PHP : première partie

I. Introduction

Historique de PHP

PHP, qui désignait à l'origine **Personal Home Page**, a été créé en 1994 par Rasmus Lerdorf. Ce projet est né de son besoin de suivre les visiteurs de son CV en ligne. En 1997, Zeev Suraski et Andi Gutmans ont amélioré le langage, menant à la première version officielle, PHP3. Ce dernier est devenu **PHP Hypertext Preprocessor**.

Évolutions majeures :

- **PHP 4** : Développé avec le moteur Zend Engine.
- **PHP 5** : Introduit la programmation orientée objet (POO) de manière avancée.
- **PHP 7** : Finalisé en décembre 2015, remplaçant PHP 5.
- **PHP 8** : Actuellement en version 8.x, apportant des améliorations, notamment la compilation JIT (Just In Time).

PHP et les sites Web dynamiques

Un **site Web dynamique** est un site où le contenu change de manière autonome, en fonction de divers facteurs tels que la date, l'utilisateur ou son historique de navigation. À l'inverse, un site statique a un contenu fixe qui nécessite une intervention manuelle pour être modifié.

Particularités du PHP

- **Langage côté serveur** : PHP s'exécute sur le serveur.
- **Code source confidentiel** : Le code PHP n'est pas visible par le client, protégeant ainsi les données sensibles.
- **Open source** : PHP est libre de droits.
- **Multi-plateforme** : Fonctionne sur divers systèmes d'exploitation (Unix/Linux, Windows, Mac OS, BSD).
- **Syntaxe intuitive** : Facile à apprendre pour ceux familiarisés avec C ou JavaScript.
- **Interfaçage avec les bases de données** : Peut interagir avec de nombreux systèmes de gestion de bases de données (SGBD), en particulier MySQL.
- **Richesse fonctionnelle** : Supporte de nombreuses bibliothèques pour divers traitements.
- **Modes d'exécution** : Peut s'exécuter en mode CGI ou CLI (ligne de commande).

Frameworks PHP

Plusieurs frameworks sont disponibles pour les développeurs, notamment :

- **Zend Framework**
- **Symfony**
- **CodeIgniter**
- **Laravel**

II. De quoi aura-t-on besoin pour coder en PHP ?

1. Serveur Web

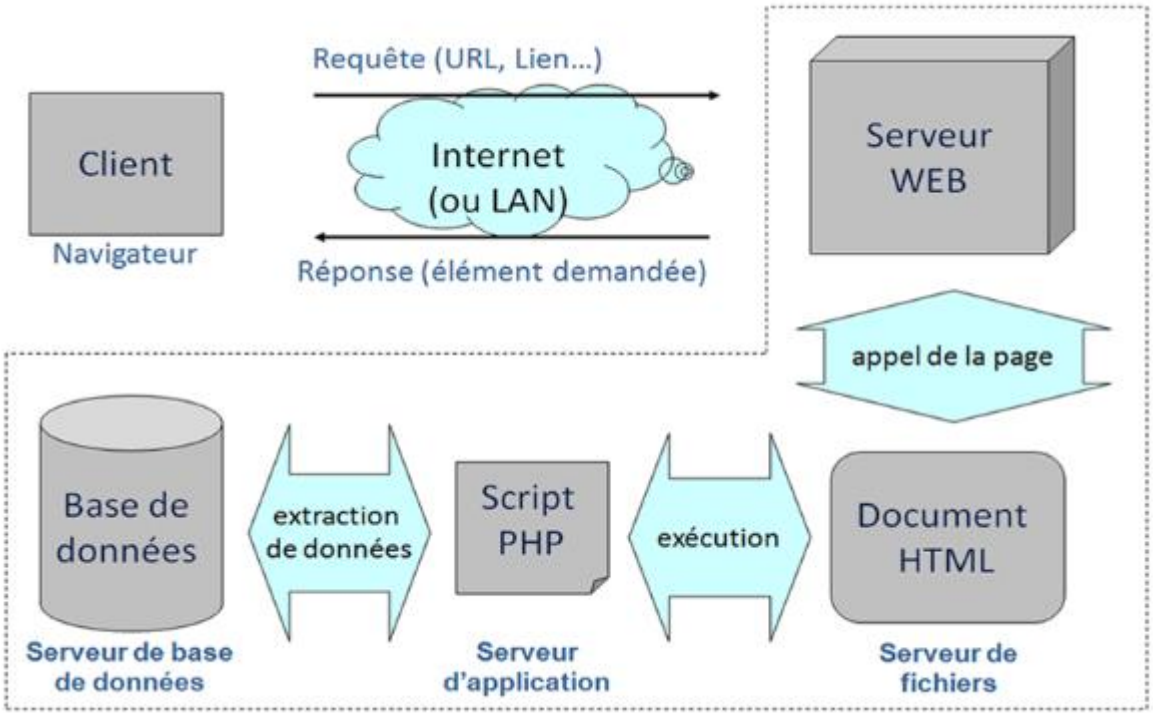
Un **serveur web** est conçu pour gérer et servir des pages web aux clients (navigateurs web) via le protocole HTTP/HTTPS. Il traite principalement des requêtes statiques et dynamiques en interaction avec un langage côté serveur comme PHP, Python ou Ruby.

