**Table des matières**

[1 Compétences pédagogiques : 4](#_Toc144236724)

[2 PHP c’est quoi ? 5](#_Toc144236725)

[3 Spécificités du langage PHP 6](#_Toc144236726)

[PHP est un langage « interprété » 6](#_Toc144236727)

[PHP est un langage « à typage faible » ou « non-typé » 7](#_Toc144236728)

[4 Evolution du PHP 8](#_Toc144236729)

[5 Environnement de travail : 9](#_Toc144236730)

[6 Syntaxe du langage : 10](#_Toc144236731)

[Emplacement des fichiers, Exemple : 10](#_Toc144236732)

[Balises : 10](#_Toc144236733)

[Exemple : 11](#_Toc144236734)

[Exemple de code d’une page : 11](#_Toc144236735)

[Syntaxe générale : 12](#_Toc144236736)

[Commenter des lignes de codes : 13](#_Toc144236737)

[Création de notre premier programme en php : 14](#_Toc144236738)

[7 Les variables : 15](#_Toc144236739)

[Une variable ça sert à quoi ? 15](#_Toc144236740)

[Les types de variables : 15](#_Toc144236741)

[Nommer une variable : 16](#_Toc144236742)

[Déclaration d’une variable : 16](#_Toc144236743)

[Exemple : 17](#_Toc144236744)

[Afficher le contenu d’une variable : 17](#_Toc144236745)

[Afficher le type d’une variable : 18](#_Toc144236746)

[Exercice variables : 18](#_Toc144236747)

[8 Les constantes : 19](#_Toc144236748)

[9 Les opérateurs : 20](#_Toc144236749)

[Les opérateurs arithmétiques : 20](#_Toc144236750)

[Exercices Opérateurs arithmétiques : 21](#_Toc144236751)

[Les opérateurs d’incrémentation : 22](#_Toc144236752)

[10 Concaténation : 23](#_Toc144236753)

[Exemple : 23](#_Toc144236754)

[Exercices : 24](#_Toc144236755)

[Les opérateurs d’affectation : 25](#_Toc144236756)

[11 Les Fonctions : 25](#_Toc144236757)

[Création d’une fonction : 25](#_Toc144236758)

[Appel d’une fonction : 26](#_Toc144236759)

[Exemple : 26](#_Toc144236760)

[Création d’une fonction avec des paramètres : 26](#_Toc144236761)

[Exemple : 26](#_Toc144236762)

[Exercices : 27](#_Toc144236763)

[Typage des paramètres : 27](#_Toc144236764)

[Exercices : 28](#_Toc144236765)

[12 Les structures conditionnelles : 28](#_Toc144236766)

[Instruction if : 28](#_Toc144236767)

[Instruction else : 28](#_Toc144236768)

[Instruction elseif : 29](#_Toc144236769)

[Opérateur ternaire : 29](#_Toc144236770)

[Instruction switch : 30](#_Toc144236771)

[Opérateurs de comparaison : 31](#_Toc144236772)

[Opérateurs logiques : 31](#_Toc144236773)

[Exemple : 32](#_Toc144236774)

[Exercices : 32](#_Toc144236775)

[13 Les boucles : 33](#_Toc144236776)

[Boucle for : 33](#_Toc144236777)

[Exemple de boucle for : 33](#_Toc144236778)

[Boucle while : 34](#_Toc144236779)

[Exemple de boucle while : 34](#_Toc144236780)

[Boucle do…while : 34](#_Toc144236781)

[Exemple de boucle do…while : 34](#_Toc144236782)

[Boucle foreach : 35](#_Toc144236783)

[Exemple de boucle foreach : 35](#_Toc144236784)

[Exercices sur les boucles : 35](#_Toc144236785)

*Emplacement table des matières suite.*

## 1 Compétences pédagogiques :

Être capable de comprendre le fonctionnement des variables

Être capable de manipuler les opérateurs

Être capable d'utiliser les instructions conditionnelles

Être capable de manipuler un tableau

Être capable de comprendre les boucles

Être capable de créer et d'utiliser des fonctions

Être capable de comprendre le fonctionnement et l'intérêt de la programmation orienté objet

Être capable de créer et utiliser les classes

Être capable de créer et utiliser des objets

Être capable de comprendre les notions d'héritage

Être capable de comprendre les notions de polymorphisme

Être capable de créer des pages Web Dynamique

Être capable de mettre en place un système d'API

Être capable de connecter une application serveur à une base de données côté Back-end

Être capable de gérer des requêtes HTTP d'interaction côté Back-end

## 2 PHP c’est quoi ?

**PHP** (**PHP Hypertext Preprocessor**) est un langage de script libre conçu pour le développement d’applications web dynamiques.

