PHP

Les variables

Une variable est un conteneur ou un espace de stockage temporaire qui va pouvoir stocker une valeur qu'on va lui assigner.

Les règles de déclaration des variables en PHP

Nous allons pouvoir créer différentes variables dans un script pour stocker différentes valeurs et pouvoir les réutiliser simplement ensuite. Lorsqu'on crée une variable en PHP, on dit également qu'on « déclare » une variable.

On va pouvoir choisir le nom qu'on souhaite donner à chacune de nos variables. Cependant, il y a quelques règles à respecter et à connaitre lors de la déclaration d'une nouvelle variable :

- Toute variable en PHP doit commencer par le signe $ qui sera suivi du nom de la variable ;
- Le nom d'une variable doit obligatoirement commencer par une lettre ou un underscore (_) et ne doit pas commencer par un chiffre ;
- Le nom d'une variable ne doit contenir que des lettres, des chiffres et des underscores mais pas de caractères spéciaux ;
- Le nom d'une variable ne doit pas contenir d'espace.

De plus, notez que le nom des variables est **sensible à la casse** en PHP. Cela signifie que l'usage de majuscules ou de minuscules va créer des variables différentes. Par exemple, les variables $texte, $TEXTE et $tEXTe vont être des variables différentes.

Par convention, si un nom de variable possède plusieurs mots nous préférerons le camel case par exemple :

$monPremierParagraphe

Les types de données en PHP

Les variables PHP vont pouvoir stocker différents types de valeurs, comme du texte ou un nombre par exemple. Par abus de langage, nous parlerons souvent de « types de variables » PHP.

En PHP, contrairement à d'autres langages de programmation, nous n'avons pas besoin de préciser à priori le type de valeur qu'une variable va pouvoir stocker. Le PHP va automatiquement détecter quel est le type de la valeur stockée dans telle ou telle variable, et nous allons ensuite pouvoir performer différentes opérations selon le type de la variable.

Une conséquence directe de cela est qu'on va pouvoir stocker différents types de valeurs dans une variable au fil du temps sans se préoccuper d'une quelconque compatibilité. Par exemple, une variable va pouvoir stocker une valeur textuelle à un moment dans un script puis un nombre à un autre moment.

Les variables en PHP vont pouvoir stocker 8 grands types de données différents :

- Le type « chaine de caractères » ou `String` en anglais ;
- Le type « nombre entier » ou `Integer` en anglais ;
- Le type « nombre décimal » ou `Float` en anglais ;
- Le type « booléen » ou `Boolean` en anglais ;
- Le type « tableau » ou `Array` en anglais ;
- Le type « objet » ou `Object` en anglais ;
- Le type « NULL » qui se dit également `NULL` en anglais ;
- Le type « ressource » ou `Resource` en anglais ;

Le type chaîne de caractères ou String

Le premier type de données qu'une variable va pouvoir stocker est le type `String` ou chaîne de caractères. Une chaine de caractères est une séquence de caractères, ou ce qu'on appelle communément un texte.

Notez que toute valeur stockée dans une variable en utilisant des guillemets ou des apostrophes sera considérée comme une chaine de caractères.

$presentation = “Je suis une marmotte”;

$tempsHibernation = “6”;

L'utilisation de guillemets ou d'apostrophe fait qu'une valeur est immédiatement considérée comme une chaine de caractères, quelle que soit cette valeur.

Les types de données nombre entier (Integer) et nombre décimal (Float ou Double)

En PHP, on va pouvoir stocker deux types différents de donnée numéraires dans nos variables : le type Integer, qui contient tous les nombres entiers positifs ou négatifs et le type Float ou Double, qui contient les nombres décimaux (nombres à virgule) positifs ou négatifs.

On va donc pouvoir stocker un entier ou un nombre décimal dans une variable. Pour cela, il suffit d'affecter le nombre à stocker à notre variable, sans guillemet ni apostrophe.

Attention cependant : lorsque l'on code, on utilise toujours les notations anglo-saxonnes. Ainsi, il faudra préciser des points à la place de nos virgules pour les nombres relatifs.

