

Développement Web - INF04

Le Langage PHP

Mussab Zneika
CY-Tech

Introduction à PHP



www.php.net

Introduction à PHP

- PHP est un langage de script côté serveur.
- Les scripts PHP sont exécutés sur le serveur.
- PHP supporte de nombreuses bases de données (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.).
- PHP est un logiciel open source.
- PHP est gratuit à télécharger et à utiliser.

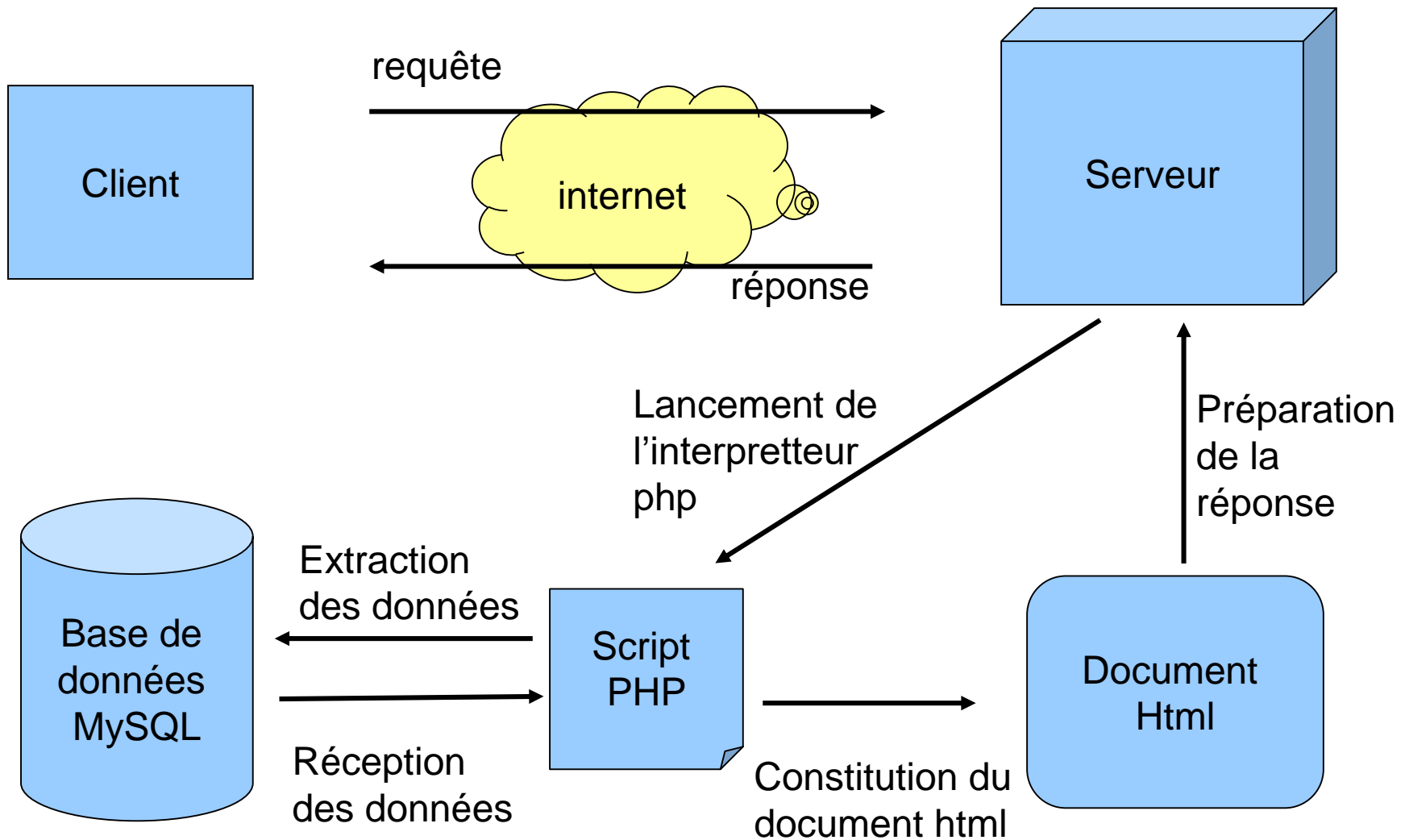
Introduction à PHP

- PHP fonctionne sur différentes plateformes (Windows, Linux, Unix, etc.).
- PHP est compatible avec presque tous les serveurs utilisés aujourd'hui (Apache, IIS, etc.).
- PHP est téléchargeable GRATUITEMENT à partir de la ressource PHP officielle : www.php.net.
- PHP est facile à apprendre et fonctionne efficacement côté serveur.

Introduction à PHP

- Le code PHP est exécuté sur le serveur, générant du HTML qui est ensuite envoyé au client.
- Le client recevrait les résultats de l'exécution de ce script, mais ne saurait pas quel était le code sous-jacent.

Introduction à PHP



6

Introduction à PHP

- Sous Windows, vous pouvez télécharger et installer WAMP. Avec une seule installation, vous obtenez un serveur Web Apache, un serveur de base de données et php.

<http://www.wampserver.com>

- Sur Mac, vous pouvez télécharger et installer MAMP.

<http://www.mamp.info/en/index.html>

- Sur Linux.

<https://doc.ubuntu-fr.org/php>

Au terminal taper **php -S localhost:8080** pour lancer le script php

Au navigateur taper **localhost:8080/leNomDeLaPage.php**

Par exemple : localhost:8080/bonjour.php

Intégration d'un script PHP dans une page

- Les pages web sont au format html.
- Les pages web dynamiques générées avec PHP sont au format php et doit enregistrer avec l'extension .php
- Le code source php est directement insérer dans le fichier html grâce au conteneur de la norme XML :

<?php ... ?>

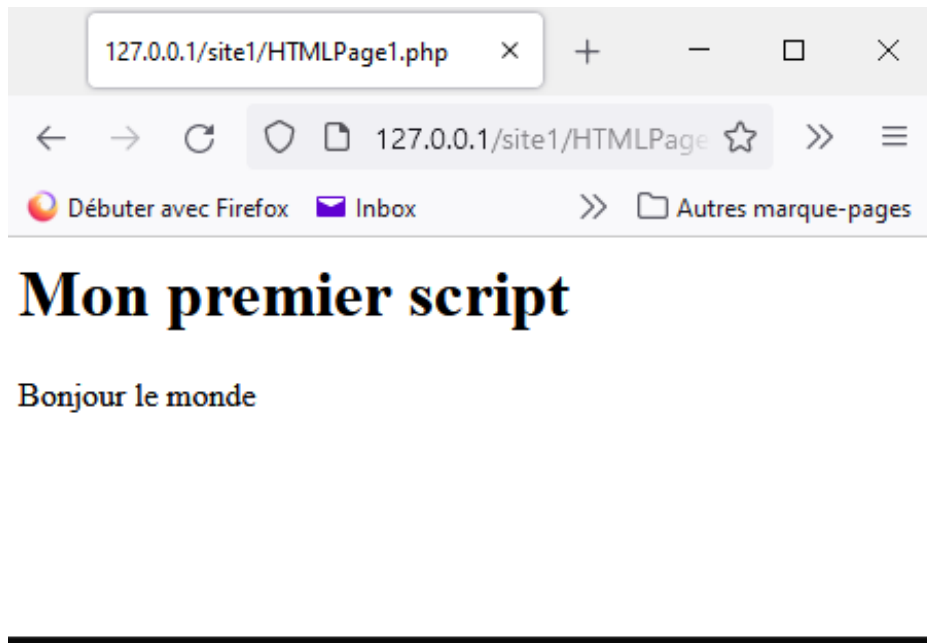
- Autres syntaxes d'intégration :
 - ▶ **<? ... ?>**
 - ▶ **<script language="php"> ... </script>**
 - ▶ **<% ... %>**

Intégration d'un script PHP dans une page

Exemple de script, code source (côté serveur) :

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

Résultat affiché par le navigateur :



Code source (côté client)

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

Affichage de texte

- echo : écriture le paramètre dans le navigateur

```
<html>
  <head><title>
    <?php
      echo "title";
    ?>
  </title></head>
  <body>
    <?php
      echo "<p>Bonjour le monde !</p>";
    ?>
  </body>
</html>
```

- print() : écriture dans le navigateur
- printf([\$format, \$arg1, \$arg2]) : écriture formatée comme en C, i.e. la chaîne de caractère est constante et contient le format d'affichage des variables passées en argument

Commentaires

- En PHP, on utilise **//** pour faire un commentaire sur une seule ligne ou **/* et */** pour faire un gros bloc de commentaire.

