IIT
INSTITUT INTERNATIONAL TECHNOLOGIE
Université Nord Américaine Privée

Programmation web II

Mohamed Karim Abdmouleh
karim.abdelmoula@iit.ens.tn

1ère année Génie Informatique

(Version 0)

Organisation du cours

Objectif général	Développer des pages Web **interactives** et **dynamiques**
Volume horaire	28 heures
Méthode pédagogique	Méthode active participative / Apprentissage par la pratique / Mini projet
Date de démarrage du module	25 janvier 2025
Date de la fin du module	10 mai 2025
Grille d'évaluation	Comptes rendus : **10**% de la note Mini-projet ou DS : **20**% de la note Examen : **70**% de la note

Objectifs

- Concevoir et développer un site web interactif et dynamique
- Manipuler le langage PHP côté serveur
- Gérer les formulaires HTML avec PHP
- Manipuler la base de données MySQL
- Utiliser les sessions et les cookies avec PHP
- Se familiariser avec l'orienté objet en JavaScript côté client
- Comprendre la notion du DOM et utiliser les éléments et les nœuds du DOM

Chapitre 1 : Notions de base du langage PHP

Introduction

- PHP est un langage open source créé en **1995** par **Rasmus Lerdorf**. Le nom vient de **Personal Home Page** Tools – un ensemble de scripts utilisés par Rasmus pour suivre les visites sur son site.
- Avec le lancement de PHP 3.0, le langage a obtenu un acronyme inversé : **PHP : Préprocesseur Hypertexte**. De nos jours, il est simplement connu sous le nom de PHP.
- Selon les données de **W3Techs** [2024], PHP est utilisé par **77,4% de tous les sites web** . Ainsi, près de 8 sites Web sur 10 que vous visitez sur Internet utilisent PHP d'une manière ou d'une autre.
- **WordPress**, le système de gestion de contenu le plus populaire qui existe, utilise PHP.

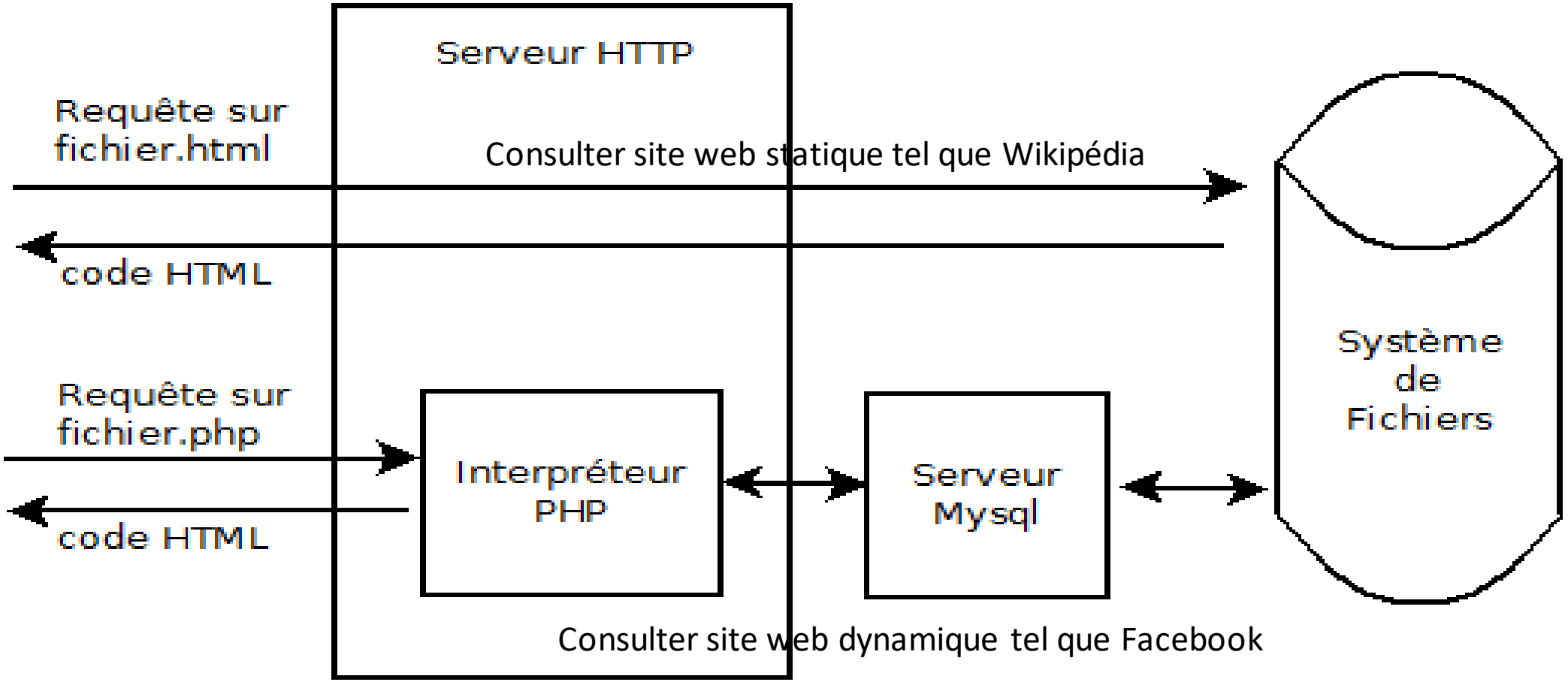
Introduction

- Les sites les plus connus comme Facebook , Wikipedia ou Youtube sont en langage PHP. Tout comme les Plateformes Ecommerce Magento ou Prestashop. Ainsi que les CMS les plus utilisés comme Drupal, Joomla ou **WordPress** (25% des sites web dans le monde). **Tous, sous PHP**, pour **produire les pages web dynamiques** que nous consultons tous les jours.
- ***une page web dynamique*** *est une page dont le contenu varie à chaque fois qu'elle est générée, par opposition à une page web statique*