$tempsHibernation = 6;

$dsitanceParcourue = 56.7;

Le type de données booléen (Boolean)

Une variable en PHP peut stocker une valeur de type booléen (Boolean en anglais).

Le type booléen est un type qui ne contient que deux valeurs : les valeurs true (vrai) et false (faux). Ce type n'est pas courant dans la vie de tous les jours mais est très (très) utilisé en informatique.

$aHibernee = true;

Nous aurons le temps d'utiliser les booléen dans plusieurs cas d'utilisation par la suite.

Le type de données Null

Le type de données Null est un type un peu particulier puisqu'il correspond à l'absence de valeur et sert donc à représenter des variables vides en PHP.

Ce type de valeur ne contient qu'une seule valeur : la valeur NULL qui correspond elle-même à l'absence de valeur.

Notez que si vous déclarez une nouvelle variable sans lui affecter de valeur (ce qui est déconseillé de manière générale), cette variable sera automatiquement de type Null.

```
$vide = NULL;
```

Les types de données PHP tableau (Array) et objet (Object)

Les types de données Array et Object sont des types de données complexes particuliers qui méritent de faire chacun l'objet de chapitres séparés.

Nous n'étudierons donc pas ces deux types pour le moment car n'avons pas les connaissances suffisantes pour bien les comprendre.

Sachez simplement que l'on va pouvoir stocker plusieurs valeurs d'un coup à l'intérieur d'une variable en lui assignant des valeurs de type Array (tableau) ou Object (objet).

Le type de données ressource (Resource)

Une ressource est une variable particulière qui contient une référence vers une ressource externe au PHP, comme dans le cas d'une variable qui représente la connexion vers une base de données par exemple.

Là encore, ce type de données est complexe et nécessite d'avoir une bonne vision d'ensemble du langage pour être bien compris. Nous l'étudierons donc plus tard.

Opérateur et concaténation

Un opérateur est un symbole qui va être utilisé pour effectuer certaines actions notamment sur les variables et leurs valeurs.

Par exemple, l'opérateur + va nous permettre d'additionner les valeurs de deux variables, tandis que l'opérateur = va nous permettre d'affecter une valeur à une variable.

La documentation officielle de PHP classe les différents opérateurs qu'on va pouvoir utiliser selon les groupes suivants :

- Opérateurs arithmétiques ;
- Opérateurs d'affectation ;
- Opérateurs sur les bits ;
- Opérateurs de comparaison ;
- Opérateur de contrôle d'erreur ;
- Opérateur d'exécution ;
- Opérateurs d'incrémentation et décrémentation ;
- Opérateurs logiques ;
- Opérateurs de chaînes ;
- Opérateurs de tableaux ;
- Opérateurs de types ;

Dans un premier temps, nous allons nous concentrer sur les opérateurs arithmétiques, les opérateurs de chaines et les opérateurs d'affectation.

Les opérateurs de chaines et la concaténation en PHP

Concaténer signifie littéralement « mettre bout à bout ». L'opérateur de concaténation qui est le point (.) va donc nous permettre de mettre bout à bout deux chaines de caractères.

Cet opérateur va s'avérer particulièrement utile lorsqu'on voudra stocker le contenu de plusieurs variables qui stockent des données de type chaine de caractères ou pour afficher différentes données au sein d'une même instruction echo.

Pour bien comprendre comment fonctionne l'opérateur de concaténation et son intérêt, il est nécessaire de connaitre les différences entre l'utilisation des guillemets et des apostrophes lorsqu'on manipule une chaine de caractères en PHP.

Sur ce sujet, vous pouvez retenir que la différence majeure entre l'utilisation des guillemets et d'apostrophes est que tout ce qui est entre guillemets va être interprété tandis que quasiment tout ce qui est entre apostrophes va être considéré comme une chaine de caractères.

Ici, « interprété » signifie « être remplacé par sa valeur ». Ainsi, lorsqu'on inclut une variable au sein d'une chaine de caractères et qu'on cherche à afficher le tout avec un echo et en utilisant des guillemets, la variable va être remplacée par sa valeur lors de l'affichage.

C'est la raison pour laquelle il faut échapper le $ si on souhaite afficher le nom de la variable comme chaine de caractères plutôt que sa valeur.