Exemple :

```
<?php
```

// commentaire de fin de ligne

/* commentaire
sur plusieurs
lignes ***/**

... # commentaire de fin de ligne
comme en Shell
?>

Variables

- Les variables sont utilisées pour stocker des valeurs, comme des chaînes de texte, des nombres ou des tableaux.
- Le typage des variables est implicite en php. Il n'est donc pas nécessaire de déclarer leur type au préalable ni même de les initialiser avant leur utilisation.
- Toutes les variables en PHP commencent par un symbole \$.
- La bonne façon de déclarer une variable en PHP :

```
$var_name = value;
```

Variables

- Un nom de variable doit commencer par une lettre ou un trait de soulignement "_" -- pas un nombre.
- Un nom de variable ne peut contenir que des caractères alphanumériques, des traits de soulignement (a-z, A-Z, 0-9 et _).
- Un nom de variable ne doit pas contenir d'espaces.
- Les variables peuvent être de type entier (integer), réel (double), chaîne de caractères (string), tableau (array), objet (object), booléen (boolean).
- Il est possible de convertir une variable en un type primitif grâce au cast (comme en C).
 - Exemple :
 - ✓ `$str = "12";` // \$str vaut la chaîne "12"
 - ✓ `$nbr = (int)$str;` // \$nbr vaut le nombre 12
- Le cast est une conversion de type. L'action de caster consiste en convertir une variable d'un type à un autre.

Gestion des variables

- Quelques fonctions :

- **empty(\$var)** : renvoie vrai si la variable est vide
 - **isset(\$var)** : renvoie vrai si la variable existe
 - **unset(\$var)** : détruit une variable
 - **gettype(\$var)** : retourne le type de la variable
 - **settype(\$var, "type")** : convertit la variable en type type (cast)
 - **is_long(), is_double(), is_string(), is_array(), is_object(), is_bool(), is_float(), is_numeric(), is_integer(), is_int()...**
 - **intval(\$val,\$base)** : retourne la valeur numérique entière équivalente d'une variable.
 - **Strval(\$val)**: retourne la valeur d'une variable, au format chaîne.
 - **Boolva(\$val)** retourne la valeur booléenne d'une variable.
 - **floatval (\$val)**:Convertit une chaîne en nombre à virgule flottante
- et beaucoup plus: : <https://www.php.net/manual/fr/ref.var.php>

Gestion des variables

- Exemple:

- `$var = 5.55;`
- `$var2 = strval($var); // '5.55'`
- `$var3 = intval($var); // 5`
- `$var4 = boolval($var); // true`
- `$var5 = floatval($var); // 5.55`
- `echo gettype($var); // double`
- `settype($var, "integer");`
- `echo $var; // 5`
- `echo gettype($var); // integer`

Concaténation PHP

- L'opérateur de concaténation (.) est utilisé pour assembler deux valeurs de chaîne.
- Pour concaténer deux variables de chaîne ensemble, utilisez l'opérateur de concaténation :

```
<?php  
$var1 = "Mussab";  
$var2 = "Zneika";  
$Prenom_nom=$var1." ".$var2;  
echo $Prenom_nom; // Mussab Zneika  
?>
```


Les opérateurs

Les opérateurs	Mode
+, -, *, /, %	Actes numériques
, &&, !	Actes logiques
==, !=, <, >, <=, >=	Comparaison
++, --	Augmentation, diminution
.	concaténation
=, +=, -=, *=, /=, %=	Affectation

Les opérateurs

code	Résultat
<code>3 + 5</code>	8
<code>2 - 9</code>	-7
<code>1 / 2</code>	0.5
<code>5 * 7</code>	35
<code>102 % 5</code>	2
<code>true false</code>	true
<code>!true</code>	false
<code>true && true</code>	true
<code>3 < 5</code>	true
<code>3 != 3</code>	false
<code>"Hello " . ', world!'</code>	Hello, world!

Structures de contrôle: if...else

- Très souvent, lorsque vous écrivez du code, vous souhaitez effectuer différentes actions pour différentes décisions.
- Vous pouvez utiliser des instructions conditionnelles dans votre code pour ce faire.
- En PHP, nous avons les instructions conditionnelles suivantes:
 - **if** : exécuter du code uniquement si la condition est vraie
 - **if...else**: exécuter un code si une condition est vraie et un autre code si la condition est fausse
 - **if...elseif....else**: utilisez cette instruction à envoyer pour élire l'un des plusieurs blocs de code à exécuter .

Structures de contrôle: if...else

- Exemple1 Cet exemple affichera pas passé

```
$a = "10";  
if ( $a > 10 ) {  
    echo "passes";  
}  
else {  
    echo "échoué" ;  
}
```

- Cet exemple affichera **échoué**

Structures de contrôle: if...else

- Exemple2

```
if(0) echo 1;      // faux
if("") echo 2;     // faux
if("0") echo 3;    // faux
if("00") echo 4;
if('0') echo 5;    // faux
if('00') echo 6;
if(" ") echo 7;
```

Cet exemple affiche 467. Donc l'espace ou la chaîne "00" ne sont pas considérés FALSE.

Structures de contrôle: switch

- Utilisez l'instruction **switch** pour sélectionner l'un des nombreux blocs de code à exécuter.

```
switch ($var) {  
    case valeur1:  
        traitement1  
        break;  
    case valeur2:  
        traitement2  
        break;  
    ...  
    default:  
        ...  
}
```

Structures de contrôle: switch

Exemple

```
$a=10;  
switch ( $a ) {  
    case 5:  
        echo "a is 5";  
        break;  
    case 10:  
        echo "a is 10";  
        break;  
    case 15:  
        echo "a is 15";  
        break;  
    default:  
        echo "a is neither 5, nor 10, nor 15";  
}
```

- Cet exemple affichera **a is 10**

Structures de contrôle: boucle(for, while, do,while

- Structures de boucle (même syntaxe qu'en langage C) :

- **for(... ; ... ; ...) {**
- ...
- **}**

- **while(...) {**
- ...
- **}**

- **do {**
- ...
- **} while(...);**

Structures de contrôle : break

- Syntaxe comme en C, C++, Java...
- Peut apparaître dans l'un des contrôles suivants :
for, while, do... while, switch
- Il arrête le flux et continue immédiatement après
- Plus de répétitions après **break**

```
<?php  
  
for($i=0;$i<6;$i++){  
    if($i==4){ break;}  
    echo $i."<br>";  
}  
echo "i after the for ". $i."<br>";  
  
?>
```

php

Le Résultat

```
0  
1  
2  
3  
i after the for 4
```

Structures de contrôle : continue

- Syntaxe comme en C, C++, Java...
- Peut apparaître dans l'un des contrôles suivants :
for, while, do... while
- Il arrête le flux et continue de vérifier la condition.
- D'autres répétitions peuvent se produire après **continue**.

```
<?php  
  
for($i=0;$i<6;$i++){  
    if($i==4){ continue;}  
    echo $i."<br>";  
  
}  
echo "i after the for ". $i."<br>";  
  
?>
```

php

Le Résultat

```
0  
1  
2  
3  
5  
i after the for 6
```

Fonctions

- Comme tout langage de programmation, php permet l'écriture de fonctions.
- Les fonctions peuvent prendre des arguments dont il n'est pas besoin de spécifier le type. Elles peuvent de façon optionnelle retourner une valeur.
- L'appel à une fonction peut ne pas respecter son prototypage (nombre de paramètres).
- Les identificateurs de fonctions sont insensibles à la casse.

