numero in base decimale e ci restituisce il numero 11

  - 011 rappresenta un numero in base ottale, in ottale le cifre possono arrivare ad un massimo di 7 e

  per essere distinte dai decimali devono iniziare con uno 0. L'ottale è un modo compatto per rappresentare  numeri binari con tre bit alla volta.

Come va letto: la prima cifra (si parte sempre con la lettura da destra verso sinistra) vale quello che sta

indicato cioè nel nostro caso 1; il secondo 1 corrisponde a 1\*8^1 cioè 8: quindi 1+8 =9

 - 0x11 rappresenta un numero esadecimale, dove la cifra più grande che può apparire è 15.

 Quindi i numeri che iniziano con 0x rappresentano i numeri esadecimali.

Siccome i nostri numeri arabi terminano a 9 dobbiamo utilizzare delle lettere

(dalla A che vale 10 fino alla F che vale 15)

Come va letto: la prima cifra è 1; la seconda cifra è 1\*16^1 cioè 16: 1 + 16 = 17

 - 0b00000011 rappresenta un numero binario, quindi un numero che inizia con 0b rappresenta un numero binario.

Come va letto: la prima cifra è 1; la seconda cifra è 1\*2^1 cioè 2: 1 + 2 = 3

# --- Argomento 8 ---

floating point e utilizzo delle notazioni scientifiche esponenziali.

I numeri floating point sono i numeri che hanno una parte intera e una parte decimale

echo 3.14, "<br>" ;

echo .14, "<br>" ;

 - .14 la parte intera del floating point può anche essere omessa e in questo caso nella stampa

 sarà considerato come unità lo zero.

echo 31.4E-1, "<br>" ;

echo 0.0314E+2, "<br>" ;

echo 1E+9 ;

Una notazione alternativa è la notazione scientifica esponenziale.

La notazione scientifica è una rappresentazione compatta di certi numeri.

 - 31.4E-1 nella notazione scientifica esponenziale c'è una prima parte che è un numero

 floating point (31.4) che va moltiplicato per 10 elevato all'esponente indicato dopo la E.

 Come va letto: 31.4\*10^-1 che possiamo tradurre in modo equivalente 31.4\*(1/10), quindi dividere per 10;

 allo stesso modo elevare un numero alla -3 vorrà dire dividerlo per mille.

 Questo come si traduce: se l'esponente è negativo dobbiamo spostare il numero (verso sinistra) di tante posizioni corrispondenti al valore dell'esponente

 - 0.0314E+2 nel caso in cui l'esponente è positivo vorrà dire moltiplicare  per dieci, per cento...

 in questo caso la moltiplicazione è \*100

Questo come si traduce: se l'esponente è positivo si sposta (verso destra) di tante posizioni corrispondenti

al valore dell'esponente

 - 1E+9 corrisponde a un miliardo (9 zeri). Qui non essendoci una parte decimale da considerare

 questa notazione comporta l'aggiunta di zeri

# --- Argomento 9 ---

utilizzo della funzione number\_format

La funzione number\_format() in PHP è una funzione che consente di formattare i numeri

in modo più leggibile e presentabile. Questa funzione è molto utile per la formattazione di numeri

a scopo di visualizzazione, ad esempio per mostrare i prezzi in un formato usabile per l'utente.

sintassi completa della funzione

number\_format ($numero, $quanti\_decimali[facoltativo],

$separatore\_decimali[facoltativo], $separatore\_migliaia[facoltativo])

echo $invio;

echo number\_format("1000000")."<br>";

echo number\_format("1000000",2)."<br>";

echo number\_format("1000000",2,",",".");

# --- Argomento 10 ---

Esempio di espressioni nelle variabili

Alle variabili possiamo assegnare altre espressioni.

Le stesse variabili possono contenere altre variabili

echo $invio;

$costo = 120;

$iva = 120\*.22;

$costo\_ivato = $costo + $iva ;

echo "Costo senza iva: $costo. IVA: $iva. Costo ivato: $costo\_ivato";

echo $invio;

Prima particolarità importante del php è quella di poter utilizzare le variabili all'interno delle stringhe

echo "Costo senza iva: \$costo. IVA:  \$iva. Costo ivato: \$costo\_ivato";

Con il \ vi è una forzatura a non far considerare una variabile in quanto tale ma come testo

echo $invio;

echo 'Costo senza iva: $costo. IVA: $iva. Costo ivato: $costo\_ivato';

L'utilizzo del'apice singolo comporta la lettura del contenuto solo come stringa

echo $invio;

echo "Costo senza iva: {$costo}€. IVA: {$iva}€. Costo ivato: {$costo\_ivato}€";

L'utilizzo delle {} come delimitatori ci permettono di utilizzare le variabili senza che il loro utilizzo venga interferito con altre stringhe