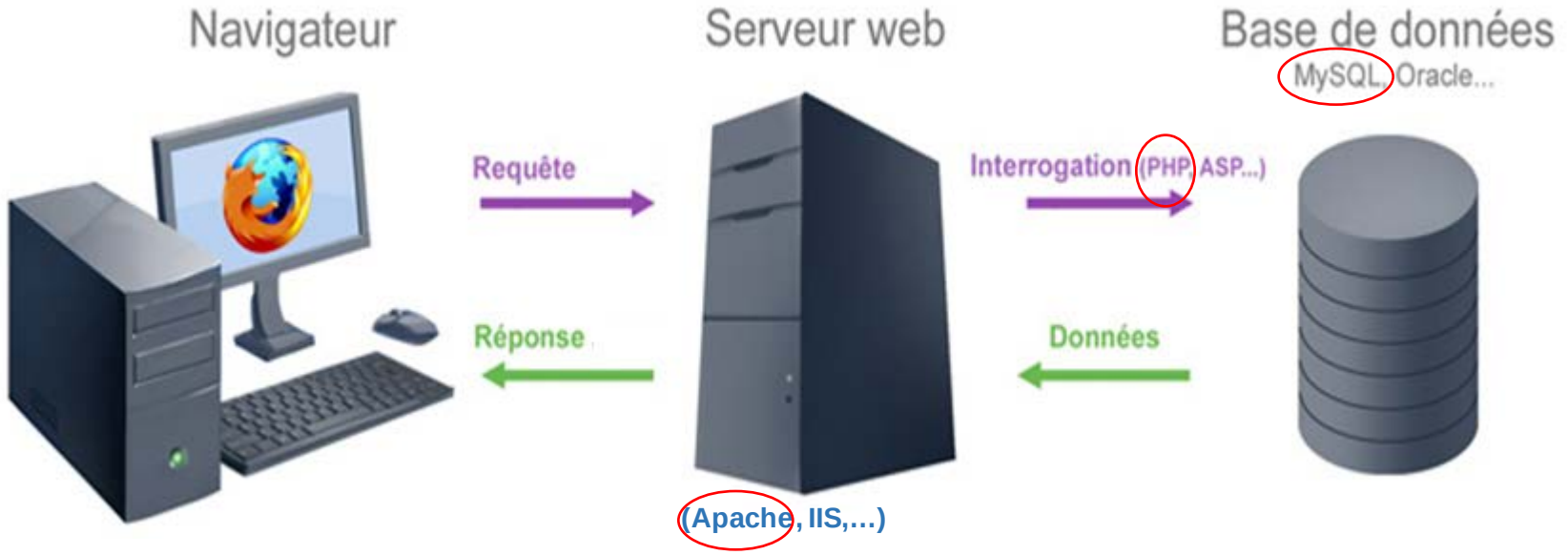
Introduction

- Dernière version PHP 8.4.3 (16 janvier 2025)
- Paradigmes : orienté objet, procédural
- Site web officiel : https://www.php.net/
- Site web recommandé : https://www.w3schools.com/php/default.asp
- Balise : Le code PHP doit être inséré entre les balises **<?php** et **?>**
- Où écrire PHP ? peut être utilisé dans différentes parties d'un document HTML:
  - Dans le <head>,
  - Dans le <body>
  - Ou en dehors du <html>

traitement d'une requête côté serveur (2 cas)



Web dynamique



les autres langages ? les concurrents ?

- **C# (ASP.net)**
- **Java (spring)**
- **Ruby (Rails)**
- **Javascript (nest, astro)**
- **python (django, flask)**
- **Etc.**

Remarque : Ne confondez pas **langage** et **framework**

Le laravel est un framework php pour le back-end

Le node.js est une plateforme de développement des applications web qui permet d'exécuter du code JavaScript côté serveur.

Express.js, Nest.js, Hapi.js, etc. ce sont des framework js backend

React bibliothèque js frontend

Écosystème PHP

- **Visual Studio Code (**gratuit et Open Source**):** est un éditeur pour ecrire du code php.
- **Xampp :** est un ensemble de logiciels (tels que Apache et MySQL) permettant de mettre en place un serveur Web local. Grace à Xampp, on peut executer du code php.

- **PhpStorm** est un éditeur offrant énormément de fonctionnalités (vérification du code, navigation intelligente, autocomplétion, refactorisation, débogage etc). Malheureusement, cet éditeur nécessite une licence (environ 200€ la première année, 160€ la seconde et 120€ les suivantes).

