# Opérateurs et concaténation en PHP

Télécharger le PDF du cours


Cours complet PHP et MySQL



Dans cette nouvelle leçon, nous allons définir ce qu'est un opérateur, établir la liste des types d'opérateurs disponibles en PHP et apprendre à en manipuler certains.

Nous allons également préciser les différences entre l'utilisation des apostrophes ou des guillemets lorsqu'on manipule une valeur de type chaine de caractères.

## Qu'est-ce qu'un opérateur ?

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 :

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

Dans cette leçon, nous allons nous concentrer sur les opérateurs arithmétiques, les opérateurs de chaines et les opérateurs d'affectation.

Nous verrons les autres types d'opérateurs au fil de ce cours lorsque cela fera le plus de sens (c'est-à-dire lorsqu'on en aura besoin).

## Les opérateurs de chaines et la concaténation en PHP

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 me semble 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.

Regardez plutôt l'exemple suivant :

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Titre principal</h1>
        <?php
            $prenom = "Pierre";
            $nom = "Giraud";
            $age = 28;

            echo "Je m'appelle $prenom et j'ai $age ans <br>";
            echo "Je m'appelle {$prenom} et j'ai {$age} ans <br>";
            echo 'Je m\'appelle $prenom et j\'ai $age ans <br>';

            $prez = "Je suis $prenom $nom, j'ai $age ans <br>";
            $prez2 = "Je suis {$prenom} {$nom}, j'ai {$age} ans <br>";
            $prez3 = 'Je suis $prenom $nom, j\'ai $age ans';

            echo $prez;
            echo $prez2;
            echo $prez3;
        ?>
        <p>Un paragraphe</p>
```

Ici, nous déclarons trois variables $prenom, $nom et $age.

On essaie ensuite d'afficher du texte avec des echo en incluant nos noms de variables au sein du texte.

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 $prez, $prez2 et $prez3 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 $prez et $prez2 stockent le texte donné avec le contenu des variables $prenom, $nom et $age tandis que la variable $prez3 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 :

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>Cours PHP & MySQL</title>
5.          <meta charset="utf-8">
```

```
10.         <h1>Titre principal</h1>
11.         <?php
12.             $prenom = "Pierre";
13.             $nom = "Giraud";
14.             $age = 28;
15.             $prez = "Je suis " .$prenom. " " .$nom. ", j'ai " .$age. " ans";
16.             $prez2 = 'Je suis ' .$prenom. ' ' .$nom. ', j\'ai '.$age. ' ans';
17.
18.
19.             echo "Je m'appelle " .$prenom. " et j'ai " .$age. " ans <br>";
20.             echo 'Je m\'appelle ' .$prenom. ' et j\'ai ' .$age. ' ans <br>';
21.
22.             echo $prez. '<br>' .$prez2;
23.         ?>
24.         <p>Un paragraphe</p>
25.     </body>
26. </html>
```

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.

A ce niveau, il est probable que vous vous demandiez l'intérêt d'utiliser l'opérateur de concaténation qui semble ici compliquer inutilement le code plutôt que simplement des guillemets et des accolades.

Ma réponse va être très simple : ici, vous pouvez utiliser l'une ou l'autre de ces méthodes pour un résultat identique. Cependant, rappelez-vous que l'utilisation d'apostrophes ou de guillemets n'est pas identique au sens où ce qui est entre guillemets va être interprété tandis que la grande majorité de ce qui est entre apostrophes ne le sera pas.

Ainsi, parfois, 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

## 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 |
|---|---|
| + | Addition |
| – | Soustraction |
| * | Multiplication |
| / | Division |
| % | Modulo (reste d'une division euclidienne) |
| ** | Exponentielle (élévation à la puissance d'un nombre par un autre) |

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.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

```

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

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

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

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

            $m = 5 % 3; //$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>';

            $p = $z ** 4; //$p stocke 2^4 = 2 * 2 * 2 * 2 = 16
            echo 'La variable $p stocke le résultat de 2 puissance 4 = ' .$p;
        ?>
        <p>Un paragraphe</p>
    </body>
 </html>
```

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.

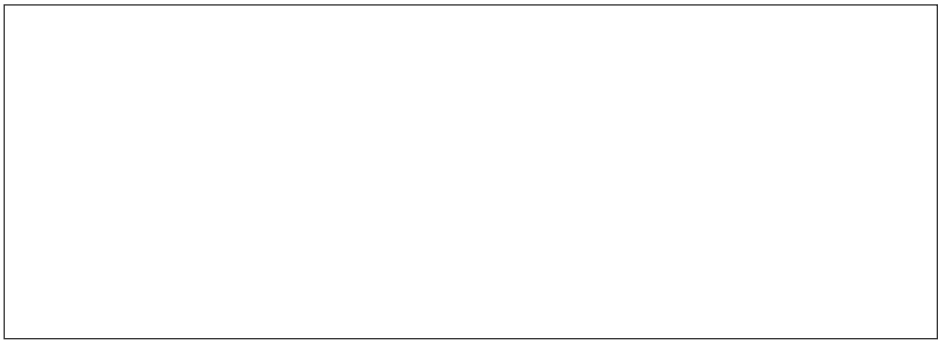
Ainsi, si on écrit $x = 1 – 2 – 3, la variable $x va stocker la valeur -4 (les opérations se font de gauche à droite). En revanche, si on écrit $x = 2 ** 3 ** 2, la variable $x stockera 512 qui correspond à 2 puissance 9 puisqu'on va commencer par calculer 3 ** 2 = 9 dans ce cas.