Il s’intègre facilement dans du contenu html.

PHP est multiplateforme (Windows, Linux, Mac Os...).

Pour fonctionner PHP a besoin d’être installé sur un serveur web (Apache, NGINX, IIS sont les plus connus).

PHP est un langage qui s’exécute côté serveur et permet la génération de page web dynamique.

L’interpréteur PHP va alors générer une page web html.

De par sa polyvalence, PHP est également un langage de programmation à part entière qui peut être utilisé indépendamment d’un serveur web par exemple pour faire du scripting ou des applications graphiques (par exemple, via la librairie PHP-GTK : <http://gtk.php.net/>)

<https://www.php.net/manual/fr/intro-whatcando.php>

## 3 Spécificités du langage PHP

### PHP est un langage « interprété »

On peut distinguer 2 grands types de langages de programmation : les langages interprétés et les langages compilés.

* **Langages interprétés** :
  + L’interpréteur lit le code et exécute les commandes ligne par ligne.
  + Le principal avantage de ces langages est leur côté multiplateforme : dès lors qu’un interpréteur existe sur la plateforme, un même code pourra y être exécuté.
  + Toutefois, cela a un coût en matière de ressources, puisqu’à chaque exécution du programme, l’interpréteur devra traduire les commandes avant qu’elles puissent être exécutées par la machine.
  + Exemples : PHP, Java, Python, JavaScript
* **Langages compilés** :
  + Les instructions sont compilées et traduites en langage machine avant de pouvoir exécuter le programme.
  + Le programme devra donc être compilé sur chaque machine sur lequel on souhaite l’exécuter.
  + Le principal avantage est l’optimisation des ressources nécessaires pour exécuter le programme, que ça soit au niveau de la mémoire nécessaire que de la vitesse d’exécution.
  + Exemples : C, C++, Pascal

Afin d’améliorer les performances des langages interprétés, la plupart de ces langages proposent en réalité une « pré-compilation ».

* **Java** utilise une **machine virtuelle**. Le compilateur Java va en réalité compiler le code en **byte code** et la machine virtuelle sera ensuite chargée d’interpréter ce byte code en langage machine.
* L’interpréteur **Python** propose (sans l’imposer) une option de précompilation des scripts en byte code.
* C’est un peu différent pour **PHP**. Le script est compilé à la volée lors de chaque appel serveur et est ensuite exécuté. Afin d’améliorer les performances et limiter les ressources, des **PHP cache code** ont été développés, permettant de mettre en cache sur le serveur le code compilé.

### PHP est un langage « à typage faible » ou « non-typé »

Une autre grande différence des langages de programmation est le typage des variables, celui-ci pouvant être « fort » ou « faible ».

Un langage à typage fort vérifiera que le type de variable attendu est bien celui reçu, et renverra une erreur dans le cas contraire. De plus, les opérations sur des variables de types différents seront interdites.

A contrario, un langage faiblement typé sera beaucoup plus permissif, ce qui peut parfois entrainer des comportements surprenants.

* Exemples de langages à typage fort : C++, Java, Python
* Exemples de langages à typage faible : PHP, JavaScript, C

## 4 Evolution du PHP

PHP a été créé en 1994 par Rasmus Lerdorf pour la création de son site web personnel.

Son travail a été repris en 1997 par deux étudiants : Andi Gutmans et Zeev Suraski qui ont alors publié la version 3 de PHP.

Ils ont ensuite réécrit le moteur interne de PHP, aboutissant au moteur Zend Engine qui a servi de base à la version 4 de PHP. Cette version 4 est également celle qui a commencé à intégrer la programmation orientée objet au langage PHP.

La version 5 a été la version intégrant un modèle de programmation orientée objet complet.

A la suite de ça, de nombreuses évolutions et réécritures du moteur ont permis à PHP de gagner en performance et robustesse en s’inspirant des langages comme Java : support des tests unitaires, typage des méthodes et propriétés, standards de développement (PSR).

Voici une frise illustrant son évolution au fil des années : <https://www.jetbrains.com/fr-fr/lp/php-25/>

## 5 Environnement de travail :

Pour développer en PHP nous allons avoir besoin :

D’un serveur, WAMP (Windows) ou LAMP(Linux) suivant notre environnement de travail.