Écosystème PHP

- **Les frameworks**
  - Les principaux frameworks PHP sont **Symfony** (2005), **CakePHP** (2005), **Laravel** (2011) et **CodeIgniter** (2006).

  Avantages

  - beaucoup moins de code à écrire (plein de fonctionnalités sont déjà codées de manière optimisées),
  - avoir un cadre solide pour travailler à plusieurs grâce à une architecture recommandée (chaque développeur ne fait pas son architecture dans son coin au risque d'avoir de graves problèmes de maintenabilité)
  - avoir une meilleure sécurité (de nombreuses configurations empêchant diverses attaques sont déjà incluses).

Symfony

Laravel

CodeIgniter

CakePHP

Récap

logo

- **PHP : Hypertext PreProcessor.**
- c’est un langage de back-end
- c’est un langage de programmation ***Web dynamique***.
- PHP est un langage de **script serveur** conçu spécifiquement pour agir au niveau des serveurs web.
- PHP est un langage ***Open Source***
- PHP est totalement gratuit.
- PHP est très majoritairement installé sur un serveur **Apache**, mais peut être installé sur les autres principaux serveurs HTTP du marché, par exemple IIS.
- C'est un langage peu typé, souple et facile à apprendre.
- L'accès aux bases de données est possible une fois l'installation des modules correspondants est effectuée sur le serveur.
- Communauté Active

Script PHP

- Un script php a la structure suivante :

  <?php ……?>

- Les fichiers php ont l'extension **.php**

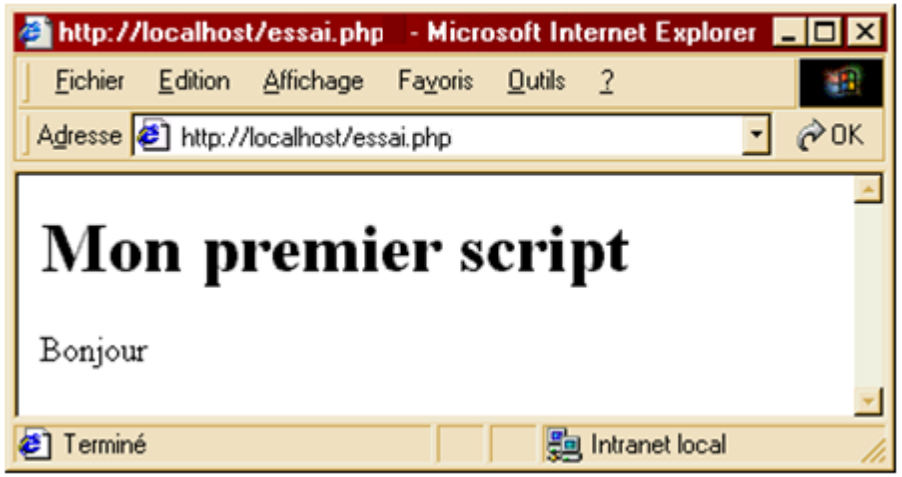
Exemple de script, code source (côté serveur) :

```
<html>
<body>
<h1>Mon premier script</h1>
<?php echo ''Bonjour\n''; ?>
</body>
</html>
```

Résultat d'exécution par le serveur

```
<html>
<body>
<h1>Mon premier script</h1>
Bonjour
</body>
</html>
```

Résultat affiché par le navigateur :



Procédure d'exécution:
Lorsque le serveur web reçoit une demande pour un fichier PHP, il **exécute le code PHP** à l'intérieur de celui-ci et **génère du contenu HTML** à envoyer au **navigateur**. Le HTML est alors interprété et affiché par le navigateur.

Différence entre les fichiers .html et .php

- **L'extension .php** : Le serveur web exécutera le code PHP dans le fichier, puis renverra le résultat sous forme de HTML au navigateur.
- **L'extension .html** : si on utilise uniquement du HTML et aucun code PHP dans le fichier, on peut utiliser l'extension .html. Dans ce cas, le serveur web ne traitera pas le fichier avec le **moteur PHP** et renverra simplement le code HTML au navigateur.
- Dans un fichier PHP, on inclut des balises HTML car le langage PHP est conçu pour générer du contenu **HTML dynamiquement et pour styliser ce contenu**. Cela peut inclure l'utilisation de feuilles de style CSS, de balises de mise en forme, de balises de liste, de tableaux et plus encore.

La question qui se pose

- Comment j'exécute un fichier .php ?
- Dans un environnement de développement local, on met les fichiers PHP dans le répertoire racine du serveur web, qui est généralement appelé "htdocs" pour Apache ou "wwwroot" pour IIS.

On trouve **htdocs** sous C:/xampp.

- pour **exécuter un fichier php,** il faut tout d'abord **démarrer le serveur** Apache à partir de son xampp control panel et puis accéder au contenu du serveur (les fichiers .php) :

il y a 2 façons pour accéder au serveur

- Soit en mettant localhost dans le navigateur et puis le chemin du fichier à exécuter
- Soit en mettant 127.0.0.1 puis le chemin du fichier à exécuter

NB: **localhost** équivalent à **C:/xampp/ htdocs**

Le port de l'écoute **80**

- Le **problème majeur** rencontré au moment du démarrage du serveur est un problème **d'utilisation de port**
- Apache utilise le port de l'écoute **80** (par défaut) qui parfois peut être utilisé par une autre application.
- Les solutions

1. arrêter les services qui écoutent sur le même port (tels que le service Oracle listener et le service virtual box) ;

Comment ? on écrit **services** dans la zone de recherche du menu démarrer puis on cherche, par exemple, le service de Oracle listener. **click droit sur le service** et on clique sur « **Arrêter** ».

