

Les fonctions

Table des matières

I. Contexte	3
II. Notions de base : arguments et valeurs de retour	3
A. Les arguments	3
B. La valeur de retour	6
C. Exercice : Quiz	7
III. Les fonctions internes à PHP	9
A. Introduction.....	9
B. Les fonctions sur les chaînes de caractères	9
C. Les fonctions sur les tableaux.....	10
D. Les fonctions mathématiques	11
E. Débugguer	12
F. Exercice : Quiz	12
IV. Syntaxe : fonctions variables, fonctions anonymes, fonctions fléchées	13
A. Syntaxe : fonctions variables, fonctions anonymes, fonctions fléchées	13
B. Exercice : Quiz	14
V. Essentiel	15
VI. Auto-évaluation	15
A. Exercice	15
B. Test.....	15
Solutions des exercices	17

I. Contexte

Durée : 1 h 00

Prérequis : aucun

Environnement de travail : Visual Studio Code avec l'extension PHP IntelliSense

Contexte

Une fonction, en langage de programmation informatique, est une série d'instructions qui produit un résultat. Le plus souvent, cette fonction a un nom. Le nom des fonctions, par convention, est explicite : il décrit ce que fait la fonction, le plus souvent grâce à un verbe qui décrit l'action (get, set, find, replace, etc.).

Pour exécuter la fonction, on dit qu'on l'appelle ou qu'on l'invoque grâce à son nom, suivi de parenthèses. Ces parenthèses peuvent comprendre les arguments de la fonction qui sont nécessaires à sa bonne exécution. Une fois appelée, la fonction exécute les instructions qu'elle contient, et produit toujours le même résultat. Elle est donc réutilisable : elle peut être appelée autant de fois que nécessaire. Elle permet d'éviter de répéter inutilement le même code, puisqu'il suffit d'appeler de nouveau la fonction.

Objectif

L'objectif de ce cours est de comprendre les éléments de base des fonctions, ainsi que de découvrir les fonctions internes à PHP.

II. Notions de base : arguments et valeurs de retour

A. Les arguments

Pour s'exécuter correctement, une fonction a parfois besoin d'arguments. Ce sont des informations qui sont passées entre parenthèses, séparées par une virgule s'il y en a plusieurs.

Exemple

Prenons par exemple la fonction `strlen()`, qui permet de connaître la taille d'une chaîne de caractère. Cette fonction prend obligatoirement un paramètre qui doit être de type string : c'est la chaîne de caractère que l'on veut mesurer. Pour connaître la taille de la chaîne de caractère suivante : *"Don't repeat yourself"*, vous pouvez appeler la fonction comme dans l'exemple ci-dessous.

```
1 <?php
2
3 strlen("Don't repeat yourself");
```

La chaîne de caractère passée entre les parenthèses est l'argument de la fonction. Il est aussi possible d'assigner cette chaîne de caractère à une variable, et de passer ensuite la variable comme argument.

```
1 <?php
2
3 $sentence = "Don't repeat yourself";
4 strlen($sentence);
```

Complément

Vous avez peut-être déjà entendu parler des paramètres d'une fonction. Les paramètres sont les noms listés lors de la définition de la fonction. Ainsi, la définition de la fonction `strlen()` comprend un paramètre, une chaîne de caractère. L'argument est la variable que vous passez quand vous appelez la fonction. Dans l'exemple précédent, la variable `$sentence` est l'argument de la fonction.

Attention

Si la fonction prend plusieurs arguments, ils doivent être passés dans le même ordre que les paramètres de la fonction.

Exemple

Imaginons une application web qui salue un utilisateur connecté en utilisant son prénom puis son nom.

```
1 <?php
2
3 function sayHello($firstName, $lastName) {
4     echo "Bonjour " . $firstName . " " . $lastName . "!";
5 }
6
7 sayHello("John", "Doe");
```

Dans cet exemple, le programme saluera l'utilisateur par un *"Bonjour John Doe!"*. Vous pouvez constater que les arguments passés à la fonction `sayHello()` respectent l'ordre des paramètres définis en amont.

Lors de la définition d'une fonction, il est possible de spécifier le type des paramètres. C'est le cas lorsque vous consultez la documentation officielle des fonctions internes de PHP. Cela vous fournit une information sur les arguments attendus.

Exemple

Dans le cas de la fonction `sayHello()`, il peut être utile de spécifier que les variables attendues en arguments doivent toujours être des chaînes de caractères. Si les arguments fournis sont d'un autre type, le programme renverra une erreur.

```
1 <?php
2
3 function sayHello(string $firstName, string $lastName) {
4     echo "Bonjour " . $firstName . " " . $lastName . "!";
5 }
```

Dans cet exemple, si la fonction `sayHello()` est appelée avec un tableau comme argument, le programme renverra une erreur de type `TypeError`.

Attention

À noter que, dans certains contextes, PHP va convertir le type de l'argument pour le faire correspondre au type attendu. Ainsi, si vous passez un integer au lieu d'une chaîne de caractère, celui-ci sera traité comme une chaîne de caractère.