* Linux / Windows : Le système d’exploitation hébergeant le serveur
* Apache (serveur web pour héberger nos différents fichiers),
* MySQL (serveur de base de données, pour héberger nos bdd),
* PHP (interpréteur PHP),

Pour concevoir nos différents fichiers :

* Un éditeur de code (Visual studio code, PhpStorm, Notepad++, Bracket, Sublime Text, etc…),

Pour tester notre code :

* Un navigateur web pour afficher nos pages tester et contrôler le rendu. (Chrome, Mozilla Firefox, Edge, Safari etc…).

## 6 Syntaxe du langage :

Pour intégrer du code PHP nous écrivons nos scripts à l’intérieur de fichier avec l’extension **php**.

### Emplacement des fichiers, Exemple :

Dans le dossier **C:\wamp64\www\exemple\*index.php*** (exemple du chemin avec WAMP) du serveur apache (WAMP, XAMPP, Docker, LAMP, etc…)) nous allons créer un fichier **index.php**.

Par défaut, le serveur web renverra le contenu du fichier index.php (ou index.html) lorsque l’on visitera le site sans préciser de fichier en particulier.

### Balises :

Lorsque PHP traite un fichier, il cherche les balises d’ouverture et de fermeture lui indiquant le code qu’il devra interpréter.

Nos scripts PHP devront être rédigés entre les balises :

* **< ?php** …code PHP… **?> :** ce sont les balises normales.
* **< ?** …code PHP… **?>  :** ce sont les balises courtes. Toutefois ces dernières pouvant être désactivées, il est conseillé de n’utiliser que les balises normales.
* **< ?=** « du texte »**?>**  : ces balises courtes permettent d’afficher du texte. Elles sont l’équivalent de

**< ?php** echo « du texte » **?>**

### Exemple :

*La page sera accessible dans le navigateur web à l’adresse suivante :*

***localhost/exemple/index.php***

*NB : le fichier doit être exécuté et se trouver sur le serveur, si on ouvre simplement le fichier celui ne retournera rien.*

*Depuis l’exemple précédent nous devons avoir le fichier à l’intérieur du répertoire WWW de Wamp ou HTDOCS de Xamp et créer un sous dossier (dans le dossier à la racine de* ***www*** *ou* ***htdocs*** *en fonction du logiciel) exemple, enfin créer un fichier index.php dans celui-ci. On saisira dans le navigateur web l’adresse suivante (url)* ***localhost/exemple/index.php****, pour exécuter le fichier. L’interpréteur PHP du serveur va alors lire le fichier* ***.php*** *et exécuter le code contenu dans celui-ci.*

### Exemple de code d’une page :

<html lang="fr">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>ma première page php</title>

</head>

<body>

<h1>mon premier programme</h1>

<?php

//le script php se trouvera entre ces balises

?>

</body>

</html>

### Syntaxe générale :

* Les instructions sont séparées par des « ; ». Bien que ça ne soit pas obligatoire, il est conseillé d’effectuer un retour à la ligne après chaque instruction.

<?php

*//script php;*

?>

* Les blocs de code (classes, boucles, fonctions) sont délimités par des accolades : « { », « } ».

<?php

Class MaClasse

{

function maFonction()

{

*// Code de la fonction*

}

}

?>

* Indentation : Bien que non obligatoire, on indente nos blocs de code pour une meilleure lisibilité (4 espaces).

### Commenter des lignes de codes :

<?php

*//commentaire sur une ligne*

*/\**

*-------------------------------------*

*Commentaire sur plusieurs lignes*

*-------------------------------------*

*\*/*

*/\*\**

*\* -------------------------------------*

*\* PHPDoc*

*\* -------------------------------------*

*\*/*

?>

### Création de notre premier programme en php :

Nous allons créer un programme qui va afficher dans le navigateur internet.

***Hello World***

-Créer une page index.php dans votre éditeur de code et déposer là à l’intérieur de votre dossier **www/cours** du serveur apache (ou **htdocs/cours** si vous utilisez **xampp**).

-A l’intérieur de la page saisir le code ci-dessous :

<html lang="fr">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Ma première page PHP</title>

</head>

<body>

<h1>mon premier programme</h1>

<?php

*//programme Hello Word*

*//La commande echo permet d’afficher du contenu dans une page html.*

echo "Hello World";

?>

</body>

</html>

**Pourquoi Hello World ?:**

<https://deux.io/pourquoi-hello-world/>

## 7 Les variables :

### Une variable ça sert à quoi ?

*Les variables permettent de stocker des valeurs (Saisies, Résultat d’un sous-programme)*

*Elles vont pouvoir contenir des valeurs de types différents (texte, numérique…)*

*Une variable est une sorte de boite étiquetée avec un contenu.*

*Pour avoir accès à son contenu nous utiliserons son étiquette (son nom).*

### Les types de variables :

Bien que le typage en PHP soit un typage faible et dynamique (c’est-à-dire que le type est défini lorsque l’on donne une valeur à la variable), une variable en PHP possède un type de donnée.

