

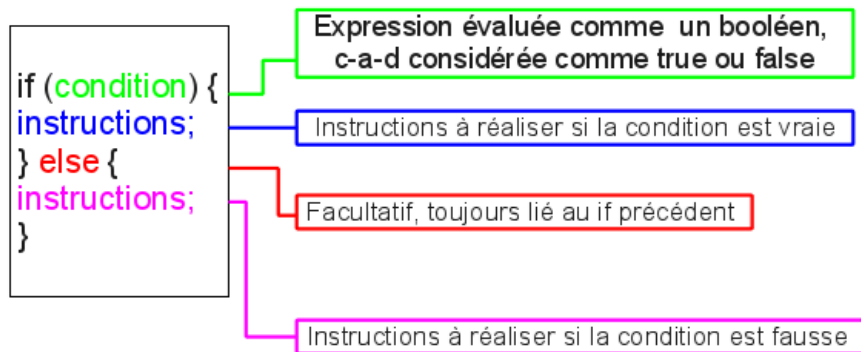


Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les structures de contrôle](#) ► [La condition if](#)

La condition if

La structure conditionnelle `if` de PHP a le même rôle que la structure SI d'AlgoBox : elle permet, si une condition est vraie, d'appliquer un traitement et, si elle est fausse, d'appliquer éventuellement un autre traitement. La syntaxe du `if` est :



Par exemple, un programme qui teste si un nombre est supérieur ou égal à 5 :

```

1  <?php
2
3  // if_1.php
4  $nb = 15;
5  if ($nb >= 5) {
6      echo "$nb est supérieur ou égal à 5";
7  } else {
8      echo "$nb est inférieur à 5";
9  }
10 echo PHP_EOL;
  
```

Cela affichera : 15 est supérieur ou égal à 5.

Le bloc `else` est facultatif :

```

1  <?php
2
3  // if_2.php
4  $nb = 15;
5  if ($nb >= 5) {
6      echo "$nb est supérieur ou égal à 5";
7  }
8  echo PHP_EOL;
  
```

Le bloc `else` est toujours lié au bloc `if` qui le précède. Ce qui peut parfois engendrer des comportements étranges si l'on ne fait pas attention. Illustrons cela par un exemple : je souhaite écrire un programme qui affiche si un nombre est positif, négatif, ou nul. Je pourrais être tenté d'écrire cet algorithme :

```

VARIABLES
nb  EST_DU_TYPE NOMBRE
message EST_DU_TYPE CHAINE
DEBUT_ALGORITHME
nb  PREND_LA_VALEUR -15
SI (nb > 0) ALORS
    DEBUT_SI
        message PREND LA VALEUR nb + " est positif"
    FIN_SI
FIN_ALGORITHME
  
```

```

AFFICHER message
FIN_SI
SI (nb < 0) ALORS
DEBUT_SI
message PREND_LA_VALEUR nb + " est négatif"
AFFICHER message
FIN_SI
SI (nb == 0) ALORS
DEBUT_SI
message PREND_LA_VALEUR nb + " est nul"
AFFICHER message
FIN_SI
FIN_ALGORITHME

```

Ce qui donnerait en PHP :

```

1 <?php
2
3 // if_3.php
4 $nb = -15;
5 if ($nb > 0) {
6     echo "$nb est positif";
7 }
8 if ($nb < 0) {
9     echo "$nb est négatif";
10 }
11 if ($nb == 0) {
12     echo "$nb est nul";
13 }
14 echo PHP_EOL;

```

Ce qui afficherait : -15 est négatif.

Maintenant si je souhaite utiliser le else, je serais tenté d'écrire cette bêtise (notez que la valeur de nb est 15) :

```

VARIABLES
nb EST_DU_TYPE NOMBRE
message EST_DU_TYPE CHAINE
DEBUT_ALGORITHME
nb PREND_LA_VALEUR 15
SI (nb > 0) ALORS
DEBUT_SI
message PREND_LA_VALEUR nb + " est positif"
AFFICHER message
FIN_SI
SI (nb < 0) ALORS
DEBUT_SI
message PREND_LA_VALEUR nb + " est négatif"
AFFICHER message
FIN_SI
SINON
DEBUT_SINON
message PREND_LA_VALEUR nb + " est nul"
AFFICHER message
FIN_SINON
FIN_ALGORITHME