2. changer le numéro de port.

Comment ? Sur la fenêtre de Xampp Control Panel, cliquer sur le bouton config de Apache et choisir le fichier Appache (httpd.conf), chercher le port 80 (il existe dans 3 emplacements) et puis changer-les avec un numéro de port de votre choix. Avec cette solution il faut indiquer le numéro de port dans l'exécution de chaque fichier .php.

On écrit, par exemple, localhost :82/etudiant.php

Vérification du port avec la commande "netstat« ou "netstat -an"

Installation

Ajouter dans VSCode les extensions suivantes:

- PHP IntelliSense
- Auto Complete Tag
- Php Auto Completion
- Html Auto Completion

Puisse je inspecter le code PHP d’un site web?

- Non, il n'est pas possible d'inspecter le code source PHP d'un site web à partir du navigateur.
- Le code PHP est exécuté côté serveur, et le navigateur ne reçoit que le résultat HTML généré par l'exécution du code PHP.
- On peut voir visualiser que le code source HTML, les feuilles de style CSS et le code JavaScript qui sont envoyés au navigateur
- La sécurité des applications web repose en partie sur le fait que le code source du serveur, y compris le code PHP, est confidentiel et ne peut pas être consulté par des utilisateurs externes.

Etapes d'exécution

- Démarrer Apache
- Accéder à C:/xampp/htdocs et supprimer le fichier index.
- Créer un dossier « TPCours » sous C:/xampp/htdocs
- Ouvrir le dossier « TPCours » avec VSCode et créer un premier fichier page.php
- Ouvrir le navigateur et écrire dans la barre d'adresse : localhost
- Le navigateur affiche le contenu du serveur
- Aller à localhost/TPCours/page.php

La fonction phpinfo()

- **phpinfo()** est une fonction PHP qui affiche des informations concernant la **configuration PHP** de votre site, y compris :
- La version actuelle de PHP de votre site est en cours d'exécution.
- Les informations et l'environnement de votre serveur.
- L'environnement PHP.
- Les informations relatives à la version de votre système d'exploitation (OS).
- les chemins, y compris l'emplacement de php.ini.
- Valeurs maître et locales pour les options de configuration PHP.
- En-têtes HTTP.
- La licence PHP.
- Modules et extensions actuellement utilisés.

phpinfo()

- **Etape 1 : Créer un fichier phpinfo.php et le téléverser sur le serveur**

```
<?php

phpinfo( );

?>
```

- **Etape 2 : Accéder à la page phpinfo dans le navigateur**
- Une fois téléversée sur votre serveur, votre page **phpinfo** sera accessible au public. Cela signifie que vous (et n'importe qui d'autre) pouvez la voir dans un navigateur en ajoutant */phpinfo.php* à la fin du domaine de votre site :

Les commentaires

Un script php se commente comme en C.

***Exemple** :*

```
<?php
// commentaire de fin de ligne
/* commentaire
sur plusieurs
lignes */
# commentaire de fin de ligne comme en Shell
?>
```

Les variables

- ***Le typage des variables est implicite en php*** : il n'est donc pas nécessaire de déclarer leurs types au préalable ni même de les initialiser avant leurs utilisations.

- Les identificateurs de variable sont précédés du symbole « **$** » (dollars).

Exemple :

```
$var=12;
```

- Les variables peuvent être de type entier (**integer**), réel (**double**), chaîne de caractères (**string**), tableau (**array**), objet (**object**), booléen (**boolean**).

Les constantes

- L'utilisateur peut définir des constantes dont la valeur est fixée une fois pour toute.

- Les constantes ne portent pas le symbole **$** (dollars) en début d'identificateur et ne sont pas modifiables.

***Syntaxe* :**

define("cte",val) : définit la constante **cte** de valeur **val**

L'affichage du texte

echo() : écriture dans le navigateur

Exemples :

```
echo "Bonjour $name";
```

Les doubles quotes permettent l'évaluation des variables et caractères spéciaux contenus dans la chaîne alors que les simples ne le permettent pas.

