



Introduction au langage de programmation PHP


[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fonctions](#) ► [Valeurs et références](#)

Valeurs et références

Avant de poursuivre sur les fonctions, nous allons faire un petit détour par les notions de passage par valeur et de passage par référence. Prenons un exemple simple : je vais déclarer une variable `nb` et lui assigner la valeur 100. Puis je vais déclarer une variable `autre_nb` et lui assigner la valeur de `nb`, ce qui me donne en PHP :

```
1 <?php
2
3 // valeurs_references_1.php
4 $nb = 100;
5 $autre_nb = $nb;
6 echo "\$nb a pour valeur $nb et \$autre_nb a pour valeur $autre_nb";
7 echo PHP_EOL;
```

Ce qui affiche :

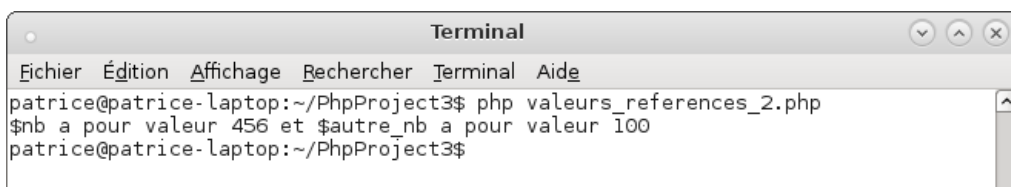


```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php valeurs_references_1.php
$nb a pour valeur 100 et $autre_nb a pour valeur 100
patrice@patrice-laptop:~/PhpProject3$
```

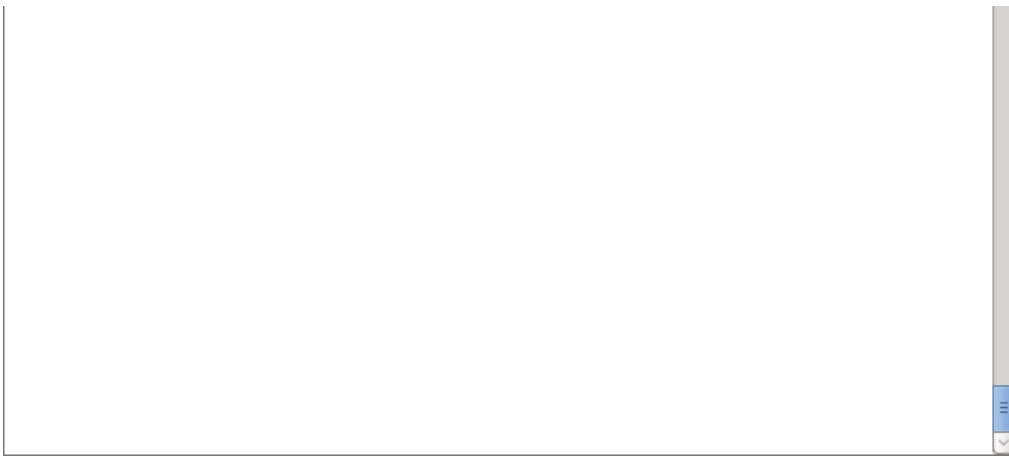
Maintenant si je modifie la valeur de `nb` juste avant l'affichage :

```
1 <?php
2
3 // valeurs_references_2.php
4 $nb = 100;
5 $autre_nb = $nb;
6 $nb = 456;
7 echo "\$nb a pour valeur $nb et \$autre_nb a pour valeur $autre_nb";
8 echo PHP_EOL;
```

Cela affiche :



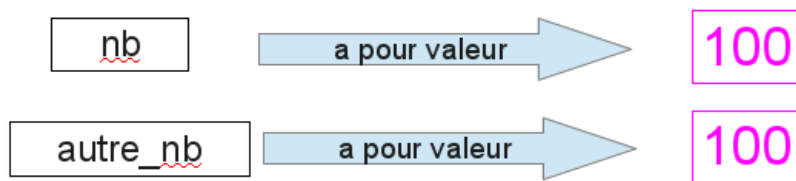
```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php valeurs_references_2.php
$nb a pour valeur 456 et $autre_nb a pour valeur 100
patrice@patrice-laptop:~/PhpProject3$
```



Cela semble normal, puisque nb et autre_nb sont deux "boîtes" différentes, chacune contient sa valeur, et si je modifie la valeur d'une des boîtes cela ne modifie pas la valeur présente dans l'autre boîte. La valeur de nb et de autre_nb sont donc stockées dans deux endroits différents. Le code :

```
1 <?php
2
3 // valeurs_references_3.php
4 $nb = 100;
5 $autre_nb = $nb;
```

correspond au schéma :



