



# Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fonctions](#) ► [Les fonctions prédéfinies](#)

## Les fonctions prédéfinies

PHP nous propose un nombre très important de fonctions prédéfinies, sous le terme fonction je regroupe ici les fonctions et les procédures. Ces fonctions vont nous permettre d'effectuer de nombreux traitements sur nos données. Vous verrez que la [liste des fonctions prédéfinies](#) que propose la documentation officielle est tout à fait impressionnante. Lorsque vous devez effectuer un traitement, demandez-vous toujours s'il n'existe pas une fonction prédéfinie qui fait le travail. En un mot, préférez les fonctions prédéfinies à vos fonctions (celle que vous allez bientôt créer avec vos doigts).

Nous n'aborderons pas toutes fonctions, nous verrons ensemble quelques fonctions utiles. Les fonctions sont très bien décrites dans la documentation officielle qui représente votre source principale d'information.

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

---

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



# Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fonctions](#) ► [Opérations sur les types et variables](#)

## Opérations sur les types et variables

Voici quelques fonctions qui permettent de travailler sur les types et [les variables](#), appelées par PHP [fonctions de gestion des variables](#). La fonction `gettype()` : renvoie une chaîne de caractères qui correspond au type de la variable passée en argument. La documentation officielle décrit la fonction de cette façon :

```
string gettype ( mixed $var )
```

Nous trouvons dans l'ordre :

- le type de retour de la fonction : string, cela signifie que la fonction renvoie une donnée de type string (chaîne de caractères)
- le nom de la fonction : `gettype`
- entre parenthèses les paramètres de la fonction avec
  - le type du paramètre : ici `mixed` signifie que le paramètre `var` accepte plusieurs types
  - le nom du paramètre : `var`

Nous trouvons ensuite dans la documentation la liste des paramètres et leurs descriptions. Puis la liste des valeurs de retour possibles suivie de quelques exemples d'utilisation. Quelques exemples :

```
1 <?php
2
3 // operations_types_variables_1.php
4 $nb = 123;
5 echo gettype($nb);
6 echo PHP_EOL;
7 echo gettype('azerty');
8 echo PHP_EOL;
9 $tab = [1, 2, 3];
10 $type = gettype($tab);
11 echo "le type de \$tab est $type" . PHP_EOL;
```

Autre fonction très intéressante, `var_dump()`, qui affiche le type et la valeur d'une variable.

```
void var_dump ( mixed $expression [, mixed $... ] )
```

Le type de retour est `void`, ce qui signifie que la fonction ne renvoie aucune valeur, c'est donc une procédure. Le paramètre `expression` peut être de n'importe quel type et peut être accompagné d'autres paramètres facultatifs. Les paramètres facultatifs sont toujours écrits entre crochets. Quelques exemples :

```
1 <?php
2
3 // operations_types_variables_2.php
4 $nb = 12;
5 var_dump($nb);
6 var_dump('azerty');
7 $tab = array(array(1.1, 2, 3), array('a', 56.8, 'c'));
8 var_dump($tab);
```

Trois autres fonctions à retenir :

1. `isset()` : vérifie si une variable a été définie (déclaration + assignation) et si elle est différente de `null`
2. `unset()` : détruit une variable
3. `empty()` : vérifie si une variable est vide, sont considérés comme vides "" (une chaîne vide), 0, 0.0, "0", NULL, FALSE, `array()` et `[]` (un tableau vide), `var $var;` (une variable déclarée, mais sans valeur dans une classe)

Exemple :

```
1 <?php
2
3 // operations_types_variables_3.php
4 $message = '';
5 $var = 125;
6 if (isset($var)) {
7     $message = 'la variable $var a été définie et est différente de null, ';
8     if (empty($var)) {
9         $message .= 'mais est vide';
10    } else {
11        $message .= "et a pour valeur $var";
12    }
13 } else {
14     $message = 'la variable n\'a pas été définie ou est égale à null';
15 }
16 echo $message . PHP_EOL;
```

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](#)

---

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



# Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fonctions](#) ► [Opérations sur les chaînes](#)

## Opérations sur les chaînes

Voici quelques fonctions parmi les nombreuses [opérations sur les chaînes](#) que proposent PHP.