En revanche, lorsqu'on utilise des apostrophes, les variables ne vont pas être interprétées mais leur nom va être considéré comme faisant partie de la chaine de caractères.

Créons 2 variables nom et prénom de type string et une variable age de type integer et affichons les avec des instructions echo, comme ceci :

```
echo "Bonjour je m'appelle $prenom $nom j'ai $age ans <br>";
```

```
echo "Bonjour je m'appelle {$prenom} {$nom} j'ai {$age} ans <br>";
```

```
echo 'Bonjour je m\'appelle $prenom $nom j\'ai $age ans <br>';
```

Nous pouvons aussi stocker ces phrases dans des variables comme ceci :

```
$presentation = "Bonjour je m'appelle $prenom $nom j'ai $age ans <br>";
```

Exercice 1 :

Affecter les 2 autres phrases dans deux autres variables comme sur le modèle précédant.

Et les afficher comme ceci :

```
echo $presentation;
```

Exercice 2 :

Afficher les 2 autres variables.

Que constez vous lors de l'affichage du rendu dans le navigateur ?

Pour notre premier `echo`, on utilise des guillemets pour entourer le texte. Les variables dans le texte vont être interprétées et c'est leur contenu qui va être affiché.

Notez cependant ici que la syntaxe avec les noms de variables directement au milieu du texte est déconseillée aujourd'hui et qu'on préfèrera utiliser la syntaxe de de notre deuxième `echo` qui utilise des accolades pour entourer les variables.

Dans notre troisième `echo`, on utilise cette fois-ci des apostrophes. Les noms des variables ne vont donc pas être interprétés mais être considérés comme du texte et s'afficher tel quel.

Finalement, on crée de la même façon trois variables `$presentation`, `$presentation2` et `$presentation3` qui stockent à nouveau du texte au sein duquel on inclut les noms de nos variables.

On `echo` alors le contenu de nos trois variables. Sans surprise, les variables `$presentation` et `$presentation2` stockent le texte donné avec le contenu des variables `$prenom`, `$nom` et `$age` tandis que la variable `$presentation3` stocke le nom de ces variables plutôt que leurs valeurs.

L'opérateur de concaténation va nous permettre de mettre bout à bout les différentes données tout en faisant en sorte que chaque donnée soit interprétée par le PHP.

Nous allons l'utiliser pour séparer nos différentes variables des chaines de caractères autour. Regardez l'exemple suivant pour bien comprendre :

```
$presentation4 = 'Bonjour je m\'appelle '.$prenom.' '.$nom.' j\'ai '.$age.' ans <br>';

$presentation5 = "Bonjour je m'appelle ".$prenom." ".$nom." j'ai ".$age." ans <br>";
```

Exercice 3 :

En reprenant la syntaxe ci-dessus afficher les variables `$presentation4 et $presentation5 en` une seule instruction echo qui permet l'affichage sur 2 lignes.

Pour concaténer correctement avec l'opérateur de concaténation, la règle est de séparer les différentes variables avec l'opérateur de concaténation (le point) des textes autour. Chaque texte devra être entouré de guillemets ou d'apostrophes selon ce qu'on a choisi.

On voudra utiliser des apostrophes plutôt que des guillemets et dans ce cas, si on souhaite que certaines de nos variables soient interprétées, il faudra utiliser l'opérateur de concaténation.

De manière générale, il est conseillé de toujours utiliser l'opérateur de concaténation lorsqu'on souhaite mettre bout-à-bout plusieurs chaines de caractères (qui seront généralement séparées par des variables), et ceci qu'on utilise des guillemets ou des apostrophes.

Les opérateurs arithmétiques

Les opérateurs arithmétiques vont nous permettre d'effectuer toutes sortes d'opérations mathématiques entre les valeurs contenues dans différentes variables lorsque ces valeurs sont des nombres.

Le fait de pouvoir réaliser des opérations entre variables va être très utile dans de nombreuses situations. Par exemple, si un utilisateur commande plusieurs produits sur notre site ou plusieurs fois un même produit et utilise un code de réduction, il faudra utiliser des opérations mathématiques pour calculer le prix total de la commande.