Exemples :

```
$prenom= 'Ali';
echo "Prenom: $prenom";          // affiche Prenom: Ali
echo 'Prenom: $prenom';                    // affiche Prenom: $prenom
```

Structures conditionnelles

- **L'instruction if : if (condition réalisée) { liste d'instructions }**

- **L'instruction if … else : if (condition réalisée) { liste d'instructions }**
  else { autre série d'instructions }

- **L'instruction if … elseif … else :**

```
if (condition réalisée) { liste d'instructions }
    elseif (autre condition ) { autre série d'instructions }
        else { série d'instructions }
```

- **Opérateur ternaire : (condition) ? instruction si vrai : instruction si faux**

- **L'instruction switch :**

```
switch (Variable) {
case Valeur1: Liste d'instructions; break;
case Valeur2: Liste d'instructions; break;
case Valeurn: Liste d'instructions; break;
default: Liste d'instructions break;
}
```

Application 1

- Rédiger une expression conditionnelle pour tester si un nombre est à la fois un multiple de 3 et de 5.
- Solution :

```
<?php
$x=1245;
if($x%3==0 AND $x%5==0)
{
echo "$x est multiple de 3 et de 5 <br />";
}
else
{
echo "$x n'est pas multiple de 3 et de 5 <br />";
}
?>
```

Structures de boucle

- **La boucle for**
  for (condition) {bloc d'instructions ;}

- **La boucle while**
  while(condition) {bloc d'instructions ;}
  while (condition) :Instruction1 ;Instruction2 ;
  …. endwhile ;

- **La boucle do…while**
  do {bloc d'instructions ;} while(condition) ;

- **La boucle foreach**
  foreach ($tableau as $valeur) {insts utilisant $valeur ;}
  Exemple

```
$tab=array(1,2,3);
foreach ($tab as $val) {   $val = $val * 2; }
```

Application 2

- Effectuer une suite de tirages de nombres aléatoires (à l'aide de la fonction rand()) jusqu'à obtenir une suite composée d'un nombre pair suivi de deux nombres impairs.
- Chaque tentative doit être affiché comme suit :

194, 285, 494
435, 759, 162
237, 292, 768
366, 533, 397
Résultat obtenu en 4 coups

Utilisation de rand() : **rand(min, max)**

```
$x=rand(0,1000) génère un nombre de 0 à 1000.
 rand(100, 1000) générera un nombre aléatoire entre 100 et 1000 inclus.
```

Solution

```
<?php
$compteur=0;
do
{
$x=rand(100,999);
$y=rand(100,999);
$z=rand(100,999);
$compteur++;
echo "$x, $y, $z <br>";}
while($x%2==1 OR $y%2==0 OR $z%2==0);
//Ou   while (!($x%2==0 AND $y%2==1 AND $z%2==1));
echo "Résultat obtenu en $compteur coups";
?>
```

Application 3

- Choisir un nombre de trois chiffres.
- Effectuer ensuite des tirages aléatoires et compter le nombre de tirages nécessaire pour obtenir le nombre initial. Arrêter les tirages et afficher le nombre de coups réalisés.
- Réaliser ce script d'abord avec l'instruction **while** puis avec l'instruction **for**.

Solution - Avec une boucle while

```
<?php
//Nombre à trouver
$nb=789;
//compteur
$coup=0;
$x=0
//boucle  de tirage
while($x!=$nb)
{
 echo $x."<br />"; //pour afficher tous les tirages
$x=rand(100,999);
 $coup++;
}
echo "$nb trouvé en $coup coups ";   ?>
```

Solution - Avec une boucle for

```
<?php
//Nombre à trouver
$nb=789;
$x=0;
//boucle de tirage
for($coup=0;$x!=$nb;$coup++)
{
 $x=rand(100,999);
 echo $x."<br />";//pour afficher tous les tirages
}
echo "$nb trouvé en $coup coups ";
?>
```

Les tableaux

PHP supporte deux types de tableaux : **indexé** et **associatif**.

Un tableau peut être initialisé avec la syntaxe **array**.

Tableaux indexés :

- Les tableaux indexés utilisent des indices numériques (entiers) pour accéder aux éléments.
- Les indices commencent généralement à zéro (0) et augmentent de manière séquentielle.
- Les éléments sont accessibles par leur position dans le tableau.

```
$tableauIndexe = array("valeur1", "valeur2", "valeur3");
echo $tableauIndexe[1]; // affiche "valeur2"
```

Les tableaux

- ***Exemple :***

  $tab_colors = array('red', 'yellow', 'blue', 'white');

  $tab = array('Ali', 2002, 20.5, $name);
- Mais il peut aussi être initialisé au fur et à mesure.

Exemples :

$prenoms[0] = "Ali";

$prenoms[1] = "Salah";

$prenoms[2] = "Mohamed";

- L'appel d'un élément du tableau se fait à partir de son indice (dont l'origine est zéro comme en C).

Exemple :

echo $prenoms[1] ; *// pour afficher* **"Salah"**

Les tableaux

Tableaux associatifs:

- Les tableaux associatifs utilisent des clés (noms) pour accéder aux éléments.
- Chaque élément du tableau est associé à une clé unique.
- Les clés peuvent être des chaînes de caractères ou d'autres types, mais elles ne sont pas nécessairement numériques ni séquentielles.

```
$tableauAssociatif = array("cle1" => "valeur1", "cle2" => "valeur2", "cle3" => "valeur3");
echo $tableauAssociatif["cle2"]; // affiche "valeur2"
```

Les tableaux

Exemple

```
// Déclaration d'un tableau associatif
$informationsPersonne = array( "nom" => "Doe", "prenom" => "John", "age" => 25, "ville" => "Paris");

// Accès aux éléments du tableau associatif
echo "Nom : " . $informationsPersonne["nom"] . "<br>";
echo "Prénom : " . $informationsPersonne["prenom"] . "<br>";
echo "Âge : " . $informationsPersonne["age"] . "<br>";
echo "Ville : " . $informationsPersonne["ville"] . "<br>";
```

