PHP : PHP Hypertext Preprocessor

1. Présentation

PHP est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web. Environ 80% des sites disponibles sur le Web l'utilisent actuellement.
Des sites connus utilisent PHP comme FaceBook, Blablacar, Spotify, Vogue, PrestaShop, Dailymotion etc
PHP est simple et facile à utiliser, ce qui est l'une des principales raisons pour lesquelles de nombreux développeurs préfèrent ce langage de programmation.

Exemple de script PHP pour afficher **Hello World !** :

```
<?php
    echo("Hello World !");
?>
```

Premières différences avec Python, un script PHP est encadré des balises de début **<?php** et de fin **?>**, chaque instruction se termine par un **;** et le ficher dans lequel se trouve le script doit avoir l'extension **php**, exemple : **bonjour.php**

2. De Python à PHP

2.1. Les variables

Exemple d'utilisation :

- en Python

```
a = 1 # a est un int
a = "Hello World ! " # a est devenu un string
```

- en PHP

```
$a = 1; # a est un int
$a = "Hello World ! "; # a est devenu un string
```

PHP comme Python est un langage typé dynamiquement, c'est-à-dire qu'on peut changer le type d'une variable.
PHP ne nous oblige pas à indiquer le type qu'une variable va contenir, mais le déduit de la valeur que nous lui attribuons. Il met aussi automatiquement à jour le type de la variable à chaque fois que nous lui attribuons une nouvelle valeur. Pour indiquer qu'il s'agit d'une variable, il faut utiliser le symbole **$** et chaque instructions doit se terminer par le **;**

Afin d'éviter les problèmes, il peut être bon de forcer une variable à un type spécifique, d'une manière explicite, ce qui nous permettra de savoir que lorsque le flux du programme atteint un point donné, cette variable aura le type de données attendu. En PHP, il y a plusieurs façons de forcer une variable à un type.

Définir le type avec settype(). Nous pouvons forcer une variable à changer de type avec la fonction settype().

Il y a une autre façon d'exécuter un forcé, de sorte qu'une variable se comporte comme un certain type. Maintenant nous allons voir un autre mécanisme de forçage qui est similaire à d'autres langages tels que C, Java, Python.

- en pyhon

```
number = int(input("Please enter an integer: "))
number_decimal =  float(input("Enter any number: "))
```

- en php

```
<?php
    $name = (string)readline("Your name: ");
    $int = (int)readline('Enter an integer: ');
    $float = (float)readline('Enter a floating');
?>
```

2.2. La concaténation

Concaténer signifie littéralement « mettre bout à bout ». L'opérateur de concaténation qui est la virgule en Python est le point (.) en PHP, il va donc nous permettre de mettre bout à bout deux chaines de caractères.

- en python

```
a = 10
b = 5
print("a*b = (a*b)")#affiche a*b = (a*b)
```

```
print(a,'*',b,'=',a*b)#affiche 10*5=50
print(f"{a}*{b}={( a * b)}")#affiche 10*5=50
```

- en php

```
<?php
    $a = 10  ;
    $b = 5 ;
    echo("$a*$b=($a*$b) \n") ; //affiche 10*5=(10*5)
    echo("$a*$b=".($a*$b)."\n") ; //affiche 10*5=50
?>
```

2.3. les conditions : if... elseif...else

On veut écrire un programme qui teste la température de l'eau et détermine son état (glace < 0, liquide >=0 et <100, vapeur >= 100) :

temp.py

```
temp_eau = int(input(" entrez la température de l'eau  "))

if temp_eau < 0 :
    print("c'est de la glace")

elif temp_eau < 100 :
    print(" c'est liquide")

else :
    print("c'est de la vapeur")
```

temp.php

```
$tempEau = readline(" entrez la température de l'eau  ");

if ($tempEau < 0 ) {
    echo("c'est de la glace \n");
}

elseif ($tempEau < 100) {
    echo("c'est liquide \n");
}
else {
    echo("c'est de la vapeur \n");
}
```

L'indentation est primordiale avec Python car elle sert à déterminer les blocs qui constituent votre code là où d'autres langages comme PHP privilégient les accolades **{ }** pour spécifier ces blocs. Lorsque l'indentation n'est pas nécessaire, elle est quand même utilisée pour une meilleure lisibilité du programme, car l'oubli d'une accolade provoquera une erreur.