☞ **Exemples de serveurs web :**

- Apache HTTP Server
- Nginx
- Microsoft IIS

☞ **Fonctionnalités principales :**

- Gérer les requêtes HTTP (GET, POST, etc.)
- Servir des fichiers statiques (HTML, CSS, JavaScript, images)
- Exécuter du code côté serveur via des modules (PHP, Python, etc.)
- Agir comme reverse proxy vers un serveur d'application



2. Préparation du serveur

Pour développer en PHP, il est crucial d'avoir un serveur Web configuré pour exécuter des requêtes HTTP. Le protocole HTTPS est également pris en charge pour la sécurité.

Principaux serveurs Web

- **Apache** : Le serveur Web le plus populaire et adapté pour PHP, supportant de nombreux modules et systèmes d'exploitation.
- **IIS** : Serveur de Microsoft, principalement pour Windows.

3. Installation d'Apache Server

Pour créer un environnement de développement, il est recommandé d'utiliser des solutions comme :

- **WAMP** : Pour Windows, intégrant Apache, MySQL et PHP.
- **XAMPP** : Multi-plateforme, supporte Apache, MySQL et PHP.
- **MAMP** : Pour MacOS, intégrant Apache, MySQL et PHP.
- **LAMP** : Pour Linux, intégrant Apache, MySQL et PHP.

III. Intégration du code PHP

Où faut-il placer nos documents PHP ?

Pour exécuter du code PHP, il faut appeler le serveur d'application via l'URL `http://localhost`. Contrairement aux fichiers HTML ou JavaScript, vous ne pouvez pas simplement cliquer sur un document PHP pour l'exécuter dans votre navigateur.

Extension des documents PHP

Un document PHP doit toujours avoir l'extension `.php`. Cela indique au serveur Web qu'il doit exécuter le document avec le moteur PHP avant de le renvoyer au client.

Document Root

Tous les documents PHP doivent être placés dans un emplacement précis appelé **Document Root**. Pour WAMP Server, le Document Root est le dossier `www`, généralement situé dans le chemin d'installation de WAMP (par défaut `C:/wamp/www`).

Il est possible de changer cet emplacement dans la configuration de WAMP, mais il est conseillé de laisser les paramètres par défaut pour se concentrer sur l'apprentissage de PHP.

Création d'un dossier pour votre projet

Pour organiser votre projet, créez un dossier nommé `formation` dans le dossier `www`. Vous y placerez le fichier `index.php` que vous créerez.

Pour exécuter ce fichier, utilisez l'URL suivante : `http://localhost/formation/index.php` ou simplement `http://localhost/formation`, car le serveur cherchera le fichier `index.php` par défaut.

À quoi ressemble un document PHP ?

