Royaume du maroc
ⵜⴰⴳⵍⴷⵉⵜ ⵏ ⵍⵎⵖⵔⵉⴱ

Ministère de l'Education Nationale
du préscolaire & des sports
ⵜⴰⵎⴰⵡⴰⵙⵜ ⵏ ⵜⵎⴰⵡⴰⵙⵜ ⴰⵏⴰⵎⵓⵔ
ⴰⵏ ⵜⵎⵎⴰⵔ ⴰⵎⵙⵡⴰⵔⴷ ⴰⵏ ⵜⵙⵏⵙⵉⵜ

Lycée Technique de Taza

شهادة التقني العالي

Brevet de Technicien Supérieur

Multimédia et Conception Web

Module : Développement Multimédia

Année professionnelle : 2024 / 2025

Professeur : HICHAM ZAARAOUI

Chapitre 6

Les fonctions en PHP

I. Les fonctions natives de PHP

- PHP propose en standard une multitude de fonctions natives écrites en langage C.
- Pour vérifier la disponibilité d'un module, vous disposez de la fonction: `get_loaded_extensions()`, qui retourne un tableau contenant les noms de tous les modules d'extension installés sur le serveur.

Exemple :

```php
<?php
$tabext = get_loaded_extensions();
natcasesort($tabext); //trie en odre croissant
foreach($tabext as $cle=>$valeur) {
    echo "$valeur <br>";
    }
?>
```

I. Les fonctions natives de PHP

Pour obtenir la liste des fonctions d'un module, vous disposez de la fonction suivante :

```
array get_extensions_funcs("nom_module").
```

Exemple

```
<?php
$tab = get_extension_funcs("mysqli");
natcasesort($tab);
foreach($tab as $cle=>$valeur) {
echo "$valeur <br>"; }
?>
```

Remarque

Pour vérifier la disponibilité d'une fonction native de PHP ou d'une fonction personnalisée, on utilise la fonction `function_exists("nom_fonction")`, qui renvoie la valeur TRUE si la fonction passée en paramètre existe et FALSE dans le cas contraire.

II. Créer des Fonctions personnalisées

1) Définition d'une fonction

- Une fonction est un bloc de code qui n'est pas exécuté de manière linéaire dans un script. Ce code ne le sera que lors de l'appel explicite de la fonction. Écrit une seule fois, ce code peut être exécuté aussi souvent que nécessaire.

2) Création d'une fonction

- Pour définir une fonction, on utilise la structure suivante:

Syntaxe:

```
function nom_fonction($paramtre1,$paramtre2,..){
            //code à exécuter
}
```

Appel de la fonction:

```
nom_fonction($argument1,$argument2,..);
```

II. Créer des Fonctions personnalisées

3) Les fonctions qui ne retournent pas de valeurs

Exemple 1:

```php
<?php
function EcrireMsg() {
  echo "Hello world!";
}
EcrireMsg(); // appel de la fonction
?>
```

Exemple 2:

```php
<?php
function Nom_age($nom, $age) {
echo "Il s'appel $nom, il est agé de $age ans<br>";
}
Nom_age("Ahmed", 33);
Nom_age("Samira", 24);
?>
```

II. Créer des Fonctions personnalisées

4) Les fonctions qui retournent une valeur

Exemple :

```php
<?php
function Produit($x,$y) {
    $p=$x*$y;
    return $p;
}
Produit(5,8); // appel de la function
Echo Produit(5,8); // affichage du resultat
?>
```

II. Créer des Fonctions personnalisées

5) Les fonctions qui retournent plusieurs valeurs

- PHP n'offre pas la possibilité de retourner explicitement plusieurs variables.
- Il suffit de retourner une variable de type ***array*** contenant autant de valeurs que désiré.