Les tableaux

Dans certains cas, un tableau peut également être à la fois indexé et associatif. Par exemple :

```
$tableauMixte = array(0 => "valeur1", "cle2" => "valeur2", 2 => "valeur3");
echo $tableauMixte[0];      // affiche "valeur1"
echo $tableauMixte["cle2"]; // affiche "valeur2"
echo $tableauMixte[2];      // affiche "valeur3"
```

Parcours d'un tableau

- **Utilisation de la structure for pour parcourir un tableau indexé**

```
$tabIndexe = array("valeur1", "valeur2", "valeur3");
$longueur = count($tabIndexe);
for ($i = 0; $i < $longueur; $i++) { echo $tabIndexe[$i] . "<br>"; }
```

- **Utilisation de la structure foreach pour parcourir un tableau indexé**

```
$tabIndexe = array("valeur1", "valeur2", "valeur3");

foreach ($tabIndexe as $valeur) {
   echo $valeur . "<br>";
}
```

- **Utilisation de la structure foreach pour parcourir un associatif**

```
$tabAssociatif = array("cle1" => "valeur1", "cle2" => "valeur2", "cle3" => "valeur3");
foreach ($tabAssociatif as $cle => $valeur) { echo "Clé: $cle, Valeur: $valeur <br>"; }
```

Exemple

```
<?php

$fruits = array("Pomme", "Banane", "Orange",
"Fraise");

foreach ($fruits as $fruit) {
   echo $fruit . "<br>";
}
?>
```

Donne le résultat suivant au navigateur :

Pomme
Banane
Orange
Fraise

Exemple

```php
<?php

// Définition d'un tableau associatif
$personne = array(
    "nom" => "Doe",
    "prenom" => "John",
    "age" => 30,
    "ville" => "Paris"
);

// Utilisation de la boucle foreach pour parcourir le tableau associatif
foreach ($personne as $cle => $valeur) {
    echo $cle . ": " . $valeur . "<br>";
}

?>
```

Donne le résultat suivant au navigateur :

nom: Doe
prenom: John
age: 30
ville: Paris

Afficher un tableau dans un tableau HTML

```
<!DOCTYPE html>
<html lang="en">
<body>
<?php
$tab= array("1"=>"un",  "2"=>"deux", "3"=>"trois");
?>
<table border="1">
   <thead>
      <tr>
         <th>En chiffre</th>
         <th>En lettre</th>
     </tr>
   </thead>
 <?php
foreach ($tab as $chiffre => $lettre){
         echo "<tr>";
         echo "<td>" . $chiffre . "</td>";
         echo "<td>" . $lettre. "</td>";
echo "</tr>";
      }
      ?>
</table>
</body>
</html>
```

En chiffre	En lettre
1	un
2	deux
3	trois

Afficher un tableau dans un tableau HTML

```
<!DOCTYPE html>
<html lang="en">
<body>
<?php
// Définition d'un tableau indexé
$personnes = array(
   array("nom" => "Doe", "prenom" => "John", "age" => 30, "ville" =>
"Paris"),
   array("nom" => "Smith", "prenom" => "Jane", "age" => 25, "ville" =>
"New York"),
   array("nom" => "Brown", "prenom" => "Mike", "age" => 35, "ville" =>
"London")
);
?>
<table border="1">
   <thead>
      <tr>
         <th>Nom</th>
         <th>Prénom</th>
         <th>Âge</th>
         <th>Ville</th>
      </tr>
   </thead>
```

```
<?php
    // Utilisation de la boucle foreach pour parcourir le tableau indexé
    foreach ($personnes  as $personne) {
       echo "<tr>";
       echo "<td>" . $personne["nom"] . "</td>";
       echo "<td>" . $personne["prenom"] . "</td>";
       echo "<td>" . $personne["age"] . "</td>";
       echo "<td>" . $personne["ville"] . "</td>";
       echo "</tr>";
    }
    ?>
</table>
</body>
</html>
```

Nom	Prénom	Âge	Ville
Doe	John	30	Paris
Smith	Jane	25	New York
Brown	Mike	35	London

Les tableaux

Quelques fonctions

- **count($tab)**, **sizeof** : retournent le nombre d'éléments du tableau.
- **in_array($var,$tab)** : dit si la valeur de **$var** existe dans le tableau **$tab**.
- **list($var1,$var2…)** : transforme une liste de variables en tableau.
- **sort($tab)** : trie alphanumérique les éléments du tableau.
- **rsort($tab)** : trie alphanumérique inverse les éléments du tableau.
- **array_merge($tab1,$tab2,$tab3…)** : concatène les tableaux passés en arguments.

Exemples de fonctions Mathématiques

- **rand([$x[,$y])** ou **mt_rand([$x[,$y])** : valeur entière aléatoire entre 0 et RAND_MAX si x et y ne sont pas définis, entre x et RAND_MAX si seul x est défini, entre x et y si ces deux paramètres sont définis.
- **ceil($x)** : arrondi supérieur.
- **floor($x)** : arrondi inférieur.
- **max($a, $b, $c …)** : retourne l'argument de valeur maximum.