Le elif en Python est remplacé par elseif → sinon si. En PHP, vous devez mettre la ou les **conditions entre parenthèses**.

Les opérateurs de comparaison en Python et PHP sont identiques :

- == (égal à) : renvoie True si les deux valeurs sont égales, False sinon.
- != (différent de) : renvoie True si les deux valeurs sont différentes, False sinon.
- > (strictement supérieur à) : renvoie True si la première valeur est strictement supérieure à la seconde, False sinon.
- < (strictement inférieur à) : renvoie True si la première valeur est strictement inférieure à la seconde, False sinon.
- >= (supérieur ou égal à) : renvoie True si la première valeur est supérieure ou égale à la seconde, False sinon.
- < = (inférieur ou égal à) : renvoie True si la première valeur est inférieure ou égale à la seconde, False sinon.
- || ou OR : OU logique Vérifie qu'une des conditions est réalisée
- && ou AND : ET logique Vérifie que toutes les conditions sont réalisées

PHP accepte les 2 notations pour le **ET** et le **OU** alors que Python n'accepte que and et or et d'aures langage comme JAVA ou Kotlin n'accepte que || et &&

2.4. les boucles : for...while...do.while

2.4.1. for.....

Exemple1 afficher les nombres de 0 à 20

- en python

```
for i in range (21) : ①
    print(i)
```

① En appelant range avec un seul paramètre, on lui donne un stop. Ainsi, **range(3)** nous renvoie : **[0, 1, 2]**.
En appelant range avec deux paramètres, on lui donne un start et un stop. Ainsi, **range(2, 8)** nous renvoie : **[2, 3, 4, 5, 6, 7]**.
Et finalement en appelant range avec trois paramètres, on lui donne un start, un stop et un step. Ainsi, **range(3, 10, 2)** nous renvoie : **[3, 5, 7, 9]**.

- en php

```php
for ($i = 0 ; $i <= 20 ; $i++ ) { ①
    echo ("$i \n");
}
```

① En PHP comme dans de nombreux langages vous devez obligatoirement indiquer une valeur de départ : $i = 0 , une condition de sortie de boucle $i < = 20 et une incrémentation $i++

$i++ signifie augmente la valeur de **$i de 1** à chaque tour de boucle, cela correspond au **step ou pas** avec Python, elle peut également s'écrire comme en Pyhton **$i** += **1**, ou encore **$i** = **$i** + **1**. Ainsi $i-- va signifier diminue la valeur de **$i de 1**.

Exemple2 afficher les nombres pairs de 50 à 100 inclus

python

```python
for i in range (50,101,2) :
    print(i)
```

php

```php
for ($i = 50 ; $i <= 100 ; $i+=2 ) {
    echo ("$i \n");
}
```

Exemple3 afficher les nombres 100 à 1 inclus

python

```python
for i in range (100,0,-1) :
    print(i)
```

php

```php
for ($i = 100 ; $i >= 1 ; $i-- ) {
    echo ("$i \n");
}
```

2.4.2. La boucle While

while signifie « Tant que ». On dit donc à l'ordinateur :
« Tant que la condition est vraie, répèter les instructions ». La syntaxe est identique en python et php, il suffit juste de privilégier les accolades **{ }** pour spécifier les blocs de programme à exécuter.

python

```
x = 1
while x < 10:
    print("x a pour valeur", x)
    x = x * 2
```

php

```
    $x = 1 ;
    while ($x < 10) {
        echo("x a pour valeur $x \n");
        $x = $x * 2 ;
    }
```

① **\n** pour forcer le saut de ligne en mode console

2.4.3. La boucle do....while

do...while signifie « Faire .....Tant que ». On dit donc à l'ordinateur :
« Faire les instructions....Tant que la condition est vraie ». La condition n'est pas vérifiée en début de boucle mais à la fin, la boucle sera donc exécutée au moins une fois.

