

# **Les opérateurs de comparaison et les expressions booléennes**

# Table des matières

<b>I. Contexte</b>	<b>3</b>
<b>II. Qu'est-ce qu'un booléen ?</b>	<b>3</b>
<b>III. Exercice : Appliquez la notion</b>	<b>5</b>
<b>IV. Opérateurs booléens</b>	<b>7</b>
<b>V. Exercice : Appliquez la notion</b>	<b>12</b>
<b>VI. Opérateurs de comparaison</b>	<b>13</b>
<b>VII. Exercice : Appliquez la notion</b>	<b>16</b>
<b>VIII. Auto-évaluation</b>	<b>17</b>
A. Exercice final.....	17
B. Exercice : Défi.....	19
<b>Solutions des exercices</b>	<b>20</b>

## I. Contexte

**Durée** : 45 min

**Environnement de travail** : Repl.it

**Pré-requis** : Bases de PHP

### Contexte

L'algèbre booléenne est une partie des mathématiques traitant de la logique.

Du point de vue de la programmation informatique, elle est utilisée dans des structures conditionnelles et itératives (boucles) et permet d'indiquer à un programme le "chemin" qu'il doit suivre lors de son exécution.

Pour cela, il compare une ou plusieurs valeurs en regardant si elles sont **vraies** ou **fausses**.

Dans une structure itérative, un booléen pourra servir de point d'arrêt, tandis que, dans une structure conditionnelle, plusieurs booléens pourront former une expression booléenne à tester.

## II. Qu'est-ce qu'un booléen ?

### Objectifs

- Savoir ce qu'est un booléen
- Savoir ce qu'est une expression booléenne

### Mise en situation

Un programme doit répondre à plusieurs cas d'utilisation alternatifs ou se comporter différemment selon des données utilisateurs. Les booléens sont un des moyens de répondre à cette problématique. Ils pourront être utilisés, par exemple, pour répondre à des conditions ou stopper des itérations.

### Définition Booléen

Un booléen est un type de donnée ou de variable pouvant prendre deux valeurs : **vrai** ou **faux**.

Une variable booléenne se verra assigner soit la valeur **true**, soit la valeur **false**. **true** et **false** sont des mots clefs logiques qui représentent des valeurs qui peuvent être manipulées en PHP (affectées, calculées...), à l'instar des entiers ou des chaînes de caractères.

### Exemple

```
1 <?php
2 $var1 = true;
3 $var2 = false;
```

## Syntaxe À retenir

Les booléens sont des informations purement logiques : il n'est pas possible de les afficher simplement en utilisant `echo`. Il faut utiliser la fonction `var_dump` pour visualiser leur contenu.

C'est une fonction fondamentale pour tout développeur PHP ! En effet, contrairement à `echo` qui ne retourne uniquement la valeur d'une variable non booléenne, le `var_dump`, lui, permettra non seulement de visualiser le contenu d'une variable booléenne, mais il affichera en plus son type et sa taille. Il est essentiel à ce qu'on appelle le debug, c'est lorsque nous corrigeons les erreurs du code.

```
1 <?php
2 $var1 = 3;
3 $var2 = "Hello !";
4 $var3 = false;
5
6 echo ($var1); //affiche: 3
7 echo ($var2); //affiche: Hello !
8 echo ($var3); //n'affiche rien car booléen
9 var_dump($var1); //affiche: (integer) 3
10 var_dump($var2); //affiche: (string) 'Hello !' (length=7)
11 var_dump($var3); //affiche: (boolean) false
```

## Rappel

Les booléens sont des informations purement logiques : il n'est pas possible de les afficher simplement en utilisant `echo`. Il faut utiliser la fonction `var_dump` pour visualiser leur contenu.

## Exemple Booléen dans une structure conditionnelle

```
1 <?php
2 $isUserLogged = true;
3
4 if($isUserLogged) {
5     echo 'Bienvenue, utilisateur !'; // Ce code n'est exécuté que si $isUserLogged est vraie,
    // donc si l'utilisateur est authentifié
6 } else {
7     echo 'Bienvenue, inconnu !'; // Ce code n'est exécuté que si $isUserLogged est faux
8 }
```

## Remarque

La structure "**if ... else**" est un élément du langage PHP qui permet d'exécuter certains morceaux de code plutôt que d'autres.

## Complément

Algèbre booléenne<sup>1</sup>

<sup>1</sup> [https://fr.wikipedia.org/wiki/Alg%C3%A8bre\\_de\\_Boole\\_\(logique\)](https://fr.wikipedia.org/wiki/Alg%C3%A8bre_de_Boole_(logique))

### III. Exercice : Appliquez la notion

Dans cet exercice, nous allons manipuler les booléens pour voir l'impact qu'ils peuvent avoir sur notre site.

Pour cela, vous allez participer à une aventure. Un peu comme pour les "livres dont vous êtes le héros", vous allez participer au "script PHP dont vous êtes le héros".

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



#### Question

[solution n°1 p.21]

Des rumeurs prétendent qu'un vil sorcier, enfermé dans son antre, a pour plan d'invoquer des démons afin de détruire le monde. Vos exploits passés sont parvenus aux oreilles du roi, qui vous a demandé d'enquêter sur cette rumeur et de mettre un terme aux agissements du sorcier, si les rumeurs étaient fondées.