La liste des arguments n'a pas toujours une taille fixe. Une fonction peut accepter un nombre illimité d'arguments si elle utilise le token `"..."`. Les arguments doivent être séparés par des virgules. La fonction les traite alors comme un tableau.

Exemple

La fonction `sayHello()` sert désormais à saluer une liste de gens qui lui sont fournis. Quel que soit le nombre d'arguments, elle les parcourt un par un et salue chaque personne.

```
1 <?php
2
3 function sayHello(...$persons) {
4     foreach($persons as $person) {
5         echo "Bonjour " . $person . "!\n";
6     }
7 }
8
9 sayHello("Agnès", "Kenza", "Nour");
```

C'est le token `"..."` placé devant le paramètre `$persons` de la fonction `sayHello()` qui vous indique qu'elle accepte un nombre illimité d'arguments. À l'intérieur de la fonction, `$persons` est un tableau, ce qui permet à la fonction d'itérer sur chaque élément grâce à l'utilisation de la boucle `foreach`.

Une fonction peut recevoir des arguments optionnels. S'ils ne sont pas présents lors de l'appel de la fonction, c'est leur valeur par défaut qui sera utilisée.

Exemple

```
1 <?php
2
3 function sayHello($name = "illustre inconnu") {
4     echo "Bonjour " . $name . "!\n";
5 }
6
7 sayHello();
```

Dans cet exemple, l'application saluera l'utilisateur par un *"Bonjour illustre inconnu!"*, car aucun nom n'a été passé en argument. C'est donc la valeur par défaut qui a été utilisée.

C'est ici que l'autocomplétion d'un éditeur de texte comme Visual Studio Code est précieuse, grâce à l'extension PHP IntelliSense. Elle permet de savoir rapidement le nombre d'arguments qui doivent être passés à la fonction interne que vous êtes en train d'appeler, leur type, et s'ils sont optionnels, quelle est leur valeur par défaut. Elle vous signale également si des arguments sont manquants, ou d'un type erroné.

Par défaut, les variables passées comme arguments à une fonction le sont par valeur, et non par référence. Cela signifie qu'une variable définie globalement, et modifiée à l'intérieur de la fonction, ne voit pas sa valeur modifiée en dehors de la fonction.

Exemple

La fonction `sayHello()` doit passer la première lettre du prénom en majuscule avant de saluer l'utilisateur. Pour cela, elle utilise la fonction interne de PHP `ucfirst()`, qui passe le premier caractère d'une chaîne en majuscule.

```
1 <?php
2
3 $name = "hector";
4
5 function sayHello($name) {
6     $name = ucfirst($name);
7     echo "Bonjour " . $name . "!\n";
8 }
```

```
9
10 sayHello($name);
```

Dans cet exemple, le programme saluera bien l'utilisateur avec une majuscule à son prénom ("*Bonjour Hector!*"). Mais hors de la fonction `sayHello()`, la variable `$name` garde sa valeur initiale ("*hector*").

Un argument est passé par référence si un `&` figure devant l'argument devant la déclaration de la fonction.

```
1 <?php
2
3 $name = "hector";
4
5 function sayHello(&$name) {
6     $name = ucfirst($name);
7     echo "Bonjour " . $name . "!";
8 }
9
10 sayHello($name);
```

Dans cet exemple, après l'exécution de la fonction, la variable `$name` a changé. Sa valeur est désormais "*Hector*".

Attention

Il est important d'être attentif à ce symbole, par exemple lorsque vous consultez la documentation d'une fonction interne à PHP. Si ce symbole `&` est présent devant le paramètre de la fonction, notez bien que la variable passée en argument sera passée par référence et donc modifiée par l'exécution de la fonction.

B. La valeur de retour

Lorsqu'une fonction est exécutée, elle peut produire un résultat (modifier une variable, afficher du texte, lire un document pdf, etc.), mais elle peut aussi renvoyer un résultat. Ce comportement est obtenu avec la commande `return`. Lorsque cette commande est appelée, la fonction cesse son exécution. Cela signifie que toutes les instructions après un `return` ne seront jamais exécutées. La valeur renvoyée par la commande `return` peut être enregistrée dans une variable, pour être réutilisée.

Exemple

Prenons l'exemple d'une fonction qui vérifie si une personne est majeure ou non.

```
1 <?php
2
3 function isOver18($age) {
4     if($age < 18) {
5         return false;
6     } return true;
7 }
8
9 $result = isOver18(17);
```

Dans cet exemple, on voit que le résultat de l'appel de la fonction avec la valeur 17 est enregistré dans la variable `$result`. La valeur de la variable `$result` est donc la valeur retournée par la fonction avec 17 pour argument (`false`). Dans cet exemple, la condition `$age < 18` est vraie, la fonction cesse donc de s'exécuter à l'intérieur du `if`, au premier `return`. Tant que cette condition est vraie, la fonction cesse toujours son exécution au même endroit et ne renverra jamais `true`.

Attention

Si la fonction n'a pas de commande `return`, sa valeur de retour sera toujours `null`.