Par exemple on veut forcer l'utilisateur à entrer un nombre positif :

```
do {
    $nb = readline('Entrer un nombre positif: ');

} while ( $nb <= 0 );
```

Cette structure n'existe pas en Python mais elle peut être facilement mise en place avec un while :

```
while True:
```

```
    number = int(input("Entrer un nombre positif : "))
    if  number > 0 :
         break
```

2.5. Les collections

En Python, les listes, les dictionnaires, les tuples et les ensembles (set) sont des collections prédéfinies que vous pouvez utiliser pour stocker un ensemble d'éléments.
En PHP vous avez les tableaux avec des indices classiques et les tableaux associatifs, à une clé on associe une valeur (dictionnaire en python) . En fait un tableau classique est également un tableau associatif mais avec des clés numériques.

Les tableaux ou les array, permettent de stocker plusieurs valeurs dans une seule variable. Imaginons avoir besoin de stocker une liste de villes. Si cette liste est très limitée, on pourrait stocker chaque ville dans une variable, cette solution fonctionne pour des volumes faibles de données mais elle est peu évolutive. En effet, si on souhaite ajouter une nouvelle ville, il sera nécessaire de créer une nouvelle variable. La solution, c'est donc de créer un tableau ou un array.

```
<?php
   // déclaration classique
   $villes = array('Fontainebleau', 'Paris', 'Lyon','Nice');

   // désormais une syntaxe plus proche de celle de Python est possible
   $villes = ['Fontainebleau', 'Paris', 'Lyon','Nice'];

?>
```

```
<?php
   $villes = ['Fontainebleau', 'Paris', 'Lyon','Nice'];
   echo $villes;
?>
```

Dans l'exemple ci-dessus, on cherche à afficher les valeurs de la variable $villes avec l'instruction echo. Mais on ne peut pas afficher les valeurs d'un array de cette manière. L'exemple ci-dessus le prouve bien, le résultat de l'instruction echo sur un tableau produit l'affichage **Array**, comme pour nous indiquer que ce n'est pas la bonne manière de procéder avec ce type de variable.

AFFICHER LES VALEURS D'UN TABLEAU EN PHP

Il existe plusieurs méthodes pour afficher les valeurs d'un tableau. Commençons par la méthode la plus simple. Conservons le même exemple que précédemment, à savoir un tableau numéroté. Chaque valeur possède une clé unique. Avec la syntaxe ci-dessous, on peut donc afficher la valeur d'une clé précise.

```
<?php
```

```
   $villes = ['Fontainebleau', 'Paris', 'Lyon'];
   echo $villes[0].' - '.$villes[1].' - '.$villes[2];
?>
```

Une approche assez simple et courante consiste à ajouter des éléments à un array en utilisant le $array[] = $var

```
<?php
   $villes[] = 'Melun'; //ajout de la Ville de Melun
?>
```

Pour afficher tous les éléments du tableau, on peut utiliser une boucle for, le premier élément étant à l'indice 0 et le dernier sera à la longueur du tableau (count) -1 (si on a 4 éléments dans un tableau les indices commençant à 0, le 4e élement sera à l'indice 3)

```
<?php
   $villes = ['Fontainebleau', 'Paris', 'Lyon'];
   for($i = 0 ; $i < count($villes) ; $i++){
        echo "$villes[$i] \n" ;
    }
?>
```

LA BOUCLE FOREACH

La boucle foreach permet de parcourir simplement les tableaux.
Pour la boucle foreach, pas besoin de condition qui permet de stopper la boucle. Elle commence donc au premier et s'arrête automatiquement après la lecture du dernier. On précise donc en premier lieu le tableau que l'on souhaite parcourir. Dans notre exemple, la variable $villes. À chaque itération de la boucle, la clé de l'élément parcouru est assignée dans la variable $key et la valeur de l'élément est assignée à la variable $laVille.