* Le type « chaine de caractères » ou String en anglais
* Le type « nombre entier » ou Integer en anglais,
* Le type « nombre décimal » ou Float en anglais,
* Le type « booléen » ou Boolean en anglais,
* Le type « tableau » ou Array en anglais,
* Le type « objet » ou Object en anglais,
* Le type « NULL » qui se dit également NULL en anglais.
* Le type « ressource » ou Resource en anglais faisant référence à une ressource externe (ex : stream FTP, PDF document, …)

### Nommer une variable :

En PHP, les variables sont représentées par un signe « $ » suivi du nom de la variable.

De plus, le nom d’une variable en PHP doit répondre à certaines contraintes.

* Le nom d’une variable doit impérativement commencer par une lettre ou le symbole underscore « \_ »
* Le nom d’une variable ne peut être composé que de lettres, chiffres et de symboles underscore.
* La casse des caractères est prise en compte. Ainsi, les variables $variable et $varIABLE sont différentes.
* Par convention, une variable commencera par une lettre en minuscule, et sera écrite en camelCase. De plus, on essayera toujours de donner un nom compréhensible à la variable pour faciliter la compréhension du code.

### Déclaration d’une variable :

En PHP une variable s’écrira comme ci-dessous :

*$nomVariable = valeur;*

Le symbole dollars **$** désignera une variable au moment de sa création et quand on l’utilisera.

Le type de la variable est défini lorsque l’on lui attribue une valeur.

Exemple d’utilisation d’une variable :

<?php

$variable = 10;

$total = $variable + 10; *//total vaut 20 (10 de variable + 10 en numérique).*

?>

### Exemple :

<?php

*//Définition d'une variable de type int ayant pour valeur 0*

$varInt = 0;

*// Définition une variable de type float (nombre à virgule flottante)*

$varFloat = 3.4;

*// Définition d'une variable de type bool (booléen)*

$varBool1 = true;

$varBool2 = false;

*/\**

*Définition d'une variable de type string*

*On peut utiliser le symbole simple quote ('') ou double quote ("") pour définir une chaine de caractères*

*\*/*

$varString1 = 'Une chaine de caractères';

$varString2 = "Une autre chaine de caractères";

*// Définition d'une variable de type array*

$varArray1 = array();

$varArray2 = [];

?>

### Afficher le contenu d’une variable :

Pour afficher le **contenu** d’une variable nous utiliseront le code ci-dessous :

<?php

*//initialisation d’une variable*

$nbr =2 ;

*//la fonction php echo permet d’afficher le contenu de la variable nbr*

echo $nbr ;

?>

### Afficher le type d’une variable :

Pour afficher le **type** d’une variable nous utiliserons la fonction « gettype($maVariable) :

<?php

*//initialisation d’une variable*

$nbr =2;

*//affichage dans la page web avec la fonction echo*

echo $nbr ;

*//utilisation de la fonction gettype pour afficher le type de la variable*

echo gettype($nbr); *//Ceci affichera "integer"*

*// Debugger une variable.*

var\_dump($nbr); *// Affichera "int(2)"*

?>

### Exercice variables :

**Exercice 1** :

-Créer une variable de type int avec pour valeur 5,

-Afficher le contenu de la variable (utilisation de la fonction php **echo**),

-Afficher son type (utilisation de la fonction php **gettype**),

-Créer une variable de type String avec pour valeur votre prénom,

-Afficher le contenu de la variable (utilisation de la fonction php **echo**),

-Créer une variable de type booléen avec pour valeur false,

-Afficher son type (utilisation de la fonction php **gettype**).

## 8 Les constantes :

Une constante est un identifiant permettant de stocker une donnée simple.

Une fois définie, cette donnée ne peut plus être modifiée.

Par convention, le nom d’une constante est toujours écrit en majuscules.