La fonction `strlen()` permet de connaître la longueur d'une chaîne et peut donc s'avérer très pratique lorsqu'on souhaite parcourir une chaîne avec une boucle :

```
1 <?php
2
3 // operations_chaines_1.php
4 $message = 'azerty';
5 $str_length = strlen($message);
6 for ($index = 0; $index < $str_length; $index++) {
7     echo $message[$index] . PHP_EOL;
8 }
```

Vous avez noté que pour accéder à un caractère d'une chaîne nous utilisons l'opérateur `[]`, de la même façon que pour les tableaux. Le premier caractère de la chaîne est à l'index 0.

La fonction `strpos()` cherche la première occurrence d'une sous-chaîne dans une chaîne, et renvoie sa position. Il existe, comme souvent, des variantes à cette fonction : `strpos()` rend le même service mais est insensible à la casse, `strrpos()` cherche la dernière occurrence et non la première, `stripos()` cherche la dernière occurrence et est insensible à la casse.

La fonction `substr()` renvoie un morceau d'une chaîne à partir d'une position et, éventuellement, d'une certaine longueur. Voici un exemple qui extrait d'une adresse mail le nom de l'hébergeur (après le @ et avant le . du nom de domaine) :

```
1 <?php
2
3 // operations_chaines_2.php
4 $email = 'jack.sparrow@yahoo.com';
5 $pos1 = strpos($email, '@') + 1;
6 $pos2 = strrpos($email, '.');
7 echo substr($email, $pos1, $pos2 - $pos1) . PHP_EOL;
```

La fonction `str_replace()` permet de remplacer toutes les [occurrences](#) d'une chaîne par une autre, et renvoie la chaîne modifiée :

```
1 <?php
2
3 // operations_chaines_3.php
4 $message = 'Le moteur de recherche Google assure la confidentialité des données.';
5 $message = str_replace('Google', 'startpage', $message);
6 echo $message . PHP_EOL;
```

La fonction `strstr()` permet de renvoyer la chaîne à partir de l'occurrence trouvée (occurrence incluse) ou, avec un paramètre, la chaîne avant l'occurrence trouvée (occurrence exclue). Par exemple pour extraire la chaîne avant le @ :

```
1 <?php
2
3 // operations_chaines_4.php
```

```
4 $email = 'jack.sparrow@yahoo.com';
5 echo strpos($email, '@', true) . PHP_EOL;
```

La fonction `trim()` permet de supprimer les caractères blancs de part et d'autre d'une chaîne, renvoie la chaîne de caractères nettoyée. La fonction `explode()` permet de découper une chaîne selon un séparateur, renvoie un tableau contenant les "morceaux".

La fonction `strcmp()` compare alphabétiquement deux chaînes de caractères. Elle renvoie un entier négatif si le premier argument est plus petit que le second, une valeur positive si le premier argument est plus grand, 0 si les deux arguments sont égaux. Exemples :

```
1 <?php
2
3 // operations_chaines_5.php
4 echo strcmp('azerty', 'uiop') . PHP_EOL;
5 echo strcmp('truc', 'truc') . PHP_EOL;
6 echo strcmp('2', '123') . PHP_EOL; // oups...
```

La dernière comparaison affirme que 2 est plus grand que 123 ce qui est vrai alphabétiquement. Pour effectuer une comparaison plus "naturelle", nous pouvons utiliser la fonction `strnatcmp()` :

```
1 <?php
2
3 // operations_chaines_6.php
4 echo strnatcmp('2', '123') . PHP_EOL;
```

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](#)

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





# Introduction au langage de programmation PHP

[Accueil](#) ▶ [Mes cours](#) ▶ [Développement logiciel](#) ▶ [Intro PHP](#) ▶ [Les fonctions](#) ▶ [T.P. voyelles et consonnes](#)

## T.P. voyelles et consonnes

Écrivez un programme qui comptera le nombre de voyelles et le nombre de consonnes d'un mot saisi par l'utilisateur.

Par exemple, si la chaîne saisie est "azerty", le programme affichera :

La chaîne "azerty" contient 3 voyelles et 3 consonnes.

Le nom du script sera de la forme "voyelles\_*nom\_prenom*.php". Vous remettrez une archive contenant ce fichier, le nom de l'archive sera de la forme "voyelles\_*nom\_prenom*.zip".

## État du travail remis

Numéro de tentative	Ceci est la tentative 1.
Statut des travaux remis	Remis pour évaluation
Statut de l'évaluation	Noté
Dernière modification	vendredi 27 mars 2015, 09:17
Remises de fichiers	

Modifier le travail

Modifier votre travail remis

## Feedback

Note	100,00 / 100,00
Évalué le	vendredi 27 mars 2015, 10:10
Évalué par	 Patrice Fernandez

Feedback par commentaires    Tu aurais pu aussi utiliser la fonction `in_array()` pour vérifier si une lettre appartient au tableau de voyelles.