Exemple :

```
function mafonction($toto) {  
    $toto += 15;  
    echo "Salut !";  
    return ($toto+10);  
}
```

```
$nbr = MaFonction(15.1);
```

```
/* retourne 15.1+15+10=40.1, les majuscules n'ont pas d'importance */
```

Fonctions

Exemple 1: Fonction de calcul de la moyenne de deux nombres

```
function avg( $a, $b ) {  
    $c = $a + $b;  
    return $c / 2;  
}
```

```
echo 'The average of 3, 5: ' . avg(  
3, 5 );  
echo "\n";  
echo 'The average of 1, 9: ' . avg(  
1, 9 );
```

Fonctions

Exemple 2: Fonction pour tester si le nombre est premier ou non
..

```
function is_prime( $a ) {  
    for ( $i = 2; $i < $a; ++$i )  
    {  
        if ( $a % $i == 0 ) {  
            return false;  
        }  
    }  
    return true;  
}  
  
if ( is_prime( 5 ) ) {  
    echo "5 is a prime number."  
}
```

Fonctions

- On peut donner une valeur par défaut aux arguments lors de la déclaration de la fonction.

Tous les arguments sont optionnels ↓

```
function makeCoffee  
( $type = "frappe", $milk = true ) {  
  
    $str = "Making a cup Coffe of type ".$type;  
    if ( $milk ) {  
        $str = $str." with milk";  
    }  
    $str .= ".\n";  
    return $str;  
}  
  
echo makeCoffee( "Arabic" );
```

- Cet exemple affichera **Making a cup Coffe of type Arabic with milk.**

Fonctions

- Valeurs par défaut uniquement pour les derniers arguments

```
function makeCoffee  
( $type = "frappe", $milk) {  
  
    $str = "Making a cup Coffe of type ".$type;  
    if ( $milk ) {  
        $str = $str." with milk";  
    }  
    $str .= ".\n";  
    return $str;  
}  
echo makeCoffee( "Arabic" );
```

- Cet exemple affichera rien .

Fonctions: la portée des variables

- On peut modifier la portée des variables locales à une fonction.
- `global` permet de travailler sur une variable de portée globale au programme.

Exemple 1:

```
$a = 5;  
echo $a."<br>";  
function tata(){  
    $a=10;  
    echo $a."<br>";  
}  
tata();  
echo $a;
```

5
10
5

Exemple 2:

```
$a = 5;  
echo $a."<br>";  
function tata(){  
    global $a;  
    $a=10;  
    echo $a."<br>";  
}  
tata();  
echo $a;
```

5
10
10

Fonctions: la portée des variables

- **static** permet de conserver la valeur d'une variable locale à une fonction

Exemple :

```
function tata(){  
    static $a=0;  
    $a=$a+5;  
    echo $a."<br>";  
}  
tata();//5  
tata();//10  
tata();//15
```

5
10
10

Fonctions:Mathématiques

- Quelques fonctions:
 - **abs(\$x)** : valeur absolue
 - **ceil(\$x)** : arrondi supérieur
 - **floor(\$x)** : arrondi inférieur
 - **pow(\$x,\$y)** : x exposant y
 - **round(\$x,\$i)** : arrondi de x à la ième décimale
 - **max(\$a, \$b, \$c ...)** : retourne l'argument de valeur maximum
 - **pi()** : retourne la valeur de Pi
 - **cos, sin, tan, exp, log, min, pi, sqrt...**
 - **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.
 - **srand()** : initialisation du générateur aléatoire
 - **getrandmax()** : retourne la valeur du plus grand entier pouvant être généré

Fonctions:Mathématiques

Exemple 1

php

```
<?php
$var=35.78910897;
$x=10; $y=20; $z=10;
echo floor(1.9) . "<br>" ; // 1
echo ceil(1.9) . "<br>" ; // 2
echo ceil(1.3) . "<br>" ; // 2
echo round(1.9) . "<br>" ; // 2
echo floor($var) . "<br>" ; // 35
echo ceil($var) . "<br>" ; // 36
echo round($var,1) . "<br>" ; // 35.8
echo round($var,2) . "<br>" ; // 35.79
echo max($x,$y,$z) . "<br>" ; // 20
echo min($x,$y,$z) . "<br>" ; // 10
echo abs($x) . "<br>" ; // 10
echo abs(-$x) . "<br>" ; // 10
?>
```

Fonctions:Mathématiques

Exemple 2

```
<?php  
echo rand() . "<br>";  
Echo rand(10,50) . "<br>";  
echo getrandmax(); //2147483647  
?>
```

php

Fonctions:String

- 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.
 - **strrev(\$str)**: Inverser une chaîne
 - **substr(\$str,\$i,\$j)** : retourne une sous chaîne de **\$str** de taille **\$j** et débutant à la position **\$i**
 - **strnatcmp(\$str1,\$str2)** : comparaison de 2 chaînes
 - **addslashes(\$str)** : désécialise les caractères spéciaux (', ", \)
 - **ord(\$char)** : retourne la valeur ASCII du caractère **\$char**.
 - **str_replace(\$str1,\$str2,\$str3[, \$count])**: remplacer str 1 par str2 dans str3
 - **explode(séparateur, \$str)**: retourne un tableau de chaînes de caractères, chacune d'elle étant une sous-chaîne du paramètre string extraite en utilisant le séparateur séparateur.
 - Plus : <https://www.php.net/manual/fr/ref.strings.php>

Fonctions:String

Exemple 1

php

```
<?php
$nom="Mussab Zneika";
echo strlen($nom) . "<br>"; //13
$nomupper=strtoupper($nom);
echo $nom."<br>"; //Mussab Zneika
echo $nomupper."<br>"; //MUSSAB ZNEIKA
$nomlow=strtolower($nom);
    echo $nom."<br>"; //Mussab Zneika
echo $nomlow."<br>"; //mussab zneika

if (strnatcmp($nomupper,$nomlow)) //yes
    { echo "yes"; }

if ($nomupper==$nomlow) //No
    { echo "yes"; }
    else { echo "No"; }
?>
```

Fonctions : String

Exemple 2

php

```
<?php
$count=0;
echo str_replace("world","Mussab","Hello world world!",
$count) . "<br>"; //Hello Mussab Mussab!
echo $count . "<br>"; //2

echo str_repeat("cergy ",5) . "<br>"; // cergy cergy cergy
cergy cergy

print_r (explode(" ","CY Cergy Paris Université en
chiffres")); //Array ( [0] => CY [1] => Cergy [2] =>
Paris [3] => Université [4] => en [5] => chiffres )

echo ord('A') . "<br>"; // 65
echo ord('a') . "<br>"; // 97

echo chr(65) . "<br>"; // A
?>
```

Fonctions : String

Exercise 1

Ecrivez une fonction **PHP** qui vérifie si une chaîne des caractères passée (ne contient que les caractères alphabétiques) est palindrome ou non ?

Remarque : Un palindrome est un mot, une phrase ou une séquence qui se lit de la même manière que l'avant, par exemple, radar , rotor ou stats .