Il existe deux façons de déclarer une constante :

<? php

define('MA\_CONSTANTE\_1', 'Hello');

const MA\_CONSTANTE\_2 = 'World!';

echo MA\_CONSTANTE\_1 . ' ' . MA\_CONSTANTE\_2;

*//affichera "Hello World!"*

?>

## 9 Les opérateurs :

### Les opérateurs arithmétiques :

Pour effectuer des opérations mathématiques sur des types numériques (int, long, float, etc…)

On utilise les opérateurs mathématiques suivants :

Addition :

***$a + $b***

Soustraction :

***$a - $b***

Multiplication :

***$a \* $b***

Division :

***$a / $b***

Modulo :

***$a % $b*** (reste de la division de **$a** divisé par **$b**)

Exponentielle :

***$a \*\* $b*** (Résultat de l'élévation de **$a** à la puissance **$b**)

### Exercices Opérateurs arithmétiques :

**Exercice 1 :**

-Créer 2 variables $a et $b qui ont pour valeur 12 et 10,

-Stocker le résultat de l’addition de $a et $b dans une variable $total,

-Afficher le résultat (utilisez la fonction **echo**)

**Exercice 2 :**

-Créer 3 variables $a, $b et $c qui ont pour valeur $a =5, $b =3 et $c = $a+$b,

-Afficher la valeur de chaque variable (utilisez la fonction **echo**).,

-passer la valeur de $a à 2,

-Afficher la valeur de $a,

-passer la valeur de $c à $b - $a,

-Afficher la valeur de chaque variable (utilisez la fonction **echo**).

**Exercice 3 :**

-Créer 2 variables $a et $b qui ont pour valeur 15 et 23,

-Afficher la valeur de chaque variable (utilisez la fonction **echo**).,

-Intervertissez les valeurs de $a et $b,

-Afficher la valeur de $a et $b (utilisez la fonction **echo**).

**Exercice 4 :**

-Ecrire un programme qui prend le prix HT d’un article (de type float), le nombre d’articles (de type integer) et le taux de TVA (de type float), et qui fournit le prix total TTC (de type float) correspondant.

-Afficher le prix HT, le nbr d’articles et le taux de TVA (utilisez la fonction **echo**),

-Afficher le résultat (utilisez la fonction **echo**).

### Les opérateurs d’incrémentation :

Pré-incrémentation :

***++$a*** *(incrémente $a de 1 ($a = $a + 1) puis retourne la valeur de $a)*

Post-incrémentation :

***$a++*** *(retourne la valeur de $a puis incrémente de 1 ($a = $a+1))*

Pré-décrémentation :

***--$a*** *(décrémente $a de 1 ($a = $a – 1) puis retourne la valeur de $a)*

Post-décrémentation :

***$a--*** *(retourne la valeur de $a puis décrémente de 1 ($a = $a – 1))*

## 10 Concaténation :

En PHP nous pouvons concaténer des valeurs entres elles. C’est à dire ajouter des chaines de caractères, des nombres, valeurs de variables au sein d’une même chaine de caractères.

### Exemple :

Ecrire le nom d’une variable dans une page web :

<?php

$nom = 'test';

*//on va utiliser le symbole \devant le nom de la variable, ce caractère permet //d’annuler l’interprétation du caractère qui va suivre, dans ce cas il va afficher le nom de la variable et non son contenu.*

echo 'affichage de la variable s’appelant \$test' ;

?>

**Ecrire la valeur d’une variable dans une page web :**

<?php

$nom = 'test';

echo "affichage du contenu de la variable \$nom : $nom ";

?>

**Concaténer des chiffres, des chaines de caractères et les afficher dans une page web :**

<?php

$str = 'une chaine de caractères';

$strLength = strlen($str);

echo "<br>La chaine de caractères '$str' contient $strLength caractères.";

?>

**Concaténer des variables dans des chaines de caractères :**

<?php

$var = 'du texte';

*//version avec encadrement ""*

$concat1 = "ma chaine $var";

*//version avec encadrement ''*

$concat2 = 'ma chaine ' . $var;

?>

### Exercices :

**Exercice 1 :**

-Créer une variable $a qui a pour valeur ***« bonjour »,***

-Afficher le **nom de la variable et sa valeur**.

**Exercice 2 :**

-Créer 1 variable $a qui a pour valeur **« bon »**,

-Créer 1 variable $b qui a pour valeur **« jour »**,

-Créer 1 variable $c qui a pour valeur **10**,

-Concaténer **$a, $b et $c + 1**,

-Afficher le **résultat** de la concaténation.

