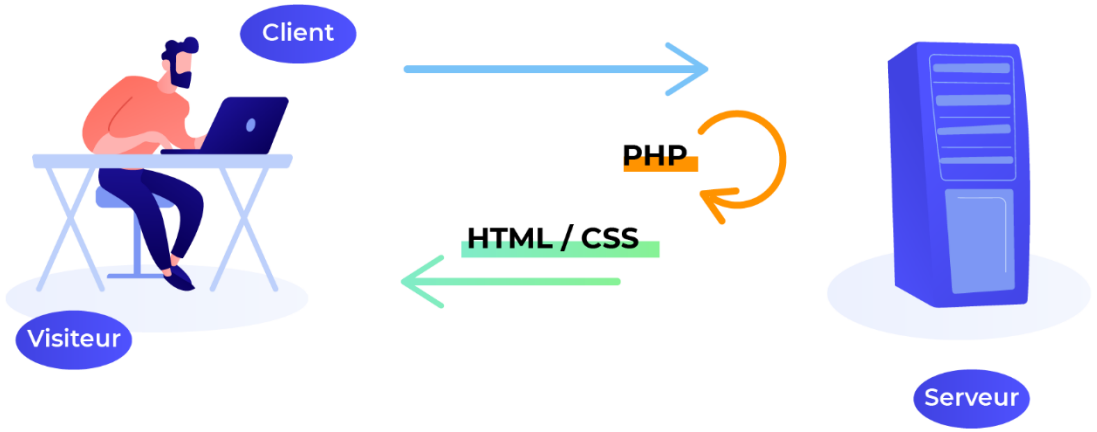
مدرسة علوم المعلومات
ⵜⵉⵣⵔⴰ ⵏⵜⵓⵙⵙⵉⵔⵉⵏ ⵉⵏⴼⵓⵔⵎⴰⵜⵉⵣⵏ
ECOLE DES SCIENCES DE L'INFORMATION
ISCHOOL.MA

# Cours Programmation web en PHP




## Semestre 2

2022/2023

# Modalités d’enseignement et d’évaluation

**Modalités d’enseignement**

Cours magistraux + Travaux dirigée (TD) + Ateliers pratiques (TP).

Volume horaire: 48

- 24h pour chaque groupe
- Questions / Réponses

**Modalités d’évaluation**

- Contrôle continu
- Examen final

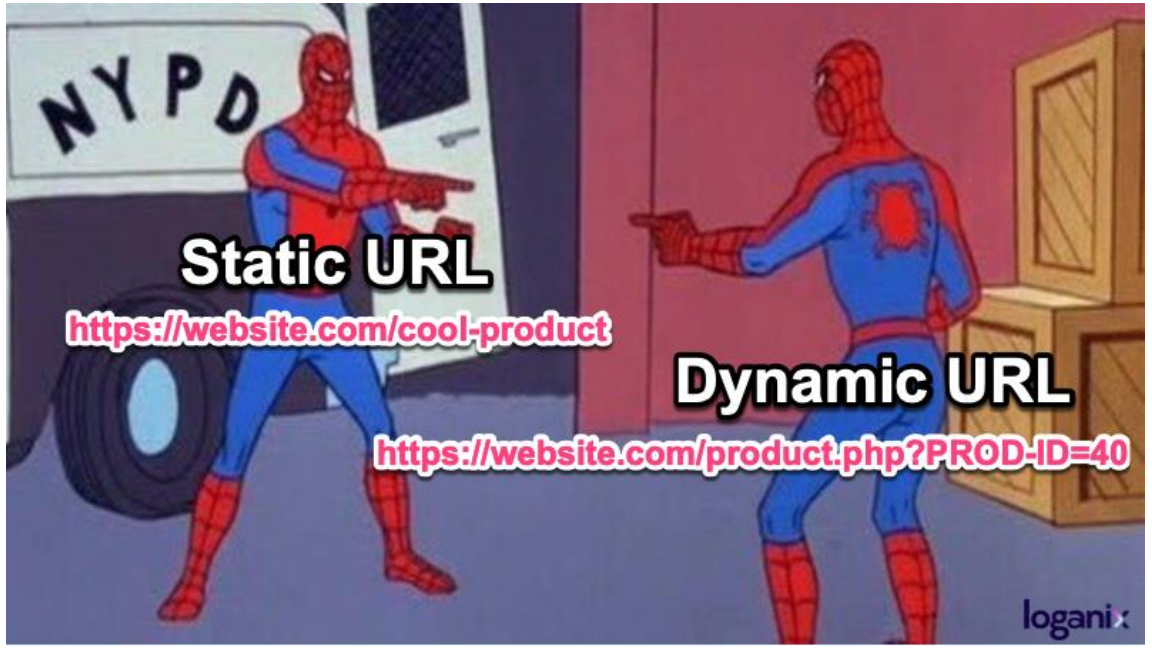
# Plan

**Partie 1: Introduction aux technologies web & syntaxe de base**

- La communication sur le web
- Technologies du web
- Instructions d'affichage
- Les variables
- Les structures de contrôle
- Les structures répétitives
- Les fonctions

# Introduction – la communication sur le web

- Il existe deux types de sites web : **les sites statiques** et **les sites dynamiques.**
- Les sites dynamiques utilisent d'autres langages en plus de HTML et CSS, tels que PHP, JAVA, Python ou autres.
- Le HTML n'est pas un langage de programmation c'est un langage de balisage d'hypertexte qui est utilisé pour créer des pages web.

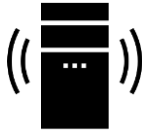


# Introduction – la communication sur le web

Navigateur


Internet


Serveur



Disque dur


1. L'utilisateur tape URL du site à visiter: esi.ac.ma

2. Le navigateur recherche l'adresse IP de esi.ac.ma

3. Le navigateur émet une requête pour afficher la page d'accueil de esi.ac.ma

http

4. La requête http traverse internet et arrive au serveur esi.ac.ma

5. Le serveur web, ayant reçu la demande, recherche la page web sur son disque.

6. Le serveur web récupère la page et la renvoie au navigateur.

http

7. Le navigateur affiche la page web (HTML)

# Technologies du web

- Plusieurs technologies peuvent être utilisées pour construire une application web dynamique.
- On distingue entre les technologies web utilisées côté client (Front-end) et coté serveur (Back-end)
- Le back-end et le front-end sont complémentaires.

| Technologies web (Front-end) | Technologie web (Back-end) |
|---|---|
| HTML | PHP |
| JavaScript | ASP |
| CSS | Java |
| React (Javascript) | Python |
| