# Fonctionnalités du langage PHP

# I. Premiers éléments du langage

➢ **Syntaxe de base:**

- Le code doit être placé entre les balises <?php  et ?>
- Chaque instruction PHP doit se terminer par un point-virgule.

```
<?php
        echo 'bonjour';  //écriture du mot bonjour sur la page html
?>
```

➢ **Les Commentaires:**

```
<?php
        // un commentaire sur une ligne
        # un autre commentaire sur une ligne
        /* un commentaire
        sur plusieurs lignes*/
        $a=5+ /* +15 */ + 3;        // un commentaire au milieu d'une instruction
?>
```

# I. Premiers éléments du langage

➢ **Intégration de PHP dans une page HTML :**

Le code PHP doit être enregistré dans un fichier avec l'extension .php

**Exemple**

```
<html>
    <body>
        <h2 id="titre2" class="c2">Le texte est en HTML</h2> <br>
        <?php
                echo " <h2 id=\"titre2\" class=\"c2\">  Le texte est en PHP.</h2>";
        ?>
        <br>
        <h2 id="titre2" class="c2"> <?php echo 'Encore du texte en PHP' ; ?></h2>
    </body>
</html>
```

Le résultat obtenu sera :

**Le texte est en HTML**
**Le texte est en PHP**
**Encore du texte en PHP**

# II. Les fonctions include() et include_once()

- Il est possible d'inclure des fichiers externes (php, html, javascript,..) au moment de l'exécution du script.
- **Exemple:** un bloc HTML ou PHP répétitif qui apparaît dans toutes les pages de votre site (menu par exemple) pourra être isolé dans un fichier PHP. Un appel de ce fichier grâce à la fonction include() par exemple apparaîtra dans toutes les pages de votre site. Ainsi si le menu doit être par exemple modifié il suffira uniquement de changer le fichier contenant le menu.

**Fichier menu.php**

```
<a href= "menu1.php" > Menu1 </a><br>
<a href="menu2.php" > Menu2 </a><br>
<a href="menu3.php" > Menu3 </a><br>
```

**Fichier page.php**

```
<?php
    include ("menu.php") ; / /inclusion du fichier contenant le menu
?>
```

# II. Les fonctions include() et include_once()

- **La fonction include()**
  - Si le fichier n'est pas trouvé:
    - un simple avertissement sera affiché
    - Le script ne s'interrompt pas
  - Les fonctions et les classes appelées ont une portée globale
  - Les variables sont accessibles par le fichier appelé à partir du point d'inclusion
- **La fonction include_once()**
  - une seule inclusion dans un script sera faite
- **La fonction require()**
  - Identique à include()
  - Si le fichier appelé n'est pas trouvé→erreur + arrêt d'exécution du script appelant
- **La fonction require_once()**
  - une seule inclusion dans un script sera faite

# III. Variables, chaines et concaténation

- Une variable est définie sous la forme: **$variable_nom**
- Le nom commence par une lettre ou _ suivi par des lettres et/ou des chiffres
- le nom d'une variable est sensible à la casse
- Pas de typage lors de la déclaration
- Affectation: **$nom = valeur**

**Exemple:**

```
<?php
	$var1 = 'Bonjour 1'; //Affectation d'une chaîne
	$Var1 = "Bonjour 2"; // Affectation d'une chaîne
	$var3 = 5; // Affectation d'un entier
	$var4 = 2.5 + (3 * 5); // Affectation d'un résultat d'une opération (17.5)
	$var5 = true; // Affectation Booléenne
	$_var6="test"; // var6 vaut "test"
?>
```

# III. Variables, chaines et concaténation

- L'affichage des variables combinées à des chaînes de caractères peut se faire de plusieurs manières en utilisant les cotes simples ( ' ) ou les doubles cotes ( '' ).
- La concaténation se fait avec le caractère point (.)