**Exercice 3 :**

-Créer une variable $a qui a pour valeur **bonjour**,

-Afficher un paragraphe (**balise html**) et à l’intérieur la phrase suivante :

***l’adrar***

-Ajouter la variable $a avant la phase dans le paragraphe,

-Cela doit donner :

***<p>bonjour l’adrar</p>***

### Les opérateurs d’affectation :

***$a = 5*** *(affecte la valeur 5 à la variable $a)*

***$a += 3*** *(affecte la valeur $a+3 à la variable $a : équivalent à $a = $a + 3)*

***$b =* 'Bonjour ';** (affecte la valeur 'Bonjour ' à la variable $b)

**$b .= 'le monde.';** (Concatène ‘le monde.’ Et l’affecte à la variable $b : équivalent à $b = $b . 'le monde.')

## 11 Les Fonctions :

Les fonctions permettent de générer du code qui va être exécuté chaque fois que l’on va appeler la fonction par son nom. Les fonctions sont créées pour exécuter une tâche précise et permettre de mutualiser le code afin d’améliorer la qualité et maintenabilité du code.

Nous avons déjà utilisé la fonction gettype() qui est une fonction interne à PHP.

Nous pouvons trouver la liste des fonctions internes sur la documentation officielle de PHP (<https://www.php.net/manual/fr/funcref.php>) ou sur le site w3school (<https://www.w3schools.com/php/default.asp>).

Une fonction peut prendre des paramètres en entrée et retourner un résultat.

### Création d’une fonction :

Pour créer une fonction en php nous allons utiliser la syntaxe suivante :

<?php

function nomDeLaFonction()

{

}

?>

Le nom d’une fonction devra répondre au mêmes critères de nommage que les variables (commencer par une lettre ou le symbole underscore et ne comporter que lettres, chiffres ou symboles underscore. Par convention le nom d’une fonction sera écrit en camelCase.

### Appel d’une fonction :

Pour appeler une fonction on va saisir le nom de la fonction suivi de ***()***

### Exemple :

<?php

function maFonction()

{

}

maFonction();

?>

### Création d’une fonction avec des paramètres :

Une fonction avec des paramètres va nous permettre d’exécuter le code de celle-ci et d’adapter son traitement en fonction de ce que l’on va passer en paramètre. Le mot clé **return** permet de retourner (int, string, boolean etc..).

### Exemple :

<? php

maFonction(10,5);

function maFonction($a,$b)

{

$result= $a+$b ;

return $result ;

}

?>

### Exercices :

**Exercice 1** :

-Créer une fonction qui soustrait à **$a** la variable **$b** (*2 paramètres en entrée*),

-la fonction doit retourner le résultat (**return**).

**Exercice 2** :

-Créer une fonction qui prend en entrée un nombre à virgule (**float**),

- la fonction doit retourner l’arrondi (**return**) du nombre en entrée (utiliser une fonction interne au langage).

**Exercice 3** :

-Créer une fonction qui prend en entrée **3 valeurs** et retourne **somme** des 3 valeurs.

**Exercice 4** :

-Créer une fonction qui prend en entrée **3 valeurs** et retourne la **valeur moyenne** des 3 valeurs (saisies en paramètre).

### Typage des paramètres :

Le version 7 de PHP introduit le typage des paramètres d’entrée et de sortie des fonctions.

Cela permet un meilleur contrôle des entrées, ce qui est impératif pour du code de qualité et devra donc être utilisé autant que possible. En effet, si le paramètre reçu par la fonction n’est pas du bon type, le serveur lancera une erreur.

Le type de sortie d’une fonction en retournant aucune valeur est « void ».

Si un paramètre peut être de plusieurs types, on peut utiliser le symbole pipe « | » pour indiquer plusieurs paramètres (exemple : int|float).

Dans l’exemple ci-dessous, on a défini que les paramètres $a et $b étaient de type « float », et que le paramètre de sortie était également de type « float ».

<? php

maFonction(10,5);

function maFonction(float $a,float $b):float

{

return $a+$b ;

}

?>

### Exercices :

**Exercice :**

Modifier les fonctions précédemment écrites en typant les paramètres d’entrée et de sortie.

## 12 Les structures conditionnelles :

Les structures conditionnelles, ou conditions vont nous permettre de conditionner l’exécution de certains blocs de codes si certaines conditions sont remplies (ou si elles ne le sont pas.

Pour cela nous allons utiliser les instructions ***if*** (***si***), ***elseif*** (**sinon si**) ***else*** (***sinon***).

### Instruction if :

Instruction fondamentale dans un programme, elle permet l’exécution d’un bloc de code si une condition est remplie.

<?php

if ($heure > 6 && $heure < 21) {

echo 'Il fait jour';

}

?>

### Instruction else :

Permet d’exécuter un bloc de code dans le cas où la condition du « if » ne soit pas remplie.