Beaucoup de héros avant vous sont entrés dans cet antre, mais personne n'en est ressorti. Les seules informations dont on dispose sont les suivantes :

- Une protection magique entoure l'antre du sorcier : quiconque possède un objet de métal sera foudroyé sur le pas de la porte. Cela ne vous plaît guère, mais vous allez devoir entrer sans épée ni bouclier et faire avec les moyens trouvés sur place. Cependant, vous pouvez remplir votre bourse de pièces d'or : la cupidité du sorcier fait qu'il autorise les petits objets en or.
- De l'extérieur, les chevaliers ont aperçu un coffre suspect dans la première salle.
- L'antre du sorcier possède un étage, ainsi qu'un sous-sol.
- Enfin, des gobelins ont été aperçus dans l'antre du sorcier. Leur allégeance n'est pas certaine : ils pourraient se révéler de précieux alliés contre le sorcier, mais également être de terribles ennemis à sa solde.

En utilisant ces informations, vous allez devoir mettre au point votre plan de bataille avant de rentrer dans l'antre du sorcier : **combien de pièces d'or** allez-vous prendre, allez-vous **ouvrir le coffre suspect**, allez-vous vous rendre au **sous-sol ou au premier étage** et est-ce que vous allez **faire confiance aux gobelins** ou non ?

Ces quatre éléments seront symbolisés sous la forme de 4 variables PHP :

```
1 <?php
2 // Mettez ici votre plan de bataille
3 $goldCoinsInYourPocket = 0;
4 $heroOpensTheChest = false;
5 $goDownAtTheStairs = false;
6 $doYouTrustGoblins = false;
```

- `$goldCoinsInYourPocket` représente le nombre de pièces d'or que vous souhaitez avoir. C'est un nombre entier.
- `$heroOpensTheChest` est un booléen (donc soit *false*, soit *true*) qui va définir votre comportement envers le coffre : allez-vous l'ouvrir (*true*) ou pas (*false*) ?
- `$goDownAtTheStairs` est un booléen qui indique si vous allez choisir le sous-sol ou non (ou aller à l'étage, dans le cas contraire).
- `$doYouTrustGoblins` est un booléen qui indique si vous faites confiance aux gobelins ou non.

Les valeurs de ces variables ne sont que des exemples : c'est à vous de définir votre plan de bataille en les modifiant.

---

1 <https://repl.it/>

Une fois votre plan de bataille prêt, lancez un repl.it, mettez-y votre plan de bataille et ajoutez le script suivant :

```

1 <?php
2 // Mettez ici votre plan de bataille
3 $goldCoinsInYourPocket = 0;
4 $heroOpensTheChest = false;
5 $goDownAtTheStairs = false;
6 $doYouTrustGoblins = false;
7
8 // Ne touchez plus à rien à partir de là
9 if ($goldCoinsInYourPocket > 10) {
10     echo 'Votre bourse déborde et toutes les pièces tombent sur le sol. Il ne vous en reste plus
    que 10.<br>';
11     $goldCoinsInYourPocket = 10;
12 }
13 echo 'Vous pénétrez dans l\'antre du sorcier... Un coffre se situe à droite et une porte est à
    gauche<br>';
14 $hasSword = false;
15 if ($heroOpensTheChest && $goldCoinsInYourPocket > 3) {
16     echo 'Le coffre était en réalité un monstre qui avait l\'apparence d\'un coffre ! Il se
    réveille grâce à la forte odeur d\'or qui émane de votre bourse et vous dévore.<br>FIN =(';
17     return;
18 } elseif ($heroOpensTheChest && $goldCoinsInYourPocket <= 3) {
19     echo 'En ouvrant le coffre, vous vous rendez compte qu\'il est rempli de dents. C\'est en
    réalité un monstre ayant l\'apparence d\'un coffre ! Heureusement, il semble endormi, ce qui
    vous permet de récupérer son contenu : <b>une épée</b> !<br>';
20     $hasSword = true;
21 }
22
23 echo 'Vous vous dirigez vers la porte et l\'ouvrez pour vous retrouver face à un escalier en
    colimaçon. Vous pouvez soit monter à l\'étage, soit descendre au sous-sol.<br>';
24
25 $hasShield = false;
26 if (!$goDownAtTheStairs) {
27     echo 'Vous montez les escaliers et ouvrez la porte du premier étage. Un gobelin vous y
    attend et vous propose de vous vendre un objet utile pour 2 pièces d\'or.<br>';
28     if ($doYouTrustGoblins && $goldCoinsInYourPocket < 2) {
29         echo 'Vous n\'avez pas assez d\'argent pour le gobelin. De rage, ce dernier sort une dague
    et vous la plante dans le ventre.<br>FIN =(';
30         return;
31     } elseif ($doYouTrustGoblins && $goldCoinsInYourPocket >= 2) {
32         echo 'Vous lui donnez deux pièces d\'or. Un peu étonné, il les prend et disparaît derrière
    une armoire quelques instants, avant de ressortir avec <b>un bouclier</b> flambant neuf ! Il
    n\'y a plus rien à faire à l\'étage, vous redescendez au sous-sol.<br>';
33         $hasShield = true;
34         $goldCoinsInYourPocket -= 2;
35     } else {
36         echo 'Vous ne faites pas confiance au gobelin. Vous fermez la porte et redescendez au
    sous-sol.<br>';
37     }
38 }
39
40 echo 'Vous descendez en direction du sous-sol. À chaque marche, des chants démoniaques se font
    entendre de plus en plus clairement. Le sorcier est en bas, et, si vous n\'intervenez pas, les
    démons envahiront notre monde. Vous arrivez finalement en bas et voyez le sorcier au milieu
    d\'un pentagramme illuminé d\'une couleur rouge sang. Il vous regarde en souriant, puis, sans
    crier gare, incante une boule de feu qu\'il envoie dans votre direction !<br>';
41
42 if (!$hasShield) {
43     echo 'Sans protection, vous vous prenez la boule de feu de plein fouet. Vous n\'êtes plus
    qu\'un petit tas de cendres fumant.<br>FIN =(';
44     return;
45 }
46