PHP : première partie

Une page PHP peut contenir plusieurs langages : HTML, CSS, JavaScript et PHP. Voici un exemple de document PHP :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <style>
    * {
      font-family: verdana;
      font-size: 10pt;
    }
    </style>
  </head>
  <body>
    <div>C'est du HTML.</div>
    <script language="javascript">
      document.write("C'est du Javascript.");
    </script><br />
    <?php
      echo "Et là, du PHP!";
    ?>
  </body>
</html>
```

Délimiteurs PHP

Pour indiquer au serveur où se trouve le code PHP, utilisez des délimiteurs. Tout code entre ces balises sera exécuté par le serveur.

Types de délimiteurs

1. **Délimiteur long** : <?php ... ?> (le plus recommandé).

2. **Délimiteur court** : <? ... ?> (peut être désactivé).
3. **Délimiteur de script** : <script language="PHP"> ... </script> (rarement utilisé).
4. **Délimiteur ASP** : <% ... %> (peut être activé ou désactivé).

IV. Les bases du PHP - Les variables, constantes et opérateurs

Généralités

Dans ce cours, nous allons explorer la syntaxe de base de PHP en utilisant son aspect procédural, similaire à la programmation en langage C. Le PHP est un langage compilé depuis sa version 4, ce qui signifie que le code source est traduit en langage machine avant d'être exécuté. Une erreur dans le code PHP empêche l'exécution de toute la page.

Les commentaires

Les commentaires en PHP sont ignorés lors de l'exécution. Ils peuvent être de deux types :

- **Commentaire de fin de ligne** : Utilisez // pour commenter jusqu'à la fin de la ligne.
- **Commentaire sur plusieurs lignes** : Utilisez /* pour commencer et */ pour terminer le commentaire.

```php
<?php
  // Commentaire de fin de ligne
  /*
    Bloc entier
    vu comme un
    commentaire
  */
?>
```

Variables PHP

Les variables en PHP permettent de stocker des valeurs qui peuvent changer au cours de l'exécution. Contrairement à JavaScript, il n'est pas nécessaire de déclarer le type d'une variable. Les variables sont préfixées par le symbole $ et peuvent contenir des lettres, des chiffres (mais pas au début) et des underscores.

Exemples de variables

```php
<?php
  $a = 10;      // Juste
  $_a9 = true;  // Juste
  //$9a = "Bonjour"; // Faux
  //$a b = 5.3;     // Faux
?>
```

Variables scalaires

Les variables scalaires peuvent être :

- **Nombres entiers** (positifs ou négatifs).
- **Nombres décimaux** (type double).
- **Chaînes de caractères**.
- **Booléens** (true ou false).

Si une variable est appelée mais n'existe pas, PHP lui affecte la valeur 0 par défaut.

Assignation par référence

L'assignation par référence permet à deux variables de pointer vers le même contenu :

```
$a = &$b; // $a et $b pointent vers le même contenu
```

Constantes PHP

Les constantes stockent des valeurs qui ne changent pas au cours de l'exécution. Elles sont définies à l'aide de la fonction define().

Exemple de constante

```
<?php
  define("cte", "Bonjour");
  echo cte; // Affiche Bonjour
  define("cte", "Bonsoir"); // Ne change pas la valeur
?>
```

Les opérateurs

Les opérateurs sont des symboles permettant d'effectuer des opérations sur les variables. PHP dispose de plusieurs types d'opérateurs :

1. Opérateurs arithmétiques

Effectuent des calculs mathématiques :

- Addition (+)
- Soustraction (-)
- Multiplication (*)
- Division (/)
- Modulo (%)

Exemple

```
$a = 10;
$b = 20;
$c = $a + $b; // $c vaut 30
```

2. Opérateurs d'incrémentation

Modifient la valeur d'une variable en l'augmentant ou la diminuant de 1 :

- Incrémentation (++)
- Décrémentation (--)

Exemple

```
$a = 10;
$a++; // $a vaut 11
$a--; // $a vaut à nouveau 10
```

3. Opérateurs d'assignement

Permettent d'affecter une valeur à une variable, avec diverses opérations :

- Affectation directe (=)
- Affectation avec addition (+=)
- Affectation avec soustraction (-=)
- Affectation avec multiplication (*=)
- Affectation avec division (/=)
- Affectation avec modulo (%=)

Exemple

```
$a = 10;
$a += 5; // $a vaut 15
```

4. Opérateurs de comparaison

Testent des conditions :

- Égal (==)
- Strictement inférieur (<)
- Inférieur ou égal (<=)
- Strictement supérieur (>)
- Supérieur ou égal (>=)
- Différent (!=)
- Identique (===)
- Non identique (!==)

Exemple

```
$a = 10;
if ($a % 2 == 0) {
```

```
  $res = "Nombre pair";
}
```

5. Opérateurs logiques

Regroupent plusieurs conditions :

- ET logique (&&)
- OU logique (||)
- NON logique (!)

Exemple