<?php

if ($heure > 6 && $heure < 21) {

echo 'Il fait jour.';

} else {

echo 'Il fait nuit.';

}

?>

### Instruction elseif :

elseif est une combinaison de if et else. Si la condition précédente n’était pas remplie, l’interpréteur va vérifier si celle-ci l’est, et exécuter le bloc de code correspondant si c’est le cas.

<?php

*//$i est un nombre*

if ($i < 0) {

echo '$i est inférieur à 0';

} elseif ($i > 50) {

echo '$i est supérieur à 50';

} else {

echo '$i est compris entre 0 et 50 inclus';

}

?>

### Opérateur ternaire :

Un operateur permet de réaliser certaines conditions if…else de manière abrégée.

Par exemple, voici deux façons de vérifier si une personne est majeure, l’une avec une structure if…else classique, l’autre avec un opérateur ternaire.

<?php

function isAdult(int $age): bool

{

if ($age >= 18) {

return true;

} else {

return false;

}

}

function isAdultV2(int $age) : bool

{

return ($age >= 18) ? true : false;

}

function isAdult200IQ (int $age): bool

{

return $age >= 18;

}

?>

### Instruction switch :

L’instruction switch équivaut une succession d’instructions if. Elle permet de comparer une variable à des valeurs, et exécuter un (ou plusieurs) bloc(s) de code en conséquence.

Attention, le switch effectue une comparaison large (il ne compare pas le type de donnée. Equivalent à ==).

<?php

*//$i est un nombre*

switch($i) {

case 1:

echo '$i vaut 1';

break; *//l'instruction break permet de sortir de la structure de contrôle et donc de ne pas effectuer les instructions suivantes.*

case 2:

echo '$i vaut 2';

break;

case 3:

echo '$i vaut 3';

break;

default: *//Bloc exécuté dans tous les cas, sauf si on a eu un break avant*

echo '$i ne vaut ni 1, ni 2, ni 3';

}

?>

### Opérateurs de comparaison :

Nous allons utiliser dans nos conditions les opérateurs de comparaisons.

|  |  |  |
| --- | --- | --- |
| **Exemple** | **Nom** | **Résultat** |
| $a == $b | Egal | true si $a est égal à $b après le transtypage (comparaison large). |
| $a === $b | Identique | true si $a est égal à $b et qu'ils sont de même type. |
| $a != $b | Différent | true si $a est différent de $b après le transtypage (comparaison large). |
| $a <> $b | Différent | true si $a est différent de $b après le transtypage (comparaison large). |
| $a !== $b | Différent | true si $a est différent de $b ou bien s'ils ne sont pas du même type. |
| $a < $b | Strictement inférieur | true si $a est strictement inférieur à $b. |
| $a > $b | Strictement supérieur | true si $a est strictement supérieur à $b. |
| $a <= $b | Inférieur ou égal | true si $a est inférieur ou égal à $b. |
| $a >= $b | Supérieur ou égal | true si $a est supérieur ou égal à $b. |
| $a <=> $b | Combiné / Spaceship operator | Retourne -1 si $a est < à $b, 1 si $a est > à $b et 0 si $a == $b, après transtypage (comparaison large). |

### Opérateurs logiques :

Voici les différents opérateurs logiques qui peuvent être utilisés :

**Exemple Nom Résultat**

$a && $b And (Et) true si $a ET $b sont true.

$a and $b And (Et) true si $a ET $b valent true.

$a || $b Or (Ou) true si $a OU $b est true.

$a or $b Or (Ou) true si $a OU $b valent true.

$a xor $b XOR true si $a OU $b est true, mais pas les deux en même temps.

!$a Not (Non) true si $a n'est pas true.

### Exemple :

<?php

$a = 6;

if($a <= 3 && $a > 0) {

*//teste si $a est plus petit que 3 et est supérieur à 0*

echo "la valeur de la variable \$a est plus petite que 3";

} elseif($a >= 3 && $a < 5) {

*//teste si $a est plus grand ou égal et 3 et inférieur à 5*

echo "la valeur de la variable \$a est comprise entre 3 et 5";

} else { *//sinon*

echo "la valeur de la variable \$a est supérieur à 5";

}

?>

### Exercices :

**Exercice 1 :**

- Créer une fonction qui teste si un nombre est **positif** ou **négatif**

**Exercice 2 :**

- Créer une fonction qui prend en entrée **3 valeurs** et retourne le nombre le plus **grand**

**Exercice 3 :**

- Créer une fonction qui prend en entrée **3 valeurs** et retourne le nombre le plus **petit**

**Exercice 4 :**

- Créer une fonction calculePrixFinal qui prend en entrée un paramètre $prix de type float et retournera le prix final.