```

```

47 echo 'Vous vous protégez de justesse avec votre bouclier, qui se désintègre sous la violence
    du choc. Le sorcier ne s\'attendait pas à vous voir encore en vie, ce qui vous permet de
    foncer vers lui avant qu\'il ne relance un sort.<br>';
48
49 if ($hasSword) {
50     echo 'D\'un coup d\'épée net, vous lui coupez la tête. La lumière du pentagramme diminue
    progressivement avant de s\'éteindre. Vous avez sauvé le monde de l\'invasion des démons !
    Vous sortez de l\'antre du sorcier en sueur.<br>';
51 } else {
52     echo 'Vous le plaquez au sol et le rouez de coups de poings. Hélas, ça ne suffit pas et il
    se dégage d\'un coup de pied. Vous tombez au sol pendant qu\'il incante une nouvelle boule de
    feu qui, cette fois, vous touche de plein fouet.<br>FIN =((';
53     return;
54 }
55
56 if ($goldCoinsInYourPocket > 0) {
57     echo 'Il vous reste un peu d\'argent : vous décidez donc de les dépenser à la taverne pour
    vous remettre de cette aventure. Le peuple se souviendra de votre acte héroïque pour les
    siècles à venir.<br>FIN =D !';
58 } else {
59     echo 'Vous décidez de rentrer chez vous pour prendre un peu de repos bien mérité. Vous ne
    savez pas de quelles aventures demain sera fait, mais au moins, il ne sera pas fait de démons.
    <br>FIN =) !';
60 }

```

Exécutez le script pour voir si votre plan de bataille était le bon.

Dans le cas contraire, rien ne vous empêche de recommencer avec un autre plan de bataille.

Une dernière chose : ce qui fait de vous un héros si réputé, c'est votre don de clairvoyance.

Même si nous n'avons pas encore vu en détails l'utilisation des booléens, rien de vous empêche de jeter un coup d'œil au code pour essayer d'avoir des pistes sur ce qui pourrait fonctionner ou pas.

Beaucoup d'éléments de code vous sont encore inconnus, mais vous devriez quand même avoir quelques pistes en essayant d'analyser ce qu'il se passe.

### Indice :

Il existe plusieurs "mauvaises fins" (signalées par un smiley triste "=(") et une "bonne fin" (signalée par un smiley content "=D)").

Cependant, pour les plus habiles d'entre vous, il existe également une "meilleure fin" (signalée par un smiley très content "=D") !

## IV. Opérateurs booléens

### Objectifs

- Connaître les opérateurs booléens
- Écrire une expression booléenne

#### Définition Opération booléenne

Une opération booléenne est un calcul retournant la valeur "vrai" ou "faux". Elle peut comporter une ou plusieurs opérands et utilise des opérateurs booléens comme ET (noté && en PHP) et OU (noté || en PHP), mais aussi des opérateurs de comparaison, comme supérieur, inférieur.

## Syntaxe À retenir

- Un booléen est représenté par une variable prenant la valeur true ou false.  
`$variable = true`
- Une expression booléenne est un calcul retournant la valeur vrai ou faux.

## Mise en situation

Comme indiqué précédemment, une expression booléenne peut impliquer un ou plusieurs paramètres, ainsi que des opérateurs booléens. Nous allons voir lesquels et comment les utiliser.

## Opérateur &&

En logique, cet opérateur représente le **ET**. C'est-à-dire que l'expression booléenne retournera **vrai** si, et seulement si, ses opérandes sont toutes vraies.

## Fondamental

L'expression **a ET b** est vraie si **a** est vraie et **b** aussi, et faux dans les autres cas.

## Table de vérité

Le tableau ci-dessous présente les différentes combinaisons possibles pour une expression booléenne basique à deux opérandes.

ET LOGIQUE (&&)		
a	b	a && b
FAUX	FAUX	FAUX
FAUX	VRAI	FAUX
VRAI	FAUX	FAUX
VRAI	VRAI	VRAI

## Complément

Ces expressions ne se limitent pas forcément à deux paramètres, nous pourrions trouver : `$a && $b && $c && $d`.

Le principe reste le même : dans le cas du ET, toutes les opérandes doivent être vraies pour que l'expression soit vraie.

## Exemple

```
1 <?php
2 $isUserConnected = true;
3 $isUserActivated = true;
4
5 $userCanAccessPage = $isUserConnected && $isUserActivated;
6
7 var_dump($userCanAccessPage); // Affiche bool(true)
```



La variable `$userCanAccessPage` vaut `true`, car `$isUserConnected` est à `true` et `$isUserActivated` est à `true`, donc `true && true` donne `true`.

Ainsi, pour qu'un utilisateur puisse accéder à notre page, il doit être connecté et son compte activé.

#### Exemple

```
1 <?php
2 $isUserConnected = true;
3 $isUserAdmin = true;
4 $isUserActivated = false;
5
6 $userCanAccessAdminPage = $isUserConnected && $isUserAdmin && $isUserActivated;
7
8 var_dump($userCanAccessAdminPage); // Affiche bool(false)
```

La variable `$userCanAccessAdminPage` vaut `false`, car `$isUserConnected` et `$isUserAdmin` sont à `true`, mais `$isUserActivated` est à `false`, donc `true && true && false` donne `false`.