Nous allons finalement, comme en mathématiques, pouvoir forcer l'ordre de priorité en utilisant des couples de parenthèses pour indiquer qu'une opération doit se faire avant toutes les autres :

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Titre principal</h1>
        <?php
            $x = 2 + 3 * 4; //$x stocke 14
            $y = (2 + 3) * 4; //$y stocke 20
            $z = 2 ** 3 - 4 * 4 / 8; //$z stocke 6

            echo '$x : ' .$x. '<br>$y : ' .$y. '<br>$z : ' .$z;
        ?>
        <p>Un paragraphe</p>
    </body>
</html>
```



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.

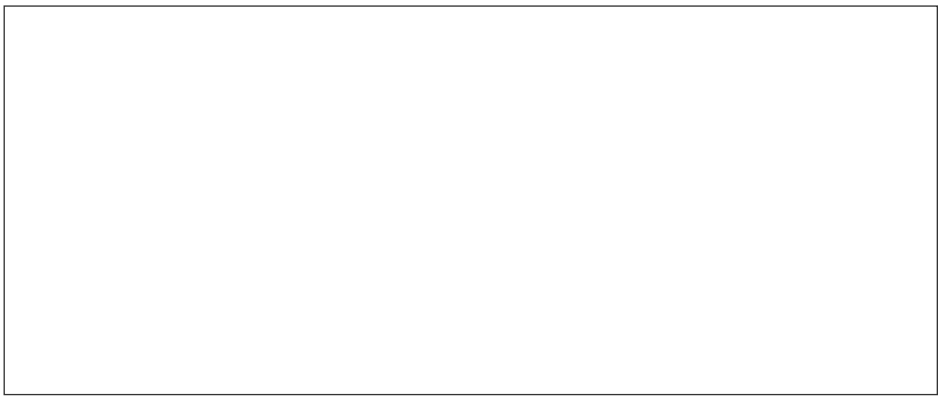
Finalement, $z stocke la valeur 6. En effet, on commence ici par calculer 2 puissance 3 (2 * 2 * 2 = 8). Ensuite, on calcule 4 * 4 / 8 = 16 / 8 = 2 car la multiplication et la division sont prioritaires sur la soustraction. Finalement, on calcule 8 – 2 = 6.

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. Nous aurons l'occasion de rencontrer ce cas plus tard dans ce cours.

```
<!DOCTYPE html>
<html>
```

```
7.      </head>
8.
9.      <body>
10.         <h1>Titre principal</h1>
11.         <?php
12.             $x = "2";
13.             $y = "3.14";
14.
15.             echo '$x stocke la valeur ' .$x. ' de type ' .gettype($x).
    '<br>';
16.             echo '$y stocke la valeur ' .$y. ' de type ' .gettype($y).
    '<br>';
17.
18.             $x = +$x;
19.             $y = -$y;
20.             $z = +"3";
21.
22.             echo '$x stocke la valeur ' .$x. ' de type ' .gettype($x).
    '<br>';
23.             echo '$y stocke la valeur ' .$y. ' de type ' .gettype($y).
    '<br>';
24.             echo '$z stocke la valeur ' .$z. ' de type ' .gettype($z);
25.         ?>
26.         <p>Un paragraphe</p>
27.     </body>
28. </html>
```



## 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 |

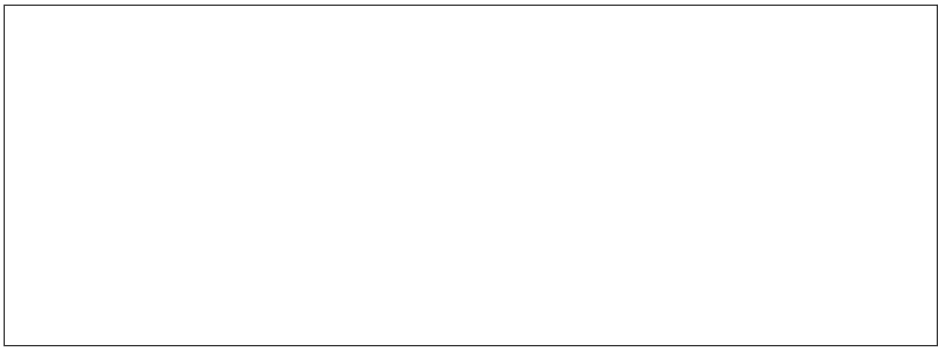
Illustrons immédiatement cela et voyons comment se servir de ces opérateurs :

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Titre principal</h1>
        <?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;
        ?>
        <p>Un paragraphe</p>
    </body>
</html>
```



Ce qu'il faut bien comprendre dans l'exemple précédent est que les opérateurs d'affectation combinés font deux choses à la fois : ils exécutent une opération puis ils affectent une valeur.

Au début, notre variable $a stocke Bonjour. Ensuite, on utilise l'opérateur d'affectation concaténant .= qui va concaténer la valeur à droite avec la valeur contenue dans la variable à gauche avant de lui affecter le résultat.

Ici, on concatène donc la chaine de caractères le monde avec la valeur Bonjour et on affecte le résultat (c'est-à-dire les deux chaines concaténées) dans la variable $a. La variable $a va donc désormais stocker Bonjour le monde.

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.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Titre principal</h1>
        <?php
            $x = 1;
            $y = 2;
            $z = 3;
            $a = 5;

            $x = $z += 2; //$z stocke 5 et $x stocke 5
            echo '$x stocke : ' .$x. ' et $z stocke : ' .$z. '<br>';

            $y += $z -= 2; //$z stocke 5 - 2 = 3 et $y stocke 2 + 3 = 5
            echo '$y stocke : ' .$y. ' et $z stocke : ' .$z. '<br>';

            $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;
        ?>
        <p>Un paragraphe</p>
    </body>
</html>
```

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

Précédent Suivant

## Laisser un commentaire

Vous devez vous connecter pour publier un commentaire.