### 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](#)

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



# Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fonctions](#) ► [Opérations sur les tableaux](#)

## Opérations sur les tableaux

Les [fonctions sur les tableaux](#) sont aussi très nombreuses. En voici quelques unes.

Les fonctions `sizeof()` et `count()` renvoie le nombre d'éléments présents dans un tableau :

```
1 <?php
2
3 // operations_tableaux_1.php
4 $tab = [132, 789, 'zert', 45.6, 'uiop'];
5 $tab_length = count($tab);
6 for ($index = 0; $index < $tab_length; $index++) {
7     echo $tab[$index] . PHP_EOL;
8 }
```

La fonction [print\\_r\(\)](#) permet d'afficher récursivement le contenu d'un tableau, c'est-à-dire le tableau et ses sous-tableaux. La fonction [array\\_push\(\)](#) permet d'ajouter un élément à la fin du tableau, tout comme l'opérateur []. La fonction [array\\_pop\(\)](#) renvoie le dernier élément du tableau et le supprime. La fonction [array\\_unshift\(\)](#) ajoute un élément au début du tableau. La fonction [array\\_shift\(\)](#) renvoie le premier élément du tableau et le supprime.

```
1 <?php
2
3 // operations_tableaux_2.php
4 $tab = [132, 789, 'zert', 45.6, 'uiop'];
5 array_push($tab, 852); // ajoute 852 à la fin du tableau
6 print_r($tab);
7 echo array_pop($tab) . PHP_EOL; // affiche 852 et le supprime du tableau
8 print_r($tab);
9 array_unshift($tab, 'machin'); // ajoute 'machin' au début du tableau
10 print_r($tab);
11 echo array_shift($tab) . PHP_EOL; // affiche 'machin' et le supprime du tableau
12 print_r($tab);
```

La fonction [array\\_values\(\)](#) renvoie un tableau de toutes les valeurs après avoir opéré une ré-indexation, ce qui est très utile après une suppression de valeur dans un tableau simple (non-associatif) :

```
1 <?php
2
3 // operations_tableaux_3.php
4 $tab = [132, 789, 'zert', 45.6, 'uiop'];
5
6 print_r($tab);
7 unset($tab[2]);
8 print_r($tab); // l'index 2 est manquant
9 $tab = array_values($tab); // ré-indexation
10 print_r($tab);
```

La fonction [array\\_splice\(\)](#) permet aussi d'effectuer des suppressions dans un tableau et opère elle-même la réindexation des éléments. Elle renvoie un tableau contenant les éléments supprimés. Par exemple si je souhaite supprimer à l'index 1 du tableau `tab` deux éléments:

```
1 <?php
2
3 // operations_tableaux_4.php
```

```
4 $tab = [132, 789, 'zert', 45.6, 'uiop'];
5 array_splice($tab, 1, 2);
6 print_r($tab);
```

La fonction `in_array()` renvoie vrai si une valeur est dans le tableau, false sinon. La fonction `array_search()` renvoie l'index de l'élément recherché sinon false. La fonction `array_keys()` renvoie un tableau des clés. La fonction `array_key_exists()` renvoie vrai si la clé existe dans le tableau, false sinon.

PHP propose un ensemble de fonctions de tri permettant d'ordonner un tableau en prenant comme critère les valeurs ou les clés :

- `sort()` : tri en ordre croissant selon les valeurs
- `ksort()` : tri en ordre croissant selon les clés
- `rsort()` : tri en ordre décroissant selon les valeurs
- `krsort()` : tri en ordre décroissant selon les clés

Des fonctions de tri sur les valeurs permettent de conserver les clés :

- `asort()` : tri en ordre croissant selon les valeurs avec conservation des clés
- `arsort()` : tri en ordre décroissant selon les valeurs avec conservation des clés

Certaines fonctions permettent d'effectuer un tri sur les valeurs ou les clés en utilisant une fonction de comparaison personnalisée passée en paramètre (fonction de callback) :

- `usort()` : tri selon les valeurs et selon une fonction de comparaison personnalisée
- `uasort()` : tri selon les valeurs avec conservation des clés et selon une fonction de comparaison personnalisée
- `uksort()` : tri selon les clés et selon une fonction de comparaison personnalisée

Les fonctions `sort()` (avec le paramètre `SORT_NATURAL`) et `natsort()` permettent de réaliser un tri selon un ordre "naturel".