Ainsi, pour qu'un utilisateur puisse accéder à notre page, il doit être connecté, que son compte soit activé et il doit être un administrateur, ce qui n'est pas le cas pour notre utilisateur.

#### Opérateur ||

En logique, cet opérateur représente le **OU**. C'est-à-dire que l'expression booléenne retournera **vrai** si seulement l'une des opérandes est vraie.

#### Fondamental

L'expression **a OU b** est vraie si **a** est vraie, ou bien si **b** est vraie.

#### Table de vérité

OU LOGIQUE (II)		
a	b	a    b
FAUX	FAUX	FAUX
FAUX	VRAI	VRAI
VRAI	FAUX	VRAI
VRAI	VRAI	VRAI

Le tableau ci-dessus présente les différentes combinaisons possibles pour une expression booléenne basique à deux opérandes.

#### Complément

Comme pour l'opérateur `&&`, nous pouvons avoir plusieurs opérandes : `$a || $b || $c || $d`.

Dans ce cas, un opérande, `$a` ou `$b` ou `$c` ou `$d`, doit être vrai pour que l'expression soit vraie.

### Exemple

```
1 <?php
2 $isUserModerator = true;
3 $isUserAdmin = false;
4
5 $scanUserAccessPage = $isUserModerator || $isUserAdmin;
6
7 var_dump($scanUserAccessPage); // Affiche bool(true)
```

Ici, `$isUserModerator` est vrai, mais `$isUserAdmin` est faux. Cependant, l'expression `$isUserModerator || $isUserAdmin` est vraie, car il suffit qu'un seul opérande soit vraie.

Ainsi, notre page est ici accessible aux administrateurs, ainsi qu'aux modérateurs : il suffit d'être l'un des deux pour y accéder.

### Exemple

```
1 <?php
2 $isUserModerator = false;
3 $isUserAdmin = false;
4
5 $scanUserAccessPage = $isUserModerator || $isUserAdmin;
6
7 var_dump($scanUserAccessPage); // Affiche bool(false)
```

Ici, les deux opérandes sont faux donc l'expression est fausse.

### Opérateur !

Cet opérateur est un peu différent des autres. Il ne s'utilise pas pour lier des opérandes, mais s'applique à un opérande en particulier. Il signifie : **n'est pas**. On le désigne également comme **NON** et permet de vérifier la valeur inverse du booléen : `!true` devient `false`, et inversement.

### Fondamental

L'expression `!a` est vraie si `a` n'est pas vrai, autrement dit si `a = false`.

### Table de vérité

A	!A
VRAI	FAUX
FAUX	VRAI

### Complément

Cet opérateur est en fait la négation d'une opérande. Il doit être placé avant celle-ci.

```
!$a && !$b && !$c && !$d
```

**Exemple**

```
1 <?php
2 $isAdmin = true;
3 $isNormalUser = !$isAdmin;
4
5 var_dump($isNormalUser); // Affiche bool(false)
6
```

Ici, \$a est vraie, donc !\$a est fausse.

**Priorité des opérateurs**

Les opérateurs mathématiques traditionnels (+, -, \*, /) ont une priorité : la multiplication et la division sont prioritaires sur l'addition et la soustraction, c'est-à-dire qu'elles vont se réaliser en premier. Les opérateurs logiques ont également une priorité à laquelle il faut faire attention.

Cet ordre est le suivant : **! est prioritaire sur &&, qui est prioritaire sur ||**.

Les parenthèses permettent de changer la priorité par défaut.

**Exemple**

```
1 <?php
2
3 var_dump(true || true && false); // Affiche bool(true)
4 var_dump((true || true) && false); // Affiche bool(false)
```

Dans le premier cas, `true && false` est évalué en premier, ce qui est égal à `false`. On calcule ensuite `true || false` qui est égal à `true`.

Dans le deuxième cas, c'est `true || true` est évalué en premier, ce qui est égal à `true`. On calcule ensuite `true && false` qui est égal à `false`.

En cas de doute, et pour des raisons de lisibilité, il est recommandé de mettre des parenthèses, même inutiles, dans les expressions booléennes.

**Syntaxe**   **À retenir**

- Le **ET** logique s'écrit `&&`
- Le **OU** logique s'écrit `||`
- Le **NON / N'EST PAS** s'écrit `!`

Les opérateurs peuvent être utilisés dans une même expression, mais ont une priorité à prendre en compte. `!` est d'abord appliqué à une opérande, puis `&&` a la priorité sur `||`.

**Complément**

Opérateurs logiques && || <sup>1</sup>

Priorité des opérateurs <sup>2</sup>

1 <https://www.php.net/manual/fr/language.operators.logical.php>

2 <https://www.php.net/manual/fr/language.operators.precedence.php>

## V. Exercice : Appliquez la notion

Dans cet exercice, nous allons manipuler les opérateurs booléens en gérant des groupes de voyageurs.

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



### Question 1

[solution n°2 p.21]

Une agence de voyage gère les départs de ses groupes de voyageurs selon trois critères :

- Est-ce que le groupe est au complet ou non ?
- Est-ce qu'un avion est disponible ? Dans le cas contraire, un bateau sera préparé.
- Est-ce que la destination est dangereuse ?