**Exemple:**

```
<?php
        $nom = 'visiteur';
        echo "bonjour $nom";  //affiche bonjour visiteur
        echo 'bonjour '.$nom; /*affiche bonjour visiteur (Le point est un opérateur de
                                                    concaténation pour les chaînes de caractères*/
        echo 'bonjour $nom'; // Affiche bonjour $nom (La variable $nom ne sera pas interprétée)
?>
```

- **Remarque sur l'usage des cotes simples ou doubles**: Pour que PHP interprète les caractères ' et " en tant que caractères et non des délimiteurs de chaines, utiliser le caractère de déspécialisation antislash \

```
<?php
        echo 'vous n\'êtes pas inscrit';       //affiche Vous n'êtes pas inscrit
        echo "<a href=\"http://www-etu\">lien.php</a>";      //affiche Lien.php
?>
```

# III. Variables, chaines et concaténation

- **les types de données**
  - PHP gère implicitement les types de données (PHP est faiblement typé)
  - Les types supportés par PHP:
    - Entier: 5
    - Float: 5.6
    - Chaine: "bonjour"
    - Booléen: true(1)/false(0)
    - Tableaux: array()
    - Objets
    - NULL
- **Fonctions utiles**:
  - **gettype($var)** : retourne le type du contenu d'une variable
  - **settype($var,"type")**: définit le type d'une variable
  - **is_numeric(), is_string(), is_integer(), is_bool(), is_int(), is_long(), is_double(), is_flaot(), is_array(), is_scalar()**: teste le type du contenu d'une variable
  - ....

# III. Variables, chaines et concaténation

- **Modification (transtypage ) explicite de types de données:**
  - La modification de types en PHP fonctionne globalement de la même façon qu'en C : le nom du type désiré est écrit entre parenthèses avant la variable à traiter.

```
<?php
        $a = 10;              // $a est un entier
        $b = (boolean) $a;    // $b est un booléen
?>
```

  - Les préfixes autorisés sont :
    - (int), (integer) : modification en integer
    - (bool), (boolean) : modification en boolean
    - (float), (double), (real) : modification en float
    - (string) : modification en string
    - (array) : modification en array
    - (object) : modification en object
    - (unset) : modification en NULL (PHP 5)

# III. Variables, chaines et concaténation

- **Modification (transtypage ) implicite de types de données**
  - PHP définit implicitement le type de variable. Le type d'une variable est déterminé par son contexte d'utilisation.

```
<?php
	$var1 = "0";			// $var1 est une chaîne de caractères (ASCII 48)
	$var1 += 2;			// $var1 est maintenant un entier (2)
	$var1 = $var1 + 1.3;		// $var1 est maintenant un nombre à virgule flottante (3.3)
	$var1 = 5 + "10 petites fleurs"; // $var1 est un entier (15)
	$var1 = 5 + "10  petites roses";	// $var1 est un entier (15)
?>
```

# III. Variables, chaines et concaténation

- **Portée (scope) d'une variable:** PHP gère trois portée de variable: local, global et static.
- **Portée locale:** une variable déclarée dans une fonction ne sera visible que dans cette fonction.

```
<?php
      $y=8;
      function Test() {
         $x = 5; // variable locale
         echo "<p>La variable x dans la function est: $x</p>";
         echo "<p>La variable y dans la function est: $y</p>"; // l'utilisation de y dans la fonction
                                                                 // génère  une erreur
      }
      Test();

      // l'utilisation de x en dehors de la fonction génère une erreur
      echo "<p>La variable x en dehors de la function est: $x</p>";
?>
```

# III. Variables, chaines et concaténation

➢ **Portée globale (1):**

- une variable qui est déclarée dans un script (avant d'être utilisée) sera visible dans tout le script mais pas dans les fonction.

```
<?php
	$x = 5; // variable globale au script
	function Test() {
		// l'utilisation de x dans la fonction génère une erreur
		echo "<p>La variable x dans la function est: $x</p>";
	}
	Test();

	echo "<p>La variable x en dehors de la function est: $x</p>";
	// affiche La variable x dans  la function est: 5
?>
```

# III. Variables, chaines et concaténation

➢ **Portée globale (2):**

▪ L'utilisation du mot clé **global** dans une fonction permet de rendre une variable globale du script visible à l'intérieur d'une fonction, et de rendre une variable d'une fonction visible partout dans le script.

```
<?php
    $x = 5; // variable globale au script
    $y = 10; // variable globale au script
    function Test() {
       global $x, $y, $z;
       $y = $x + $y;
       $z = 8;
    }

    Test();
    echo $y; // affiche 15
    echo $z; // affiche 8
?>
```

# III. Variables, chaines et concaténation

➢ **Portée globale (3): utilisation du tableau $GLOBALS[]**

- PHP stocke également toutes les variables globales dans un tableau appelé **$GLOBALS[index]**.
- L'index contient le nom de la variable. Ce tableau est également accessible depuis les fonctions et peut être utilisé pour manipuler les variables globales directement.

```
<?php
      $x = 5;
      $y = 10;
      function Test() {
         $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
      }

      Test();
      echo $y; // affiche 15
?>
```

