

PHP & SQL - Bases - 4 - Les Opérateurs et la Condition

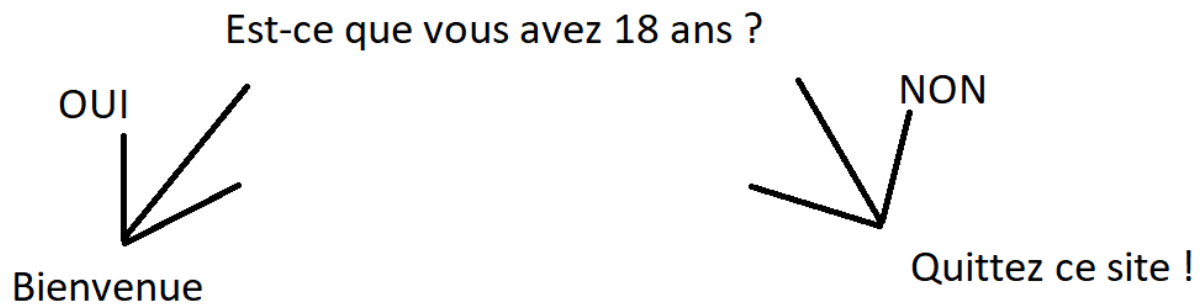
📄 Cours	PHP
📄 Type	Guide
🔗 Supports	https://www.php.net/manual/fr/language_operators.php
📄 Difficulté	Facile

Normalement, si vous avez déjà vu l'algorithmie, le mot condition (ou structure de contrôle) devrait vous parler.

On va utiliser ces structures pour découvrir ensemble les opérateurs et manipuler votre code afin d'écrire les prémices d'un premier programme !

Vous connaissez sans doute la règle, vous n'avez pas le droit à l'alcool avant 18 ans (théoriquement).

Le programme d'un site de vente d'alcools se présentera donc comme suit :



L'objectif de ce programme ultra-sécurisé (🤖) est de s'assurer que vous ayez l'âge requis pour commander des boissons. Voici à quoi ce programme ressemblerait en PHP :

```
<?php
// Amusez-vous à changer l'âge pour voir le résultat changer sur la page
$age = 21;

if ($age >= 18){
    echo "Vous pouvez entrer."; // Résultat qui sera retourné dans ce cas.
} else {
    echo "Sortez immédiatement";
}
```

Le mot clé `if` est une expression permettant de définir une condition. Ce qui est à l'intérieur des accolades `{ ... }` s'exécutera si la condition demandée est vraie. Dans notre cas, on souhaite vérifier si `$age` est supérieur ou égal à 18. `$age` étant égal à 21, on entre à l'intérieur du bloc d'instructions pour exécuter le code à l'intérieur.

Dans le cas de `else`, c'est un mot clé permettant lui aussi de définir une condition. Cette condition n'est vraie QUE et seulement SI le `if` précédent est faux. (Oui, on parle toujours de condition vraie, on va éviter de vous retourner le cerveau, mais reprenez bien qu'une condition doit être respectée, donc vraie !)

Si vous changez la valeur de l'âge pour quelque chose comme 17 ou moins, vous aurez alors le premier bloc (`if`) qui sera `false`. La condition du bloc `else` sera donc `true` (il est vrai qu'elle est bien fausse cette condition 🤖).

i Le bloc `else` est optionnel. Vous pouvez tout simplement décider de ne rien faire si la première condition n'est pas remplie.

Le bloc `else` peut d'ailleurs causer certains problèmes parfois car il veut bien dire : “Dans TOUS les autres cas”. Ce qui peut parfois lancer l'instruction alors que vous ne vous y attendiez pas. Faites donc bien attention à utiliser les `else` consciencieusement.

Exercice 4 :

Écrivez un algorithme permettant d'écrire le message suivant : “Vous êtes accepté” si le prénom de l'utilisateur (supposons que l'utilisateur a rentré cette valeur) s'appelle Kevin. Dans le cas contraire, vous lui demandez de changer de prénom.

