Université de Paris **Internet et outils**
L1 **Année 2022-2023**

TP n° 4

Fonctions et tableaux en PHP

En plus du cours, on pourra consulter `https://www.php.net/manual/fr/` (en français). Avoir recours à la documentation de référence doit être le premier réflexe en cas d'interrogation.

Vous trouverez notamment sur `https://www.php.net/manual/fr/language.types.array.php` un manuel d'utilisation des tableaux et sur `https://www.php.net/manual/fr/ref.array.php` une liste de fonctions sur les tableaux.

1 Fonctions et boucles

Exercice 1 — *Fonctions simples*

1. Écrire une fonction récursive `factorielle` prenant en argument un nombre et renvoyant la factorielle de ce nombre. Par exemple `factorielle(5)` renverra 120.
2. Écrire une fonction `lien` prenant en argument une chaîne de caractères contenant une adresse web et affichant un lien HTML vers cette adresse web et l'utilisant aussi comme texte du lien. Par exemple, le code PHP suivant

```php
lien("https://moodle.u-paris.fr/course/view.php?id=1622");
```

   devra produire le code HTML suivant

```html
<a href="https://moodle.u-paris.fr/course/view.php?id=1622">
    https://moodle.u-paris.fr/course/view.php?id=1622
</a>
```

3. Écrivez une fonction `add` prenant deux arguments `$x` et `$y` ayant le comportement suivant : `add(1, 2)` affiche `1 + 2 = 3`. Testez la fonction avec des entiers, des chaînes et un mélange des deux.
4. Réécrivez la fonction pour qu'au lieu d'afficher le résultat elle retourne une chaîne de caractères que vous afficherez par la suite. En général, c'est cette façon de faire qui est préférée car elle sépare la logique (en l'occurrence, calculer une addition) de son affichage.

Exercice 2 — *Boucles*

1. Écrire une fonction `ligne` prenant en argument un entier `$n` et un caractère `$c`, et renvoyant une chaîne de caractère composée de `$n` fois le symbole `$c`. Vous pourrez pour cela utiliser les boucles sur les entiers, dont la syntaxe est identique à celle de Java. Par exemple, l'exécution du programme suivant affiche `0123456789` :

```php
<?php
for ($i=0; $i<10; $i++)
{
        echo $i;
}
?>
```

2. En utilisant la fonction précédente, écrivez maintenant une fonction qui prend en entrée un entier `$n`, et affiche une pyramide de signes $*$ de hauteur `$n`. Par exemple, pour $n = 3$, vous afficherez

```
*
**
***
```

3. Modifier la fonction précédente, de sorte que la pyramide ne soit construite que si l'entier donné en paramètre est inférieur à 10. Elle affichera un message d'erreur dans le cas contraire.
4. En utilisant deux boucles, écrivez une fonction qui prend un entier n inférieur à 10 et affiche pour $n = 3$ par exemple :

```
*
**
***
**
*
```

5. Réécrivez cette même fonction mais en utilisant `foreach` à la place de `for` cette fois-ci. `foreach` est à préférer systématiquement à `for` dans la mesure du possible car, sémantiquement, elle rend mieux compte de la volonté du programmeur qui est souvent d'itérer sur les éléments d'un domaine (par exemple un tableau) qui n'est pas forcément une séquence d'entiers. Elle permet ainsi d'éviter nombre d'erreurs comme l'accès invalide à des éléments d'un tableau (par exemple accéder au 5e élément d'un tableau à 3 éléments).

   Vous pourrez utiliser la fonction `range` [1] pour créer un tableau représentant un intervalle d'éléments.

2 Tableaux

Vous utiliserez la fonction `print_r` pour afficher chacun des tableaux que vous créerez de façon lisible. Le cas échéant, ces appels devront être effectués dans un bloc `<pre> ... </pre>` qui permet d'afficher du texte préformatté.

Exercice 3 — *Création*

1. Créez les tableaux qui s'affichent de la manière suivante à l'aide de `print_r`.

```
Array
(
    [0] => toto
    [1] => titi
    [2] => tutu
)
```

```
Array
(
    [1] => toto
    [3] => tutu
    [4] => tata
)
```

```
Array
(
    [toto] => titi
    [2] => tutu
    [3] => 3
)
```

2. Attention, la conversion de type automatique à la volée peut vous jouer des tours. D'après vous, que fait le code suivant ?

1. http://php.net/manual/fr/function.range.php

```php
$tableau = array(
    0 => "toto",
    "0" => "titi",
    "00" => "tutu"
);$
```

Exercice 4 — *Accesseur* Créez (ou réutilisez) un tableau et, en utilisant une boucle, affichez pour chacune de ses clés k et de ses valeurs v : "`La valeur associée à` k `est` v". Par exemple, sur le deuxième tableau de la question précédente la boucle devra afficher

> La valeur associée à 1 est toto.
> La valeur associée à 3 est tutu.
> La valeur associée à 4 est tata.

Exercice 5 — *Fonctions sur des tableaux*

1. Écrire une fonction PHP prenant en argument un tableau et renvoyant sa taille.
2. Écrire une fonction prenant en argument un tableau, et renvoyant un nouveau tableau inversant les clés et les valeurs.
   Que se passe-t-il lorsque le tableau passé en entrée possède deux fois la même valeur associée à deux clés différentes ?
3. Écrire une fonction prenant en argument deux tableaux et retournant leur union. C'est à dire que le tableau résultant doit avoir les éléments de l'un et de l'autre. Si les deux tableaux ont une clé en commun, vous garderez la dernière clé et vous afficherez un avertissement avec le nom de la clé problématique.
4. Écrire une fonction prenant en paramètres deux tableaux et retournant un tableau sans index explicite contenant les clés communes des deux tableaux.

3 Tableaux multidimensionnels

Exercice 6 — *Tableaux multidimensionnels*

1. Écrire un tableau associatif multidimensionnel `$pays` dont les clés sont les noms de quatre pays de votre choix et dont les valeurs sont des tableaux associatifs. Ces tableaux auront pour clés la capitale, le nombre d'habitants, et le continent.
2. Écrire une fonction `findArray($tab, $index, $val)` qui renvoie, pour un tableau multidimensionnel, un index, et une valeur, un tableau sans index explicite contenant toutes les clés dont la valeur correspondant à l'index est la valeur spécifiée. Par exemple, `findArray($pays, "continent", "Europe")` nous renverra un tableau de tous les pays dont la valeur associée à la clé `"continent"` est `"Europe"`.
3. Écrivez une fonction `flatten` qui aplatit les deux premiers niveaux d'un tableau de dimension n (on suppose $n > 1$). Les nouvelles clés seront composées des clés originelles ainsi que des clés des tableaux correspondant aux clés originelles (par exemple, à la clé `"France/population"`, on fera correspondre le tableau `$pays["France"]["population"]`).