```
$a = 10;
$b = 5;
if ($a % 2 == 0 || $b % 2 == 0) {
  $res = "Le résultat de la multiplication des deux nombres sera pair";
Echo $res
```

V. Les structures de contrôle - Conditions et boucles

Structures conditionnelles et boucles

Les structures de contrôle en PHP sont similaires à celles de JavaScript, avec quelques ajouts spécifiques à PHP.

Structure `if else`

La structure `if else` permet de vérifier une condition et d'exécuter un traitement basé sur le résultat de cette condition.

Exemple :

```
<?php
  $a = 1;
  if ($a == 1) {
    echo "Un";
  } else {
    if ($a == 2) {
      echo "Deux";
    } else {
      echo "Autre";
    }
  }
?>
```

Simplification avec `elseif`

Si plusieurs conditions doivent être vérifiées, vous pouvez utiliser `elseif` pour simplifier le code :

```php
<?php
  $a = 1;
  if ($a == 1) {
    echo "Un";
  } elseif ($a == 2) {
    echo "Deux";
  } else {
    echo "Autre";
  }
?>
```

Opérateur ternaire

L'opérateur ternaire permet de réduire la syntaxe de `if else` :

```php
<?php
  $a = 1;
  echo ($a % 2 == 0) ? '$a est paire' : '$a est impaire';
?>
```

Résultat : `$a est impaire`

Structure `switch case`

La structure `switch case` permet de tester plusieurs cas pour une même variable.

Exemple :

```php
<?php
  $a = 1;
  switch ($a) {
    case 1:
      echo "Un";
      break;
    case 2:
      echo "Deux";
      break;
    default:
      echo "Autre";
  }
?>
```

Boucle `for`

La boucle `for` exécute un bloc de code un nombre défini de fois.

Exemple :

```php
<?php
  for ($i = 0; $i < 10; $i++) {
    echo "Bonjour<br />";
  }
?>
```

Résultat :

```
Bonjour
Bonjour
...
(10 fois)
```

Boucle `while`

La boucle `while` exécute un bloc de code tant qu'une condition est vraie.

Exemple :

```php
<?php
  $i = 0;
  while ($i < 10) {
    echo "Bonjour<br />";
    $i++;
  }
?>
```

Boucle `do while`

La boucle `do while` exécute le bloc de code au moins une fois, puis vérifie la condition.

Exemple :

```php
<?php
  $i = 0;
  do {
    echo "Bonjour<br />";
    $i++;
  } while ($i < 10);
?>
```

Mots-clés `break` et `continue`

- **break** : Met fin à une boucle prématurément.

Exemple :

```
<?php
  for ($i = 0; $i < 10; $i++) {
    if ($i == 5) break;
    echo "Bonjour<br />";
  }
?>
```

Résultat :

```
Bonjour
...
(5 fois)
```

- **continue** : Ignore l'itération courante et passe à l'itération suivante.

Exemple :

```
<?php
  for ($i = 0; $i < 10; $i++) {
    if ($i == 5) continue;
    echo "$i<br />";
  }
?>
```

VI. Formulaires et fonctions agissant sur les variables

Récupération des champs de formulaire

Supposons que nous ayons un formulaire simple en HTML :

```
<form method="post" action="">
  <input type="text" name="prenom" /><br />
  <input type="submit" name="valider" value="Vérifier" />
</form>
```

Ce formulaire contient un champ de texte pour le prénom et un bouton de soumission. Les données sont envoyées à la même page via la méthode POST.

Variables $_POST et $_GET

Pour récupérer les valeurs envoyées par le formulaire, nous utilisons les tableaux superglobaux $_POST et $_GET.

- **$_POST** : Contient les valeurs des champs de formulaire lorsque la méthode POST est utilisée.
- **$_GET** : Utilisé lorsque la méthode GET est employée.

Par exemple, pour récupérer la valeur du champ prenom, vous pouvez utiliser :

```
$prenom = $_POST["prenom"];
```

Fonctions agissant sur les variables

Il existe plusieurs fonctions utiles pour manipuler et vérifier les variables en PHP :

- **empty($var)** : Vérifie si la variable est vide.
- **isset($var)** : Vérifie si la variable existe.
- **unset($var)** : Supprime la variable.
- **gettype($var)** : Retourne le type de la variable.
- **is_numeric($var)** : Vérifie si la variable est numérique.
- **is_int($var)** : Vérifie si la variable est un entier.

Exemple d'application

Vérifions si le champ prénom n'est pas vide. Si le formulaire est soumis, nous afficherons un message indiquant si le prénom est valide ou non.

Voici le code complet :