Fonctions : String

Exercise 1: solution

PHP

```
function palindrome($str1) {  
    $str1=str_replace(" ", "", $str1);  
    $str1=strtolower($str1);  
    $str2=strrev($str1);  
    if ($str1==$str2) {  
        echo $str1." est un palindrome<br>";  
    }  
    else {  
        echo $str1.' n\'est pas un palindrome<br>';  
    }  
}  
  
palindrome("radar"); // radar est un palindrome  
palindrome("a Laval elle lavala"); //  
alavalellelavala est un palindrome
```

Fonctions:header()

- Il est possible d'envoyer des entêtes particuliers du protocole HTTP grâce à la fonction header.
- **Syntaxe : header(\$str);**
- Les entêtes doivent obligatoirement être envoyées avant l'affichage de tout caractère dans la page en cours. Car l'affichage force l'envoi des entêtes de base.
- **headers_sent()** : Retourne TRUE si les entêtes ont déjà été envoyées, FALSE sinon.
- **Exemple:**
 - **header("Location: http://www.example.com/");** /* Redirection du navigateur */
 - **header('Content-type: text/html; charset=utf-8')** <=>
 <meta
 http-equiv="Content-type"
 content="text/html; charset=utf-8"
 />

Tableaux

- Une variable tableau est de type **array**.
- Un tableau accepte des éléments de tout type.
- Les éléments d'un tableau peuvent être de types différents et sont séparés d'une virgule.
- Un tableau peut être initialisé avec la syntaxe **array**.
 - Exemple :
 - `$tab_colors = array('red', 'yellow', 'blue', 'white');`

Valeurs	Red	yellow	blue	white
index	0	1	2	3

- Mais il peut aussi être initialisé au fur et à mesure.
 - Exemples :
 - `$prenoms[] = "Mussab";`
 - `$prenoms[] = "Wassim";`
 - `$prenoms[] = "Marc";`
 - `$villes[0] = "Paris";`
 - `$villes[1] = "Cergy";`
 - `$villes[2] = "Madrid";`

Tableaux

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

Exemple :

```
$toto = array( 0 => 1, 1 => 'tata' );  
echo $toto[ 0 ]; // 1  
echo $toto[ 1 ]; // tata
```

```
$foo = array( 0 => 1, 'bar' );  
echo $foo[ 0 ]; // 1  
echo $foo[ 1 ]; // bar
```

Tableaux

- Parcours d'un tableau. : **For, while, do.. while**
 - **\$tab = array("Mussab", "Marc", "Juilan");**

Exemple 1 :

```
$i=0;  
while($i < count($tab)) {  
  // count() retourne le nombre  
  d'éléments  
  echo $tab[$i]."<br>";  
  $i++;  
}
```

Exemple 2 :

```
for($i=0;$i<count($tab); $i++)  
{  
  echo $tab[$i]."<br>";  
}
```

- Parcours d'un tableau. : **foreach**
 - **\$tab = array("Mussab", "Marc", "Juilan");**

Exemple :

```
foreach($tab as $elem) {  
  echo $elem. "<br>";  
}
```

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
- **range(\$i,\$j)** : retourne un tableau contenant un intervalle de valeurs
- **shuffle(\$tab)** : mélange les éléments d'un tableau
- **sort(\$tab)** : trie alphanumérique les éléments du tableau
- **rsort(\$tab)** : trie alphanumérique inverse les éléments du tableau
- **implode(\$str,\$tab), join** : retournent une chaîne de caractères contenant les éléments du tableau **\$tab** joints par la chaîne de jointure **\$str**
- **explode(\$delim,\$str)** : retourne un tableau dont les éléments résultent du hachage de la chaîne **\$str** par le délimiteur **\$delim**
- **array_merge(\$tab1,\$tab2,\$tab3...)** : concatène les tableaux passés en arguments
- **array_rand(\$tab)** : retourne un élément du tableau au hasard

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
- **range(\$i,\$j)** : retourne un tableau contenant un intervalle de valeurs
- **shuffle(\$tab)** : mélange les éléments d'un tableau
- **sort(\$tab)** : trie alphanumérique les éléments du tableau
- **rsort(\$tab)** : trie alphanumérique inverse les éléments du tableau
- **implode(\$str,\$tab), join** : retournent une chaîne de caractères contenant les éléments du tableau **\$tab** joints par la chaîne de jointure **\$str**
- **explode(\$delim,\$str)** : retourne un tableau dont les éléments résultent du hachage de la chaîne **\$str** par le délimiteur **\$delim**
- **array_merge(\$tab1,\$tab2,\$tab3...)** : concatène les tableaux passés en arguments

Tableaux

❑ Quelques fonctions:

- **array_push(\$tab, \$e1,\$e1..)** : ajoute les éléments \$e1, \$e2, ... a la fin du tableau.
- **array_pop(\$tab)**: retourne le dernier élément du tableau et le supprimer du tableau.
- **array_shift(\$tab)**: dépile un élément au début d'un tableau.
- **array_unshift((\$tab, \$e1,\$e1..)** empile un ou plusieurs éléments \$e1, \$e2,.. au début d'un tableau.
- **array_flip(\$tab)**: remplace les clés par les valeurs, et les valeurs par les clés.
- **array_reverse(\$tab, bool)**: Inverse l'ordre des éléments d'un tableau.
- **slice(\$tab ,begin[,length],bool)**: retourner un nouveau tableau constitue d'une sous-partie d'un tableau existant.
- **array_splice(\$tab1, begin[,length],[\$tab])**:**Supprime** des éléments d'un tableau et remplacez-le par de nouveaux éléments :

Tableaux

Exemple 1

Ajouter des éléments dans un tableau (push,unshift):

PHP

```
<?php
$prenoms=array('Mussab', 'Marc','julian','Naya');
//On ajoute les valeurs Sarra et Francia en fin de
tableau
array_push($prenoms, 'Sarra', 'Francia'); //On
ajoute deux éléments en début de tableau
array_unshift($prenoms,'Flora', 'claire');

//On affiche les valeurs de notre tableau
$p = "";
for($i = 0; $i <count($prenoms); $i++) {
    $p .= 'Prénom n' . $i . ' : ' . $prenoms[$i] .
    '<br>'; }
//On affiche la taille du tableau et les prénoms
echo 'Le tableau contient ' .count($prenoms) .
éléments : <br>' . $p;
?>
```

Le Résultat

Le tableau contient 8 éléments :

Prénom n1 : Flora
Prénom n2 : claire
Prénom n3 : Mussab
Prénom n4 : Marc
Prénom n5 : julian
Prénom n6 : Naya
Prénom n7 : Sarra
Prénom n8 : Francia

Tableaux

Exemple 2

Supprimer des éléments dans un tableau(pop,shift):

PHP

Le Résultat

```
<?php
$prenoms=array('Mussab',
'Marc','julian','Naya');
//On supprime le premier prénom du tableau
$supprp = array_shift($prenoms);
// On supprime le dernière prénom du tableau
$supprd = array_pop($prenoms);

//On récupère les valeurs de notre tableau
$p = "";
for($i = 0; $i <count($prenoms); $i++) { $p .=
'Prénom n' . ($i+1) . ' : ' . $prenoms[$i] .
'<br>'; }

//On affiche la taille du tableau et les prénoms
echo 'Nouveau tableau :<br>'. $p. '<br> Prénom
supprimé : <br>'.$supprp . '<br>'.$supprd ;
?>
```

Nouveau tableau :
Prénom n°1 : Marc
Prénom n°2 : julian

Prénom supprimé :
Mussab
Naya

Tableaux

Exemple 3

Ajouter des éléments choisis dans un tableau(splice)