L'instruction

```
$autre_nb = $nb;
```

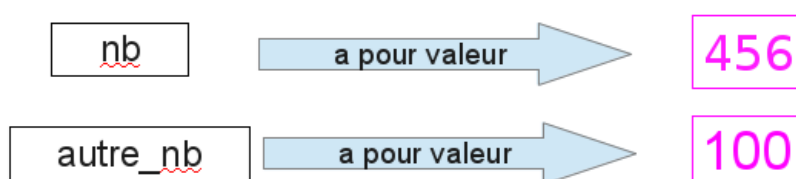
assigne à autre_nb une copie de la valeur de nb. Je souhaite maintenant, à la suite du code, modifier la valeur de \$nb :

```
1 <?php
2
3 // valeurs_references_4.php
4 $nb = 100;
5 $autre_nb = $nb;
6 $nb = 456;
```

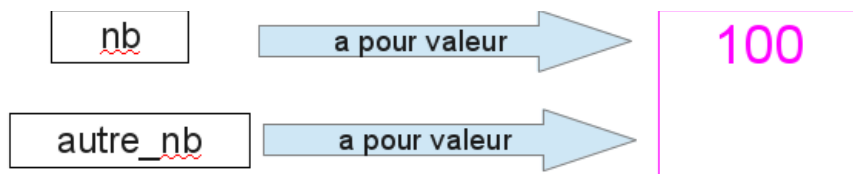
L'instruction

```
$nb = 456;
```

ne modifie pas la valeur de autre_nb, seulement la valeur de nb :



Je pourrais demander à PHP que mes deux variables nb et autre_nb partagent la même boîte, de façon à obtenir ceci :



Nous allons juste créer une flèche de la variable `autre_nb` vers la boîte de `nb`. La variable `autre_nb` fera ici référence à la variable `nb`. Pour faire référence à une variable nous utiliserons en PHP l'opérateur `&` devant la variable à référencer :

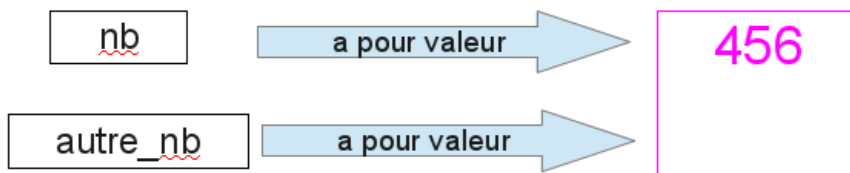
```

1 <?php
2
3 // valeurs_references_5.php
4 $nb = 100;
5 $autre_nb = &$nb;
6 echo "\$nb a pour valeur $nb et \$autre_nb a pour valeur $autre_nb";
  
```

Cette fois mes deux variables font boîte commune, si je modifie la valeur de l'une, je modifie aussi la valeur de l'autre. Ce qui revient à modifier la valeur qui se trouve dans la boîte commune.

```

1 <?php
2
3 // valeurs_references_6.php
4 $nb = 100;
5 $autre_nb = &$nb;
6 $nb = 456;
7 echo "\$nb a pour valeur $nb et \$autre_nb a pour valeur $autre_nb";
  
```



En somme lorsque j'écris

```
$autre_nb = $nb;
```

j'assigne à `autre_nb` une copie de la valeur de `nb` : c'est un passage par valeur, ou encore par copie.

Lorsque j'écris

```
$autre_nb = &$nb;
```

je dis que `autre_nb` fait référence à la variable `nb` : c'est un passage par référence.

Maintenant faisons une petite expérience :

- nous allons créer une fonction `doubler()` qui prendra en paramètre un entier `n`, qui doublera sa valeur et qui l'affichera
- nous déclarerons ensuite une variable `nb` et lui assignerons la valeur 100
- nous appellerons la fonction `doubler()` avec en argument `nb`
- nous afficherons la valeur de `nb`.

```

1 <?php
2
3 // valeurs_references_7.php
4 function doubler($n) {
5     $n *= 2;
6     echo "dans la fonction doubler() \$n vaut $n" . PHP_EOL;
7 }
8
9 $nb = 100;
10 doubler($nb);
11 echo "Après l'appel de la fonction doubler(), \$nb vaut $nb";
  
```

```
12 echo PHP_EOL;
```

Ce qui affiche :

```

Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php valeurs_references_7.php
dans la fonction doubler() $n vaut 200
Après l'appel de la fonction doubler(), $nb vaut 100
patrice@patrice-laptop:~/PhpProject3$

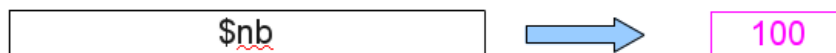
```

Lorsque j'appelle la fonction `doubler()`, je lui demande de travailler avec la variable `nb` : je passe en argument à la fonction `doubler()` la variable `nb`. Dans la fonction le paramètre `n` prendra donc la valeur de `nb`. Le traitement de doubler consiste à doubler la valeur de la variable qui lui est passée en argument : ce qui est effectivement réalisé puisque, dans ma fonction, la valeur a bien été doublée. Par contre, après l'appel de la fonction, je m'aperçois que la valeur de `nb` est restée la même et n'a pas été doublée. La variable que j'ai passée en argument n'a donc finalement pas été modifiée. La valeur qui a été doublée au sein de ma fonction n'était donc pas la "vraie" variable `nb` mais une copie.