Opérateur de chaînes

L'opérateur de chaînes (ou de concaténation) est très simple et vous l'avez déjà aperçu lors du chapitre précédent. Il s'agit du simple `.`

Je vous invite d'ailleurs à cliquer sur le titre pour ouvrir la documentation le concernant qui me paraît personnellement très simple à comprendre. (L'objectif est aussi de vous apprendre à voler de vos propres ailes par la suite !)

Opérateur Arithmétique

Même chose, la documentation est amplement suffisante. Je vous laisse fouiller un peu et tester quelques calculs par vous-mêmes.


i Vous ne pouvez pas calculer avec des string. Faites-donc attention si vous avez déclaré un nombre en chaîne de caractères comme ceci : “42”.

Pour vérifier quel type vous avez fourni à votre calcul, je vous invite à utiliser `var_dump()` à la place de `echo`. Pour le moment, retenez juste que ça fait + de choses que `echo` mais que c'est très similaire. Nous apprendrons à utiliser cette “fonction” par la suite :

```
<?php
$numberOne = 12;
$numberTwo = 12;

$variable = $numberOne + $numberTwo;

var_dump($variable);
```

 Il est possible d'ajouter ou de réduire une valeur en utilisant un raccourci. Si vous souhaitez ajouter un nombre à votre variable, vous penserez sûrement à écrire :

`$number = $number + 4` . Sachez que vous pouvez écrire `$number += 4` à la place.

Opérateurs d'incrémentation et décrémentation

Incrémentation veut dire "+1".

Décrémentation veut dire "-1".

Grâce à ces opérateurs, il est possible d'ajouter ou de réduire une valeur à une variable après l'exécution d'un script. On l'utilise dans de nombreux cas, notamment celui des boucles (que l'on verra par la suite).

On peut facilement imaginer tout un tas de programme où l'on utilise l'incrémentation : Par exemple, un compteur de visiteurs, un compteur de clics, un compteur qui prend -1 à chaque fois qu'un joueur tire une carte dans un jeu-vidéo, etc.

On utilisera le `++` pour incrémenter et le `--` pour décrémentation.

Exercice 5 :

En vous basant sur le code suivant, essayez d'incrémenter le nombre pour qu'il soit égal à 10 et de l'afficher.

```
<?php
$number = 9;

var_dump($number);
```

Opérateurs de comparaison

Rappelez-vous, nous avons utilisé l'un de ces opérateurs au début de ce chapitre pour vérifier l'âge d'un utilisateur.

Je vous laisse une fois de plus parcourir la documentation pour bien les découvrir.

i Il est important de comprendre la différence entre `==` et `===`. Le premier permet de vérifier si les valeurs sont égales (Ce qui veut dire que `42` et `"42"` en chaîne de caractères seront égaux). Le second vérifie si le type est aussi valide. Ce qui dans notre cas n'est pas True.

Je vous invite à tester cette ligne de code dans votre fichier `index.php` :

```
<?php  
  
var_dump(0 == false)
```

Curieux non ? Ce `var_dump()` vous indique ici que vous êtes en train de vérifier une condition (après tout c'est ce que vous faites non ? Vous lui demandez de vous écrire `0 == false` !).

Si vous n'avez pas compris pourquoi il est alors écrit True au moment de l'affichage, c'est tout simplement parce que cette condition est vraie. 0 veut dire "off" (ou false) en langage binaire. On verra plus tard que nous pouvons nous mêmes utiliser la valeur 0 et 1 pour des booléennes (notamment en SQL).

💡 Essayez-donc de comparer `[]` avec `false` puis `0` avec `false`. Il y a une petite subtilité entre l'absence de valeur n'est pas la même chose que 0. Car 0 est aussi une valeur ! 😬

Opérateurs logiques

Les opérateurs logiques permettent de comparer plusieurs conditions entre elles. Par exemple, on peut vérifier si un utilisateur possède bien une carte d'identité ET (`&&`) s'il a 18 ans. On peut aussi vérifier s'il possède soit l'un, soit l'autre (avec le OU qui s'écrit `||` en PHP).

```
<?php
```

```
$john = [  
    'age' => 19,  
    'card' => true  
];  
  
if ($john['age'] >= 18 && $john['card'] === true) {  
    echo "John peut entrer dans la boîte de nuit";  
}
```

Normalement, vous devriez avoir déjà vu les priorités sur la table de vérité sur les opérateurs logiques. Si vous avez besoin d'un rappel, [cliquez-ici](#).