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](#)

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



# Introduction au langage de programmation PHP

[Accueil](#) ▶ [Mes cours](#) ▶ [Développement logiciel](#) ▶ [Intro PHP](#) ▶ [Les fonctions](#) ▶ [T.P. occurrences](#)

## T.P. occurrences

Affichez le nombre d'occurrences de chaque lettre de l'alphabet contenue dans une chaîne de caractères saisie par l'utilisateur (en minuscule). C'est-à-dire combien il y a de lettres 'a' dans la chaîne, combien il y a de lettres 'b' dans la chaîne, et ainsi de suite jusqu'à 'z'. Pour stocker le nombre d'occurrences vous utiliserez un tableau associatif dont les clés seront les lettres et les valeurs les nombres d'occurrences de ces lettres dans la chaîne. Le nom du script sera de la forme "occurrences\_*nom\_prenom*.php". Vous remettrez une archive zip contenant ce fichier, le nom de l'archive sera de la forme "occurrences\_*nom\_prenom*.zip"

## État du travail remis

Numéro de tentative	Ceci est la tentative 1.
Statut des travaux remis	Remis pour évaluation
Statut de l'évaluation	Noté
Dernière modification	vendredi 27 mars 2015, 10:04
Remises de fichiers	

Modifier le travail

Modifier votre travail remis

## Feedback

Note	100,00 / 100,00
Évalué le	vendredi 27 mars 2015, 10:56
Évalué par	 Patrice Fernandez
Feedback par commentaires	C'est bon.

### 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](#)

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



# Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fonctions](#) ► [Les fonctions utilisateurs](#)

## Les fonctions utilisateurs

PHP nous permet de construire nos propres fonctions. Une fonction se déclare avec le mot clé "function". Ce mot clé sera suivi du nom de la fonction (par exemple `mafonction` ou `ma_function`) puis des paramètres toujours écrits entre parenthèses. S'il n'y a pas de paramètres, les parenthèses resteront vides mais seront toujours présentes. On ne pourra pas créer deux fonctions portant le même nom. Enfin les instructions à réaliser seront écrites entre `{}` dans le bloc d'instructions :

```
function ma_fonction($param1, $param2){  
    instruction_1 ;  
    instruction_2 ;  
    ...  
}
```

Une fois cette fonction créée, nous l'appellerons comme une fonction "normale" c'est-à-dire via son nom et en lui fournissant les arguments nécessaires. Par exemple la fonction `verifier_majorite()` pourrait être implémentée comme cela :

```
1 <?php  
2  
3 // fonctions_utilisateurs_1.php  
4 function verifier_majorite($age) {  
5     if ($age >= 18) {  
6         echo 'Vous êtes majeur';  
7     } else {  
8         echo 'Vous êtes mineur';  
9     }  
10 }  
11  
12 verifier_majorite(25); // appel de la fonction  
13 echo PHP_EOL;
```

Ou encore `additionner()` qui prendra en paramètre un nombre `nb1` et un nombre `nb2` :

```
1 <?php  
2  
3 // fonctions_utilisateurs_2.php  
4 function additionner($nb1, $nb2) {  
5     echo "$nb1 + $nb2 = " . ($nb1 + $nb2);  
6 }  
7  
8 additionner(10, 15);  
9 echo PHP_EOL;
```

Pour renvoyer une valeur à la fin de la fonction nous utiliserons le mot clé "return". Il est possible de mettre plusieurs `return` dans une fonction mais l'exécution d'un `return` met toujours fin à la fonction. Par exemple si nous souhaitons transformer la fonction `verifier_majorite()` pour qu'elle renvoie le résultat :

```
1 <?php  
2  
3 // fonctions_utilisateurs_3.php  
4 function verifier_majorite($age) {  
5     $resultat = '';
```



```

6     if ($age >= 18) {
7         $resultat = 'Vous êtes majeur';
8     } else {
9         $resultat = 'Vous êtes mineur';
10    }
11    return $resultat;
12 }
13
14 echo verifier_majorite(25) . PHP_EOL;

```

Nous pourrions également écrire :

```

1 <?php
2
3 // fonctions_utilisateurs_4.php
4 function verifier_majorite($age) {
5     if ($age >= 18) {
6         return 'Vous êtes majeur';
7     } else {
8         return 'Vous êtes mineur';
9     }
10 }
11
12 echo verifier_majorite(25) . PHP_EOL;