PHP

```
<?php
$prenoms=array('Mussab', 'Marc','julian','Naya');
$tab2=array('Claire', 'Anne');
//On ajoute les valeurs Claire et Anne à partir de
l'index 2
array_splice($prenoms,2, 0,$tab2) ; $p ="";

//On affiche les valeurs de notre tableau
for($i = 0; $i <count($prenoms); $i++) {
    $p .= 'Prénom n' . ($i+1) . ' : ' . $prenoms[$i] .
'<br>'; }

//On affiche la taille du tableau et les prénoms
echo 'Le tableau contient ' .count($prenoms).'
éléments : <br>' . $p;
?>
```

Le Résultat

Le tableau contient 6 éléments :
Prénom n1 : Mussab
Prénom n2 : Marc
Prénom n3 : Claire
Prénom n4 : Anne
Prénom n5 : julian
Prénom n6 : Naya

Tableaux

Exemple 4

Supprimer des éléments choisis dans un tableau(Splice)

php

```
<?php
$prenoms=array('Mussab',
'Marc','julian','Naya');

//On supprime deux valeurs à partir de
l'index 2
array_splice($prenoms,2, 2) ;

$p = "";
//On récupère les valeurs de notre tableau
for($i = 0; $i <count($prenoms); $i++) {
    $p .= 'Prénom n' . ($i+1) . ' : ' .
    $prenoms[$i] . '<br>'; }

//On affiche la taille du tableau et les
prénoms echo 'Le tableau contient '
.count($prenoms).' éléments : <br>' . $p;
?>
```

Le Résultat

Le tableau contient éléments :

Prénom n1 : Mussab

Prénom n2 : Marc

Tableaux

Exemple 5

Trier les éléments d'un tableau(sort)

PHP

```
<?php
$prenoms=array('Mussab', 'Marc','julian','Naya',
'anne');
//On trie le tableaux dans l'ordre alphabétique
sort($prenoms) ;

//On récupère les valeurs de notre tableau
$p = "";
for($i = 0; $i <count($prenoms); $i++) {
    $p .= 'Prénom n' . ($i+1) . ' : ' . $prenoms[$i]
    . '<br>'; }
//On affiche la taille du tableau et les prénoms
echo 'Le tableau contient ' .count($prenoms).'
éléments : <br>' . $p;
?>
```

Le Résultat

Le tableau contient 5 éléments :
Prénom n1 : Marc
Prénom n2 : Mussab
Prénom n3 : Naya
Prénom n4 : anne
Prénom n5 : julian

Tableaux

Exemple 6

Trier les éléments d'un tableau(sort)

php

```
<?php
$numb=array(10,20,3, 30,4, 1,35,98);
sort($numb);
for($i = 0; $i <count($numb); $i++) {
echo $numb[$i]." , \t"; }
?>
```

Le Résultat

1 , 3 , 4 , 10 , 20 , 30 , 35 , 98 ,

php

```
<?php
$numb=array(10,20,3, 30,4, 1,35,98);
rsort($numb);
for($i = 0; $i <count($numb); $i++) {
echo $numb[$i]." , \t"; }
?>
```

Le Résultat

98 , 35 , 30 , 20 , 10 , 4 , 3 , 1

Exercise 2

Ecrire une fonction php qui prend comme paramètres un tableau "tab" et une variable "v" et renvoie true si v dans tab et false sinon

Tableaux

Exercise 2: solution

PHP

```
<?php
$nums=array(10,20,3, 30,4, 1,35,98);
function rechercheArray($tab,$v) {

    for($i=0;$i<count($tab);$i++) {
        if($tab[$i]==$v)
            return true;
        } return false;
    }

    echo rechercheArray($nums,100);

    // echo in_array(1,$nums)

    ?>
```


Exercise 3

Ecrivez une fonction PHP `toto($str)` qui prend une chaîne des caractères en paramètre (ne contient que les caractères alphabétiques) et convertir la première lettre de chaque mot en majuscule et les autres lettres en minuscule.

Testez cela avec : `toto("wELCome mUssAB ZneIKA ")`;

Doit afficher : `Welcome Mussab Zneika`

Tableaux

Exercise 3: solution

PHP

```
<?php
function toto($str1) {
    $str1=trim($str1);
    $str2= strtolower($str1);
    $res="";
    $s=explode(" ", $str2);
    foreach ($s as $elm) {
        $res=$res.strtoupper($elm[0]);
        $res=$res.substr($elm,1, strlen($elm));
        $res=$res." ";
    }
    return $res;
}
echo toto("wELCome mUssAB ZneIKA"); ?>
```

Tableaux Associatif

- Un tableau associatif est appelé aussi *dictionnaire* ou *hashtable*. On associe à chacun de ses éléments une clé dont la valeur est de type chaîne de caractères.
- L'initialisation d'un tableau associatif est similaire à celle d'un tableau normal.

Exemple 1 :

```
$personne = array("Nom" => 'Zneika', "Prénom" => 'Mussab');
```

Exemple 2 :

```
$personne["Nom"] = 'Zneika';  
$personne["Prénom"] = 'Mussab';
```

Tableaux Associatif

- Parcours d'un tableau associatif.
 - `$personne = array("Nom" => 'Zneika', "Prénom" => 'Mussab');`

Exemple 1 :

```
foreach($personne as $elem) {  
    echo $elem;  
}
```

Exemple 2 :

```
foreach($personne as $key => $elem) {  
    echo "$key : $elem";  
}
```

- Quelques fonctions alternatives pour le parcours de tableaux (normaux ou associatifs) :
 - **reset(\$tab)** : place le pointeur sur le premier élément
 - **current(\$tab)** : retourne la valeur de l'élément courant
 - **next(\$tab)** : place le pointeur sur l'élément suivant
 - **prev(\$tab)** : place le pointeur sur l'élément précédant
 - **each(\$tab)** : retourne la paire clé/valeur courante et avance le pointeur

Tableaux Associatif

Exemple

php

```
<?php
$famille = array("pere" =>'robisson', "mere"=>'gilda',
"soeur"=>'Laurence', "frere" => 'Ludovic', "cousin1" =>'Guillaume',
0=>50) ;
```

```
foreach ($famille as $elm)
{ echo $elm."<br>"; }
```

```
robisson
gilda
Laurence
Ludovic
Guillaume
50
```

```
foreach ($famille as $key =>$elm)
{ echo "$key => $elm.<br>"; }
```

```
pere => robisson.
mere => gilda.
soeur => Laurence.
frere => Ludovic.
cousin1 => Guillaume.
0 => 50.
```

?>

Tableaux multidimensionnels

- Aussi simple qu'un tableau () dans un tableau ()
- pour créer une matrice de 3*3 on peut écrire

```
$toto = array(  
    array( 1, 2, 3 ),  
    array( 4, 5, 6 ),  
    array( 7, 8, 9 )  
);
```

- Accéder aux éléments de Matrix

```
echo $toto[ 1 ][ 2 ]; // 6  
echo $toto[ 2 ][ 0 ]; // 7
```

- Parcours d'un tableau

```
foreach ($toto as $row ) {  
    foreach ( $row as $element ) {  
        echo "$element ";  
    }  
    echo "<br>";  
}
```

```
1 2 3  
4 5 6  
7 8 9
```

Les Formulaires

- Pour obtenir des données à partir de HTTP GET :
 - Variable `$_GET`
 - `$_GET["nom_paramètre"]`
- Pour obtenir des données à partir de HTTP POST :
 - Variable `$_POST`
 - `$_POST["nom_paramètre"]`
- Défini automatiquement par PHP

Les Formulaires: Exemple1

HtmlPage1.html

```
<center> <h1> Login Form </h1> </center>
<form action="Page1.php" method="GET"> <div
class="container">
<label>Username : </label> <input type="text"
placeholder="Enter Username" name="username"
required>

<label>Password : </label> <input
type="password" placeholder="Enter Password"
name="password" required>

<button type="submit">Login</button> </div>
</form>
```

