Exercices PHP

Pour bien débuter avec PHP

CONTENU

CONSIGNES

Créez un répertoire « php_intro_exercices » dédié à ces exercices. Chaque exercice de ce document doit être réalisé dans un fichier à l'intérieur de ce répertoire. Le nom du fichier est indiqué pour chaque exercice.

Règles :

- Si précisés, respectez scrupuleusement les noms des fichiers, fonctions, variables, classes…
- Le code doit être commenté en Français.
- Le nom des variables, fonctions, classes doivent être en Anglais.

SERIE 1 : INTRODUCTION

Reproduisez ces 2 exemples avant de commencer.

Dans un fichier « 01-hello.php », créer une fonction « helloWorld() » qui **affiche** « Hello World ! ».

```php
<?php
/**
* Affiche "Hello World !"
*/
function helloWorld() : void
{
   echo "Hello World !";
}

// test de la fonction (la fonction affiche directement le résultat)
helloWorld();
```

Dans le fichier « 01-hello.php », créer une fonction « hello() » qui accepte un argument de type string et qui **retourne** « Hello » suivi de la valeur de l'argument.

```php
<?php
/**
* Retourne "Hello $name !"
* @param string $name le nom à afficher
*/
function hello(string $name) : string
{
   return "Hello $name";
}

// test de la fonction (la fonction n'affiche rien. "echo" affichera la valeur retournée par la fonction)
echo hello("Mike");
```

EXERCICE 1.A

Améliorez la fonction Hello() en prenant en compte cette condition :

- Si $name est vide, retourner "Nobody".

SERIE 2 : NOMBRES

Les fonctions de la série 2 doivent être implémentées dans le fichier « 02-numbers.php ».

EXERCICE 2.A

Créer une fonction « getSum() » qui accepte deux arguments de type int. Elle devra **retourner** la somme des deux valeurs.

Exemple :
getSum(5, 3); // retourne 8

EXERCICE 2.B

Créer une fonction « getSub() » qui accepte deux arguments de type int. Elle devra **retourner** la soustraction des deux valeurs.

Exemples :
getSub (5, 3); // retourne 2
getSub (3, 5); // retourne -2

EXERCICE 2.C

Créer une fonction getMulti() qui accepte deux arguments de type float. Elle devra **retourner** la multiplication des deux valeurs. Limitez le résultat à 2 décimales.

Exemples :
getMulti (5.6, 3); // retourne 16.8
getMulti (5.6, -3.7); // retourne -20.72

EXERCICE 2.D

Créer une fonction getDiv() qui accepte deux arguments de type int. Elle devra **retourner** la division des deux valeurs. Limitez le résultat à 2 décimales.

Rappel : une division par zéro est impossible. Dans ce cas, **retourner** la valeur « 0 ».

Exemples :
getDiv (20, 3); // retourne 8.33
getDiv (20, 0); // retourne 0

SERIE 3 : CHAINES DE CARACTERES

Les fonctions de la série 3 doivent être implémentées dans le fichier « 03-strings.php ».

EXERCICES 3.A

Créer une fonction « getMC2() ».

Cette fonction doit **retourner** Le nom de l'inventeur de la formule « $E = MC^2$ ».

EXERCICE 3.B

Créer une fonction « getUserName() » qui accepte deux arguments (prénom et nom) de type string.
Cette fonction doit retourner la concaténation des deux valeurs.

Exemple :
getUserName ('mickaël', 'devoldère'); // retourne « mickaëldevoldère »

EXERCICE 3.C

Créer une fonction « getFullName() » acceptant deux arguments (nom et prénom) de type string.
Cette fonction doit **retourner** la concaténation des deux valeurs avec un espace entre les 2, le prénom en minuscule et le nom en MAJUSCULE.

Exemple :
getFullName ('devoldère', 'mickaël'); // retourne « mickaël DEVOLDÈRE »

EXERCICE 3.D

Créer une fonction « askUser() » acceptant deux arguments (nom et prénom) de type string.
Cette fonction doit **retourner** une chaîne de caractères sous la forme :

> « Bonjour prénom, nom. Connaissez-vous Einstein ? »

Cette fonction doit **réutiliser** les fonctions

- « getFullName() » pour obtenir la concaténation des deux valeurs.
- « getMC2() » pour obtenir le nom de l'inventeur de la formule $E = MC^2$.