Exemple : Fonction de calcul du module et d'argument d'un nombre complexe (a+b*i)

```
<?php
function modarg($reel,$imag){
        $mod =sqrt($reel*$reel + $imag*$imag);
        $arg = atan2($imag,$reel);
        return array("module"=>$mod,"argument"=>$arg);}
        $a=5; $b=8; $complex=modarg($a,$b);
        echo "Nombre complexe $a + $b i:<br>
        module= {$complex["module"]} <br>
        argument ={$complex["argument"]} radians";
```

II. Créer des Fonctions personnalisées

6) Les fonctions qui ont des paramètres par défaut :

Lorsque vous créez une fonction personnalisée, vous pouvez procéder de même et donner des valeurs par défaut aux paramètres de la fonction.

Exemple : Fonction qui calcul le prix hors taxes à partir du prix TTC et le taux de TVA.

```php
<?php
function prixht($ptt,$tax=20) {
      $pht= round($ptt*(1-$tax/100),2);
      return $pht; }
      $prixtt=154;
      echo "Prix TTC= $prixtt DH ",prixht($prixtt)," DH <br>";
      echo "Prix TTC= $prixtt DH ",prixht($prixtt,20),"DH <br>";
      echo "Prix TTC= $prixtt HD ",prixht($prixtt,5.5),"DH";
?>
```

II. Créer des Fonctions personnalisées

7) Les fonctions avec un nombre de paramètres variables :

Pour créer une fonction acceptant un nombre variable de paramètres, il suffit de passer un tableau comme paramètres a cette fonction.

Exemple : Fonction qui calcul le produit de N nombres.

```php
<?php
function prod($tab){
$prod = 1;
foreach($tab as $val){ $prod*=$val; }
return $prod;}
$tab1= range(1,10);
echo "Produit des nombres de 1 à 10 :", prod($tab1),"<br>";
$tab2 = array(7,12,15,3,21);
echo "Produit des éléments du tableau 2 est: ",prod($tab2),"<br>";
?>
```

II. Créer des Fonctions personnalisées

8) Les fonctions particulières de PHP :

- **`integer func_num_args()`**: retourne le nombre d'arguments passés à la fonction, et s'utilise seulement dans le corps même d'une fonction.

- **`divers func_get_arg(integer $N)`** retourne la valeur du paramètre passé à la position $N. le premier paramètre a l'indice 0.

- **`array func_get_args():`** retourne un tableau indicé contenant tous les arguments passés à une fonction personnalisée.

III. Portée des variables

1) Les variables locales et globales:

- Toutes les variables utilisées **dans la déclaration d'une fonction** sont, sauf indication contraire, **locales** au bloc de définition de la fonction.
- Toutes les variables qui sont définies **en dehors d'une fonction** ou sont **globales** et accessibles partout dans le script qui les a créées.

Exemple :

```php
<?php
$x = 5; // x est une variable globale
function test() {
  // l'utilisation de x à l'intérieur de cette fonction va générer une erreur
  echo " La valeur x est : $x <br>";
}
myTest();
echo " La valeur de x est : $x ";
?>
```

Remarque : Pour éviter l'erreur, il faut ajouter l'instruction `global $x` à l'intérieur de la fonction myTest() .

III. Portée des variables

2) Les variables statiques

- Lors de l'appel d'une fonction, les variables locales utilisées dans le corps de la fonction ne conservent pas la valeur qui leur est affectée par la fonction.
- Pour conserver la valeur précédemment affectée entre deux appels d'une même fonction, il faut déclarer la variable comme statique en la faisant précéder du mot-clé ***static***

Exemple :

```php
<?php
function fct(){
        static $cpt=1;
        $cpt++;
        echo $cpt."<br>";
}
fct();
fct();
fct();
?>
```

IV. Passage des arguments par valeur et par référence

1) Passage des arguments par valeur

Les modifications des valeurs des arguments ne sont pas visibles à l'extérieur de la fonction.

2) Passage des arguments par référence

Les modifications effectuées dans le corps de la fonction sont répercutées à l'extérieur, c'est-à-dire, la fonction peut modifier la variable.