exemple1 : si on ne souhaite afficher la valeur tous les éléments du tableau

```
<?php
   $villes = ['Fontainebleau', 'Paris', 'Lyon'];
   foreach($villes as $laVille){
        echo "$laVille \n" ;
    }
?>
```

exemple2 : si vous souhaitez également afficher les clés (les indices dans un tableau classique) la syntae est la suivante :

```
<?php
   $villes = ['Fontainebleau', 'Paris', 'Lyon'];
   foreach($villes as $key=>$laVille){
```

```
        echo "$laVille est à la position $key \n" ;
    }
?>
```

LES TABLEAUX ASSOCIATIFS

Nous avons pour le moment uniquement travaillé avec des tableaux numérotés. Les clés étaient numériques. Il existe un autre type de tableau où les clés sont des chaînes de caractères. On parle alors de tableau associatif.

```
<?php
   $villes = [
      'Fontainebleau' => 77,
      'Paris' => 75,
      'Lyon' => 69
   ];
?>
```

La différence entre un tableau classique et associatif se situe au niveau du type de la clé. Un tableau classique utilise des clés numériques qui ne sont pas indiquées dans le tableau, tandis qu'un tableau associatif utilise des chaînes de caractères.

MODIFIER UN TABLEAU

- modifier une valeur d'un tableau classique, on souhaite remplacer Fontainebleau par Melun

```
   $villes = ['Fontainebleau', 'Paris', 'Lyon'];
   $villes[0] = 'Melun';
```

- modifier une valeur d'un tableau associatif, on souhaite remplacer 77 par "Seine-et-Marne" pour la clé Fontainebleau

```
$villes = ['Fontainebleau' => 77, 'Paris' => 75, 'Lyon' => 69];
   $villes['Fontainebleau'] = 'Seine-et-Marne';
```

- ajouter la ville de Nice dans le tableau classique

```
   $villes = ['Fontainebleau', 'Paris', 'Lyon'];
   $villes[] = 'Nice';
```

- ajouter la ville la clé Nantes avec pour valeur 44 dans le tableau associatif

```
   $villes = ['Fontainebleau' => 77, 'Paris' => 75, 'Lyon' => 69];
   $villes['Nantes'] = 44;
```

- supprimer un élément du tableau unset($villes[0])
- compter les éléments d'un tableau count($villes)
- rechercher une valeur dans un tableau : in_array(10,$tab) retourne **TRUE** si c'est le cas, et **FALSE** dans le cas inverse.
- rechercher si une clé existe dans le tableau : array_key_exists('Paris') retourne **TRUE** si c'est le cas, et **FALSE** dans le cas inverse.
- trier les tableaux, il existe plusieurs façons de trier un tableau, par les valeurs, les clés, en ordre inverse, en conservant les associations clé-valeur.
  - trier un tableau par ordre alphabétique ou numérique avec la fonction sort($tab)
  - Vous pouvez trier un tableau en ordre inverse avec la fonction rsort($tab)
  - Si vous voulez trier en conservant les associations clé-valeur, utilisez asort() au lieu de sort() pour trier par ordre alphabétique, et arsort() au lieu de rsort() pour trier en ordre inverse.
- pour les tableaux avec des donées numériques pour calculer la somme **array_sum($tab)**, le minimum **min($tab)**, le maximum **max($tab)** etc etc

Les fonctionnalités autour de la manipulation des tableaux sont nombreuses. Des fonctions existent nativement pour répondre à des besoins très précis. N'hésitez pas à consulter la documentation officielle.

2.6. Des tableaux dans les tableaux

Un tableau multidimensionnel est un tableau qui va lui-même contenir d'autres tableaux en valeurs.

On appelle ainsi tableau à deux dimensions un tableau qui contient un ou plusieurs tableaux en valeurs, tableau à trois dimensions un tableau qui contient un ou plusieurs tableaux en valeurs qui contiennent eux-mêmes d'autres tableaux en valeurs et etc.

Les « sous » tableaux vont pouvoir être des tableaux numérotés ou des tableaux associatifs ou un mélange des deux.