Exemple

Si l'on reprend l'exemple de la fonction `sayHello()`, on note que celle-ci affiche une chaîne de caractère grâce à la commande `echo`, mais elle ne renvoie aucune valeur. Dans l'exemple ci-dessous, la valeur de `$result` sera donc `null`.

```
1 <?php
2
3 function sayHello($firstName, $lastName) {
4     echo "Bonjour " . $firstName . " " . $lastName . "!";
5 }
6
7 $result = sayHello("John", "Doe");
```

Tout comme les paramètres d'une fonction, sa valeur de retour peut être typée. Cela signifie que lors de la définition, le type de la valeur de retour a été précisé. Cette information est à prendre en compte lorsque vous consultez la documentation de PHP, ou que vous prototypez vos propres fonctions. Le type de la valeur de retour est précisé lors de la définition de la fonction, juste après les paramètres.

```
1 <?php
2
3 function isOver18($age): bool {
4     if($age < 18) {
5         return false;
6     } return true;
7 }
```

Si la fonction ne retourne rien, le type de la valeur de retour sera `void`. Une fonction dont la valeur de retour est de type `mixed` peut retourner différents types de variables.

C. Exercice : Quiz

[solution n°1 p.19]

Question 1

Une fonction ne peut prendre qu'un nombre limité d'arguments.

- ☐ Vrai
- ☐ Faux

Question 2

Quelle est la valeur de la variable `$result` à l'exécution du code suivant ?

```
1 <?php
2
3 function isItLunchTime(int $time): bool {
4     if($time >= 12 && $time <= 14) {
5         return true;
6     } return false;
7     return "no";
8 }
9
10 $result = isItLunchTime(13);
```

- ☐ True
- ☐ False
- ☐ "no"
- ☐ Void

Question 3

Quelle sera la valeur de \$newBudget à l'exécution du code suivant ?

```
1 <?php
2
3 function decrementBudget(int $budget, int $expenditure = 10): int {
4     return $budget -= $expenditure;
5 }
6
7 $newBudget = decrementBudget(200);
```

- ☐ 200
- ☐ 190
- ☐ Error

Question 4

Quelle est la valeur de retour d'une fonction dans laquelle ne figure pas la commande return ?

- ☐ false
- ☐ undefined
- ☐ null
- ☐ 0

Question 5

Quelle est la valeur de la variable \$age après l'exécution du code PHP suivant ?

```
1 <?php
2 $age = 17;
3
4 function getOlder(int $age): int {
5     $age = $age + 1;
6     return $age;
7 }
8
9 getOlder($age);
```

- ☐ 18
- ☐ Error
- ☐ 17

III. Les fonctions internes à PHP

A. Introduction

PHP dispose de nombreuses fonctions internes, déjà définies. On a vu dans la première partie des fonctions de ce type, telles que `strlen()` ou `count()`.

Attention

Certaines fonctions internes à PHP ne sont disponibles qu'avec des extensions de PHP. Nous verrons ici les fonctions de base, pour travailler par exemple sur les chaînes de caractères ou les tableaux ou réaliser des opérations mathématiques.

B. Les fonctions sur les chaînes de caractères

Ces fonctions permettent de manipuler des variables de type chaîne de caractère. Leur liste complète figure sur la documentation officielle de PHP¹.

Exemple

Prenons l'exemple d'une application qui reçoit des chaînes de caractères et qui les affiche dans un bloc à taille fixe, n'acceptant que 20 caractères. Toute chaîne qui dépasse cette limite est tronquée, et se termine par "...". Il existe deux fonctions internes qui permettent de réaliser cette opération. La première, on l'a vu, c'est `strlen()`, qui retourne le nombre de caractères que contient la chaîne. La deuxième, c'est `substr()`, qui retourne une sous-section d'une chaîne de caractère, et qui prend trois arguments. Voyons sa définition.

```
1 <?php
2 substr(string $string, int $offset, ?int $length = null): string
```

Le premier argument, `$string`, est la chaîne de caractère sur laquelle réaliser l'opération. Le deuxième, `$offset`, est l'indice de départ à partir duquel tronquer la chaîne.

Attention

L'indice du premier caractère d'une chaîne est 0.

Exemple

Le troisième argument, `$length`, est optionnel. S'il n'est pas spécifié, sa valeur par défaut est nulle. La valeur de retour de `substr()` est de type chaîne de caractère. Il suffit ensuite de concaténer les points de suspension à la fin de la valeur de retour.

```
1 <?php
2 $sentence = "It's raining cats and dogs";
3 if (strlen($sentence) > 20) {
4     $sentence = substr($sentence, 0, 17) . "...";
5 }
```

Voici quelques-unes des fonctions internes concernant les chaînes de caractères qui sont utiles à connaître :

- `str_split()` : convertit une chaîne de caractère en tableau associatif. Valeur de retour : le tableau.
- `implode()` : rassemble les éléments d'un tableau en une chaîne. Valeur de retour : la chaîne de caractère.
- `explode()` : scinde une chaîne de caractère selon le séparateur passé en argument, et retourne un tableau. Valeur de retour : le tableau.

