Fonctions

- PHP possède plus de 1000 fonctions prédéfinies
- Une fonction est un bloc d'instructions
- Elle ne s'exécute pas automatiquement
- On peut créer nos propres fonctions

```
function NomFonction (parameteres) {
        // code à executer;
}
```

- *NomFonction* doit commencer par une **lettre** ou « _ »
- Une fonction doit être définie avant d'être appelée

Exemple

- Ecrire une fonction qui détermine le maximum entre deux nombres
- On peut la nommer Max

```
<?php
    Function Max (int $x, int $y) {
        If ($x>$y){
            echo "le maximum est: ".$x;
        else
            echo "le maximum est: ".$y;
                }
        }
    Max(12, 3); // appel de la fonction Elle affiche le maximum est:12
    ?>
```

```php
<?php
function maxi(int $x, int $y=3) {
  if ($x>$y){
  echo "le maximum est:".$x;
  }
   else {
   echo "le maximum est: ".$y;
  }
}
maxi(4,5); // affiche le maximum est:5
maxi(2); // affiche le maximum est:3 utilise la valeur par défaut
?>
```

- La fonction qui retourne la somme de deux entiers

```
<?php
  function somme(int $a, int $b) {
  return $a + $b;
 }
 echo somme(2, “3 ");
 ?>
```

- Sortie 5 // comme strict n’est pas activé, “3" est transformé en int(3), et la fonction retourne 5

- Langage PHP est faiblement typé

```
<?php
    declare(strict_types=1); // types doivent être respectés
    function somme(int $a, int $b) {
    return $a + $b;
    }
    echo somme(2,“3 ");
?>
```

Sortie: PHP Fatal error

Quelques fonctions mathématique

- **pi():** renvoie la valeur de **pi**
- **min():** renvoie la valeur **minimale** d'un tableau
- **max():** renvoie la valeur **maximale** d'un tableau
- **abs():** calcule la valeur **absolue** d'un nombre
- **sqrt():** calcule la **racine carrée** d'un nombre
- **round():** arrondit un nombre à la valeur **la plus proche**
- **Rand():** renvoie un nombre **aléatoire**
- **etc**

Tableaux (array)

- Un tableau stocke **plusieurs** valeurs dans une seule variable
- On peut accéder aux valeurs en se référant aux numéros d'indice ou clés
- En PHP, la notation **[ ]** ou la fonction **array()** sont utilisées pour créer un tableau
- Ex. $tab=[ ] // tableau vide

  $tab=array() // tableau vide

- Il existe **trois** types de tableaux:
  - **Indexés**: utilisent des indices *numériques*
  - **Associatifs**: utilisent des *clés* nommées (clé=>valeur)
  - **Multidimensionnels**: contenant un ou *plusieurs* tableaux

Tableau indexé

- Il existe deux façons de créer un tableau indexé:
  - L'index est attribué *automatiquement*

```
$couleurs=array('rouge', 'vert', 'bleu');
```

  - L'index est affecté *manuellement*

```
$couleurs[0]='rouge';
$couleurs[1]='vert';
$couleurs[2]='bleu';
```

  - On peut utiliser la boucle **for** pour parcourir tous les éléments d'un tableau indexé.
  - Ex.

```
For($x=0; $x<count($couleurs);$x++)
    echo $couleurs[$x];
```

Tableau associatif

- Utilise des *clés nommées*
- Il existe deux façons de créer ce type de tableau :
  - $age = array("Peter"=>“25", "Ben"=>"30", "Jean"=>"40");
  - $age['Peter'] = “25";
    $age['Ben'] = "30";
    $age['Jean'] = "40";
- On peut utiliser la boucle foreach pour parcourir les éléments d’un tableau associatif.