Selon les groupes de voyageurs, ces critères pourront, ou non, être respectés. Ainsi, le code suivant est utilisé pour définir le départ d'un groupe au complet, pour lequel un avion est disponible et dont la destination n'est pas dangereuse :

```
1 <?php
2
3 $isGroupFull = true;
4 $isPlaneAvailable = true;
5 $isDestinationDangerous = false;
6
7 // Cette variable décide du départ
8 $canWeGo = !$isDestinationDangerous && $isPlaneAvailable && $isGroupFull;
9
10 if ($canWeGo) {
11     echo 'Départ imminent !';
12 }
```

L'affichage du message "Départ imminent !" est déterminé par le résultat de l'opération logique stockée dans la variable `$canWeGo`.

Lancez ce code dans un repl.it. En manipulant les valeurs des variables `$isGroupFull`, `$isPlaneAvailable` et `$isDestinationDangerous`, assurez-vous qu'il n'est possible de partir (donc que le message "Départ imminent !" s'affiche) uniquement quand le groupe est au complet, que l'avion est disponible et que la destination n'est pas dangereuse.

### Question 2

[solution n°3 p.21]

Un groupe de courageux se présentent à l'agence. Ils ne craignent aucune destination tant qu'ils sont ensemble, mais veulent une destination sans danger s'ils ne sont pas au complet. Dans tous les cas, l'avion ou le bateau leur vont.

Écrivez l'expression permettant de calculer `$canWeGo` dans ce cas.

Testez le programme correspondant.

#### Indice :

La variable `$isPlaneAvailable` n'a aucune influence sur le résultat : elle peut être enlevée.

1 <https://repl.it/>

### Question 3

[solution n°4 p.21]

Un groupe de casse-cous se présente à l'agence de voyage. Ce groupe n'a pas froid aux yeux et ne veut participer qu'à des expéditions dans des zones dangereuses. De plus, ils sont impatients : si leur groupe est au complet, ils n'attendront pas que l'avion soit prêt pour partir et prendront le bateau. En revanche, si leur groupe n'est pas au complet, mais que l'avion est disponible, alors ils partiront : tant pis pour les retardataires !

Modifiez la règle de la variable `$canWeGo` pour leur permettre de partir.

Modifiez et testez le programme.

#### Indice :

Les casse-cous sont prêts à partir si l'avion est prêt OU que leur groupe est au complet.

#### Indice :

Attention, selon l'ordre dans lequel vous utilisez vos variables, des parenthèses pourraient être utiles.

## VI. Opérateurs de comparaison

### Objectifs

- Connaître les opérateurs de comparaison
- Intégrer les opérateurs de comparaison aux expressions booléennes

### Mise en situation

À l'instar des opérateurs booléens, les opérateurs de comparaison peuvent être utilisés dans les expressions booléennes, conjointement ou non aux premiers nommés. Nous allons voir pourquoi et comment les utiliser.

Les opérateurs de comparaison permettent de comparer deux valeurs numériques. Contrairement aux opérateurs mathématiques qui retournent des entiers, ils retournent des booléens. C'est ainsi qu'il est possible de les utiliser dans les expressions booléennes.

Exemple	Preuve par l'exemple
<pre>1 &lt;?php 2 \$userCount = 10; 3 4 var_dump(\$userCount === 10); // Affiche bool(true) 5 var_dump(\$userCount &gt; 100); // Affiche bool(false)</pre>	

### Liste des opérateurs de comparaison

- > strictement supérieur à
- >= supérieur ou égal à
- < strictement inférieur
- <= inférieur ou égal à
- == égal à
- != différent de
- <> différent de

**Comparaison stricte** : les types sont comparés en plus des valeurs.

- === valeur égale et du même type
- !== valeur différente **OU** type différent

#### Attention

Lorsque l'on compare, non strictement, des entiers et des chaînes de caractères, ces dernières seront converties en valeurs numériques par PHP.

```
1 <?php
2 $userCount = 10;
3
4 if ($userCount == "10") {
5     echo 'Il n\'y a plus de place pour les utilisateurs';
6 }
```

Pour évaluer la condition du "if", `$userCount = "10"` est devenu `$userCount = 10`.

#### Exemple Opérateurs > et >=

```
1 <?php
2 $userCount = 15;
3
4 if ($userCount > 0) {
5     echo 'Voici la liste des utilisateurs :';
6 }
7
8 echo "<br>";
9
10 if ($userCount >= 11) {
11     echo '<a href="users?page=2">Page suivante</a>';
12 }
```

`$userCount` est effectivement supérieur à 0, donc la phrase *"Voici la liste des utilisateurs :"* sera affichée. Ces comparaisons sont, entre autres, utilisées pour les systèmes de pagination : dans cet exemple, on peut imaginer une page listant les utilisateurs 10 par 10. Ainsi, si le nombre d'utilisateurs est supérieur ou égal à 11, on affiche un lien vers la page suivante.

Les opérateurs "<" et "<=" s'utilisent de la même manière.

#### Exemple Opérateur ==

```
1 <?php
2 $userCount = 1;
3
4 if ($userCount == 1) {
5     echo 'Un utilisateur trouvé !';
6 }
7
8 echo "<br>";
9
10 if ($userCount == '1') {
11     echo 'Un utilisateur trouvé !';
12 }
13
14 echo "<br>";
15
16 if ($userCount == true) {
17     echo 'Un utilisateur trouvé !';
18 }
```

```
18 }
19 ?>
```

La comparaison en utilisant `==` étant **non stricte**, les trois conditions sont vraies. Effectivement, dans notre cas, `$userCount` est égal à 1, mais aussi à "1" car "1" sera converti en entier. On remarque également que PHP convertit automatiquement les booléens `true` en la valeur 1 : la troisième condition est donc vraie également.