¹ <https://www.php.net/manual/fr/ref.strings.php>

- `str_contains()` : permet de vérifier si une chaîne contient la portion de chaîne passée en argument. Valeur de retour : booléen.
- `str_replace()` : permet de remplacer une sous-section d'une chaîne de caractère par une autre. Valeur de retour : une chaîne de caractère.
- `str_repeat()` : répète une chaîne de caractère le nombre de fois désirées. Valeur de retour : une nouvelle chaîne de caractère.

Exemple

Imaginons un programme qui masque les noms d'animaux. La fonction `str_replace()` permet de chercher dans une phrase les mots à remplacer. Les arguments sont dans l'ordre : `$search`, la ou les valeurs recherchées (un tableau peut être utilisé), `$replace`, la ou les valeurs à remplacer, `$subject`, la chaîne de caractère sur lequel effectuer le remplacement, et en option, `$count`, le nombre de modifications à effectuer.

```
1 <?php
2 $sentence = "It's raining cats and dogs";
3 $redactedSentence = str_replace(["cats", "dogs"], "***", $sentence);
```

Dans cet exemple, la valeur de `$redactedSentence` est : *"It's raining *** and ***"*.

C. Les fonctions sur les tableaux

PHP dispose également d'une série de fonctions internes propres aux manipulations de tableaux (ajouter/supprimer des éléments, les trier, etc.).

Leur liste complète se trouve ici : PHP¹.

Attention

Pensez à consulter la documentation de chaque fonction pour vérifier si les tableaux passés en arguments le sont par valeur ou par référence.

Attention

- Fonctions qui modifient l'argument, passé par référence :
 - `array_push()` : ajoute un ou plusieurs éléments à la fin d'un tableau. Valeur de retour : le tableau modifié.
 - `array_pop()` : supprime le dernier élément d'un tableau. Valeur de retour : l'élément supprimé.
 - `array_shift()` : supprime le premier élément d'un tableau. Valeur de retour : l'élément supprimé.
 - `array_unshift()` : ajoute un ou plusieurs éléments au début d'un tableau. Valeur de retour : le nombre d'éléments du tableau modifié.
 - `array_splice()` : extrait un ou plusieurs éléments d'un tableau, et les remplace par les éléments passés en arguments. Valeur de retour : un tableau qui contient les éléments extraits.
 - `sort()` : classe les éléments d'un tableau par ordre croissant.
- Fonctions qui ne modifient pas l'argument, passé par valeur :
 - `count()` : compte les éléments d'un tableau. Valeur de retour : le nombre d'éléments.
 - `array_slice()` : extrait des éléments d'un tableau. Valeur de retour : un tableau comprenant les éléments extraits.
 - `array_sum()` : additionne toutes les valeurs d'un tableau. Valeur de retour : somme des valeurs.

¹ <https://www.php.net/manual/fr/ref.array.php>

Exemple

Voilà un exemple qui permet d'illustrer la différence entre `array_slice()` et `array_splice()`. Essayez de deviner les valeurs de la variable `$animals` après chaque opération.

```
1 <?php
2 $animals = ['dogs', 'cats', 'birds'];
3 array_push($animals, 'horses', 'cows');
4 // Etape 1 : quelle est la valeur de $animals ?
5 $animalsLength = count($animals);
6 $animalsSliced = array_slice($animals, 0, ($animalsLength - 1));
7 // Etape 2 : quelle est la valeur de $animals ?
8 $animalsSpliced = array_splice($animals, $animalsLength - 2, 1, 'ducks');
9 // Etape 3 : quelle est la valeur de $animals ?
```

`$animals` est passé par référence à la fonction `array_push()`. À l'étape 1, `$animals` comprend donc ['dogs', 'cats', 'birds', 'horses', 'cows'].

`$animals` est passé par valeur à la fonction `array_slice()`. À l'étape 2, `$animals` comprend donc toujours ['dogs', 'cats', 'birds', 'horses', 'cows'].

`$animals` est passé par référence à la fonction `array_splice()`. À l'étape 3, `$animals` comprend donc ['dogs', 'cats', 'birds', 'ducks', 'cows']. C'est l'avant-dernier élément de `$animals` qui a été remplacé.

D. Les fonctions mathématiques

Les fonctions mathématiques permettent de réaliser des opérations sur des nombres en PHP.

Leur liste complète se trouve ici : PHP¹.

Parmi les plus utiles, on trouve les fonctions `round()`, `ceil()` et `floor()` qui permettent d'arrondir un nombre décimal.

Exemple

Prenons π , que l'on peut obtenir grâce à la fonction `pi()`. La fonction `round()` permet d'arrondir au nombre de chiffres après la virgule que l'on souhaite. La fonction `floor()` arrondit à l'entier inférieur. La fonction `ceil()` arrondit à l'entier supérieur.

```
1 <?php
2 $round = round(pi(), 3) // 3.142
3 $floor = floor(pi()) // 3
4 $ceil = ceil(pi()) // 4
```

Les fonctions `min()` et `max()` sont également utiles pour connaître les valeurs minimales et maximales d'une suite de nombres. L'argument peut être un tableau, ou bien une suite de nombre séparés par des virgules.