- Ex. <?php

```
$age = array("Peter"=>“25", "Ben"=>"30“);

foreach($age as $x => $value) {
        echo “Clé=" . $x . ",Valeur=" . $value;
        echo "<br>";
}
?>
```

Tableau multidimensionnel

- Dans un tableau à **deux** dimensions, on a besoin de **deux** indices pour sélectionner un élément.
- Ex. $personne=array(

  array('Peter',' 25'),

  array('Ben',' 30'));

Nom	Age
Peter	25
Ben	30

- Pour accéder à l'age de 'Peter', on utilise deux indices $personne[0][1]='25' ou deux boucles **for**

Quelques fonctions

- Count(): compte le nombre d'éléments d'un tableau (taille)
- array_reverse(): inverse les éléments d'un tableau
- array_unique(): supprime les doublons
- sort(): trie un tableau indexé dans l'ordre croissant des ses valeurs
- etc

- Ex. <?php
  $couleur = array("rouge", "bleu", "vert");
  echo count($couleur);
  ?>

- Résultat: 3

Tri des éléments d'un tableau

- Les éléments d'un tableau peuvent être triés par ordre alphabétique ou numérique, croissant ou décroissant.
- Les fonctions de tri sont:
  - **sort():** tableau indexé, ordre croissant de ses valeurs
  - **rsort()**: tableau indexé, ordre décroissant de ses valeurs
  - **asort()**: tableau associatif, ordre croissant de ses valeurs
  - **ksort()**:tableau associatif, ordre croissant de ses clés
  - **arsort()**: tableau associatif, ordre décroissant de ses valeurs
  - **krsort()**: tableau associatif, ordre décroissant de ses clés

Exemple 1

```
<?php
    $colors = array("red", "blue", "green");
    rsort($colors); // ordre alphabétique décroissant
    $taille = count($colors);
    for($x = 0; $x < $taille; $x++) {
    echo $colors[$x]; // affichage des éléments du tableau
    echo "<br>";
  }
?>
```

Résultat: red

green

blue

Exemple 2

- <?php
  $age = array("Peter"=>"35", "Ben"=>"37",
  "Joe"=>"43");
  asort($age);
  ?>

- Résultat:

  Key=Peter, Value= 35
  Key=Ben, Value= 37
  Key=Joe, Value= 43

Exemple 3

- <?php
  $age = array("Peter"=>"35", "Ben"=>"37",
  "Joe"=>"43");
  ksort($age);
  ?>
- Résultat ?

  Key=Ben, Value=37
  Key=Joe, Value=43
  Key=Peter, Value=35

Exemple 4

- <?php
  $age = array("Peter"=>"35", "Ben"=>"37",
  "Joe"=>"43");
  arsort($age);
  ?>
- Résultat ?

  Key=Joe, Value= 43
  Key=Ben, Value= 37
  Key=Peter, Value= 35

Exemple 5

- <?php
  $age = array("Peter"=>"35", "Ben"=>"37",
  "Joe"=>"43");
  krsort($age);
  ?>
- Résultat ?

  Key=Peter, Value=35
  Key=Joe, Value=43
  Key=Ben, Value=37

Formulaires

- GET et POST sont deux méthodes utilisées pour ***collecter*** les données de formulaire.
- **Ex**

```
<form method='POST' action='file.php'>
        Nom:<input type='text' name='name'>
        <input type='submit'>
</form>
```

- File.php

```
<?php
        echo 'le nom est: $_POST[name]';
?>
```

- $_POST[name] crée un tableau associatif contenant des **clés** et des **valeurs** (clé1=>valeur1, clé2=>valeur2 …)

  - Clé: nom d'un champ de formulaire
  - Valeur: valeur saisie par l'utilisateur

Get vs Post

- **GET**
  - Données **visibles** (noms et valeurs sont affichées dans l'URL)
  - Quantité de données à envoyer est **limitée** (2000 caractères)
  - Utilisée pour l'envoi des données **non sensibles**
- **POST**
  - Données sont **invisibles** pour les autres (noms et valeurs sont encapsulés dans le **corps** de la requête HTTP)
  - **Aucune** limite sur la taille des données à envoyer
  - Utilisée pour l'envoi des données **sensibles** comme les *mots de passes*.

NB: Il est donc préférable d'utiliser POST pour la soumission des données.