



PHP & SQL - Bases - 2 - Variables, Constantes & Données

⌚ Cours	PHP
⌚ Type	Guide
⌚ Supports	https://www.php.net/manual/fr/langref.php
⌚ Difficulté	Très Facile

Les variables

Lorsqu'on écrit un programme, il est utile de pouvoir sauvegarder des valeurs pour les utiliser à plusieurs reprises. Par exemple, lorsque vous souhaitez afficher le nom d'une personne, il est préférable de sauvegarder ce nom dans une variable afin de pouvoir le modifier une seule fois et de voir le nom changer à tous les endroits où nous l'avons écrit.

Si ce n'est pas encore clair pour vous, ça le sera forcément avec un peu de pratique.

Commençons par écrire ce script :

```
<?php  
$message = 'Salut';  
  
echo $message;  
echo $message;  
echo $message;
```

Vous voyez où je veux en venir ?

Normalement, sur votre page index.php, vous devez vous retrouver avec ceci :

SalutSalutSalut

Imaginez que vous avez un programme complexe qui affiche des milliards de fois votre message (ce serait totalement inutile mais vulgarisons un peu pour le moment). Est-ce que vous vous imaginez taper ce message des milliards de fois ? Non, bien sûr, vous allez simplement créer une variable. Essayez donc de modifier le message par vous-même afin d'obtenir autre chose comme :

C'est incroyable ce qu'on peut faire avec PHP. Regarde maman, je suis développeur ! C'est incroyable ce qu'on peut faire avec PHP. Regarde maman, je suis développeur ! C'est incroyable ce qu'on peut faire avec PHP. Regarde maman, je suis développeur !

Vous l'avez peut-être compris précédemment. Pour écrire une variable, il suffit d'écrire un `$` suivi d'un mot (de votre choix). Ainsi, vous pouvez créer les variables :

`$voiture` , `$laProgrammationCestSuper` , `$aQuoiCaSertToutCa` du moment que vous n'utilisez pas de chiffres en premier caractère, ni d'espaces ou de caractères spéciaux (pas d'accent, pas de etc. sauf le `_`), vous pourrez en créer à l'infini !

⚠ Pour déclarer des variables, il faut respecter certaines règles de nommage. Le premier caractère doit être obligatoirement en minuscule. Lorsque nous avons plusieurs mots dans la variable, il faut respecter le camelCase (qui consiste à écrire commeCaEnGrosTuMetsUneMajusculeAChaqueNouveauMot)

Après avoir écrit une variable, il faut lui affecter une valeur, c'est à dire ce que la variable va contenir. On peut ici imaginer que l'identifiant de la variable est le nom d'un carton, et que la valeur est l'intérieur de ce carton.

Par exemple :

```
<?php  
$livre = 'Victor Hugo - Les misérables';
```

Si l'on souhaite afficher le livre, il suffira donc de faire un simple `echo $livre;`

i Si vous essayez de faire un `echo $Livre;` PHP vous retournera l'erreur suivante :
Warning: Undefined variable \$Livre
in C:\xampp\htdocs\Decouverte\index.php on line 5
Ceci est dû au fait que vous avez mis une majuscule. Les variables sont sensibles à la casse.

Vous pouvez d'ailleurs modifier la valeur au cours du script :

```
<?php  
$livre = 'Victor Hugo - Les misérables';  
$livre = 'Margaret Atwood - La servante écarlate';  
  
echo $livre;
```

Essayez donc pour voir quel livre va s'afficher.

Exercice 1 :

Créez une variable permettant de stocker votre nom ainsi qu'une variable permettant de stocker votre prénom et demandez à PHP d'afficher votre nom suivi de votre prénom.

Il se peut que votre résultat ne soit pas espacé. Je vous laisse chercher par vous-mêmes. À l'avenir, je vous laisserai de plus en plus chercher. Être développeur, c'est aussi être capable de chercher par soi-même et de lire et comprendre une documentation.

Sachez d'ailleurs que pour chaque exercice à venir, vous pouvez m'envoyer les résultats pour obtenir la correction. 😊

Les constantes

Les variables sont modifiables, ce qui peut nous poser problème si nous souhaitons définir des valeurs qui seront supposées être toujours vraies. Par exemple, il y a 24h dans une journée, notre satellite s'appelle Lune ou encore "Les humains ont des yeux".

A priori, ces choses ne sont pas prêtes de changer et pour être sur que vous ne faites pas une mauvaise manipulation dans votre code, il est utile d'utiliser les constantes. Ces constantes sont un peu différentes, car elles peuvent se déclarer de deux manières :

```
<?php  
define('NAME', 'Frédéric');  
  
echo NAME;
```

```
<?php  
const NAME = 'Frédéric';  
  
echo NAME;
```

Nous verrons plus tard, pourquoi il est possible de les définir différemment. Mais essayez de retenir la première pour le moment car elle est un peu plus difficile à utiliser.

Le mot clé `define()` est une méthode. Nous verrons le sens de ce mot plus tard, mais sachez qu'en gros, PHP va affecter une valeur à une constante grâce à ce mot. Ensuite, entre parenthèses, c'est à vous de spécifier le nom de la constante entre guillemets, suivi d'une virgule pour le séparer de sa valeur.