Exemple

Prenons une application qui reçoit une liste de notes et affiche les notes minimales et maximales.

```
1 <?php
2 $grades = [12, 8, 18, 5, 14, 11];
3 echo 'Les notes sont comprises entre ' . min($grades) . ' et ' . max($grades) . '.'
```

Dans cet exemple, le programme affichera "Les notes sont comprises entre 5 et 18."

¹ <https://www.php.net/manual/fr/book.math>

E. Débugguer

Il existe deux fonctions utiles pour débbugguer du code, `var_dump()` et `print_r()`. Les deux affichent dans la console le contenu d'une variable. `var_dump()` est plus précise mais plus difficile à lire, car elle affiche le type et la valeur d'une variable. `print_r()` formate l'information de façon plus lisible.

F. Exercice : Quiz

[solution n°2 p.20]

Question 1

La fonction `count()` permet de connaître la longueur d'une chaîne de caractère.

- ☐ Vrai
- ☐ Faux

Question 2

Les fonctions internes ne prennent comme arguments que des variables passées par valeur.

- ☐ Vrai
- ☐ Faux

Question 3

Quel est la valeur de `$fruits` après l'exécution du code suivant ?

```
1 <?php
2 $fruits = ['banana', 'apple', 'kiwi'];
3 array_slice($fruits, count($fruits) - 1, 1);
```

- ☐ ['banana', 'apple', 'kiwi']
- ☐ ['kiwi']
- ☐ ['banana', 'apple']

Question 4

Quelle est la fonction qui permet d'arrondir un nombre décimal à l'entier inférieur ?

- ☐ `$round`
- ☐ `ceil()`
- ☐ `floor()`

Question 5

Quelle fonction transforme la chaîne de caractères suivante en tableau de prénoms ?

```
1 <?php
2 $listOfNames = 'Ahmed;Anna;Elise;Franck';
```

- ☐ `explode(';', $listOfNames)`
- ☐ `str_split($listOfNames)`
- ☐ `implode(';', $listOfNames)`

IV. Syntaxe : fonctions variables, fonctions anonymes, fonctions fléchées

A. Syntaxe : fonctions variables, fonctions anonymes, fonctions fléchées

Nous avons vu la façon classique d'écrire et d'appeler une fonction. PHP gère plusieurs syntaxes alternatives qui permettent d'écrire ou d'appeler une fonction différemment.

La première est la fonction variable. Si une variable est suivie de parenthèse, PHP interprète le code comme un appel de fonction et va rechercher dans le code la fonction du même nom.

Exemple

```
1 <?php
2 function reverseBoolean($boolean) {
3     if ($boolean) {
4         return false;
5     } return true;
6 };
7 $opposite = 'reverseBoolean';
8 $opposite(true);
```

Dans l'exemple ci-dessus, la variable `$opposite` est interprétée comme l'appel de la fonction `reverseBoolean()`, car elle est suivie de parenthèses.

Jusqu'ici, les fonctions que nous avons vu ont un nom, ce qui permet de les appeler plusieurs fois dans le code. Mais certaines fonctions peuvent être écrites sans préciser leur nom. On les appelle fonctions anonymes, mais aussi fermetures, ou closures. Elles sont notamment utilisées dans le cas des fonctions de rappel.

Définition

Une fonction de rappel est une fonction passée comme argument à une autre fonction.

Exemple

Certaines fonctions internes de PHP utilisent les fonctions de rappel. Parmi celles-ci, il y a `array_map()`, qui permet d'effectuer la même opération sur chaque élément d'un tableau. `array_map()` prend comme premier argument une fonction de rappel (aussi appelée callback) et un tableau.

```
1 <?php
2 $numbers = [1, 2, 3];
3 $doubles = array_map(function($element) {
4     return $element * 2;
5 }, $numbers);
```

Dans cet exemple, on voit que le premier argument de la fonction `array_map()` est une fonction anonyme, qui multiplie par 2 chaque élément du tableau passé en deuxième argument.

Depuis PHP 7.4, les fonctions anonymes peuvent être écrites de façon plus concise. Ce sont les fonctions fléchées. La syntaxe d'une fonction fléchée est la suivante : `fn ($argument) => valeur_de_retour`.

Exemple

Si l'on reprend l'exemple précédent, mais que la fonction anonyme est écrite cette fois-ci comme une fonction fléchée, cela donne le code suivant.

```
1 <?php
2 $numbers = [1, 2, 3];
3 $doubles = array_map(fn($element) => $element * 2, $numbers);
```

B. Exercice : Quiz

[solution n°3 p.21]

Question 1

Une fonction variable est une fonction dont la valeur de retour peut-être enregistrée dans une variable.

- ☐ Vrai
- ☐ Faux

Question 2

Quelle est la valeur de la variable \$greetings('Sara') ?

```
1 <?php
2 function sayHello($name) {
3     return 'Bonjour ' . $name . '!';
4 }
5 $greetings = 'sayHello';
6 $greetings('Sara');
```

- ☐ 'sayHello'
- ☐ 'Bonjour Sara!'
- ☐ Error

Question 3

Une fonction anonyme est une fonction de rappel.