```
<?php
  @$prenom = $_POST["prenom"];
  @$valider = $_POST["valider"];
  $message = "";

  if (isset($valider)) {
    if (empty($prenom)) {
      $message = '<font color="#FF0000">Prénom invalide.</font>';
    } else {
      $message = '<font color="#00FF00">Prénom valide.</font>';
    }
  }
?>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
  </head>
  <body>
    <form method="post" action="">
      <input type="text" name="prenom" /><br />
```

```
      <input type="submit" name="valider" value="Vérifier" />
    </form>
    <?php
      echo $message;
    ?>
  </body>
</html>
```

- **Récupération des valeurs** :
  - Les lignes @$prenom = $_POST["prenom"]; et @$valider = $_POST["valider"]; récupèrent les valeurs des champs du formulaire, avec le symbole @ pour masquer les erreurs si les clés n'existent pas.
- **Condition de traitement** :
  - if (isset($valider)) vérifie si le bouton de soumission a été cliqué.
  - Ensuite, if (empty($prenom)) vérifie si le champ prénom est vide.
- **Affichage du message** :
  - Le message est stocké dans la variable $message, qui est affichée sous le formulaire.

Masquage des messages d'erreur

Le symbole arobas (@) permet de masquer les avertissements générés par PHP concernant les clés non définies. Bien que cela ne soit pas recommandé en production, il peut être utile lors du développement pour éviter des messages d'erreur indésirables.

Fonctions mathématiques en PHP

PHP propose une série de fonctions prédéfinies pour effectuer des opérations mathématiques. Voici un aperçu des principales fonctions disponibles.

VII. Fonctions mathématiques standards

Liste des fonctions

- **abs($x)** : Retourne la valeur absolue de $x.
- **ceil($x)** : Retourne l'arrondi supérieur de $x.
- **floor($x)** : Retourne l'arrondi inférieur de $x.
- **round($x, $i)** : Retourne l'arrondi le plus proche de $x avec $i chiffres après la virgule.
- **pow($x, $y)** : Retourne $x à la puissance $y.
- **max($a, $b, $c, ...)** : Retourne le plus grand nombre parmi $a, $b, $c, ...
- **min($a, $b, $c, ...)** : Retourne le plus petit nombre parmi $a, $b, $c, ...
- **log($x)** : Retourne le logarithme naturel de $x.
- **log10($x)** : Retourne le logarithme en base 10 de $x.
- **exp($x)** : Retourne l'exponentielle de $x.
- **sqrt($x)** : Retourne la racine carrée de $x.
- **hexdec($x)** : Convertit $x de la base hexadécimale à la base décimale.

- **dechex($x)** : Convertit $x de la base décimale à la base hexadécimale.
- **bindec($x)** : Convertit $x de la base binaire à la base décimale.
- **decbin($x)** : Convertit $x de la base décimale à la base binaire.

Exemple d'utilisation

```php
<?php
  $x = -1.6;
  echo abs($x); // Affiche 1.6
  echo ceil($x); // Affiche -1
  $nbr = "A2F5";
  echo hexdec($nbr); // Affiche 41717
?>
```

Fonctions trigonométriques

Liste des fonctions trigonométriques

- **pi()** : Retourne une approximation de π (3.14159265359).
- **sin($x)** : Retourne le sinus de $x.
- **cos($x)** : Retourne le cosinus de $x.
- **tan($x)** : Retourne la tangente de $x.
- **asin($x)** : Retourne l'arc sinus de $x.
- **acos($x)** : Retourne l'arc cosinus de $x.
- **atan($x)** : Retourne l'arc tangente de $x.

Constantes prédéfinies

- **M_PI** : Valeur de π.
- **M_E** : Valeur du nombre d'Euler (2.71828182846).

Nombres aléatoires

PHP permet également de générer des nombres aléatoires, ce qui est utile dans diverses applications comme les jeux ou les CAPTCHA.

Fonctions pour les nombres aléatoires

- **mt_rand()** : Génère un nombre entier aléatoire entre 0 et RAND_MAX. Par exemple, mt_rand(1, 4) génère un nombre aléatoire entre 1 et 4.
- **mt_srand()** : Initialise le générateur de nombres aléatoires.
- **mt_getrandmax()** : Retourne la valeur maximale que mt_rand() peut générer (2147483647).

Exemple d'utilisation