En PHP, nous allons pouvoir utiliser les opérateurs arithmétiques suivants :

Opérateur	Nom de l'opération associée	Ordre de priorité des calculs
+	Addition	3
–	Soustraction	3
*	Multiplication	2
/	Division	2
%	Modulo (reste d'une division euclidienne)	2
**	Exponentielle (élévation à la puissance d'un nombre par un autre)	1

Avant d'utiliser les opérateurs arithmétiques, clarifions ce que sont le modulo et l'exponentielle.

Le modulo correspond au reste entier d'une division euclidienne. Par exemple, lorsqu'on divise 5 par 3, le résultat est 1 et il reste 2 dans le cas d'une division euclidienne. Le reste, 2, correspond justement au modulo.

L'exponentielle correspond à l'élévation à la puissance d'un nombre par un autre nombre.
La puissance d'un nombre est le résultat d'une multiplication répétée de ce nombre par lui-même. Par exemple, lorsqu'on souhaite calculer 2 à la puissance de 3 (qu'on appelle également « 2 exposant 3 »), on cherche en fait le résultat de 2 multiplié 3 fois par lui-même c'est-à-dire 2*2*2 = 8.

Exercice 4 :

Sans votre IDE :

```php
$x = 2;
$y = 3;
$z = 4;
echo '$x stocke ' .$x. ', $y stocke ' .$y. ', $z stocke ' .$z. '<br>';
```

Que devrait afficher cette instruction ?

```php
$a = $x + 1; //$a stocke 2 + 1 soit 3
$b = $x + $y; //$b stocke 2 + 3 soit 5
$c = $x - $y; //$c stocke 2 - 3 soit -1
echo '$a stocke ' .$a. ', $b stocke ' .$b. ', $c stocke ' .$c. '<br>';
```

Que devrait afficher cette instruction ?

```php
$x = $x * $y; //$x stocke désormais 2 * 3 soit 6
echo 'La variable $x stocke désormais : ' .$x. '<br>';
```

Que devrait afficher cette instruction ?

```php
$z = $x / $y; //$z stocke désormais 6 / 3 soit 2
echo 'La variable $z stocke désormais : ' .$z. '<br>';
```

Que devrait afficher cette instruction ?

```php
$m = 5 % 3; //$m stocke le reste de la division euclidienne de 5 par 3
echo 'Le reste de la division euclidienne de 5 par 3 est ' .$m. '<br>';
```

Que devrait afficher cette instruction ?

```php
$p = $z ** 4; //$p stocke 2^4 = 2 * 2 * 2 * 2 soit 16
echo 'La variable $p stocke le résultat de 2 puissance 4 = ' .$p;
```

Que devrait afficher cette instruction ?

Concernant les règles de calcul, c'est-à-dire l'ordre de priorité des opérations, celui-ci va être le même qu'en mathématiques : l'élévation à la puissance va être prioritaire sur les autres opérations, tandis que la multiplication, la division et le modulo vont avoir le même ordre de priorité et être prioritaires sur l'addition et la soustraction qui ont également le même niveau de priorité.

Si deux opérateurs ont le même ordre de priorité, alors c'est leur sens d'association qui va décider du résultat. Pour les opérateurs arithmétiques, le sens d'association correspond à l'ordre de leur écriture à l'exception de l'élévation à la puissance qui sera calculée en partant de la fin.

Ainsi, si on écrit $x = 1 – 2 – 3, la variable $x va stocker la valeur -4 (les opérations se font de gauche à droite). En revanche, si on écrit $x = 2 ** 3 ** 2, la variable $x stockera 512 qui correspond à 2 puissance 9 puisqu'on va commencer par calculer 3 ** 2 = 9 dans ce cas.

Exercice 5 :

Sans votre IDE : Que devrait afficher l'instruction echo ci-dessous ?

```php
$x = 2 + 3 * 4;
$y = (2 + 3) * 4;
$z = 2 ** 3 - 4 * 4 / 8;
echo '$x : ' .$x. '<br>$y : ' .$y. '<br>$z : ' .$z;
```

Ici, $x stocke la valeur 14. En effet, la multiplication est prioritaire sur l'addition. On va donc commencer par faire 3 * 4 puis ajouter 2 au résultat.