- ☐ Vrai
- ☐ Faux

Question 4

Ces deux fonctions retournent-elles le même résultat ?

```
1 <?php
2 function ($element) {
3     return $element + 2;
4 }
5 fn($element) => $element + 2;
```

- ☐ Oui
- ☐ Non

Question 5

Le code ci-dessous va-t-il renvoyer une erreur ?

```
1 <?php
2 $names = ['Lionel', 'Alma', 'Nour'];
3 array_map(fn($name) => 'Bonjour ' . $name, $names);
```

- ☐ Oui
- ☐ Non

V. Essentiel

Une fonction est une suite d'opérations qui produit un résultat. Elle peut accepter des arguments, qui sont des variables dont elle a besoin pour exécuter les instructions qu'elle contient. C'est la définition d'une fonction qui fixe le nombre d'arguments, leur type, et leur valeur par défaut s'ils sont optionnels. La définition d'une fonction permet aussi de savoir si la fonction a une valeur de retour, c'est-à-dire que son exécution produit une variable qu'il est possible de sauvegarder.

PHP dispose d'un grand nombre de fonctions internes. Ces fonctions déjà définies permettent d'exécuter du code parfois complexe sans avoir besoin de réécrire toute la logique nécessaire. La documentation officielle de PHP les recense toutes, et permet de connaître leur définition. Comprendre la définition d'une fonction est essentiel pour bien l'utiliser. Savoir si une fonction va modifier la variable passée en argument, par exemple, est déterminant pour la suite de l'exécution du code.

PHP accepte plusieurs syntaxes pour l'écriture d'une fonction. Ces différences ont un impact sur la concision du code, mais aussi sur la possibilité ou non de le réutiliser. Par exemple, une fonction amenée à être appelée plusieurs fois doit avoir un nom. Une fonction de rappel utilisée à l'intérieur d'une autre fonction n'a, elle, pas besoin d'en avoir un.

VI. Auto-évaluation

A. Exercice

Vous développez une application dédiée au cinéma. Pour chaque film, vous souhaitez afficher le nom du film, la moyenne des notes qui lui ont été attribuées, et les commentaires des spectateurs.

```
1 <?php
2 $movie = "Children of Men";
3 $rates = [3, 4, 2, 4, 3, 5, 0];
4 $comment = ["Albert", "Super film, j'ai adoré!"];
```

Question 1

[solution n°4 p.23]

Comment calculer la moyenne des notes contenues dans la variable \$rates ?

Question 2

[solution n°5 p.23]

Comment anonymiser l'auteur du commentaire, en ne gardant que la première lettre de son prénom et en remplaçant la suite du nom par des * ? (Exemple = 'Albert' => 'A*****')

B. Test

Exercice 1 : Quiz

[solution n°6 p.23]

Question 1

Comment s'appelle la fonction passée en premier argument de `array_map()` dans le code suivant ?

```
1 <?php
2 $ages = [23, 45, 72, 12];
3 array_map(fn($age) => $age + 1, $ages);
```

- ☐ Une fonction fléchée
- ☐ Une fonction interne
- ☐ Une fonction anonyme
- ☐ Une fonction variable

Question 2

La fonction ci-dessous s'exécutera correctement.

```
1 <?php
2 function sayHello($firstName, $lastName, $welcome = true) {
3     if ($welcome) {
4         echo 'Bonjour ' . $firstName . ' ' . $lastName . '. Bienvenue !';
5     } else {
6         echo 'Bonjour ' . $firstName . ' ' . $lastName . '.';
7     }
8 }
9 sayHello('Jane', 'Doe');
```

- ☐ Vrai
- ☐ Faux

Question 3

Quelle est le type de la valeur de retour de la fonction `sayHello()` ?

```
1 <?php
2 function sayHello($firstName, $lastName, $welcome = true) {
3     if ($welcome) {
4         echo 'Bonjour ' . $firstName . ' ' . $lastName . '. Bienvenue !';
5     } else {
6         echo 'Bonjour ' . $firstName . ' ' . $lastName . '.';
7     }
8 }
```

- ☐ Void
- ☐ Null
- ☐ Undefined

Question 4

Quelle est la valeur de la variable `$animals` après l'exécution du code suivant ?

```
1 <?php
2 $animals = ['elephant', 'tiger', 'lion'];
3 array_splice($animals, 2, 1, 'cat');
```

- ☐ ['elephant', 'tiger', 'lion']
- ☐ ['elephant', 'tiger', 'cat']
- ☐ ['lion']

Question 5

Laquelle de ces fonctions ne modifie pas \$animals ?

```
1 <?php  
2 $animals = ['elephant', 'tiger', 'lion'];
```

- ☐ array_pop(\$animals);
- ☐ array_splice(\$animals, count(\$animals) - 1, 1);
- ☐ array_slice(\$animals, count(\$animals) - 1, 1);


Solutions des exercices

Exercice p. 7 Solution n°1**Question 1**

Une fonction ne peut prendre qu'un nombre limité d'arguments.