Exemple :
askUser ('devoldère', 'mickaël'); // retourne « Bonjour mickaël DEVOLDÈRE, connaissez-vous Einstein ? »

SERIE 4 : CONDITIONS

Les fonctions de la série 4 doivent être implémentées dans le fichier « 04-conditions.php ».

EXERCICE 4.A

Créer une fonction « isMajor() » acceptant un argument de type int. Elle devra **retourner** un booléen.
Si l'âge est supérieur ou égal à 18, elle doit **retourner** true. Sinon elle doit **retourner** false.

Exemples :

isMajor (12); // retourne « false »

isMajor (18); // retourne « true »

isMajor (42); // retourne « true »

EXERCICE 4.B

Créer une fonction « getRetired() » acceptant un argument de type int. Elle devra **retourner** un string.
Cette fonction permet de calculer le nombre d'années restant avant la retraite ou le nombre d'années depuis la retraite.
Pour cet exercice, l'âge de la retraite est fixé à 60 ans.

Exemples :

getRetired (12); // retourne « il vous reste 48 ans avant la retraite »

getRetired (60); // retourne « Vous êtes à la retraite cette année »

getRetired (72); // retourne « Vous êtes à la retraite depuis 12 ans »

getRetired (-2); // retourne « Vous n'êtes pas encore né »

EXERCICE 4.C

Créer une fonction « getMax() » acceptant 3 arguments de type float.
Cette fonction doit **retourner** la valeur du plus grand des 3 nombres. Limitez le résultat à 3 décimales
Si au moins 2 des valeurs fournies sont égales la fonction **retourne** 0.

EXERCICE 4.D

Créez une fonction « capitalCity() » qui accepte un argument de type string (le pays dont on cherche la capitale).
Elle devra **retourner** le nom de la capitale des pays suivants :

France	=>	Paris
Allemagne	=>	Berlin
Italie	=>	Rome
Maroc	=>	Rabat
Espagne	=>	Madrid
Portugal	=>	Lisbonne
Angleterre	=>	Londres

Si le pays ne fait pas partie de la liste ci-dessus, la fonction retourne la valeur « Capitale inconnue ».

Note : Utilisez la structure SWITCH pour faire cet exercice.

SERIE 5 : DATE ET HEURE

Les fonctions de la série 5 doivent être implémentées dans le fichier « 05-datetime.php ».

EXERCICES 5.A

Créer une fonction « getToday() ». Cette fonction doit **afficher** et **retourner** la date du jour au format d/m/Y sous forme de chaine de caractères (exemple : 21/10/2020).

Aide et exemples :

https://www.php.net/manual/fr/function.date.php

https://www.php.net/manual/fr/datetime.format.php#refsect1-datetime.format-examples

EXERCICE 5.B

Créer une fonction « getTimeLeft() » acceptant un argument de type string et qui retourne une chaine de caractère.
La valeur de l'argument représente une date au format **Y-m-d** (ex: 2020-11-23).

La fonction doit vérifier si la date fournie est valide (bon format, date cohérente).
Si la date est ultérieure à la date du jour, la fonction retourne la différence en années/mois/jours.
Si la date est égale à la date du jour, la fonction retourne « Aujourd'hui ».
Si la date est antérieur à la date du jour, la fonction retourne « Évènement passé ».

Exemples :

Pour les exemples suivants: *DateDuJour = 2020-01-30 (30 Janvier 2020)*

getTimeLeft ("2019-09-29"); // retourne « Évènement passé »

getTimeLeft ("2020-01-30"); // retourne « Aujourd'hui »

getTimeLeft ("2020-02-15"); // retourne « Dans 16 jours »

getTimeLeft ("2020-05-16"); // retourne « Dans 4 mois et 17 jours »

getTimeLeft ("2021-05-30"); // retourne « Dans 1 an et 4 mois »

getTimeLeft ("2022-10-17"); // retourne « Dans 2 ans et 9 mois »

SERIE 6 : TABLEAUX

Les fonctions de la série 6 doivent être implémentées dans le fichier « 06-arrays.php ».

Pour les exemples et vos tests, le tableau ci-dessous sera utilisé :

- *$names = ['Joe', 'Jack', 'Léa', 'Zoé', 'Néo'];*

EXERCICE 6.A

Créer une fonction « firstItem() » qui accepte un argument de type array. Elle devra **retourner** le premier élément du tableau. Si le tableau est vide, la fonction **retourne** null.