Exemple, on souhaite créer un tableau avec comme clé le nom des élèves et comme valeur une note.

```
$tabEleves = ['alain' => 12, 'Pascal' => 15, 'Lionel' => 6, 'Pionil' => 16 ];

foreach($tabEleves as $nom=>$note){
    echo "$nom a une note de $note \n" ;
}
```

Si on souhaite associer à un élève plusieurs notes, on devra stocker les notes dans un tableau, ainsi la valeur associée à la clé élève sera un tableau

```
$tabEleves = ['alain' => [12,5,13], 'Pascal' => [15,8,12], 'Lionel' => [6,5,15] ];
```

```
foreach($tabEleves as $nom=>$lesNotes) { ①
    echo "$nom \n" ;
    foreach($lesNotes as $laNote){ ②
        echo "$laNote \n" ;
    }
}
```

① à la clé élève est associée non pas une seule note mais un ensemble de notes stockées dans un tableau

② pour afficher chaque note, on devra parcourir le tableau **$lesNotes** pour afficher chaque note **$laNote**

Parfois, on voudra simplement afficher la structure d'un tableau PHP sans mise en forme pour vérifier ce qu'il contient ou pour des questions de débogage.

Le PHP nous fournit plusieurs possibilités de faire cela : on va pouvoir soit utiliser la fonction print_r(), soit la fonction var_dump() ou encore json_encode() que nous connaissons déjà pour afficher n'importe quel type de tableaux (numérotés, associatifs ou multidimensionnels).

```
$tabEleves = ['alain' => [12,5,13], 'Pascal' => [15,8,12]];

echo($tabEleves); // affiche Array

print_r($tabEleves);// Affiche
/*
ArrayArray
(
    [alain] => Array
        (
            [0] => 12
            [1] => 5
            [2] => 13
        )

    [Pascal] => Array
        (
            [0] => 15
            [1] => 8
            [2] => 12
        )

)
*/

echo(json_encode($tabEleves)); ①
// {"alain":[12,5,13],"Pascal":[15,8,12]}
```

① JSON (JavaScript Object Notation) est un format de fichier textuel conçu pour l'échange de

données. Il représente des données structurées basées sur un sous-ensemble du langage de programmation JavaScript. JSON est populaire en raison de son style autodescriptif, facile à comprendre, léger et compact. Il est compatible avec de nombreux langages de programmation, environnements et bibliothèques.

3. Les fonctions

Comme en Python une fonction correspond à une série cohérente d'instructions qui ont été créées pour effectuer une tâche précise. Pour exécuter le code contenu dans une fonction, il va falloir appeler la fonction.

Une des forces du langage PHP est sa richesse en terme de fonctionnalités. En effet, il dispose à l'origine de plus de 3 000 fonctions natives prêtes à l'emploi garantissant aux développeurs de s'affranchir de temps de développement supplémentaires et parfois fastidieux. Ces fonctions permettent entre autre de traiter les chaînes de caractères, d'opérer mathématiquement sur des nombres, de convertir des dates, de se connecter à un système de base de données, de manipuler des fichiers présents sur le serveur...etc PHP puise aussi sa richesse dans le dynamisme de sa communauté de développeurs.

- count() qui permet de compter le nombre d'éléments d'un tableau
- strlen() qui permet de calculer la longueur d'une chaîne de caractères
- min, max, array_sum, rand, round, pow etc fonctions mathématiques.

En plus des fonctions internes, le PHP nous laisse la possibilité de définir nos propres fonctions. Pour déclarer une fonction, il faut déjà commencer par préciser le mot clef function (def en Python) qui indique au PHP qu'on va définir une fonction personnalisée.

Depuis sa dernière version majeure (PHP7), le PHP nous offre néanmoins la possibilité de préciser le type de données attendues lorsqu'on définit une fonction. Si une donnée passée ne correspond pas au type attendu, le PHP essaiera de la convertir dans le bon type et s'il n'y arrive pas une erreur sera cette fois-ci renvoyée.

temp3.py

```
def etat_eau(temp_eau : int) -> str: ①
    etat = ""

    if temp_eau < 0 :
        etat = "c'est de la glace"

    else :
        if temp_eau < 100 :
            etat = " c'est liquide"
        else :
            etat = "c'est de la vapeur"

    return etat
```