```

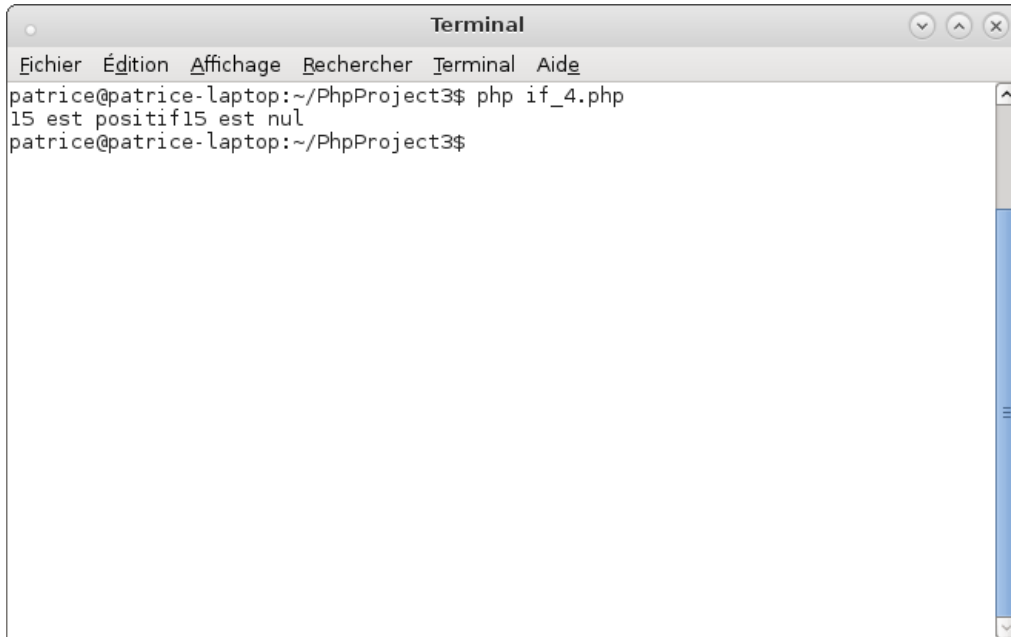
En PHP, cela donne :

```

1 <?php
2
3 // if_4.php
4 $nb = 15;
5 if ($nb > 0) {
6     echo "$nb est positif";
7 }
8 if ($nb < 0) {
9     echo "$nb est négatif";
10 } else {
11     echo "$nb est nul";
12 }
13 echo PHP_EOL;

```

Le résultat est logique mais peut surprendre :



```

Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php if_4.php
15 est positif15 est nul
patrice@patrice-laptop:~/PhpProject3$

```

Si nb vaut 15 alors l'expression `$nb > 0` du if est vraie, donc l'instruction qui se trouve dans le bloc du if (entre les accolades) sera exécutée. Le programme affiche "15 est positif" et continue son exécution. J'arrive au second if dont la condition est `$nb < 0`. Cette expression est fausse, l'instruction du bloc if ne sera pas exécutée. Mais après ce bloc if, il y a un bloc else. Comme nous l'avons dit, le bloc else est lié au bloc if qui le précède. Ce qui signifie que si je ne rentre pas dans le bloc if, je rentre nécessairement dans le bloc else qui le suit. Donc ici l'instruction du bloc else sera exécutée : le programme affiche "15 est nul". Finalement nous obtenons donc l'affichage "15 est positif15 est nul".

Corrigeons ce programme :

```

VARIABLES
nb EST_DU_TYPE NOMBRE
message EST_DU_TYPE CHAINE
DEBUT_ALGORITHME
nb PREND_LA_VALEUR 15
SI (nb > 0) ALORS
    DEBUT_SI
    message PREND_LA_VALEUR nb + " est positif"
    AFFICHER message
    FIN_SI
SINON
    DEBUT_SINON
    SI (nb < 0) ALORS
        DEBUT_SI
        message PREND_LA_VALEUR nb + " est négatif"
        AFFICHER message
        FIN_SI
    SINON
        DEBUT_SINON
        message PREND_LA_VALEUR nb + " est nul"
        AFFICHER message
        FIN_SINON
    FIN_SINON
FIN_ALGORITHME

```

Ce qui donne en PHP :

```

1 <?php
2
3 // if_5.php
4 $nb = 15;
5 if ($nb > 0) {
6     echo "$nb est positif";
7 } else {
8     if ($nb < 0) {
9         echo "$nb est négatif";
10    } else {
11        echo "$nb est nul";

```

```

12     }
13 }
14 echo PHP_EOL;

```

Ici nous avons imbriqué un if-else dans un autre if-else.