Les opérateurs `!=` et `<>` s'utilisent de la même manière. `<>` est l'équivalent de `!=`, mais nous préférons ce dernier, plus utilisé dans les scripts.

#### Exemple Égalité stricte `===`, différence stricte `!==`

Reprenons l'exemple un peu plus haut de l'égalité non stricte.

```
1 <?php
2 $userCount = 1;
3
4 if ($userCount == 1) {
5     echo 'Un utilisateur trouvé !';
6 }
7
8 echo "<br>";
9
10 if ($userCount === 1) {
11     echo 'Un utilisateur trouvé !';
12 }
13
14 echo "<br>";
15
16 if ($userCount === true) {
17     echo 'Un utilisateur trouvé !';
18 }
19 ?>
```

Cette fois-ci, seule la première condition est vraie, car la **comparaison stricte** avec `===` vérifie la valeur 1, mais aussi le type : entier. La seconde condition est fausse, car le type est *string*, et la troisième est fausse, car `true` est un booléen.

#### Conseil Utiliser de préférence `===` et `!==`

Avec l'opérateur `==`, PHP va toujours essayer de convertir les valeurs à comparer pour les mettre au même type, parfois de manière assez contre-intuitive, comme nous l'avons vu pour le booléen.

Il est ainsi préférable d'utiliser la notation stricte `===` ou `!==`, moins source d'erreurs que les notations non strictes.

#### Exemple

Prenons le cas du modérateur d'un site. Nous aimerions afficher un message pour lui signaler qu'il a des messages à modérer. Ce message ne doit apparaître que si notre utilisateur est un modérateur et que le nombre de messages à modérer est supérieur à 0.

```
1 <?php
2 $numberOfNonModeratedMessages = 1;
3 $isModerator = true;
4
5
6 $hasMessagesToModerate = $isModerator && ($numberOfNonModeratedMessages>0);
```

```
7
8 var_dump($hasMessagesToModerate); // Affiche bool(true)
9
```

#### Remarque

Dans ce cas, la priorité est donnée aux opérateurs de comparaison. Ensuite, la priorité des opérateurs booléens reste la même.

#### Syntaxe À retenir

- Les opérateurs de comparaison servent à comparer deux valeurs : `if ($nombre === 10)`, `while ($nombre <= 10)`.
- Utiliser plutôt les comparaisons strictes pour les égalités : `===` ou `!==`.

#### Complément

Opérateurs de comparaison<sup>1</sup>

Tableau de comparaison de types<sup>2</sup>

## VII. Exercice : Appliquez la notion

Les opérateurs de comparaison sont très utiles dans toutes les manipulations monétaires. Pour notre exemple, nous allons réaliser un petit script permettant à un utilisateur de gérer son porte-monnaie virtuel sur un site d'achat en ligne.

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



### Question

[solution n°5 p.22]

Un utilisateur de site d'achat en ligne possède un porte-monnaie virtuel. Selon la quantité d'argent dont il dispose, différents messages vont s'afficher :

- S'il n'y a plus d'argent dans son porte-monnaie, alors le message *"Votre porte-monnaie est vide"* doit apparaître.
- S'il possède entre 5 et 10 euros, alors le message *"Consultez notre rayon 'promotions' !"* doit apparaître.
- S'il possède strictement plus de 500 euros, alors le message *"Le rayon VIP vous attend !"* doit apparaître.

Dans les autres cas, aucun message particulier ne doit apparaître.

1 <https://www.php.net/manual/fr/language.operators.comparison.php>

2 <https://www.php.net/manual/fr/types.comparisons.php>

3 <https://repl.it/>



Le code permettant de gérer ces situations est le suivant :

```
1 <?php
2 $walletAmount = 0;
3
4 $isEmpty = false;
5 $hasLowAmount = false;
6 $isVip = false;
7
8 if ($isEmpty) {
9     echo 'Votre porte-monnaie est vide';
10 }
11
12 if ($hasLowAmount) {
13     echo 'Consultez notre rayon \'promotions\' !';
14 }
15
16 if ($isVip) {
17     echo 'Le rayon VIP vous attend !';
18 }
```

Vous devez remplacer la valeur des variables `isEmpty`, `hasLowAmount` et `isVip` par des comparaisons pour que ce script fonctionne convenablement. Assurez-vous que tout fonctionne en utilisant un repl.it et en modifiant le contenu du porte-monnaie de l'utilisateur (`$walletAmount`).

## VIII. Auto-évaluation

### A. Exercice final

#### Exercice 1

[solution n°6 p.22]

Exercice

Que retourne cette expression booléenne ?

```
1 <?php
2
3 var_dump(true && false && true);
```

- ☐ True
- ☐ False

Exercice

Que retourne cette expression booléenne ?

```
1 <?php
2
3 var_dump(true || false || true);
```

- ☐ bool(true)
- ☐ bool(false)

Exercice

Que retourne cette expression booléenne ?

```
1 <?php
2
3 var_dump(true && (false || true) || false);
```

- ☐ bool(true)
- ☐ bool(false)

Exercice

Qu'affiche ce morceau de code ?

```
1 <?php
2
3 $quantity = 10;
4 $stock = 5;
5
6 $canBuy = $quantity < $stock;
7
8 var_dump($canBuy);
```

- ☐ bool(true)
- ☐ bool(false)

Exercice

Soit la formule de gestion de stock ci-après, choisissez l'opérateur de comparaison qui convient entre \$quantity et \$stock.

```
1 <?php
2
3 $quantity = 10;
4 $stock = 5;
5
6 $canBuy = $quantity ? $stock;
7
8 var_dump($canBuy);
```