Le Résultat

Username : Password :



localhost/site1/page1.php?username=Mussab&password=Mussab

Welcom Mussab

```
<?php
$x=$_GET ["username"]; if ($x=="Mussab")
    echo "Welcom $x";
else
    header ("Location:HtmlPage1.html");
?>
```

page1.php

Les Formulaires: Exemple2

HtmlPage1.html

```
<center> <h1> Login Form </h1> </center>
<form action="Page1.php" method="POST"> <div
class="container">
<label>Username : </label> <input type="text"
placeholder="Enter Username" name="username"
required>

<label>Password : </label> <input
type="password" placeholder="Enter Password"
name="password" required>

<button type="submit">Login</button> </div>
</form>
```

Le Résultat

Username : Password :



localhost/site1/Page1.php

Welcom Mussab

```
<?php
$x=$_POST ["username"]; if($x=="Mussab")
    echo "Welcom $x";
else
    header("Location:HtmlPage1.html");
?>
```

page1.php

Les Formulaires: Exemple 3

HtmlPage1.html

```
<form action="Page1.php" method="POST"> <div
class="container">
  <h2>Que signifie HTML ?</h2>
  <input type="radio" name="ans" value="ans1" >
a. Et hyperliens texte Markup Language<br >

  <input type="radio" name="ans" value="ans2">
b. Hyper Text Markup Language <br>

  <input type="radio" name="ans" value="ans3">
c. Langage de manipulation hypertexte<br>

  <input type="radio" name="ans" value="ans4">
d. Langage de balisage de l' outil
d'accueil<br>

  <button type="submit">Login</button> </div>
</form>
```

Le Résultat

Que signifie HTML ?

- ☐ a. Et hyperliens texte Markup Language
- ☒ b. Hyper Text Markup Language
- ☐ c. Langage de manipulation hypertexte
- ☐ d. Langage de balisage de l' outil d'accueil

send

localhost/site1/Page1.php



Bravo

```
<?php
  $$x=$_POST["ans"];
  if ($x=="ans2")
    echo "Bravo";

  else
    echo "no";

?>
```

page1.php

Les Formulaires: Exemple 4

HtmlPage1.php

```
<?php
$x="mussab";
header("Location: page1.php?id=5&name=$x" );
?>
```

page1.php

```
<?php
$id=$_GET["id"];
$name=$_GET["name"];
echo "$id: $name";
?>
```

Le Résultat

localhost/site1/page1.php?id=5&name=mussab



5: mussab

Inclusions des fichiers

- On peut inclure dans un script php le contenu d'un autre fichier.
- **require** insert dans le code le contenu du fichier spécifié même si ce n'est pas du code php. Est équivalent au préprocesseur **#include** du C.
- *Exemple* : **require("fichier.php");**
- **include** évalue et insert à chaque appel (même dans une boucle) le contenu du fichier passé en argument.
- *Exemple* : **include("fichier.php");**
- Les instructions **include** et **require** sont identiques, sauf en cas d'échec :
 - **require** produira une erreur fatale (E_COMPILE_ERROR) et arrêtera le script
 - **include** ne produira qu'un avertissement (E_WARNING) et le script continuera
- **include_once, require_once**

Inclusions des fichiers

HtmlPage1.php

```
<?php
    include("page1.php");
?>
```

Le Résultat

localhost/site1/HtmlPage1.php

Welcome to our page www.example.com, date : 2021-12-19 20:01:05

```
<?php
echo "Welcome to our page www.example.com,
date : ". date("Y-m-d H:i:s");
?>
```

page1.php

Inclusions des fichiers : Exemple 2

HtmlPage1.php

```
<?php
    require("page1.php");
?>
```

Le Résultat

localhost/site1/HtmlPage1.php

Welcome to our page www.example.com, date : 2021-12-19 20:01:05

```
<?php
echo "Welcome to our page www.example.com,
date : ". date("Y-m-d H:i:s");
?>
```

page1.php

Cookies

- Un cookie est un fichier que le serveur envoie sur la machine de l'utilisateur
- Ils sont principalement utilisés pour distinguer les utilisateurs
- Il s'agit d'une liste de correspondances clé valeur
- Ils ont une date d'expiration
- Ils sont stockés sur l'ordinateur de l'utilisateur
- Ils sont envoyés à chaque demande
- Différents cookies pour différents domaines
- Ils sont précieux ! Si quelqu'un te les vole alors ça devient toi

Cookies



Salut!

Pour, rappelez-vous que vous êtes un administrateur



admin



je suis un administrateur

D'accord, fais ce que tu veux



Cookies

- Ils sont transférés dans les en-têtes de requête
- Pour définir un cookie pour un utilisateur, nous faisons :
 - **setcookie (\$nom \$, \$valeur, \$ expirer)**
- La fonction setcookie() doit être placé avant tout code HTML, car le cache du navigateur doit être vide pour que cette fonction fonctionne convenablement
- Exemple:

```
setcookie("nom_cookie", "mussab", time()+36000);
```

 - time() : renvoie l'heure au format d'horodatage Unix
 - Expire apres 3600 secondes à partir de maintenant

Cookies

- Ils sont situés dans la variable `$ _COOKIE`
- Il a les clés des noms des cookies
- Il a les valeurs des cookies
- Pour supprimer les cookies il suffit de définir le cookie avec expirer dans le passé: Exemple:
 - `setcookie("nom_cookie", "mussab", time()-36000);`
- Afficher un cookie: `echo $_COOKIE[" nom_cookie "];`
- Afficher tous les cookies: `print_r($_COOKIE);`

Cookies

Exemple

Page1.php

```
<?php  
setcookie("user", "mussab", time()+60*60);  
?>
```

```
<html> <head> </head> <body>  
  <?php  
  if(isset($_COOKIE["user"]))  
  { echo "Welcome". $_COOKIE["user"]; }  
  ?>  
  
</body> </html>
```

Sessions

- Les sessions sont un moyen de sauvegarder et de modifier des variables tout au cours de la visite d'un internaute sans qu'elles ne soient visibles dans l'URL et quelque soient leurs types (tableau, objet...).
- Cette méthode permet de sécuriser un site, d'espionner le visiteur, de sauvegarder son panier (e-commerce), etc.
- Les informations de sessions sont conservées en local sur le serveur tandis qu'un identifiant de session est posté sous la forme d'un cookie chez le client (ou via l'URL si le client refuse les cookies).

Sessions

- **Fonction `session_start()`**

- Démarre une session ou restaure la précédente si l'utilisateur a envoyé un ID de session valide
- Parce qu'il définit un cookie, nous l'appelons avant toute sortie

- **Variable `$_SESSION`**

- Nous l'utilisons après avoir appelé `session_start()`
- Les variables sont stockées et seront disponibles dans les demandes futures avec le même ID de session

Sessions

- Quelques fonctions :
 - **session_start()** : démarre une session.
 - **session_destroy()** : détruit les données de session et ferme la session. Cette fonction ne détruit pas les variables globales associées à la session, de même, elle ne détruit pas le cookie de session.
 - **session_unset()** : détruit toutes les variables de la session courante.
 - **session_id()**: récupérer ou définir l'identifiant de session pour la session courante.
 - **session_status()**: pour connaître l'état de la session courante.
 - ✓ PHP_SESSION_DISABLED si les sessions sont désactivées.
 - ✓ PHP_SESSION_NONE si les sessions sont activées, mais qu'aucune n'existe.
 - ✓ PHP_SESSION_ACTIVE si les sessions sont activées, et qu'une existe.