La variable $y stocke 20. En effet, on utilise des parenthèses pour forcer la priorité de l'addition par rapport à la multiplication.

Finalement, $z stocke la valeur 6. En effet, on commence ici par calculer 2 puissance 3 (2 * 2 * 2 = 8). Ensuite, on calcule 4 * 4 / 8 = 16 / 8 = 2 car la multiplication et la division sont prioritaires sur la soustraction. Finalement, on calcule 8 – 2 = 6.

Notez également que les opérateurs + et - peuvent également servir à convertir le type de valeur contenue dans une variable vers Integer ou Float selon ce qui est le plus approprié.

Cette utilisation des opérateurs va pouvoir nous être utile lorsqu'on aura variables contenant des « nombres » stockés sous le type de chaines de caractères et pour lesquelles on voudra réaliser des opérations mathématiques.

Les opérateurs d'affectation et opérateurs combinés

Les opérateurs d'affectation vont nous permettre, comme leur nom l'indique, d'affecter une certaine valeur à une variable.

Nous connaissons déjà bien l'opérateur d'affectation le plus utilisé qui est le signe =. Cependant, vous devez également savoir qu'il existe également des opérateurs combinés notamment pour les opérateurs arithmétiques et l'opérateur de concaténation et qui sont les suivants :

Opérateur	Définition
.=	Concatène puis affecte le résultat
+=	Additionne puis affecte le résultat
-=	Soustrait puis affecte le résultat
*=	Multiplie puis affecte le résultat
/=	Divise puis affecte le résultat
%=	Calcule le modulo puis affecte le résultat
**=	Élève à la puissance puis affecte le résultat

Exemple :

```php
$a = "Bonjour";
$a .= " le monde"; //$a stocke "Bonjour le monde"
echo '$a stocke : ' .$a. '<br>';
$x = 5;
$x -= 3; //$x stocke désormais 2
echo '$x stocke : ' .$x. '<br>';
$y = 3;
$y **= $x; //$y stocke 3^2 = 3 * 3 = 9
echo '$y stocke : ' .$y;
```

Par ailleurs, notez que tous les opérateurs d'affectation ont une priorité de calcul égale mais qui est inférieure à celle des opérateurs arithmétiques ou de concaténation.

Lorsque des opérateurs ont des ordres de priorité égaux, c'est le sens d'association de ceux-ci qui va décider du résultat. Pour les opérateurs arithmétiques, on a vu que l'association se

faisait par la gauche sauf pour l'élévation à la puissance. Pour **les opérateurs d'affectation, l'association se fait par la droite.**

```
$x = 1;
$y = 2;
$z = 3;
$a = 5;
$x = $z += 2;
echo '$x stocke : ' .$x. ' et $z stocke : ' .$z. '<br>';
Que devrait afficher l'instruction echo ci dessus ?

$y += $z -= 2;
echo '$y stocke : ' .$y. ' et $z stocke : ' .$z. '<br>';
Que devrait afficher l'instruction echo ci dessus ?

$y /= $z -= 2; //$z stocke 1 et $y stocke 5
echo '$y stocke : ' .$y. ' et $z stocke : ' .$z. '<br>';
$a *= 4 + 2; //$z stocke 30
echo '$a stocke : ' .$a;
```

Pour notre premier calcul, nous utilisons les deux opérateurs d'affectation = et +=. L'association va se faire par la droite. On commence donc à ajouter 2 à la valeur de **$z qui stocke désormais 5** et on stocke la même valeur dans $x. Faites bien attention ici : $x ne stocke bien évidemment pas la variable $z mais seulement la dernière valeur connue de $z. Si on modifie ensuite la valeur de $z, cela n'a aucun impact sur $x.

Les deux exemples suivants utilisent à nouveau deux opérateurs d'affectation. L'association va donc toujours se faire par la droite.

Dans notre dernier exemple, cependant, on utilise à la fois un opérateur d'affectation et un opérateur arithmétique. Les opérateurs arithmétiques sont prioritaires sur les opérateurs d'affectation. On commence donc par réaliser l'opération arithmétique (4 + 2 = 6) et on multiplie ensuite la valeur de $a par 6 avant d'affecter la nouvelle valeur dans $a.