# III. Variables, chaines et concaténation

➢**Variable statique**

- Lorsqu'une fonction est terminée/exécutée, toutes ses variables sont supprimées.
- Si nous voulons qu'une variable locale ne soit pas supprimée (Nous en avons besoin pour un autre travail), on utilise le mot-clé **static** pour déclarer la variable.
- **Exemple:**

```
<?php
      function Test() {
         static $x = 0;
         echo $x;
         $x++;
      }
      Test();   // affiche 0
      Test();  // affiche 1
      Test();  // affiche 2
?>
```

# III. Variables, chaines et concaténation

➢ **Quelques fonctions utiles:**

- **isset($var)**: Détermine si une variable est définie et est différente de NULL
- **unset($var1,$var2,..)**: détruit la ou les variables dont le nom a été passé en argument
- **empty($var)**: teste si la variable $var  n'existe pas ou elle est vide (0,0.0,"",tableau vide,..)
- **trim($var)**: supprime les espaces à gauche et à droite d'une chaine
- **addslashes($var)**: rajoute \ devant ' et "
- **var_dump($var)**: affiche les informations d'une variable (très utile lors du test de scripts)
- **print_r($var)**: affiche le contenue d'une variable (très utile lors du test de scripts)
- ...

# III. Variables, chaines et concaténation

- **les constantes**
  - Les constantes utilisateur sont crée par "**define**" ou "**const**".
  - Exemple: **define('PI',3.14)**;  **const PI = 3.14;**
  - Non modifiables
  - Seules les valeurs scalaires (booléen, entier, double et chaine de caractères) peuvent être affectées à des constantes
- **Les constantes PHP**

Exemple de constantes prédéfinies en PHP:

- PHP_VERSION: version php sur le serveur
- PHP_OS: le système d'exploitation du serveur
- __FILE__: nom du fichier en cours d'exécution
- __LINE__: numéro de la ligne courante
- ...

# IV. Les opérateurs

- **Les opérateurs arithmétiques:** +, -,*,/,
- **Opérateur de concaténation**: $a="Numéro". $i
- **Les opérateurs d'assignation**: +=,-=,*=, /=, ++, --
- **Les opérateurs logique:**
  - == : strictement égal
  - != : différent
  - > : plus grand que
  - < : inférieur à
  - >= : supérieur à
  - <= : inférieur à
  - && : et logique
  - || : ou logique
  - AND : et
  - OR : ou
- **Les valeurs logique:**
  - TRUE : 1 ou oui
  - FALSE : 0 ou non

# V. Les structures conditionnelles et répétitives

## ➢La structure if

```
<?php
        if(condition)
                { …..}
        else
                {……}


        if (condition1)
                {……..……}
        elseif (condition2)
                {……..}
        …
        elseif (condition3)
                {…….}
        …
        else {…….…}
?>
```

# V. Les structures conditionnelles et répétitives

➢ **La structure if (version sans accolades)**

```
<?php
    if(condition):
        .
        .
        .
    else:
        .
        .
        .
    endif;
```

```
<?php
    if (condition1):
        .
        .
        .
    elseif (condition2):
        .
        .
        .
    elseif (condition3):
        .
        .
        .
    else:
        .
        .
        .
    endif;
?>
```

# V. Les structures conditionnelles et répétitives

## ➢La structure switch

```
<?php
    switch ($variable)
    {
    case condition1:
        //Traitement de la condition 1
        break;
    case condition2:
        //Traitement de la condition 2
        break;
        ….
    case conditionN:
        //Traitement de la condition N
        break;
    default:
        //Traitement par défaut
    }
?>
```

# V. Les structures conditionnelles et répétitives

## ➢La structure switch (version sans accolades)

```
<?php
    switch ($variable):
        case condition1:
        //Traitement de la condition 1
        break;
        case condition2:
        //Traitement de la condition 2
        break;
        ….
        case conditionN:
        //Traitement de la condition N
        break;
        default:
        //Traitement par défaut
    endswitch;
?>
```

# V. Les structures conditionnelles et répétitives

## ➢La structure while()

```
<?php
      while (condition)
      {
            //Traitements
      }


            //exemple
      $x = 1;
      while($x <= 5) {
         echo "Le nombre est: $x <br>";
         $x++;
      }
?>
```

```
<?php                    //version sans accolades

      while (condition):
            //Traitements
      endwhile;


            //exemple
      $x = 1;
      while($x <= 5):
         echo "Le nombre est: $x <br>";
         $x++;
      endwhile;
?>
```