☐ Vrai

☒ Faux

 Si le token "..." est présent devant le paramètre de la fonction, cela signifie qu'elle peut recevoir un nombre illimité d'arguments. Ils seront alors traités comme un tableau.

Question 2

Quelle est la valeur de la variable \$result à l'exécution du code suivant ?


```
1 <?php
2
3 function isItLunchTime(int $time): bool {
4     if($time >= 12 && $time <= 14) {
5         return true;
6     } return false;
7     return "no";
8 }
9
10 $result = isItLunchTime(13);
```

☒ True

☐ False

☐ "no"

☐ Void

 La condition étant évaluée à vrai, c'est la valeur de retour dans la condition qui sera renvoyée par la fonction, donc true. La commande return "no" ne sera jamais exécutée, car la fonction cessera toujours son exécution avant. Le typage permet de vérifier que la fonction prend un integer en argument, et renvoie toujours un booléen.

Question 3


Quelle sera la valeur de \$newBudget à l'exécution du code suivant ?

```
1 <?php
2
3 function decrementBudget(int $budget, int $expenditure = 10): int {
4     return $budget -= $expenditure;
5 }
6
7 $newBudget = decrementBudget(200);
```

☐ 200


☒ 190

☐ Error

 La fonction decrementBudget() est appelée avec un seul argument, c'est donc la valeur par défaut du deuxième argument qui est prise en compte. 200 - 10 = 190.

Question 4


Quelle est la valeur de retour d'une fonction dans laquelle ne figure pas la commande return ?

- ☐ false
- ☐ undefined
- ☒ null
- ☐ 0
-  En l'absence de la commande return, la valeur de retour d'une fonction est toujours null.

Question 5

Quelle est la valeur de la variable \$age après l'exécution du code PHP suivant ?


```
1 <?php
2 $age = 17;
3
4 function getOlder(int $age): int {
5     $age = $age + 1;
6     return $age;
7 }
8
9 getOlder($age);
```

- ☐ 18
- ☐ Error
- ☒ 17
-  La variable \$age est passée par valeur à la fonction getOlder(), ce qui signifie que la variable \$age n'est pas affectée. À la fin de l'exécution de ce script, la valeur de \$age est toujours de 17. Pour conserver la valeur modifiée par la fonction, il faut l'enregistrer dans une nouvelle variable.

Exercice p. 12 Solution n°2


Question 1

La fonction count() permet de connaître la longueur d'une chaîne de caractère.

- ☐ Vrai
- ☒ Faux
-  C'est la fonction strlen() qui permet de connaître la longueur d'une chaîne de caractère. La fonction count() permet de connaître le nombre d'éléments d'un tableau.

Question 2

Les fonctions internes ne prennent comme arguments que des variables passées par valeur.


- ☐ Vrai
- ☒ Faux
-  De nombreuses fonctions internes qui concernent les tableaux modifient directement le tableau passé en argument.

Question 3

Quel est la valeur de \$fruits après l'exécution du code suivant ?

```
1 <?php
2 $fruits = ['banana', 'apple', 'kiwi'];
3 array_slice($fruits, count($fruits) - 1, 1);
```


- ☒ ['banana', 'apple', 'kiwi']
- ☐ ['kiwi']
- ☐ ['banana', 'apple']

 La variable passée en argument à la fonction array_slice() est passée par valeur, elle n'est donc pas transformée par l'exécution de la fonction. Pour sauvegarder l'élément extrait par array_slice(), il faut enregistrer le résultat de l'appel de la fonction dans une nouvelle variable.

Question 4

Quelle est la fonction qui permet d'arrondir un nombre décimal à l'entier inférieur ?

- ☐ \$round
- ☐ ceil()
- ☒ floor()


 floor() arrondit un nombre décimal à l'entier inférieur. ceil() arrondit un nombre décimal à l'entier supérieur.

Question 5

Quelle fonction transforme la chaîne de caractères suivante en tableau de prénoms ?

```
1 <?php
2 $listOfNames = 'Ahmed;Anna;Elise;Franck';
```

- ☒ explode(';', \$listOfNames)
- ☐ str_split(\$listOfNames)
- ☐ implode(';', \$listOfNames)


 La fonction explode() transforme en tableau une chaîne de caractère. C'est le séparateur passé en argument (ici, le point-virgule) qui permet de séparer chaque élément du suivant. str_split() transforme chaque caractère en élément du tableau. implode() transforme un tableau en chaîne de caractère.

Exercice p. 14 Solution n°3

Question 1

Une fonction variable est une fonction dont la valeur de retour peut-être enregistrée dans une variable.


- ☐ Vrai
- ☒ Faux

 Une fonction variable est une variable dont la valeur correspond au nom d'une fonction. Si la variable est suivie de parenthèses, PHP comprend qu'il s'agit d'un appel de fonction et recherche dans le code la fonction du même nom afin de l'exécuter.

Question 2


Quelle est la valeur de la variable \$greetings('Sara') ?

```
1 <?php
2 function sayHello($name) {
3     return 'Bonjour ' . $name . '!';
4 }
5 $greetings = 'sayHello';
6 $greetings('Sara');
```

- ☐ 'sayHello'
- ☒ 'Bonjour Sara!'
- ☐ Error
-  PHP interprète la variable \$greetings(Sara) comme l'appel de la fonction sayHello avec 'Sara' comme argument.