```

Si vous créez une variable dans une fonction, cette variable n'existera plus une fois la fonction exécutée. Cette variable est dite "locale" à la fonction.

```

1 <?php
2
3 // fonctions_utilisateurs_5.php
4 function do_truc() {
5     $message = 'azerty'; // création d'une variable dans une fonction
6     echo 'do_truc() est exécutée !';
7 }
8
9 do_truc();
10 echo $message; // renvoie une erreur de type NOTICE : undefined variable
11 echo PHP_EOL;

```

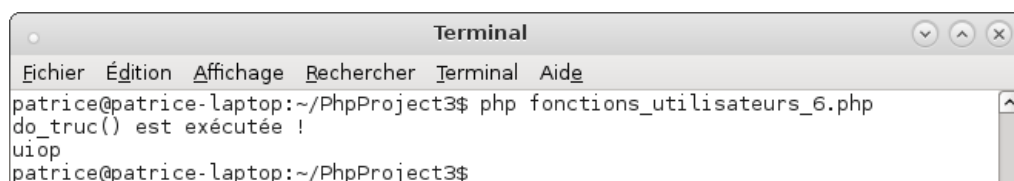
Le mot clé global permet d'utiliser au sein d'une fonction une variable définie hors de la fonction :

```

1 <?php
2
3 // fonctions_utilisateurs_6.php
4 $message = 'azerty';
5
6 function do_truc() {
7     global $message;
8     $message = 'uiop';
9     echo 'do_truc() est exécutée !' . PHP_EOL;
10 }
11
12 do_truc();
13 echo $message;
14 echo PHP_EOL;

```

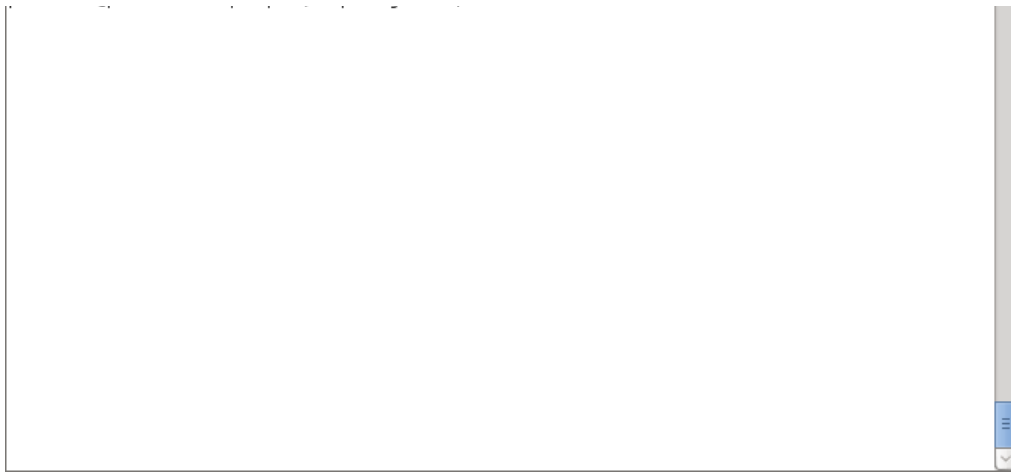
Cela affiche :



```

Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php fonctions_utilisateurs_6.php
do_truc() est exécutée !
uiop
patrice@patrice-laptop:~/PhpProject3$

```



Dans les fonctions PHP, il est possible de donner aux paramètres une valeur par défaut. Les paramètres qui possèdent une valeur par défaut deviennent facultatifs. On peut mélanger les paramètres "normaux" et les paramètres avec une valeur par défaut, mais les paramètres avec une valeur par défaut doivent toujours figurer à la fin. Quelques exemples :

```
1 <?php
2
3 // fonctions_utilisateurs_7.php
4 function se_presenter($age, $nom = 'inconnu', $prenom = 'inconnu') {
5     echo "votre nom est $nom, votre prénom est $prenom et vous avez $age ans";
6     echo PHP_EOL;
7 }
8
9 se_presenter(18);
10 se_presenter(45, 'Sparrow');
11 se_presenter(42, 'Wayne', 'Bruce');
```

Ce code affiche :

```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php fonctions_utilisateurs_7.php
votre nom est inconnu, votre prénom est inconnu et vous avez 18 ans
votre nom est Sparrow, votre prénom est inconnu et vous avez 45 ans
votre nom est Wayne, votre prénom est Bruce et vous avez 42 ans
patrice@patrice-laptop:~/PhpProject3$
```