```php
<?php
```

```
  for ($i = 1; $i <= 8; $i++) {
    $nbr = mt_rand();
    echo "$i - <b>$nbr</b><br>";
    mt_srand();
  }
?>
```

Formatage des nombres

La fonction number_format() permet de formater les nombres pour une meilleure lisibilité, en ajoutant des séparateurs de milliers.

Syntaxe de number_format()

```
number_format($nbr, $dec, $vir, $sep)
```

- **$nbr** : Le nombre à formater.
- **$dec** : Le nombre de chiffres après la virgule.
- **$vir** : Le symbole de la virgule.
- **$sep** : Le symbole du séparateur de milliers.

Exemple d'utilisation

```
<?php
  $nbr = 123456789.0123;
  echo number_format($nbr, 2, ",", " "); // Affiche 123 456 789,01
?>
```

VIII. Les chaînes de caractères en PHP

Une chaîne de caractères en PHP est une suite de caractères, sans limite de longueur. Elle est délimitée par des guillemets doubles (") ou simples (').

Exemple :

```
$str1 = "Bonjour";
$str2 = 'Bonsoir';
```

Différence entre simples et doubles cotes

- **Doubles cotes** : Les variables et les séquences d'échappement sont évaluées.
- **Simples cotes** : La chaîne est affichée telle quelle, sans évaluation.

Exemple :

```
$str = "Bonjour";
echo "$str à tous"; // Affiche: Bonjour à tous
echo '$str à tous'; // Affiche: $str à tous
```

Concaténation de chaînes de caractères

Pour concaténer des chaînes, on utilise l'opérateur point (**.**).

Exemple :

```php
<?php
$nbr = 2.3;
echo "L'arrondi supérieur de $nbr est " . ceil($nbr);
?>
```

Résultat : `L'arrondi supérieur de 2.3 est 3`

Fonctions agissant sur les chaînes de caractères

PHP propose de nombreuses fonctions pour manipuler les chaînes. Voici une liste des plus utiles :

- **strlen($str)** : Retourne le nombre de caractères de $str.
- **strtoupper($str)** : Convertit $str en majuscules.
- **strtolower($str)** : Convertit $str en minuscules.
- **ucfirst($str)** : Met en majuscule le premier caractère de $str.
- **trim($str)** : Supprime les espaces au début et à la fin de $str.
- **rtrim($str)** : Supprime les espaces à la fin de $str.
- **substr($str, $deb, $nbr)** : Extrait une partie de $str à partir de la position $deb et retourne $nbr caractères.
- **ord($car)** : Retourne le code ASCII du caractère $car.
- **chr($int)** : Retourne le caractère correspondant au code ASCII $int.
- **addslashes($str)** : Ajoute des antislashs avant les caractères spéciaux dans $str.
- **stripslashes($str)** : Supprime les antislashs dans $str.
- **strip_tags($str)** : Supprime les balises HTML dans $str.
- **htmlentities($str)** : Convertit certains caractères en entités HTML.
- **md5($str)** : Calcule le hachage MD5 de $str.
- **sha1($str)** : Calcule le hachage SHA1 de $str.
- **str_replace($occ, $rem, $str)** : Remplace toutes les occurrences de $occ par $rem dans $str.
- **explode()** transforme une chaîne en tableau.
- **implode() et join()** transforment un tableau en chaîne.

Exemple d'utilisation :

```php
<?php
$str = "Bonjour";
echo strlen($str); // Affiche: 7
echo strtoupper($str); // Affiche: BONJOUR
```

```php
echo substr($str, 0, 3); // Affiche: Bon
echo htmlentities("<font>"); // Affiche: &lt;font&gt;

for ($i = 0; $i < strlen($str); $i++) {
    echo substr($str, $i, 1) . " : " . ord(substr($str, $i, 1)) . "<br />";
}
echo md5($str); // Affiche: ebc58ab2cb4848d04ec23d83f7ddf985
echo str_replace(["o", "n", "u"], ["x", "y", "z"], "Bonjour"); // Affiche: Bxyjxzr
?>
```

Afficher les chaînes de caractères

Pour afficher des chaînes, on utilise généralement `echo`, qui n'est pas une fonction requérant des parenthèses.

Syntaxe :