Application 4

- Créer un tableau dont les indices varient de 11 à 36 et dont les valeurs sont des lettres de A à Z.
- Lire ensuite ce tableau avec une boucle **for** puis une boucle **foreach** et afficher les indices et les valeurs (la fonction **chr(n)** retourne le caractère dont le code ASCII vaut n).

Solution

Les chaînes de caractère

- Une variable chaîne de caractères n'est pas limitée en nombre de caractères. Elle est toujours délimitée par des **simples** quotes ou des **doubles** quotes.

Exemples :

$nom = "Ben Mohamed";

$prenom = 'Mohamed';

- ***Quelques caractères spéciaux*** : **\n** (nouvelle ligne), **\r** (retour à la ligne), **\t** (tabulation horizontale), **** (antislash), **\$** (caractère dollars), **\ "** (double quote).

Exemple :

echo "Hello Word !\n";

- ***Opérateur de concaténation de chaînes :*** **.** (point)

Les chaînes de caractère (suite)

- Quelques fonctions :

- **strlen($str)** : retourne le nombre de caractères d'une chaîne.
- **strtolower($str)** : conversion en minuscules.
- **strtoupper($str)** : conversion en majuscules.
- **trim($str)** : suppression des espaces de début et de fin de chaîne.
- **substr($str,$i,$j)** : retourne une sous chaîne de **$str** de taille **$j** et débutant à la position **$i.**
- **preg_match("/*pattern*/", $str)** : teste l'existence du ***pattern*** dans la chaîne **$str**.

NB : On peut ajouter le "**i**" après le délimiteur du pattern de la fonction **preg_match** pour indiquer que la recherche ne sera pas sensible à la casse.

Exemple :

```
$adresse="Rte Mharza km1 SFAX";
if (preg_match("/sfax/i", $adresse))          echo "Vous habitez Sfax. ";
```

- Les patterns peuvent être très complexes et contenir des caractères spéciaux.

Application 5

1. Transformez une chaîne écrite dans des casses différentes afin que chaque mot ait une initiale en majuscule.

Utiliser la fonction **ucwords**(); qui permet d'avoir des majuscules au début de chaque mot

ucwords("boNjouR iiT"); donne Bonjour Iit

2. En utilisant la fonction strlen() écrivez une boucle qui affiche chaque lettre de la chaîne "PHP 8" sur une ligne différente.

Solution

```
<?php
$ch="TransFOrmeZ unE ChaîNE écRITe  dans des cASses diFFéreNTes  afiN qUe chAQue MOT ait une inITiale en MAJUSCULE";
```

Les expressions régulières

Les caractères spéciaux (suite)

- **[abcdef]** : intervalle de caractères, teste si l'un d'eux est présent.
- **[a-f]** : plage de caractères : teste la présence de tous les caractères minuscules entre **'a'** et **'f'**
- **[^0-9]** : exclusion des caractères de **'0'** à **'9'**.
- **\^** : recherche du caractère **'^'** que l'on déspécialise par l'antislash ****
- **.** : remplace un caractère.
- **?** : rend facultatif le caractère qu'il précède.
- **+** : indique que le caractère précédent peut apparaître une ou plusieurs fois.
- ***** : pareil que **+** Mais le caractère précédent peut ne pas apparaître du tout.
- **{i,j}** : retrouve une chaîne contenant entre au minimum **i** et au maximum **j** fois le motif qu'il précède.
- **{i,}** : idem mais pas de limite maximum.
- **{i}** : retrouve une séquence d'exactement **i** fois le motif qu'il précède.
- **^** : le motif suivant doit apparaître en début de chaîne.
- **$** : le motif suivant doit apparaître en fin de chaîne.

Les expressions régulières

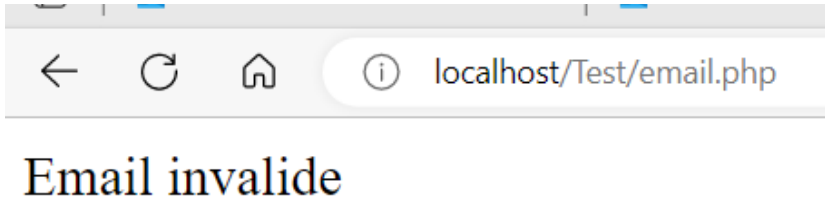
Les caractères spéciaux (suite)

Exemples de motifs :

- **"/[A-Z] /"** : recherche toutes les majuscules
- **" /[a-zA-Z]/ "** : recherche toutes les lettres de l'alphabet minuscules ou majuscules
- **"/ [^aeyuio]/ "** : exclu les voyelles
- **"/^Le/"** : toute chaîne commençant par le mot **''Le ''**

Exercice : donner le patron d'un email valide

```php
<?php
$email= "ali@eee";
if (!preg_match("/^(.+)@(.+)\.(.+)$/",$email))
echo "Email invalide";
else
echo "Email valide";
?>
```



Dates et heures

- **date("$format")** : retourne une chaîne de caractères contenant la date et/ou l'heure locale au format spécifié.

Exemple 1 :

```
echo date("Y-m-d H:i:s");
/* affiche la date au format MySQL : '2024-02-21 11:30:29' */
```

- **getdate()** : retourne un tableau associatif contenant la date et l'heure.