PHP nous propose une structure `elseif` afin d'éviter certaines imbrications. Le `elseif` permet des structures du genre :

```

SI nb > 0 ALORS
    AFFICHER nb + " est positif"
SINON SI nb < 0 ALORS
    AFFICHER nb + " est négatif"
SINON
    AFFICHER nb + " est nul"
FINSI

```

Le `else` sera lié au `elseif` qui le précède :

```

1 <?php
2
3 // if_6.php
4 $nb = 15;
5 if ($nb > 0) {
6     echo "$nb est positif";
7 } elseif ($nb < 0) {
8     echo "$nb est négatif";
9 } else {
10    echo "$nb est nul";
11 }
12 echo PHP_EOL;

```

Dans un `if`, il est possible dans la condition d'invertir l'opérateur de gauche et l'opérateur de droite. Par exemple si vous souhaitez tester un égalité vous pouvez écrire cela :

```

1 <?php
2
3 // if_7.php
4 $nb = 10;
5 if ($nb == 10) {
6     echo 'OK';
7 }
8 echo PHP_EOL;

```

ou encore :

```

1 <?php
2
3 // if_8.php
4 $nb = 10;
5 if (10 == $nb) {
6     echo 'OK';
7 }
8 echo PHP_EOL;

```

Cette inversion permet d'éviter les erreurs sémantiques dues à l'opérateur `=`. Que se passerait-il si, par étourderie, nous oublions un égal à l'opérateur d'égalité ? Celui-ci se transformerait en opérateur d'assignation. L'expression `$nb == 10` deviendrait `$nb = 10`. L'expression n'aurait donc plus le même sens, d'où l'erreur sémantique. L'expression `$nb == 10` est une expression booléenne qui vérifie l'égalité de valeur entre la variable `nb` et 10. Par contre `$nb = 10` est une assignation de valeur, et cette expression renvoie un résultat : la valeur assignée, ici 10. Les expressions `$nb == 10` et `$nb = 10` n'ont pas le même sens, mais elles renvoient bien toutes les deux une valeur : `true` ou `false` pour la première, 10 pour la seconde. En PHP, un nombre entier non

nul est considéré comme true. Donc dans le if la condition `$nb = 10` sera considérée comme vraie puisqu'elle renvoie un entier non nul, et non entrons dans le if... Voici un exemple de ce comportement :

```
1 <?php
2
3 // if_9.php
4 $nb = 5;
5 if ($nb = 10) {
6     echo 'OK';
7 }
8 echo PHP_EOL;
```

Dans le if nous avons une assignation et non une comparaison... Si nous inversons les deux opérateurs :

```
1 <?php
2
3 // if_10.php
4 $nb = 10;
5 if (10 = $nb) {
6     echo 'OK';
7 }
8 echo PHP_EOL;
```

L'éditeur nous signale une erreur de syntaxe à la ligne 5, nous devons corriger en :

```
1 <?php
2
3 // if_11.php
4 $nb = 10;
5 if (10 == $nb) {
6     echo 'OK';
7 }
8 echo PHP_EOL;
```

L'inversion des opérateurs évite donc l'erreur sémantique.

Fin

NAVIGATION

Accueil

■ Ma page

Pages du site

Mon profil

Cours actuel

Intro PHP

Participants

Le langage PHP : introduction

Les types et les variables

Les opérateurs

Les structures de contrôle

📁 Boucles et conditions

📁 **La condition if**

📁 L'opérateur ternaire

📁 T.P. conditions

📁 La structure switch

📁 La boucle for

📁 T.P. boucles for

📁 La boucle while

📁 La boucle do-while

📁 T.P. boucles while et do-while

 [Break et continue](#) [T.P. imbrications](#)[Les structure de données](#)[Les fonctions](#)[Les erreurs](#)[Les fichiers](#)[Les expressions rationnelles](#)[PHP et HTML](#)[Petite application](#)[Mes cours](#)

ADMINISTRATION

[Administration du cours](#)[Réglages de mon profil](#)

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[Intro PHP](#)