Il est possible en PHP de gérer un nombre indéfini d'arguments. La fonction `func_num_args()` permet à l'intérieur d'une fonction, de récupérer le nombre d'arguments qui lui a été passé. La fonction `func_get_args()` renvoie un tableau contenant tous les arguments qui ont été passés à la fonction. Enfin la fonction `func_get_arg()` permet de récupérer un argument selon son index.

```
1 <?php
2
3 // fonctions_utilisateurs_8.php
4 function do_truc() {
5     $nb_arg = func_num_args();
```

```

6     echo "vous avez passé $nb_arg arguments à cette fonction";
7 }
8
9 function calculer_moyenne() {
10     $somme = 0;
11     $args = func_get_args();
12     foreach ($args as $value) {
13         $somme += $value;
14     }
15     $moyenne = $somme / func_num_args();
16     echo "la moyenne des arguments est $moyenne";
17 }
18
19 do_truc('Sparrow', 45645, [1, 2, 3], 12.5);
20 echo PHP_EOL;
21 calculer_moyenne(3, 4, 5);
22 echo PHP_EOL;

```

Dans un script les fonctions peuvent être déclarées n'importe où, même après leurs appels :

```

1 <?php
2
3 // fonctions_utilisateurs_9.php
4 do_truc('Sparrow', 45645, [1, 2, 3], 12.5);
5 echo PHP_EOL;
6 calculer_moyenne(3, 4, 5);
7 echo PHP_EOL;
8
9 function do_truc() {
10     $nb_arg = func_num_args();
11     echo "vous avez passé $nb_arg arguments à cette fonction";
12 }
13
14 function calculer_moyenne() {
15     $somme = 0;
16     $args = func_get_args();
17     foreach ($args as $value) {
18         $somme += $value;
19     }
20     $moyenne = $somme / func_num_args();
21     echo "la moyenne des arguments est $moyenne";
22 }

```

Les fonctions peuvent aussi être déclarées dans des structures de contrôle, mais dans ce cas, elles doivent être déclarées avant d'être appelées :

```

1 <?php
2
3 // fonctions_utilisateurs_10.php
4 $nom = 'Wayne';
5 if ($nom == 'Sparrow') {
6
7     function direBonjour() {
8         echo 'Bonjour Sparrow';
9     }
10
11 } else {
12
13     function direBonjour() {
14         echo 'Bonjour inconnu';
15     }
16
17 }
18 direBonjour();

```

[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](#)

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

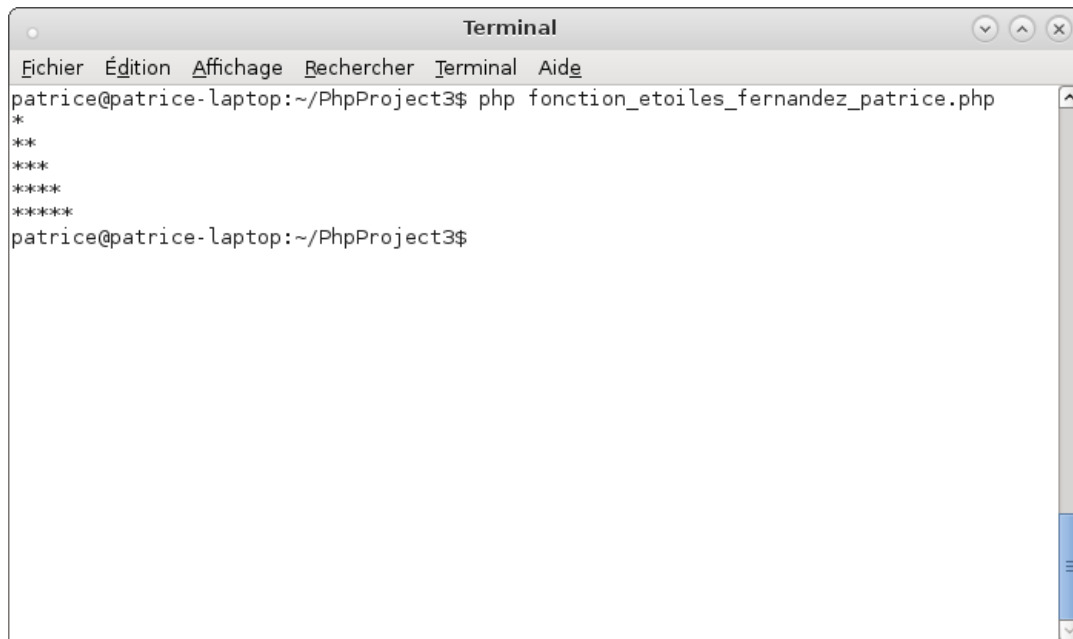


# Introduction au langage de programmation PHP

[Accueil](#) ▶ [Mes cours](#) ▶ [Développement logiciel](#) ▶ [Intro PHP](#) ▶ [Les fonctions](#) ▶ [T.P. fonction d'étoiles](#)