```php
<?php
echo "Bonjour";
?>
```

Ou avec le délimiteur court :

```php
<?="Bonjour"?>
```

`print`

La structure `print` fonctionne de manière similaire à `echo` :

```php
<?php
print "Bonjour"; // Affiche: Bonjour
?>
```

`Printf` : la fonction `printf()` permet d'afficher des chaînes formatées :

```php
<?php
printf("Salut à %s", "tous"); // Affiche: Salut à tous
?>
```

Exercice - Vérification des champs de formulaire en PHP

Énoncé

1. créer un formulaire d'inscription contenant les champs suivants :

- **Civilité** : liste de sélection (Mlle, Mme, M.)

- **Nom** : zone de texte
- **Prénom** : zone de texte
- **Email** : zone de texte
- **Mot de passe** : zone de mot de passe
- **Confirmation du mot de passe** : zone de mot de passe
- **Bouton d'envoi** : pour valider l'inscription

2. contrôler la saisie et d'afficher des messages d'erreur si des champs sont laissés vides ou si les mots de passe ne correspondent pas. Les valeurs saisies doivent être conservées dans le formulaire après soumission.

Solution

```
<?php
  @$civilite = $_POST["civilite"];
  @$nom = $_POST["nom"];
  @$prenom = $_POST["prenom"];
  @$email = $_POST["email"];
  @$pass = $_POST["pass"];
  @$repass = $_POST["repass"];
  @$valider = $_POST["valider"];

  $message = ""; // Initialisation de la variable message

  if (isset($valider)) {
    if (empty($nom)) {
      $message .= '<div class="erreur">Nom laissé vide.</div>';
    } elseif (empty($prenom)) {
      $message .= '<div class="erreur">Prénom laissé vide.</div>';
    } elseif (empty($email)) {
      $message .= '<div class="erreur">Email laissé vide.</div>';
    } elseif (empty($pass)) {
      $message .= '<div class="erreur">Mot de passe laissé vide.</div>';
    } elseif ($pass != $repass) {
```

```
      $message .= '<div class="erreur">Les mots de passe ne sont pas identiques.</div>';
    } else {
      $message .= '<div class="rappel"><b>Rappel:</b><br />';
      $message .= $civilite . ' ' . ucfirst(strtolower($prenom)) . ' ' . strtoupper($nom) . '<br />';
      $message .= 'Email: ' . $email;
      $message .= '</div>';
    }
  }
?>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="ISO-8859-1" />

  </head>
  <body>
    <?php echo $message ?> <!-- Affichage des messages d'erreur ou de rappel -->
    <form name="fo" method="post" action="">
      <fieldset>
        <legend>Nouvel utilisateur</legend>
        <div class="label">Civilité</div>
        <div class="champ">
          <select name="civilite">
            <option <?php if ($civilite == "Mme") echo "selected";?>>Mme</option>
            <option <?php if ($civilite == "Mlle") echo "selected";?>>Mlle</option>
            <option <?php if ($civilite == "M.") echo "selected";?>>M.</option>
          </select>
        </div>
        <div class="label">Nom</div>
```

```
        <div class="champ">
          <input type="text" name="nom" value="<?php echo $nom ?>" />
        </div>
        <div class="label">Prénom</div>
        <div class="champ">
          <input type="text" name="prenom" value="<?php echo $prenom ?>" />
        </div>
        <div class="label">Email</div>
        <div class="champ">
          <input type="text" name="email" value="<?php echo $email ?>" />
        </div>
        <div class="label">Mot de passe</div>
        <div class="champ">
          <input type="password" name="pass" />
        </div>
        <div class="label">Confirmer le mot de passe</div>
        <div class="champ">
          <input type="password" name="repass" />
        </div>
        <div class="champ">
          <input type="submit" name="valider" value="Valider l'inscription" />
        </div>
      </fieldset>
    </form>
</body>  </html>
```

- **Récupération des valeurs** : Les valeurs des champs du formulaire sont récupérées avec $_POST. Les arobas (@) sont utilisés pour éviter les notifications d'erreur si les champs ne sont pas définis.
- **Validation des champs** : Lors de la soumission, le code vérifie si les champs sont vides ou si les mots de passe ne correspondent pas. Les messages d'erreur sont stockés dans la variable $message.

- **Affichage des messages** : Les messages sont affichés au-dessus du formulaire, et les valeurs saisies sont conservées dans les champs en utilisant l'attribut value.

IX. Les tableaux en PHP

Variables tableau

Les tableaux en PHP permettent de stocker plusieurs valeurs dans une seule variable. Ils peuvent être **indexés** ou **associatifs**, et peuvent aussi contenir plusieurs dimensions.

Tableaux indexés

Un tableau indexé utilise des indices numériques pour accéder à ses éléments. Par défaut, les indices commencent à 0.

Méthodes de création

Première méthode :