- Si le prix est > à 2000€, la ristourne sera de 20%

- Si le prix est > à 1000€, la ristourne sera de 10%

- Sinon, la ristourne sera de 0

**Exercice 5 :**

- Créer une fonction qui prend en entrée **1 année (entier)** et qui affiche « l’année x est une année bissextile » si l’année est bissextile ou « l’année x n’est pas une année bissextile » si ce n’est pas une année bissextile

- Pour rappel une année bissextile est définie de la façon suivante (<https://fr.wikipedia.org/wiki/Ann%C3%A9e_bissextile>)  :

- Les années sont en général bissextiles si elles sont multiples de quatre

- elles ne sont pas bissextiles si elles sont multiples de cent à l'exception des années multiples de quatre cents qui le sont.

## 13 Les boucles :

Comme dans tous les langages de programmation, PHP gère les structures de boucle.

La boucle est un élément de base d'un langage de programmation.

Les boucles permettent de répéter plusieurs fois une ou plusieurs instructions tant qu'une condition est vérifiée ou bien jusqu'à ce qu'elle soit vérifiée.

En PHP, il existe 4 types de boucles différentes :

* La boucle « for »
* La boucle « while »
* La boucle « do…while »
* La boucle « foreach »

Pour écrire une boucle (**for**):

### Boucle for :

La boucle « for » prend 3 paramètres en entrée : un paramètre initial, une condition d’arrêt, et une opération à effectuer en fin d’itération.

### Exemple de boucle for :

Tant que $i est inférieur à 10 on répète l’opération :

<?php

*// for (valeur initiale ; condition ; opération)*

for ($i=0; $i < 10; $i++){

echo 'Ceci est une boucle for en PHP';

echo '<br>';

echo "\$i = $i";

echo '<br>';

}

?>

### Boucle while :

L’instruction « while » est traduite en « tant que ». Tant que la condition est remplie, la boucle continue de s’exécuter.

### Exemple de boucle while :

Tant que $i est inférieur à 10 on répète l’opération :

<?php

$i = 0; *//variable compteur*

while ($i < 10) *//boucle while*

{

echo 'Ceci est une boucle while en PHP';

echo '<br>';

echo "\$i = $i";

echo '<br>';

*//à chaque tour j’incrémente $i (+1)*

$i++;

}

?>

### Boucle do…while :

L’instruction « do…while » pourrait être traduite par « faire…tant que ». Elle est dérivée de la boucle « while ». La différence est que la boucle while vérifie la condition avant que l’on rentre dans la boucle, alors que dans le cas de la boucle do…while, on entrera toujours dans le bloc « do » de la boucle, et on en sortira si la condition n’est plus remplie.

Le choix d’utiliser « while » plutôt que « do…while » dépend de la situation.

### Exemple de boucle do…while :

Tant que $i est inférieur à 10 on répète l’opération :

<?php

$i = 0; *//variable compteur*

do {

echo 'Ceci est une boucle do...while en PHP';

echo '<br>';

echo "\$i = $i";

echo '<br>';

} while (++$i < 10);

?>

### Boucle foreach :

A la différence des autres boucles, ce type de boucle n’est pas présent dans la majorité des langages de programmation, et est pourtant d’une efficacité redoutable.

Cette boucle permet de parcourir tous les éléments d’un objet « iterable » (tableau, chaine de caractères, collection d’objets.

### Exemple de boucle foreach :

<?php

foreach ($tableau as $valeur)

{

echo $valeur.'<br />';

}

?>

### Exercices sur les boucles :

**Exercice 1 :**

- Choisir un nombre compris entre 0 et 999

- A l’aide d’une boucle while, effectuez des tirages aléatoires (utilisation de la fonction PHP « rand() » jusqu’à trouver le bon nombre.

- Affichez le nombre d’itérations nécessaires pour trouver le nombre

**Exercice 2 :**

- Choisir un nombre de lignes

- Choisir un nombre de colonnes

- A l’aide de boucles « for », obtenez le résultat suivant :

0000000000

1111111111

2222222222

3333333333

4444444444

5555555555

6666666666

**Exercice 3 :**

- Ecrivez des boucles qui affichent ce qui est demandé

- Le nombre de colonne à afficher dépend du n° de ligne, à la ligne i, il faut afficher i colonnes.

- Le résultat attendu est celui-ci :

1

22

333

4444

55555

666666

7777777