Exemple :

```php
<?php
function foo(&$var){
      $var++; }
$a=5;
echo "$a<br>";
foo($a);
echo "$a <br>";
?>
```

V. Cas particuliers

1) Fonctions dynamiques

PHP offre la possibilité de travailler avec des noms de fonctions dynamiques, qui peuvent être variables.

Code	Equivalent à
`$ch ="phpinfo";` `$ch();`	`phpinfo();`
`$ch = "date";` `echo $ch("d/M/Y H:i:s");`	`date("d/M/Y H:i:s");`

`$tabfonc= array("phpinfo","date");`
`$tabfonc[0]();`
`echo $tabfonc[1]("D d m Y H:i:s");`	

V. Cas particuliers

2) Fonctions conditionnelles:

Une fonction est dite conditionnelle si elle est définie à l'intérieur d'un bloc if.

Exemple :

```php
<?php test();
//salut(); Cet appel provoquerait une erreur fatale
$heure=date("H");
if($heure<18){
function salut(){ echo "Bonjour : Fonction accessible seulement
avant 18h00 <br>";}
}else{
function salut(){ echo "Bonsoir : Fonction accessible seulement
après 18h00 <br>"; } }
function test() {
echo "Fonction accessible partout <br>"; return TRUE;
} salut();
?>
```

V. Cas particuliers

3) Fonction définie dans une autre fonction:

La fonction incluse n'est alors utilisable que si celle qui la contient a été appelé une seule fois , sinon PHP lève une erreur fatale.

Exemple :

```php
<?php
//enfant();//ERREUR FATALE
function parent() {
      echo "Bonjour les enfants! <br>";
      function enfant(){ echo "Bonsoir papa !<br>"; }
}
parent();//Bonjour les enfants!
enfant();//Bonsoir papa !
enfant();//Bonsoir papa !
?>
```

V. Cas particuliers

4) Fonction récursives:

Une fonction est dite récursive si, à l'intérieur de son corps, elle s'appelle elle-même avec une valeur de paramètre différent (sinon elle boucle).

Exemple :

```php
<?php
function facto($n){
    if ($n==1) return 1;
    else {
    return $n*facto($n-1);
        }
                }
echo "factorielle = ",facto(6);//720
?>
```

Exercice

Créer une fonction qui calcule et retourne la mensualité à payer pour un prêt en fonction du capital emprunté, du taux d'intérêt et de la durée du prêt.

La formule de calcul d'une mensualité M est la suivante :

$$M = \frac{C \times T}{1 - (1 + T)^{-n}}$$

C: La capitale empruntée ($capitale).
T: Taux mensuel, mais le taux annuel qui est donnée en paramètre ($tauxann)
n: Durée en nombre de mois, mais la durée en année qui est donnée ($dureeann)
$assur: permet de calculer le montant de l'assurance. Il vaut 1 si le client désire s'assurer et 0 dans le cas contraire.
Calcul de l'assurance $ass soit 0,035% du capital par mois.

Correction

```
<!DOCTYPE html>
<html lang="en">
<head> <meta charset="UTF-8"><title>Mensualite</title></head>
<body>
<?php
//Definition de la fonction mensualite
function mensualite($capitale,$tann,$dann,$assur){
$tmen=$tann/100/12;
$dmois=$dann*12;
$assur=($capitale*(0.035/100))*$assur;
$men=($capitale*$tmen)/(1-pow(1+$tmen,-$dmois))+$assur;
$mens=round($men,2);
return $mens;
}
```

Correction

```
//appel de la fonction mensualite
$capitale=10000;
$tann=5;
$dann=3;
$assur=false;
$mens=mensualite($capitale,$tann,$dann,$assur=true);
echo "*Pour un capitale de $capitale d'un taux d'interet de
$tann% pour l'an sur $dann ans, sans assurance, la mensualité
est de : $mens <br>";
$assur=true;
$mensa=mensualite($capitale,$tann,$dann,$assur);
echo "*Pour un capitale de $capitale d'un taux d'interet de
$tann% pour l'an sur $dann ans, avec assurance, la mensualité
est de : $mensa <br>";
?> </body> </html>
```