## T.P. fonction d'étoiles

Implémentez une fonction qui affichera un cône d'étoiles :



```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php fonction_etoiles_fernandez_patrice.php
*
**
***
****
patrice@patrice-laptop:~/PhpProject3$
```

Vous lui passerez en argument le nombre d'étages à afficher.

Le nom du script sera de la forme "fonction\_etoiles\_*nom\_prenom*.php". Vous remettrez une archive zip contenant ce fichier, le nom de l'archive sera de la forme "fonction\_etoiles\_*nom\_prenom*.zip"

## État du travail remis

Numéro de tentative	Ceci est la tentative 1.
Statut des travaux remis	Aucune tentative
Statut de l'évaluation	Pas évalué
Dernière modification	vendredi 27 mars 2015, 10:05

Ajouter un travail

Modifier votre travail remis

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



# 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 :

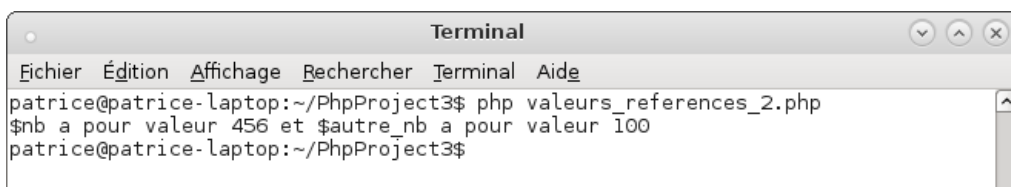


```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php valeurs_references_1.php
$nb a pour valeur 100 et $autre_nb a pour valeur 100
patrice@patrice-laptop:~/PhpProject3$
```

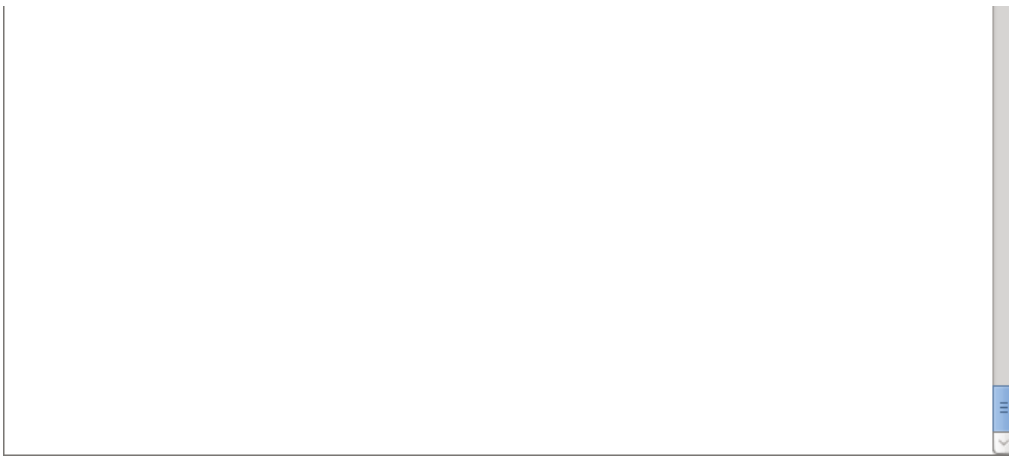
Maintenant si je modifie la valeur de `nb` juste avant l'affichage :

```
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 :



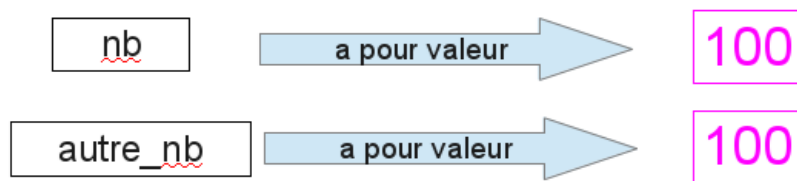
```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php valeurs_references_2.php
$nb a pour valeur 456 et $autre_nb a pour valeur 100
patrice@patrice-laptop:~/PhpProject3$
```



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;
```