Question 3


Une fonction anonyme est une fonction de rappel.

- ☐ Vrai
- ☒ Faux
-  Une fonction anonyme est une fonction qui n'a pas de nom. Elle peut souvent être utilisée comme fonction de rappel (callback). Une fonction de rappel est une fonction passée comme argument à une autre fonction.

Question 4

Ces deux fonctions retournent-elles le même résultat ?


```
1 <?php
2 function ($element) {
3     return $element + 2;
4 }
5 fn($element) => $element + 2;
```

- ☒ Oui
- ☐ Non
-  La première fonction est une fonction anonyme. Elle prend un \$element en argument et retourne la valeur de \$element + 2. La deuxième fonction est une fonction fléchée, sa syntaxe est différente mais elle produit le même résultat.

Question 5

Le code ci-dessous va-t-il renvoyer une erreur ?

```
1 <?php
2 $names = ['Lionel', 'Alma', 'Nour'];
3 array_map(fn($name) => 'Bonjour ' . $name, $names);
```

- ☐ Oui
- ☒ Non
-  La fonction array_map de PHP prend comme premier argument une fonction de rappel. Cette fonction peut-être une fonction fléchée. Elle s'exécute sur tous les éléments du tableau passé en deuxième argument.

p. 15 Solution n°4

```

1 <?php
2 $total = array_sum($rates);
3 $numberOfElements = count($rates);
4 $average = $total / $numberOfElements;

```

La fonction `array_sum()` permet d'additionner toutes les valeurs de `$rates`. La fonction `count()` permet-elle de connaître le nombre d'éléments du tableau `$rates`. La moyenne des notes est donc le total divisé par le nombre d'éléments.

p. 15 Solution n°5

Exercice p. 15 Solution n°6


Question 1

Comment s'appelle la fonction passée en premier argument de `array_map()` dans le code suivant ?

```

1 <?php
2 $ages = [23, 45, 72, 12];
3 array_map(fn($age) => $age + 1, $ages);

```

- ☒ Une fonction fléchée
 - ☐ Une fonction interne
 - ☐ Une fonction anonyme
 - ☐ Une fonction variable
-  Une fonction fléchée est une manière plus concise d'écrire une fonction anonyme, que l'on peut résumer comme ceci : `fn(argument) => valeur_de_retour`.


Question 2

La fonction ci-dessous s'exécutera correctement.

```

1 <?php
2 function sayHello($firstName, $lastName, $welcome = true) {
3     if ($welcome) {
4         echo 'Bonjour ' . $firstName . ' ' . $lastName . '. Bienvenue !';
5     } else {
6         echo 'Bonjour ' . $firstName . ' ' . $lastName . '.';
7     }
8 }
9 sayHello('Jane', 'Doe');


```

- ☒ Vrai
 - ☐ Faux
-  Le troisième paramètre de `sayHello()` a une valeur par défaut (`$welcome = true`). S'il est omis lors de l'appel de la fonction, sa valeur par défaut sera donc `true`.

Question 3

Quelle est le type de la valeur de retour de la fonction `sayHello()` ?


```
1 <?php
2 function sayHello($firstName, $lastName, $welcome = true) {
3     if ($welcome) {
4         echo 'Bonjour ' . $firstName . ' ' . $lastName . '. Bienvenue !';
5     } else {
6         echo 'Bonjour ' . $firstName . ' ' . $lastName . '.';
7     }
8 }
```

- ☐ Void
- ☒ Null
- ☐ Undefined
-  Si la commande return est omise, la valeur de retour de la fonction sera null.

Question 4

Quelle est la valeur de la variable \$animals après l'exécution du code suivant ?


```
1 <?php
2 $animals = ['elephant', 'tiger', 'lion'];
3 array_splice($animals, 2, 1, 'cat');
```

- ☐ ['elephant', 'tiger', 'lion']
- ☒ ['elephant', 'tiger', 'cat']
- ☐ ['lion']
-  La variable \$animals est passée par référence à la fonction array_splice, ce qui signifie qu'elle sera modifiée par l'exécution de la fonction. Celle-ci extrait des éléments de \$animals à partir de l'index 2. Le troisième argument est le nombre d'éléments que l'on souhaite retirer (ici, 1). Le dernier argument est l'élément à substituer. Ici, on enlève donc le dernier élément, 'lion', et on le remplace par 'cat'.

Question 5

Laquelle de ces fonctions ne modifie pas \$animals ?

```
1 <?php
2 $animals = ['elephant', 'tiger', 'lion'];
```

- ☐ array_pop(\$animals);
- ☐ array_splice(\$animals, count(\$animals) - 1, 1);
- ☒ array_slice(\$animals, count(\$animals) - 1, 1);
-  Les deux premières fonctions sont deux manières différentes de retirer le dernier élément du tableau \$animals. La troisième, array_slice(), extrait le dernier élément du tableau et le retourne. Le tableau \$animals n'est pas affecté par l'exécution de cette fonction.