Notez d'ailleurs que les constantes s'écrivent par convention en MAJUSCULES. Si vous souhaitez faire des mots composés, il faudra les déclarer `COMME_CECI_D_ACCORD`

Les Types de données

Jusqu'à présent, nous avons utilisé uniquement des chaînes de caractères lorsque nous avons déclaré nos variables.

Les chaînes de caractères sont appelées String en anglais. D'ailleurs, sachez qu'après ce chapitre, nous programmerons uniquement en anglais pour respecter les bonnes pratiques.

Il existe également tout un tas d'autres types de données :

Type	Signification	Exemple
String	Chaîne de caractères	\$name = "Victor"
Integer	Nombre entier	\$age = 12;
Float	Nombre à virgule	\$size= 1.70
Boolean	Vrai ou Faux	\$victory = true

Les string :

Comme vous l'avez déjà vu, les string sont des chaînes de caractères. Il existe plusieurs façons de déclarer les chaînes de caractères. Vous pouvez les déclarer à l'aide de guillemets simples et de guillemets doubles.

Pourquoi ?

Tout simplement parce que vous avez une différence de traitement entre les deux :

```
<?php  
$message = 'J'ai de l'avance sur mon travail.';
```

&

```
<?php  
$message = "J'ai de l'avance sur mon travail.";
```

Ces deux scripts ne sont pas interprétés de la même manière. Le premier pensera que vous avez terminé la valeur de votre variable après le J et vous retournera sans doute une erreur car il ne comprendra pas pourquoi il y a du texte superflu ensuite.

Le deuxième cas vous permettra d'utiliser les guillemets simples au sein des guillemets doubles.

D'ailleurs, il existe une méthode pour pouvoir utiliser les deux types de guillemets à l'intérieur de la même chaîne de caractères :

```
<?php  
$message = 'Je souhaiterais vous présenter l\'autre méthode : \"Tadaa!\" \n';
```

Essayez donc d'afficher ce message plusieurs fois. Vous verrez que non seulement il s'affiche bien, mais en plus, il retourne à la ligne !

C'est grâce au `\` (à ne pas confondre avec `/`) qui permet de spécifier que la caractère qui suit doit être “échappé”.

Le `\n` quant à lui est une séquence d'échappement. Celle-ci sert simplement à dire : Retourne à la ligne.

 Il y a une petite différence d'interprétation entre le simple guillemet et le double-guillemet. Le simple exécutera la chaîne de caractères telle quelle. Ce qui veut dire que vous ne pouvez pas appeler de variable au sein d'une chaîne entre double guillemet. Le code l'interprétera simplement comme du texte. À l'inverse, si vous appelez une variable au sein d'une chaîne de caractères étant définie entre simple guillemet, la valeur de la variable sera bien affichée.

Il est possible de “concaténer” plusieurs strings ensemble en utilisant le `.` qui est lu. Essayez donc :

```
<?php  
$message = "Salut";  
$message2 = "Martin";  
  
echo $message . " " . $message2;
```

Les integer :

Les integer (ou int) sont tout simplement les nombres entiers.

Il vous suffit simplement d'écrire votre variable et d'y affecter une valeur sans rien ajouter :

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

 Il est possible d'écrire de très grands nombres et de les espacer pour plus de visibilité en utilisant le séparateur underscore `_`. Exemple : `8_000_000`

Les float :

Les float sont tout simplement des nombres à virgules.

La virgule s'écrit avec un point en programmation :

```
<?php  
$nombre = 12.4;
```

 Si l'on fait la distinction entre nombres à virgule et nombres entiers, c'est parce que les nombres à virgule peuvent contenir beaucoup + de valeurs que les entiers et utiliseront donc + de mémoire. Il vous est donc conseillé d'utiliser les types de données appropriés dans la majorité des langages de programmation.

 Lorsque vous effectuez des opérations avec les nombres en PHP, il se peut que votre résultat soit parfois un peu différent de la réalité. Ceci est dû au fait que la capacité de nombre de PHP n'est pas infinie, ce qui peut causer des erreurs au moment de l'opération. Il existe évidemment des techniques pour éviter ce genre de résultat et la plupart des outils de paiement en ligne ont ces outils déjà intégrés.

Les booléens :

Les booléens sont tout simplement des valeurs Vraies ou Fausses. C'est très utile pour savoir si un utilisateur a bien coché ou non une case, s'il est bien connecté ou non, etc.

On peut les déclarer seulement en utilisant le mot clé True ou False :

```
<?php  
$win = true;  
$win = false;
```

Vous pouvez au passage écrire True ou false comme vous le souhaitez : tRue, TruE, TRUE etc. sera toujours valide pour PHP.

 Il existe d'autres types de données qui seront bien moins souvent utilisés et donc moins importants comme les binaires et les hexadécimaux.

J'ai volontairement omis certains types de données qui sont très utilisés pour vous simplifier la vie. On verra donc d'autres types importants plus tard ! 😊

Exercice 2

Écrivez un programme permettant d'afficher :

- Votre nom (dans une variable name)
- Votre prénom (dans une variable firstname)
- Votre âge (dans une variable aussi)
- Votre taille (dans une variable aussi)

dans une chaîne de caractères (qui est une nouvelle variable faisant appel aux autres variables aussi ! 😊 pfiouu).

Bonus : Essayez de le faire en utilisant les simples guillemets puis de le faire en utilisant les doubles guillemets.