Exemple :

```
firstItem ($names); // retourne « Joe »
```

EXERCICE 6.B

Créer une fonction « lastItem() » acceptant un argument de type array. Elle devra **retourner** le dernier élément du tableau. Si le tableau est vide, il faudra retourner null.

Exemple :

```
lastItem ($names); // retourne « Néo »
```

EXERCICE 6.C

Créer une fonction « sortItems() » acceptant un argument de type array.
Cette fonction **retourne** le tableau trié par ordre décroissant. Si le tableau est vide, il faudra retourner un tableau vide.

Exemple :

```
sortItems ($names); // retourne « ['Zoé', 'Néo', 'Léa', 'Joe', 'Jack'] »
```

EXERCICE 6.D

Créer une fonction « stringItems() » acceptant un argument de type array.
Cette fonction **retourne** une chaine de caractère contenant tous les éléments du tableau triés en ordre croissant et séparés par une virgule et un espace. Si le tableau est vide, il faudra retourner la valeur « Nothing to display ».

Exemple :

```
sortItems ($names); // retourne « Jack, Joe, Léa, Néo, Zoé »
```

SERIE 7 : CONTROLES DE SAISIE

Les fonctions de la série 7 doivent être implémentées dans le fichier « 07-inputs.php ».

EXERCICE 7.A

Créez une fonction « stringLength() » qui accepte un argument de type string et **retourne** un booléen qui vaut true si la chaine de caractères contient au moins 9 caractères et false si elle contient moins de 9 caractères.

Exemples :

stringLength ("motDePasse"); // retourne « true »

stringLength ("azer"); // retourne « false »

EXERCICE 7.B

Créez une fonction « passwordCheck() » acceptant un argument de type string.
Cette fonction **retourne** un booléen qui vaut true si le mot de passe respecte les règles suivantes :

- Contient au moins 9 caractères
- Contient au moins 1 chiffre
- Contient au moins une majuscule et une minuscule
- Contient au moins 1 caractère non alphanumérique

Pour cet exercice, vous devez réutiliser la fonction « stringLength () » créée dans l'exercice précédent.

EXERCICE 7.C

Vous devez créer une fonction pour permettre l'identification d'un utilisateur.
Pour cet exercice, les mots de passe ne sont pas chiffrés.

Soit le tableau d'utilisateurs suivant (récupéré depuis une base de données) :

```
$users = [
   'joe' => 'Azer1234!',
   'jack' => 'Azer-4321',
   'admin' => '1234_Azer',
];
```

Créez une fonction « userLogin() » acceptant 3 arguments :

1) le nom d'utilisateur (string).
2) le mot de passe à tester (string).
3) le tableau d'utilisateurs où effectuer la recherche (array).

Cette fonction **retourne** un booléen qui vaut true si l'utilisateur a été trouvé et que les mots de passe correspondent.
La fonction retourne false dans le cas contraire (utilisateur non trouvé OU mot de passe invalide).

Pour cet exercice, vous devez réutiliser les fonctions créées dans les exercices précédents.

SERIE 8 : HTML

Les fonctions de la série 7 doivent être implémentées dans le fichier « 08-html.php ».

EXERCICE 8.A

L'objectif de cet exercice est de générer une liste HTML (ul > li) à partir d'un tableau de données.

Créez une fonction « htmlList() » qui accepte deux arguments :

1) Un string représentant le nom de la liste.
2) Un array représentant les éléments de cette liste.

La fonction **retourne** une liste HTML (ul > li).

- Chaque élément de cette liste est extrait du tableau passé en paramètre.
- Les éléments sont affichés par ordre alphabétique.
- Si le tableau est vide, la fonction retourne « Aucun résultat » à la place de la liste HTML.

Exemples :

```
$names = ['Joe', 'Jack', 'Léa', 'Zoé', 'Néo' ];

htmlList ("Liste des personnes", $names );

/*

Résultat :

<h3>Liste des personnes</h3>
<ul>
  <li>Jack</li>
  <li>Joe</li>
  <li>Léa</li>
  <li>Néo</li>
  <li>Zoé</li>
</ul>

*/
```

```
$names = []; // le tableau est vide

htmlList ("Liste des personnes", $names );

/*

Résultat :

<h3>Liste des personnes</h3>
<p>Aucun résultat</p>

*/
```

--- FIN DU DOCUMENT ---