Lorsque j'appelle la fonction `doubler()`, je lui passe en argument la variable `nb`. J'indique donc que, dans la fonction `doubler()`, le paramètre `n` sera égal à `nb`. Tout ce passe comme si dans la fonction `doubler()`, `$n = $nb`. Mais nous avons vu précédemment que, quand j'écris cela, je ne fais qu'assigner à `$n` une copie de la valeur de `$nb`. La fonction `doubler()` ne travaillera donc pas sur la "vraie" variable `nb` mais simplement sur une copie. Au sein de la fonction la variable `n`, copie de `nb`, est bien doublée, mais la valeur de `nb`, elle, n'est pas modifiée. Lorsque je passe une variable en argument à une fonction, je passe en réalité une copie de cette variable : je réalise un passage par valeur. Lors de l'appel d'une fonction, cette dernière réalise une copie des arguments, travaille sur ces copies, et lorsque le traitement est terminé, à la fin de la fonction, les copies sont détruites.

Reprenons tout cela avec des schémas :

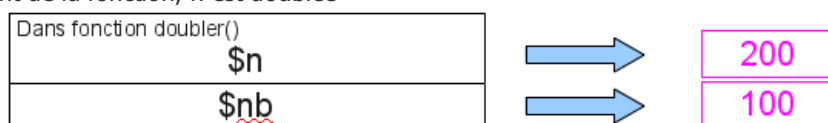
1. je déclare une variable `nb` et lui assigne la valeur 100



2. j'appelle la fonction `doubler()` en lui passant en argument `nb`, donc dans la fonction `doubler()`, le paramètre `n` sera égal à `nb`, ou plutôt à une copie de `nb`



3. lors du traitement de la fonction, `n` est doublée



4. puis la fonction affiche

```
dans la fonction doubler() $n vaut 200
```

5. la fonction `doubler()` est terminée



6. le script affiche

Après l'appel de la fonction doubler(), \$nb vaut 100

7. fin du script.

Il est tout à fait possible de passer à une fonction des arguments par référence en utilisant l'opérateur & vu précédemment. Cette opérateur doit être utilisé devant le paramètre lors de la déclaration de la fonction :

```
1 <?php
2
3 // valeurs_references_8.php
4 function doubler(&$n) {
5     $n *= 2;
6     echo "dans la fonction doubler() \$n vaut \$n" . PHP_EOL;
7 }
```

Voici le code complet :

```
1 <?php
2
3 // valeurs_references_9.php
4 function doubler(&$n) {
5     $n *= 2;
6     echo "dans la fonction doubler() \$n vaut \$n" . PHP_EOL;
7 }
8
9 $nb = 100;
10 doubler($nb);
11 echo "Après l'appel de la fonction doubler(), \$nb vaut \$nb";
12 echo PHP_EOL;
```

Ce qui affiche :

```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php valeurs_references_9.php
dans la fonction doubler() $n vaut 200
Après l'appel de la fonction doubler(), $nb vaut 200
patrice@patrice-laptop:~/PhpProject3$
```

Avec des schémas :

1. je déclare une variable nb et lui assigne la valeur 100



2. j'appelle la fonction doubler() en lui passant en argument nb, donc dans la fonction doubler(), la paramètre n fera référence à nb



3. lors du traitement de la fonction, n est doublée





4. puis la fonction affiche

dans la fonction `doubler()` `$n` vaut 200

5. la fonction `doubler()` est terminée



6. le script affiche

Après l'appel de la fonction `doubler()`, `$nb` vaut 200

7. fin du script.

Les types que nous avons rencontrés jusqu'à présent, (à savoir boolean, integer, double, string, array) sont tous passés par valeur. Le type object sera lui passé par référence.

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

Les structure de données

Les fonctions

Les fonctions prédéfinies

Opérations sur les types et variables

Opérations sur les chaînes

T.P. voyelles et consonnes

Opérations sur les tableaux

T.P. occurrences

Les fonctions utilisateurs

T.P. fonction d'étoiles

Valeurs et références

T.P. fonction doubler

T.P. fonction calculer

Les erreurs

Les fichiers

Les expressions rationnelles

PHP et HTML

Petite application

[Mes cours](#)

ADMINISTRATION



Administration du cours

Réglages de mon profil