① Définition d'une fonction nommée **etat_eau** qui va recevoir un argument (temp_eau de type int), qui retournera une chaîne de caractère (str)

temp.php

```
function etatEau(int $temp) { ①
        $etat = "";

        if ($temp < 0 )
            $etat = "c'est de la glace";

        else {
            if ( $temp < 100 )
                $etat = " c'est liquide";
            else
                $etat = "c'est de la vapeur";
        }
        return $etat ;
    }
```

① Définition d'une fonction nommée **etatEau** qui va recevoir un argument ($temp de type int). Vous pouvez remarque que les accolades pour les if - else sont facultatives, lorsqu'il n'y a qu'une seule instruction.

3.1. Exercices : variables et conditions

3.1.1. bonjour

a) Bonjour… qui ?

Écrire le programme Bonjour.php qui demande à l'utilisateur de saisir son nom et qui affiche un message personnalisé (« bonjour durand »).

b) Élémentaire mon cher

Écrire un programme qui lit deux entiers et qui affiche leur somme, leur différence, leur produit et leur quotient (la division par 0 n'est pas traitée).

3.1.2. tarif réduit

Écrire un programme qui demande l'âge de l'utilisateur et lui indique s'il a droit au tarif réduit (moins de 26 ans).

Reprendre votre programme , mais le tarif réduit s'applique pour les personnes ayant moins de 26 ans ou plus de 65 ans.

3.1.3. départements

Concevoir un programme qui détermine si l'utilisateur habite l'île de France (département 75, 77, 78, 91, 92, 93, 94). Initialiser un tableau contenant tous les départements d'Île-De-France et ensuite

utiliser la fonction in_array .

1. L'utilisateur rentre un numéro de département, le programme lui précise si il fait parti de l'île de France ou non.
2. Votre programme doit ensuite s'assurer que le code saisi est compris entre 1 et 99.Tand que le département saisi n'est pas comris entre 1 et 99, le programme demandera à l'utilisateur de saisir un département valide.

3.1.4. produit

Concevoir un programme qui affiche le signe du produit de deux nombres (positif ou négatif) saisis et ceci sans calculer leur produit. Le produit de 2 nombres est positif si les 2 nombres sont positifs ou les 2 nombres sont négatifs sinon il est négatif.

exemple :

```
chiffre 1 --> -2 chiffre2 --> 5 ----- produit négatif
```

```
chiffre 1 --> 2 chiffre2 --> 4 ----- produit positif
```

3.1.5. racine carrée

Écrire un programme qui affiche la racine carrée du produit de a par b de deux nombres positifs a et b de type float entrés par l'utilisateur. Si au moins un de ces nombres est négatif, le programme affiche un message d'erreur. La fonction sqrt() est une fonction intégrée en PHP qui renvoie la racine carrée d'un nombre.

3.1.6. bissextile

Écrire un programme qui permet de déterminer si une année saisie par l'utilisateur est bissextile. Une année est dite bissextile si c'est un multiple de 4, sauf si c'est un multiple de 100. Toutefois, elle est considérée comme bissextile si c'est un multiple de 400.

3.2. Les boucles

3.2.1. 1 : Compteur

Ecrire un programme Compteur.php qui permet de compter et d'afficher les nombres de 1 jusqu'à 20.

Améliorer le programme pour faire saisir à l'utilisateur jusqu'à quel chiffre il souhaite que le programme compte.

3.2.2. 2 : pair ou impair

Ecrire un programme qui demande à l'utilisateur de saisir un chiffre, le programme devra indiquer

si ce chiffre est pair ou impair (utilisé le modulo %) puis afficher les dix nombres pairs ou impairs suivant.

Exemple si l'utilisateur saisie 10,

le programme affichera > nombre pair >10 – 12 - 14 – 16 ……28

3.2.3. 3 : multiple

Écrire un programme qui affiche les multiples de 5 jusqu'à 50. Améliorez votre programme, cette fois il doit afficher les multiples d'un nombre saisi par l'utilisateur jusqu'à un nombre max que l'utilisateur aura aussi choisi.