# V. Les structures conditionnelles et répétitives

➢ **La structure do while()**

```
<?php
      do
      {
            //Traitements
      }
      while (condition);
?>
```

```
<?php                    //exemple
      $x = 1;
      do
      {
         echo  "Le nombre est : $x <br>";
         $x++;
      }
      while ($x <= 5);
?>
```

# V. Les structures conditionnelles et répétitives

➢ **La structures for()**

```
<?php
  for(instr. d'entrée; condition ; instr. de sortie)
  {
      //Traitements réalisés
  }

      //exemple
  for ($x = 0; $x <= 10; $x++) {
          echo  "Le nombre est: $x <br>";
      }


?>
```

```
<?php                    //version sans accolades

  for(instr. d'entrée; condition ; instr. de sortie):
      //Traitements réalisés
  endfor;

      //exemple
  for ($x = 0; $x <= 10; $x++):
          echo  "Le nombre est: $x <br>";
  endfor;
?>
```

# V. Les structures conditionnelles et répétitives

## ➢La structure foreach()

```
<?php
  foreach($tableau as $valeur)
  {
      //Traitements réalisés
  }

  //exemple
  $colors= array("red", "green", "blue", "yellow");


      foreach ($colors as $value) {
         echo "$value <br>";
      }
?>
```

```
<?php                    //version sans accolades
  foreach($tableau as $valeur):
      //Traitements réalisés
  endforeach;

  //exemple
  $colors = array("red", "green", "blue", "yellow");

      foreach ($colors as $value):
         echo "$value <br>";
      endforeach;
?>
```

# VI. Les fonctions

## ➢ Définition et appels des fonctions

```
<?php
      function nom_de_la_fonction ([$paramètres])
      {
            //traitement
            return (valeur); //optionnlle
      }
            // Appel de la fonction :
     nom_de_la_fonction ($paramètres) ;
?>
```

**Exemple1:**

```
<?php
      function afficher ($nom,$prenom) { echo 'Bonjour '.$nom. ' '.$prenom ; }
      afficher ('Mohamed','Ali') ; // Bonjour Mohamed Ali
      $nom1='Salma' ;
      $prenom1='Zouari' ;
      afficher ($nom1,$prenom1) ;// Bonjour Salma Zouari
?>
```

# VI. Les fonctions

**Exemple2:**

```
<?php
	function additionner ($variable1,$variable2)
	{
		$total = $variable1 + $variable2;
		return ($total) ;
	}
	$resultat= additionner (1,2) ; //affiche 3
	echo $resultat.' <br>' ;
	$var1=6 ;
	$var2=7 ;
	$resultat= additionner ($var1,$var2) ;
	echo $resultat.' <br>' ; // affiche 13
?>
```

➢**Fichier de fonctions**: Si les fonctions sont créées dans un fichier externe, ce fichier sera appelé à l'intérieur des autres fichiers par le biais de la fonction include (ou include_once ou require ou require_once).

# VII. Les tableaux

Il existe deux type de tableaux de variables sous PHP : Tableaux numérotés et les tableaux associatifs.

- **Les tableaux à index numériques (tableaux numérotés)**:
  - l'accès à la valeur de la variable passe par un index numérique.
  - Déclaration: **$tableau = array (valeur0,valeur1,valeur2, …,valeurN) ;**
  - Accès:
    $tableau[0] donnera valeur0
    $tableau[1] donnera valeur1
  - Exemple: $tab[0] est la valeur du premier élément du tableau $tab
- **Les tableaux à index associatifs (ou tableaux associatifs)**:
  - l'accès à la valeur de la variable passe par un index nominatif
  - Déclaration: **$tableau = array (variable1 => valeur1, variable2 => valeur2, …) ;**
  - Accès :
    $tableau[variable1] donnera valeur1
    $tableau[variable2] donnera valeur2
  - Exemple : $tableau[nom], $tableau[prénom], $tableau[adresse] ou nom, prénom et adresse sont soit des variables string soit des chaines constante.