Opérateur Null coalescent

Cet opérateur un peu particulier permet d'exécuter le script même si la chose que vous souhaitez utiliser est nulle en lui précisant quoi faire.

Exemple :

```
<?php  
  
var_dump($test ?? "ça n'existe pas");
```

Je demande à mon programme de m'afficher `$test` mais cette variable n'a été déclarée nulle part. Pour exécuter mon script sans me retourner une erreur, j'utilise l'opérateur "Null" soit `??` et je lui précise ensuite de m'écrire "ça n'existe pas" si et seulement si cette variable n'existe pas.

Nous aurons l'occasion d'utiliser cet opérateur plus tard, mais ça ne fait pas de mal de le découvrir tout de suite.

💡 Il est possible d'affecter une valeur à une variable si celle-ci n'existe pas en utilisant un raccourci : `??=` . Essayez donc par vous-mêmes de faire : `$number ??= 12;`

Ordre de priorité

Sachez qu'il existe un ordre d'exécution entre les différents opérateurs. La documentation peut vous paraître indigeste cette fois, mais en réalité ce n'est pas

bien difficile. Cet ordre fonctionne de la même manière qu'en mathématiques :

Si vous effectuez $2 + 2 * 4$ vous obtiendrez 10 car la multiplication prend la priorité.

En français :

Si vous dites : "J'aimerais avoir des pâtes et de la bolognaise ou une glace", vous aurez soit une glace, soit des pâtes bolo. Cela est dû à la priorité que vous appliquez lorsque vous énoncez cette phrase sur le mot OU.

En programmation, il vous est recommandé d'utiliser des parenthèses pour séparer les priorités non souhaitées :

```
<?php
if (($age >= 18 && $size >= 1.60) || ($card == true) {
    echo "Vous pouvez entrer";
}
```

On fait bien attention à ce que l'utilisateur mesure plus d'1M60 et ait 18 ans ou plus OU alors s'il possède une carte. Si l'on retire les parenthèses, cela donnerait " On vérifie si l'utilisateur a 18 ans ou plus et (s'il fait 1M60 minimum ou alors qu'il a une carte)". Ce qui n'est pas la même chose.

Else if

Après avoir vu un peu en détail les différents opérateurs, il est temps de vous faire découvrir la notion de `else if`. Celui-ci permet de définir une nouvelle condition si la première n'est pas respectée et donc d'itérer plusieurs conditions à la suite dans la même structure.

On peut ainsi traiter chaque cas à part :

```
<?php
if (age >= 18) {
    echo "Vous êtes majeur";
} else if (age == 17) { // traite uniquement le cas 17 ans
    echo "Il vous manque 1 an";
} else if (age == 16) { // traite uniquement le cas 16 ans
    echo "Il vous manque 2 ans";
} else { // traite tout les autres cas
    echo "Vous avez 16 ans ou moins";
}
```

💡 Vous pouvez parfois tomber sur un cas un peu plus difficile à lire que voici :

```
<?php

$age = 19;

$allowAccess = $age >= 18 ? "Accepté" : "refusé";
echo $allowAccess;
```

Cette ligne : `$allowAccess = $age >= 18 ? "Accepté" : "refusé";` est appelé opérateur ternaire. C'est un raccourci permettant de définir une condition en une seule ligne.

Exercice 6 :

Écrivez un programme qui réagit en fonction de l'heure. (Stockez l'heure dans une variable pour le moment, mais supposons que nous avons utilisé une fonction permettant de la deviner).

L'objectif est le suivant :

En fonction de l'heure, affichez ces messages :

- Avant midi: Bonjour, bonne matinée n'est ce pas ?
- Entre midi et 13h : Bon appétit !
- Avant 19h: Bonne après-midi
- sinon : Bonne nuit !