```php
$tab = array("PHP", "Coté serveur", 60);
```

Deuxième méthode :

```php
$tab = array();
$tab[0] = "PHP";
$tab[1] = "Coté serveur";
$tab[2] = 60;
```

Vous pouvez également définir des indices spécifiques :

```php
$tab = array();
$tab[0] = "PHP";
$tab[9] = "Coté serveur";
$tab[10] = 60;
```

Troisième méthode :

```php
$tab = array();
$tab[] = "PHP";
$tab[] = "Coté serveur";
$tab[] = 60;
```

Tableaux associatifs

Un tableau associatif utilise des clés (chaînes de caractères) au lieu d'indices numériques.

Exemples de déclaration

Première méthode :

```
$tab = array("langage" => "PHP", "execution" => "Coté serveur", "heures" => 60);
```

Deuxième méthode :

```
$tab = array();
$tab["langage"] = "PHP";
$tab["execution"] = "Coté serveur";
$tab["heures"] = 60;
```

Tableaux à plusieurs dimensions

Les tableaux peuvent aussi être multidimensionnels. Voici un exemple d'un tableau indexé à deux dimensions :

```
$tab1 = array("A", "B", "C");
$tab2 = array("D", "E", "F");
$tab3 = array("G", "H", "I");
$tab = array($tab1, $tab2, $tab3);
```

Pour accéder aux éléments :

```
echo $tab[0][0]; // Affiche: A
echo $tab[1][2]; // Affiche: F
```

Parcourir un tableau : structure `foreach`

La structure `foreach` est utilisée pour parcourir les tableaux de manière simple et efficace.

Exemple avec tableau indexé :

```
$tab = array("PHP", "Coté serveur", 60);
foreach ($tab as $elem) {
    echo "$elem <br />";
}
```

Exemple avec tableau associatif :

```
$tab = array(
    "langage" => "PHP",
    "execution" => "Coté serveur",
    "heures" => 60
);
foreach ($tab as $cle => $elem) {
```

```
    echo "$cle: $elem <br />";
}
```

Fonctions agissant sur les tableaux

Il existe plusieurs fonctions utiles pour manipuler les tableaux en PHP :

- **count($tab) ou sizeof($tab)** : Retourne le nombre d'éléments du tableau.
- **in_array($var, $tab)** : Vérifie si $var existe dans le tableau.
- **list($var1, $var2, ...)** : Affecte les valeurs du tableau à des variables.
- **shuffle($tab)** : Mélange les éléments du tableau.
- **sort($tab)** : Trie les éléments du tableau par ordre alphanumérique.
- **rsort($tab)** : Trie les éléments en ordre inverse.
- **array_rand($tab)** : Retourne une clé aléatoire du tableau.
- **array_merge($tab1, $tab2, ...)** : Fusionne plusieurs tableaux en un seul.
- **implode($sep, $tab)** : Retourne une chaîne de caractères à partir des éléments du tableau, séparés par $sep.
- **explode($occ, $str)** : Crée un tableau en éclatant la chaîne $str au niveau des occurrences $occ.

Exemples d'utilisation :

```
$tab = array("PHP", "Coté serveur", 60);
echo count($tab); // Affiche: 3

if (in_array("PHP", $tab))
    echo "Existe"; // Affiche: Existe

list($a, $b, $c) = $tab;
echo "$a - $b - $c"; // Affiche: PHP - Coté serveur - 60

echo implode(" - ", $tab); // Affiche: PHP - Coté serveur - 60

echo array_rand($tab); // Affiche soit: 0, 1, ou 2
echo $tab[array_rand($tab)]; // Affiche soit: PHP, Coté serveur, ou 60

$str = "Bonjour";
$tab = explode("o", $str);
echo implode("*", $tab); // Affiche: B*nj*ur
```