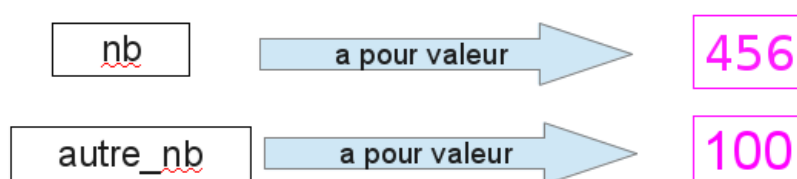
assigne à autre\_nb une copie de la valeur de nb. Je souhaite maintenant, à la suite du code, modifier la valeur de \$nb :

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

L'instruction

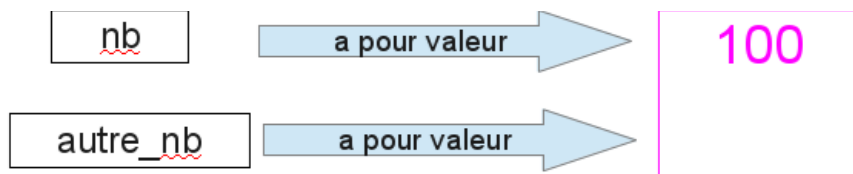
```
$nb = 456;
```

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 :

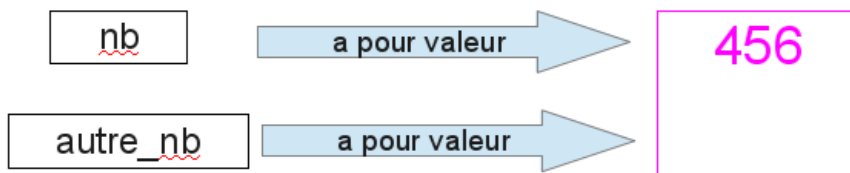
```

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(), $nb vaut 100
patrice@patrice-laptop:~/PhpProject3$

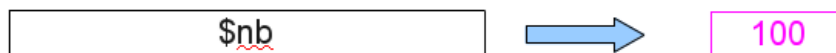
```

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 :

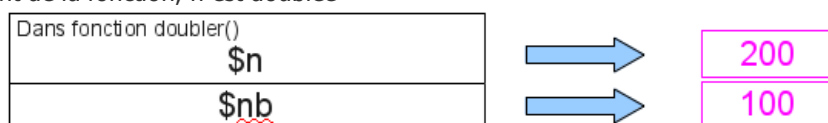
1. je déclare une variable `nb` et lui assigne la valeur 100



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



# Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fonctions](#) ► [T.P. fonction doubler](#)

## T.P. fonction doubler

Implémentez une fonction qui doublera toutes les valeurs d'un tableau d'entiers passé en argument.

Le nom du script sera de la forme "fonction\_doubler\_*nom\_prenom*.php". Vous remettrez une archive contenant ce fichier, le nom de l'archive sera de la forme "fonction\_doubler\_*nom\_prenom*.zip".

## État du travail remis

Numéro de tentative	Ceci est la tentative 1.
Statut des travaux remis	Aucune tentative
Statut de l'évaluation	Pas évalué
Dernière modification	vendredi 27 mars 2015, 13:21

Ajouter un travail

Modifier votre travail remis

### 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

---

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



# Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fonctions](#) ► [T.P. fonction calculer](#)

## T.P. fonction calculer

Implémentez une fonction `calculer()` qui prendra en argument un opérateur parmi `+`, `-`, `/`, `*` (sous forme de chaîne de caractères), puis un nombre indéfini d'entiers (au minimum 2 entiers). Par exemple, `calculer('+', 1, 2, 3, 4, 5, 6)` ou encore `calculer('-', 20, 45)`. La fonction retournera le résultat du calcul.

Le nom script sera de la forme "fonction\_calculer\_nom\_prenom.php". Vous remettrez une archive zip contenant ce fichier, le nom de l'archive sera de la forme "fonction\_calculer\_nom\_prenom.zip".

## État du travail remis

Numéro de tentative	Ceci est la tentative 1.
Statut des travaux remis	Aucune tentative
Statut de l'évaluation	Pas évalué
Dernière modification	vendredi 27 mars 2015, 14:09

Ajouter un travail

Modifier votre travail remis

### 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](#)

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