Exemple 2 :

```
$aujourdhui = getdate();
$mois = $aujourdhui['mon'];
$jour = $aujourdhui['mday'];
$annee = $aujourdhui['year'];
echo "$jour/$mois/$annee";
```

- **checkdate($month, $day, $year)** : vérifie la validité d'une date.

Exemple 3 :

```
if(checkdate(03, 31,2005))  echo "La date est valide";
```

Dates et heures (suite)

- ***Les formats pour date :***

d Jour du mois sur deux chiffres [01..31] **j** Jour du mois sans les zéros initiaux **l** Jour de la semaine textuel en version longue et en anglais **D** Jour de la semaine textuel en trois lettres et en anglais **w** Jour de la semaine numérique [0..6] (0: dimanche)	**m** Mois de l'année sur deux chiffres [01..12] **n** Mois sans les zéros initiaux **F** Mois textuel en version longue et en anglais **M** Mois textuel en trois lettres **Y** Année sur 4 chiffres **y** Année sur 2 chiffres

h Heure au format 12h [01..12] avec les zéros initiaux

g Heure au format 12h sans les zéros initiaux

H Heure au format 24h [00..23] avec les zéros initiaux

G Heure au format 24h sans les zéros initiaux

i Minutes [00..59]

s Secondes [00.59]

t Nombre de jour dans le mois donné [28..31

Dates et heures

Clé	Description	Exemple de valeur retournée
"seconds"	Représentation numérique des secondes	0 à 59
"minutes"	Représentation numérique des minutes	0 à 59
"hours"	Représentation numérique des heures	0 à 23
"mday"	Représentation numérique du jour du mois courant	1 à 31
"wday"	Représentation numérique du jour de la semaine courante	0 (pour Dimanche) à 6 (pour Samedi)
"mon"	Représentation numérique du mois	1 à 12
"year"	Année, sur 4 chiffres	Exemples : 1999 ou 2003
"yday"	Représentation numérique du jour de l'année	0 à 365
"weekday"	Version texte du jour de la semaine	Sunday à Saturday
"month"	Version texte du mois, comme January ou March	January à December
0	Nombre de secondes depuis l'époque Unix, similaire à la valeur retournée par la fonction time() et utilisée par date().	Dépend du système, typiquement de -2147483648 à 2147483647.

Fonctions prédéfinies

Fonctions	Commentaires	Exemple

gettype ($nom_var)	détermine le type de la variable ("integer", "double", etc.)	$x = 2.3;
		echo (gettype($x));
		// double

is_int ($nom_var)	détermine si la variable est un entier	$x = 2.3; $entier = is_int ($x); // false
is_string ($nom_var)	détermine si la variable est une chaîne	$x = "2.3"; $str = is_string ($x) ; // true
is_numeric ($nom_var)	détermine si une variable est un type numérique	$x = 23; is_numeric ($x) // true
isset ($nom_var)	détermine si une variable possède une valeur (true ou false)	$x = 2.3; $val = isset ($x) ; // true
unset ($nom_var)	Détruit une variable	unset ($x); // isset($x) donne false

Fonctions prédéfinies

Fonctions	Commentaires	Exemple
empty ($nom_var)	renvoie true si la variable est vide ou vaut 0, sinon revoie false	$x = 2.3; $vide = empty ($x) ; // false
ctype_alpha ($nom_var)	vérifie si tous les caractères de la variable sont des lettres alphabétiques	$x = "a3"; ctype_alpha ($x) // false
include ("nom fichier")	inclut et exécute le fichier spécifié en argument	Include(“template/ent.php")
include_once ("nom fichier")	inclut et évalue le fichier spécifié durant l'exécution du script. Le comportement est similaire à include(), mais la différence est que si le code a déjà été inclus, il ne le sera pas une seconde fois.	Include("adm/cnx.php")
require ("nom fichier")	identique à include() mise à part le fait que lorsqu'une erreur survient, il stoppera le script alors que include() n'émettra qu'une alerte de type E_WARNING, ce qui permet au script de continuer.	require("template/ent.php")

PHP	JavaScript
Scripts côté serveur	Scripts côté client
Utilisé dans l'administration	Utilisé sur l'interface publique (maintenant full stack avec Node.js)
Se combine uniquement avec le HTML	Se combine avec plusieurs langages
Partiellement sensible à la casse	Entièrement sensible à la casse
Différences de syntaxe, par exemple # est autorisé pour les commentaires	Différences de syntaxe, par exemple # nest pas autorisé pour les commentaires
Variables déclarées avec le préfixe $.	Variables déclarées avec les mots-clés var ou let
Possède des tableaux associatifs	Aucun tableau associatif
S'intègre à de nombreuses bases de données	Support de base de données faible ou inexistant
Multi-threaded	Single-threaded
Rapide si PHP 7.0 ou supérieur	Habituellement plus rapide que PHP
Utilise les gestionnaires de paquets PEAR et Composer	Utilise les gestionnaires de paquets npm, Yarn et Bower
Rapide à exécuter si la version de PHP > 7.x	Généralement plus rapide que le PHP
Utilisé sur environ 80% des sites web	Utilisé sur presque tous les sites web