# VII. Les tableaux

➢ **Exemples**:

```
<?php
    //Tableau à index numéroté
    $tableau1 = array ('château','maison','bateau') ;
    echo "Contenu du tableau 1 :<br>";
    echo $tableau1[0]."<br>";
    echo $tableau1[1]."<br>";
    echo $tableau1[2]."<br>";
    //Tableau à index associatif
    $tableau2 = array ('prenom' =>Med','nom' =>'Ali','ville' =>'Tunis') ;
    echo "Contenu du tableau 2 :<br>";
    echo $tableau2['prenom']"<br>";
    echo $tableau2['nom']."<br>";
    echo $tableau2['ville']"<br>";
?>
```

Contenu du tableau 1 :
château
maison
bateau

Contenu du tableau 2 :
Med
Ali
Tunis

# VII. Les tableaux

➢ **Parcours des tableaux**:

- **Avec for**

  $tableau = array (valeur0,valeur1,valeur2, …) ;

  for($i=0;$i<count($tableau);$i++) {……}

- **Avec foreach**

  $tableau = array (valeur0,valeur1,valeur2, …) ;

  **foreach ( $tableau as $valeur ) { //**Appeller ici la valeur courante par $valeur …**}**

**Exemple:**

```
<?php
      $tableau1 = array ('chateau','maison','bateau') ;
      foreach ( $tableau1 as $valeur ) { echo $valeur."<br>"; }

      $tableau2 = array ('prenom' =>'Med','nom' =>'Ali','ville' =>'Tunis') ;
      foreach ( $tableau2 as $valeur ) { echo $valeur."<br>"; }
?>
```

# VII. Les tableaux

**Exemple 2:** Utiliser le nom de variable et la valeur pour un tableau associatif

```
<?php
	$tableau2 = array ('prénom' =>'Med','nom' =>'Ali','ville' =>'Tunis') ;
	foreach ( $tableau2 as $variable=>$valeur )
	{
		echo $variable." a pour valeur ".$valeur."<br>";
	}
?>
```

Le résultat obtenu sera :

prénom a pour valeur Med
nom a pour valeur Ali
ville a pour valeur Tunis

# VII. Les tableaux

➢ **Quelques fonctions sur les tableaux :**

- **count($tab)**: retourner la taille d'un tableau
- **array_key_exists($var, $tab)** : permet de vérifier un tableau associatif contient une variable associative (clef). (retourne true ou false).
- **in_array(value,$tab)**: retourne si un tableau contient une valeur
- **array_search($val,$tab)**: recherche une valeur dans un tableau et retourne:
  - la position (indice) si c'est un tableau numéroté
  - le nom de la variable si c'est un tableau associatif
  - false si non la valeur n'est pas trouvée.
- **sort($tab)**: trie croissant d'un tableau numéroté
- **rsort($tab)**: trie décroissant d'un tableau numéroté
- **asort($tab)**: trie croissant sur les valeurs d'un tableau associatif
- **arsort($tab)**: trie décroissant sur les valeurs d'un tableau associatif
- **ksort($tab)**: trie croissant sur les indices (keys) d'un tableau associatif
- **krsort($tab)**: trie décroissant sur les indices (keys) d'un tableau associatif
- **array_sum($tab)**, **array_product($tab)**: retournent respectivement la somme et le produit des valeurs d'un tableau.

# VIII. Les chaines de caractères

- Une chaine de caractère est une suite de caractères délimitée par " ou '
- Elle est aussi vue en tant qu'un tableau de caractères
- Les caractères spéciaux en PHP doivent être précédés par \: \" , \\, \', \n,\r,\t

➢ **Quelques fonctions associées aux chaines de caractères**:

- **strlen(chaine)**: longueur de la chaine
- **substr(chaine,debut,[taille])**: extraire une sous-chaine spécifiée avec début et taille
- **trim(chaine)**: supprime les espaces blancs (" ", \n, \r, \t, \v, \0) de début et de fin d'une chaine et retourne une chaine nettoyée.
- **implode(separteur,array)**: retourne une chaine constituée de tous les éléments du tableau et séparé par séparateur.
- **explode(separteur,chaine)**: retourne le tableau qui contient les élément de la chaine séparés par séparateur.
- **str_replace(modèle,remplacement,chaine)**: remplace toutes les occurrences de modèle par remplacement dans chaine

# VIII. Les chaines de caractères

- **strtolower($str)**: convertit une chaine en minuscule
- **strtoupper($str)**: convertit une chaine en majuscule
- **ucfirst($str)**: convertit le premier caractère de la chaine en majuscule
- **ucwords($str)**: convertit le premier caractère de chaque mot de la chaine en majuscule
- **strcmp($str1,str2)** et **strncmp($str1,str2,n)**: comparaison de deux chaines et d'une partie de deux chaines
- **strstr($str1,str2)**:retourne la première occurrence de $str1 dans $str2
- **ord($char)**: retourne le code ASCII du caractère $char
- **strrev($str)**: retourne la chaine $str lue à l'envers
- **…**