3.2.4. 4 : devinette

Écrire un programme qui génère de manière aléatoire un nombre entier compris entre 0 et 100 ($nbAleatoire = rand(0,100);) et qui invite un joueur à deviner ce nombre en utilisant le moins d'essais possible avec 7 possibilités maximum. A chaque essai, le programme répond trop grand, trop petit, gagné en n essais si le nombre et trouvé et perdu si le nombre d'essais a été dépassé.

3.3. Trouver les bonnes fonctions

3.3.1. Exercice 1 :

Écrire un programme qui demande à l'utilisateur de saisir une phrase puis à l'aide des méthodes que vous devez trouver, votre programme affichera :

1. phrase courte si elle comporte moins de 20 caractères
2. phrase de longueur moyenne si elle comporte au moins 20 caractères et moins de 50
3. phrase longue si elle a plus de 50 caractères

3.3.2. Exercice 2 :

Écrire un programme qui permet de saisir un mot puis à l'aide des méthodes que vous devez trouver, d'afficher:

1. la première lettre de ce mot.
2. la dernière lettre de ce mot
3. le nombre de lettres de ce mot

3.3.3. Exercice 3 :

Écrire un programme qui demande à l'utilisateur son nom et son prénom. Celui-ci doit ensuite afficher :

1. Le nom en majuscule suivi du nombre de lettres
2. Le prénom en minuscule excepté la 1ère lettre en majuscule suivi du nombre de lettres

Exemple : kEYnEs jOHn ⇒ KEYNES (6) John (4)

3.3.4. Exercice 4

Couper une phrase si elle comporte plus de 10 caractères et lui ajouter trois points...

3.3.5. Exercice 5

Écrire un programme qui permet de saisir un mot puis à l'aide des méthodes que vous devez trouver, d'afficher s'il s'agit d'un palindrome, c'est à dire un mot dont l'ordre des lettres reste le même qu'on le lise de gauche à droite ou de droite à gauche, exemple ; radar, elle, kayak etc...

3.4. les collections

3.4.1. déclaration de tableaux

Initialiser un tableau ($tab) de 10 entiers compris entre 1 et 100 choisis au hasard.

```
for ($i = 0 ; $i < 10 ; $i++ ) {
    $tab[] = rand(1,100);
}
```

1. Ecrire le programme qui affiche la somme, le plus petit et le plus grand des éléments de ce tableau.(trouvez les bonnes fonctions)
2. Ajoutez la déclaration de 2 tableaux supplémentaires $tabPair et $tabImpair. Ajoutez le code qui copie les éléments pairs de $tab dans $tabPair, et les éléments impairs dans $tabImpair.
3. Afficher à l'aide de la boucle **foreach** le contenu de chaque tableau.

3.4.2. tableau associatif

Initialiser un tableau associatif, avec une dizaine de noms d'élèves et leur note :

```
$eleves = array('adrien'=>12, 'jemuel'=>8, 'tristan'=>18.........); ①
}
```

1. Ecrire le code qui affiche le nom de chaque élève et leur note.
2. Afficher la moyenne de la classe
3. Initialiser un tableau associatif, avec une dizaine de noms d'élèves et leurs notes

① déclaration classique d'un tableau. La notation [elt1, elt2...] est récente.

```
$elevesSio = array('adrien'=>array(12, 10,8), 'jemuel'=>array(8,13,15).........);
}
```

Ecrire le code qui affiche le nom de chaque élève et l'ensemble de leurs notes ainsi que la moyenne.

3.4.3. Pays - Chômage

Initialiser un tableau associatif, avec une dizaine de noms de pays et leur taux de chômage :

```
$pays = array('France'=>7, 'USA'=>4, 'Belgique'=>8.........);
}
```

1. Ecrire le code qui affiche le nom de chaque pays ayant un taux de chômage supérieur à 6.

3.5. Les fonctions

3.5.1. fonction serie

Ecrire la fonction serie() qui prend comme argument un nombre entier n et qui renvoie un tableau contenant les nombres entiers allant de 1 à n.