Sessions

- Sauvegarder des **variables de type objet dans une session est la méthode de sécurisation maximum des données** : elles n'apparaîtront pas dans l'URL et ne pourront pas être forcées par un passage manuel d'arguments au script dans la barre d'adresse du navigateur.
- Les données de session étant sauvegardées sur le serveur, l'accès aux pages n'est pas ralenti même si des données volumineuses sont stockées.
- Une session est automatiquement fermée si aucune requête n'a été envoyée au serveur par le client durant un certain temps (2 heures par défaut dans les fichiers de configuration Apache).
- Une session est un moyen simple de suivre un internaute de page en page (sans qu'il s'en rende compte). On peut ainsi sauvegarder son parcours, établir son profil et établir des statistiques précises sur la fréquentation du site, la visibilité de certaines pages, l'efficacité du système de navigation...

Sessions

Exemple

Page1.php

```
<?php
session_start();
if ( isset( $_SESSION["pageviews" ] ) )
{ ?><p>Vous avez vu la page <?php echo $_SESSION[ "pageviews"]; ?>
fois </p>

<?php ++$_SESSION["pageviews" ]; } else { $_SESSION["pageviews"] = 1;
} ?>
```


Arrêt prématuré

- Pour stopper prématurément un script, il existe deux fonctions.
- **die** arrête un script et affiche un message d'erreur dans le navigateur.
- *Exemple :*
- **if(mysql_query(\$requete) == false)**
- **die("Erreur de base de données à la requête :
\$requet");**
- **exit** l'arrête aussi mais sans afficher de message d'erreur.
- *Exemple :*
- **function foobar() {**
- **exit();**
- **}**
- Ces fonctions stoppent tout le script, pas seulement le bloc en cours.

Gestion de Fichiers

- De nombreuses applications travaillent avec des fichiers. Que ce soit pour les lire, les remplir, les supprimer ou même changer leurs attributs.
- L'accès aux fichiers locaux est très rapide. Si vous avez peu de traitements et de tris à faire sur le contenu, il est généralement plus performant d'utiliser des fichiers qu'une base de données.
- Gestion des fichiers
 - Stockage de données
 - Bien que plus lent qu'une base de données
 - Manipulation des fichiers uploadés
 - À partir de formulaires
 - Création de fichiers à télécharger

Gestion de Fichiers

Lecture et écriture: Ouvrir/fermer un fichier

- **fopen()** : ouvre un fichier en tant que « flux », et PHP renvoie un « gestionnaire » vers le fichier qui peut être utilisé pour référencer le fichier ouvert dans d'autres fonctions.
- Chaque fichier est ouvert dans un mode particulier.

'r'	Ouvert en lecture seulement. Commencer au début du fichier.
'r+'	Ouvert à la lecture et à l'écriture. Commencer au début du fichier.
'w'	Ouvert à l'écriture uniquement. Supprimez tout le contenu précédent, si le fichier n'existe pas, créez-le.
'a'	Ouvrir l'écriture, mais commencer à la FIN du contenu actuel.
'a+'	Ouvrez pour la lecture et l'écriture, commencez à FIN et créez un fichier si nécessaire.

- **fclose()** : ferme le fichier.

Gestion de Fichiers

Lecture et écriture: Ouvrir/fermer un fichier (Exemple)

```
<?php
// ouvrir le fichier à lire
$inp = fopen('some/file.ext', 'r');
// ouvrir le fichier à écrire
$out = fopen('some/file.ext', 'w');
// fermer les deux fichiers
fclose($inp);
fclose($out);
?>
```

Gestion de Fichiers

Lecture et écriture: Lecture de données

- Il existe deux fonctions principales pour lire les données :
 - **fgets(\$fp,\$bytes)** : lit jusqu'à \$bytes de données, s'arrête à la nouvelle ligne ou à la fin du fichier
 - **fread(\$fp,\$octets)**: lit jusqu'à \$ octets de données, s'arrête à la fin du fichier
- **feof(\$fp)** : renvoie true si nous avons atteint la fin du fichier

Exemple:

```
$fp = fopen('test.txt', 'r');  
while (!feof($fp)) {  
    echo fgets($fp, 1024);  
    echo '<br />';  
}  
  
fclose($fp);
```

Gestion de Fichiers

Lecture et écriture: Lecture de données

- Il existe deux fonctions « rapides » qui ne nécessitent pas l'ouverture d'un fichier :
 - **file(\$nom de fichier)**: lit le fichier entier dans un tableau avec chaque ligne une entrée distincte dans le tableau.

Exemple:

```
<?php
$tab = file('test.txt');
for($i=0 ; $tab[$i] ; $i++){
echo $tab[$i].'<br>';
}
```

- **file_get_contents(\$filename)**: lit le fichier entier en une seule chaîne

Exemple:

```
<?php
$contentu_fichier = file_get_contents('test.txt');
echo $contentu_fichier;
?>
```

Gestion de Fichiers

Lecture et écriture: Écriture de données

- **fwrite(\$fp,\$data)** : écrire des données dans un fichier

Exemple:

```
// Ouvre le fichier
$fp = fopen("test.txt", "w");
//Insère le texte « PHP » dans le fichier
fwrite( $fp , "PHP" ) ;

fclose($fp) ;
```

Gestion de Fichiers

Lecture et écriture: Écriture de données

- **file_put_contents()**: une fonction d'accès rapide pour écrire dans un fichier.
- La fonction **file_put_contents()** prend en paramètre une adresse de fichier et une chaîne de caractères.
- La chaîne est alors écrite dans le fichier. Si le fichier existait déjà, son contenu est écrasé.
- Il est toutefois possible de demander à ce que la chaîne en argument soit ajoutée au fichier au lieu de remplacer le contenu actuel. Il suffit alors de spécifier la constante **FILE_APPEND** comme troisième paramètre.

Exemple:

```
<?php
$fichier = 'test.txt';
$contenu = 'Contenu du fichier';
file_put_contents($fichier, $contenu, FILE_APPEND) ;
?>
```


Gestion de Fichiers

Quelques fonctions:

- **file_exists(\$file)** : indique si le fichier **\$file** existe
- **filesize(\$file)** : retourne la taille du fichier **\$file**
- **filetype(\$file)** : retourne le type du fichier **\$file**
- **unlink(\$file)** : détruit le fichier **\$file**
- **copy(\$source, \$dest)** : copie le fichier **\$source** vers **\$dest**
- **readfile(\$file)** : affiche le fichier **\$file**
- **rename(\$old, \$new)** : renomme le fichier **\$old** en **\$new**
- Plus: <https://www.php.net/manual/fr/ref.filesystem.php>

Exercise 1

Ecrire un script en PHP pour lire le contenu d'un fichier texte "test.txt" ligne par ligne et afficher à l'écran le nombre de mots de ce fichier.