- ☐ <
- ☐ <=
- ☐ >
- ☐ >=

Exercice

Qu'est-ce que le code suivant affiche ?

```
1 <?php
2
3 $isConnected = true;
4 $unreadMessages = 0;
5 $isAdmin = false;
6 $unreadAdminOnlyMessages = 5;
7
8 $hasMessagesToRead = ($isConnected && $unreadMessages>0) ||
    ($unreadAdminOnlyMessages>0&&$isAdmin);
9
10 var_dump($hasMessagesToRead);
```

- ☐ bool(true)
- ☐ bool(false)

### Exercice

Qu'est-ce que le code suivant affiche ?

```
1 <?php
2
3 $isConnected = false;
4 $unreadMessages = 0;
5 $isAdmin = true;
6 $unreadAdminOnlyMessages = 5;
7
8 $hasMessagesToRead=($isConnected && $unreadMessages>0) ||
  ($unreadAdminOnlyMessages>0&&$isAdmin);
9
10 var_dump($hasMessagesToRead);
```

- ☐ bool(true)
- ☐ bool(false)

### B. Exercice : Défi

Pour cet exercice final, nous allons revoir tous les types d'opérations booléennes pour compléter un script permettant de gérer une boîte de messagerie.

Notre boîte de messagerie est susceptible d'afficher des messages aux utilisateurs selon l'état de la boîte.

La messagerie est divisée en 4 boîtes :

- Les messages non lus (représentés par la variable `$unreadMessages`)
- Les messages lus (représentés par la variable `$readMessages`)
- La corbeille (représentée par la variable `$junkMessages`)
- Les spams (représentés par la variable `$spamMessages`)

Les utilisateurs peuvent être de simples utilisateurs ou des utilisateurs Premium, selon la valeur de la variable `$isPremium`.

Les utilisateurs normaux ont une limite de 50 messages, toutes boîtes confondues (représenté par la variable `$messageCountLimit`), tandis que les utilisateurs Premium ont une limite de 100 messages (`$premiumCountLimit`).

L'application possède ainsi le code suivant :

```
1 <html>
2 <head></head>
3 <body>
4 <h1>Messagerie</h1>
5 <?php
6 $isPremium = true;
7 $unreadMessages = 10;
8 $readMessages = 30;
9 $junkMessages = 10;
10 $spamMessages = 3;
11 $messagesCountLimit = 50;
12 $premiumCountLimit = 100;
13
14
15 $hasUnreadMessages = false;
16 $mustCleanMessages = false;
```

```

17 $isMessageboxFull = false;
18
19 if ($hasUnreadMessages) {
20     echo 'Vous avez ',$unreadMessages,' messages non lus<br>';
21 }
22
23 if ($mustCleanMessages) {
24     echo 'Vous avez des messages à nettoyer<br>';
25 }
26
27 if ($isMessageboxFull) {
28     echo 'Attention : vous avez dépassé la limite de messages autorisée !<br>';
29 }
30 ?>
31 </body>
32 </html>

```

Les trois variables `$hasUnreadMessages`, `$mustCleanMessages` et `$isMessageboxFull` sont responsables de l'affichage de messages sur l'application. Si elles sont à *true*, elles afficheront le message correspondant.

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



### Question 1

[solution n°7 p.25]

Lancez ce code dans un repl.it. Par défaut, la valeur des variables `$hasUnreadMessages`, `$mustCleanMessages` et `$isMessageboxFull` est *false*. Pour faire apparaître les messages, mettez-les à *true*.

### Question 2

[solution n°8 p.25]

Modifiez la valeur `$hasUnreadMessages` pour que le message ne s'affiche que lorsque la boîte des messages non lus n'est pas vide. Modifiez les valeurs de `$unreadMessages` pour tester tous les cas possibles.

### Question 3

[solution n°9 p.25]

Modifiez la valeur `$mustCleanMessages` pour que le message ne s'affiche que lorsqu'il y a besoin de nettoyer les messages, c'est-à-dire s'il y a des messages dans la corbeille ou dans les spams. Modifiez les valeurs de `$junkMessages` et `$spamMessages` pour tester tous les cas possibles.

#### Indice :

Il est possible de ne faire qu'une seule comparaison ! Plutôt que de vérifier que chaque boîte a un nombre supérieur à 0, il peut être malin de s'assurer que la somme des deux boîtes est supérieure à 0 !

### Question 4

[solution n°10 p.25]

Modifiez la valeur `$isMessageboxFull` pour que le message ne s'affiche que lorsque la somme des messages de toutes les boîtes de l'utilisateur dépasse la limite autorisée. Pensez au cas des utilisateurs Premium !

#### Indice :

Vous avez le droit d'utiliser des variables intermédiaires pour vous aider.

## Solutions des exercices

1 <https://repl.it/>

**p. 5 Solution n°1**

Voici la solution pour la bonne fin :

```
1 <?php
2
3 $goldCoinsInYourPocket = 2;
4 $heroOpensTheChest = true;
5 $goDownAtTheStairs = false;
6 $doYouTrustGoblins = true;
```

Pour la meilleure fin, il vous suffit d'ajouter une troisième pièce d'or au compteur.

**p. 12 Solution n°2**

Le message "Départ Imminent !" ne s'affiche que lorsque les variables sont égales à :

```
1 <?php
2 $isGroupFull = true;
3 $isPlaneAvailable = true;
4 $isDestinationDangerous = false;
```

Tout changement fait disparaître le message.