Exemple : serie(5)

entrée : 5
sortie : print_r (1, 2, 3, 4, 5)

3.5.2. fonction serieInverse()

Ecrire la fonction serieInverse() qui prend comme argument un entier n et qui renvoie un tableau contenant les nombres entiers allant de n à 1.

Exemple : serieInverse(5)
entrée 5
sortie :
(5, 4, 3, 2, 1)

3.5.3. fonction somme()

Ecrire la fonction somme() qui prend comme argument un entier n et qui renvoie la somme des nombres entiers allant de 1 à n.

3.5.4. fonction sommeCube()

Ecrire la fonction sommeCube() qui prend comme argument un entier n et qui renvoie la somme des nombres entiers au cube allant de 1 à n.

3.5.5. fonction produit()

Ecrire la fonction produit() qui prend comme argument un entier n et qui renvoie la factorielle de n.
n! = 1 × 2 × 3 × 4 × □ × (n−1) × n

3.5.6. pair

Ecrire la fonction tabPair() qui prend comme argument un tableau n de nombres entiers et qui renvoie un tableau ne contenant que les nombres pairs

3.5.7. impair

Ecrire la fonction tabImpair() qui prend comme argument un tableau n de nombres entiers et qui renvoie un tableau ne contenant que les nombres impairs

3.5.8. parité

Ecrire la fonction tabParite() qui prend comme argument un tableau n de nombres entiers et qui renvoie un tableau associatif ayant 2 clés : la clé 'pair' ne contenant que les nombres pairs et la clé 'impair' ne contenant que les nombres impairs.

3.5.9. droitPrime

Vous devez écrire une fonction pour les caisses d'allocations familiales : droitPrime. Celle- ci reçoit 2 arguments (le nombre d'enfant et le montant des ressources annuelles d'une personne) puis renvoie un booléen en fonction des règles ci dessous :

Nombre d'enfants	**Plafonds de ressources**
1	33 044,00 €
2	38 045,00 €
3 ou+	44 046,00 €

Explication : Si les données saisies concernent une personne ayant 1 enfant, cette personne aura le droit à la prime si ces revenus n'excèdent pas le montant de 33044€

3.5.10. listeDiviseurs

Ecrire la fonction listeDiviseurs() qui prend comme argument un nombre entier n et qui renvoie un tableau contenant tous les diviseurs du nombre n.

3.5.11. estPremier

Ecrire la fonction estPremier() qui prend comme argument un nombre entier n et qui renvoie True si n est un nombre premier et False sinon. Vous pouvez utiliser la fonction listeDiviseurs écrite précédemment sachant qu'un nombre premier n'a que 2 diviseurs (1 et lui-même).

3.5.12. estParfait

Ecrire la fonction estParfait() qui prend comme argument un nombre entier n et qui renvoie True si n est un nombre parfait et False sinon. En arithmétique, un nombre parfait est un entier naturel égal à la moitié de la somme de ses diviseurs ou encore à la somme de ses diviseurs stricts. Exemple 6 est un nombre parfait car la somme des ses diviseurs stricts (1 + 2 + 3) = 6, 28 est également un

nombre parfait (1 + 2 + 4 + 7 + 14) = 28. Les nombres parfaits sont rares, il n'en existe que trois inférieurs à 1000 qui sont 6, 28 et 496. Ensuite vient 8128, puis 33 550 336, vous pouvez donc initialiser un tableau avec ces nombres et ensuite utiliser la fonction in_array pour écrire cette fonction.

3.5.13. sommeChriffres

Écrire la fonction sommeChiffres() qui prend comme argument un nombre entier n et qui renvoie un entier représentant la somme des chiffres qui compose le nombre entier n. Exemple n → 125 somme chiffre : (1 + 2 + 5) = 8. Utiliser le modulo 10 et la fonction intval.

3.5.14. nombreAmi

Écrire la fonction nombreAmi() qui prend comme argument deux nombres entiers et qui renvoie True si ils sont amis et False sinon. Deux nombres seront amis si la somme des chiffres qui composent chaque nombre est identique. Exemple 66 (6 + 6) = 12 est ami avec 93 (9 + 3) = 12. Attention la notion mathématique des nombres amicaux est différente.