Exemple : Si le fichier "test.txt" contient les lignes suivantes :

Il y a une aire de jeux

Un avion est dans le ciel

Le ciel est bleu

La fonction doit afficher la sortie 17

Gestion de Fichiers

Exercice 1: solution1

PHP

```
<?php
// ouvrir le fichier pour lecture
$fp=fopen("test.txt","r");
$nbmots=0;
//lire le fichier ligne par ligne
while (!feof($fp)) {
    $line=fgets($fp); // lire une ligne
    $line=trim($line);
    $mots=explode(" ", $line); //diviser le texte par un
séparateur d'espace
    $nbmots=$nbmots+count($mots);
}

fclose($fp);
echo $nbmots;
?>
```

Cette solution est bonne s'il n'y a qu'un seul espace entre les mots

Gestion de Fichiers

Exercice 1: solution2

PHP

```
<?php
// ouvrir le fichier pour lecture
$fp=fopen("test.txt","r");
$nbmots=0;

//lire ligne par ligne
while (!feof($fp)) {
    $line=fgets($fp); // lire une ligne
    $line=trim($line);
    $mots=explode(" ", $line); //diviser le texte par un
    séparateur d'espace

    foreach($mots as $em) {
        if($em!="") $nbmots=$nbmots+1;
    }
}
fclose($fp);
echo $nbmots;
?>
```

Gestion de Fichiers

Exercise 1: solution 3

```
<?php $nbmots=0;
//lire le contenu complet du fichier
$contenu_fichier = file_get_contents('test.txt');

$contenu_fichier=trim($contenu_fichier);
//"\n" est le saut de ligne dans le fichier donc nous le
remplaçons par " "
$contenu_fichier=str_replace("\n"," ",$contenu_fichier);

$mots=explode(" ", $contenu_fichier);

foreach($mots as $em) {
    //echo $em."<br>";
    if($em!="") $nbmots=$nbmots+1;
}
echo $nbmots;

?>
```

PHP

Exercise 2

Le fichier "étudiants.txt" contient les notes des élèves du cours de programmation Web au format (prénom, nom, note).

Écrivez un script en PHP qui lirait le contenu du fichier "étudiants.txt" et écrirait dans un autre fichier les détails des étudiants dont la note est supérieure à 10.

Exemple : étudiants.txt

Mussab,Zneika,20

Naya, AL, 15

Marc, Zet, 13

Lois, Anab, 5

Résultats attendus : étudiants1.txt

Mussab,Zneika,20

Naya, AL, 15

Marc, Zet, 13

Gestion de Fichiers

Exercise 2: solution

PHP

```
<?php
// ouvrir le fichier pour lecture
$fp=fopen("students.txt","r");
// ouvrir le fichier pour l'écriture
$fpout=fopen("students1.txt","w");
$nb=0;
//lire le fichier ligne par ligne
while (!feof($fp)) {
    $student=fgets($fp); // lire une ligne
    $tab=explode(",",$student);
    if(isset($tab[2])) {
        if(floatval($tab[2])>=10) {
            fwrite($fpout,$student);
            $nb=$nb+1; }
    }
}
fclose($fp);
fclose($fpout);
echo $nb;
?>
```

Gestion de Fichiers

Accès aux dossiers

- Il est possible de parcourir les répertoires grâce à ces quelques fonctions :
- **chdir(\$str)** : Change le dossier courant en **\$str**. Retourne TRUE si succès, sinon FALSE.
- **getcwd()** : Retourne le nom du dossier courant (en format chaîne de caractères).
- **opendir(\$str)** : Ouvre le dossier **\$str**, et récupère un pointeur **\$d** dessus si succès, FALSE sinon et génère alors une erreur PHP
- **closedir(\$d)** : Ferme le pointeur de dossier **\$d**.
- **readdir(\$d)** : Renvoie le nom du fichier suivant dans le répertoire Les fichiers sont triés comme sur le système de fichier
- **rewinddir(\$d)** : Retourne à la première entrée du dossier identifié par **\$d**

Gestion de Fichiers

Accès aux dossiers

- Exemple: pour afficher les fichiers et dossiers dans le dossier actuel

```
<?php
$handle = opendir( './' );

$file=readdir( $handle );

while( false !== $file ) {
    $file=readdir( $handle );
    echo "$file<br />";
}

closedir( $handle );
?>
```

Gestion de Fichiers

Accès aux dossiers

- Il existe un autre moyen d'accéder aux dossiers : l'utilisation de la classe **dir**.
- la classe **dir** a les attributs :
 - **handle** : valeur du pointeur
 - **path** : nom du dossier
- la classe **dir** a les méthodes :
 - **read()** : équivalent à **readdir(\$d)**
 - **close()** : équivalent à **closedir(\$d)**
- Constructeur : **dir(\$str)** : retourne un objet **dir** et ouvre le dossier **\$str**

Gestion de Fichiers

Accès aux dossiers

- Exemple: pour afficher les fichiers et dossiers dans le dossier actuel

```
<?php
```

```
$d = dir(' . '); // ouverture du dossier courant
```

```
echo "Pointeur: " . $d->handle . "<br>";
```

```
echo "Chemin: " . $d->path . "<br>";
```

```
while ($fp = $d->read()) { // lecture d'une entrée
```

```
echo $fp . "<br>";
```

```
}
```

```
$d->close();
```

```
?>
```

Gestion de Fichiers

Manipulation des fichiers uploadés

- Ils sont envoyés au serveur avec une requête POST
- Ils sont stockés temporairement pendant l'exécution de notre programme
- Si nous ne les sauvegardons pas, ils sont perdus après la fin de l'exécution

<input type="text"/>	Parcourir...
<input type="text"/>	Parcourir...
<input type="text"/>	Parcourir...
<input type="text"/>	Parcourir...
<input type="button" value="envoyer"/>	

Gestion de Fichiers

Manipulation des fichiers uploadés

- Variable **\$ _FILES**: contient tous les fichiers téléchargés par l'utilisateur
- **userfile** est le nom du champ de fichier des formulaires
- **\$ _FILES ['userfile'] ['name']**: contient le nom réel du fichier
- **\$ _FILES ['userfile'] ['tmp_name']**: contient la localisation du fichier temporaire
- **\$ _FILES ['userfile'] ['size']**: contient la taille du fichier en octets
- **move_uploaded_file (\$ nomdefichier, \$ destination)**: vérifie si le \$nomdefichier a bien été téléchargé et enregistré

Gestion de Fichiers

HtmlPage1.html

```
<form enctype="multipart/form-data"
method="post" action="page1.php " >
<input name="file1" type="file" />
<input type="submit" value="envoyer" />
</form>
```

<?php

```
$destination = ".\Uploads\\";
if( !empty( $_FILES ) ) {
    $destination .= $_FILES[ 'file1' ][ 'name' ];
    $filename = $_FILES[ 'file1' ][ 'tmp_name' ];
    move_uploaded_file( $filename, $destination );
}
?>
```

page1.php

Les exceptions en PHP

- Les exceptions sont utilisées pour modifier le flux normal d'un script si une erreur spécifiée se produit.
- Historiquement, les erreurs PHP sont gérées via le système error reporting, c'est à dire l'affichage de messages d'erreur directement dans la page.
- Depuis PHP 5, les exceptions ont fait leur apparition.
- Ce système de gestion des erreurs est radicalement différent, c'est un mécanisme utilisé dans de nombreux langages comme, C#, Python ou Java.
- **try catch:**

```
try{  
    .....  
} catch(Exception $e){  
    print_r($e);  
}
```

Les exceptions en PHP

- Comme tous les mots-clés de programmation, ces trois mots proviennent de l'anglais et signifient :
 - Try : une fonction utilisant une exception doit se trouver dans un bloc "try". Si l'exception ne se déclenche pas, le code continuera normalement. Cependant, si l'exception se déclenche, une exception est « levée ».
 - Throw : C'est ainsi que vous déclenchez une exception. Chaque « throw » doit avoir au moins une « prise ».
 - Catch : Un bloc "catch" récupère une exception et crée un objet contenant les informations sur l'exception.

Les exceptions en PHP

```
<?php
function diviser($nb1, $nb2)
{ if($nb2==0) {
    throw new Exception("diviser sur 0 pas possible ");}
  return $nb1/$nb2;}

try{
  diviser(20,0);
}
catch(Exception $e){
  echo 'Message: ' . $e->getMessage();
}
?>
```

page1.php

Le Résultat

Message: diviser sur 0 pas possible

SOURCES D'INFORMATIONS

- <https://www.php.net/manual/fr/function.file.php>
- Cours de Ahmed Jebali : Le Langage PHP