**p. 12 Solution n°3**

```
1 $scanWeGo = $isGroupFull || !$isDestinationDangerous;
```

```
1 <?php
2
3 $isGroupFull = true;
4 $isDestinationDangerous = true;
5
6 // Cette variable décide du départ
7 $scanWeGo = $isGroupFull || !$isDestinationDangerous;
8
9 if ($scanWeGo) {
10     echo 'Départ imminent !';
11 }
```

**p. 13 Solution n°4**

```
1 $scanWeGo = $isDestinationDangerous && ($isPlaneAvailable || $isGroupFull);
```

À noter que, si on avait oublié les parenthèses, les casse-cous pourraient également partir uniquement si leur groupe est au complet, quels que soient la destination ou l'état de l'avion.

```
1 <?php
2
3 $isGroupFull = true;
4 $isPlaneAvailable = false;
5 $isDestinationDangerous = true;
6
```

```
7 // Cette variable décide du départ
8 $scanWeGo = $isDestinationDangerous && ($isPlaneAvailable || $isGroupFull);
9
10 if ($scanWeGo) {
11     echo 'Départ imminent !';
12 }
```

#### p. 16 Solution n°5

```
1 <?php
2 $walletAmount = 0;
3
4 $isEmpty = $walletAmount === 0;
5 $hasLowAmount = $walletAmount >= 5 && $walletAmount <= 10;
6 $isVip = $walletAmount > 500;
7
8
9 if ($isEmpty) {
10     echo 'Votre porte-monnaie est vide';
11 }
12
13 if ($hasLowAmount) {
14     echo 'Consultez notre rayon \'promotions\' !';
15 }
16
17 if ($isVip) {
18     echo 'Le rayon VIP vous attend !';
19 }
20
```

#### Exercice p. 17 Solution n°6


##### Exercice

Que retourne cette expression booléenne ?

```
1 <?php
2
3 var_dump(true && false && true);
```

☐ True

☒ False

 Si un seul élément d'un **ET** est faux, alors toute l'expression est fausse.

##### Exercice

Que retourne cette expression booléenne ?

```
1 <?php
2
3 var_dump(true || false || true);
```

☒ bool(true)

☐ bool(false)

Q Si un seul élément d'un **OU** est vrai, alors toute l'expression est vraie.

### Exercice

Que retourne cette expression booléenne ?

```
1 <?php
2
3 var_dump(true && (false || true) || false);
```

☒ bool(true)

☐ bool(false)

Q true && (false || true) || false  
= true && (true) || false  
= true || false  
= true

### Exercice

Qu'affiche ce morceau de code ?

```
1 <?php
2
3 $quantity = 10;
4 $stock = 5;
5
6 $canBuy = $quantity < $stock;
7
8 var_dump($canBuy);
```

☐ bool(true)

☒ bool(false)

Q Le contenu de la variable \$quantity est supérieur à celui de la variable \$stock, donc l'expression \$quantity < \$stock est fausse.

### Exercice

Soit la formule de gestion de stock ci-après, choisissez l'opérateur de comparaison qui convient entre \$quantity et \$stock.

```
1 <?php
2
3 $quantity = 10;
4 $stock = 5;
5
6 $canBuy = $quantity ? $stock;
7
8 var_dump($canBuy);
```

☐ <

☒ <=

☐ >

☐ >=

Q L'opérateur de comparaison qui convient est <=.

### Exercice

Qu'est-ce que le code suivant affiche ?

```
1 <?php
2
3 $isConnected = true;
4 $unreadMessages = 0;
5 $isAdmin = false;
6 $unreadAdminOnlyMessages = 5;
7
8 $hasMessagesToRead = ($isConnected && $unreadMessages>0) ||
9 ($unreadAdminOnlyMessages>0&&$isAdmin);
10 var_dump($hasMessagesToRead);
```

☐ bool(true)

☒ bool(false)

Q \$isConnected && \$unreadMessages > 0 || \$unreadAdminOnlyMessage > 0 && \$isAdmin  
= true && false || true && false  
= false || false  
= false

### Exercice

Qu'est-ce que le code suivant affiche ?

```
1 <?php
2
3 $isConnected = false;
4 $unreadMessages = 0;
5 $isAdmin = true;
6 $unreadAdminOnlyMessages = 5;
7
8 $hasMessagesToRead=($isConnected && $unreadMessages>0) ||
9 ($unreadAdminOnlyMessages>0&&$isAdmin);
10 var_dump($hasMessagesToRead);
```

☒ bool(true)

☐ bool(false)

Q (\$isConnected && \$unreadMessages>0) || (\$unreadAdminOnlyMessage>0&&\$isAdmin)  
= false && false || true && true  
= false || true  
= true



**p. 20 Solution n°7**

Les messages suivants devraient apparaître :

- 1 Vous avez 10 messages non lus
- 2 Vous avez des messages à nettoyer
- 3 Attention : vous avez dépassé la limite de messages autorisée !

**p. 20 Solution n°8**

```
1 $hasUnreadMessages = $unreadMessages != 0;
```

**p. 20 Solution n°9**

```
1 $mustCleanMessages = $junkMessages + $spamMessages > 0;
```

**p. 20 Solution n°10**

```
1 $totalMessages = $unreadMessages + $readMessages + $junkMessages + $spamMessages;  
2 $isMessageboxFull = !$isPremium && $totalMessages > $messagesCountLimit || $isPremium &&  
   $totalMessages > $premiumCountLimit;
```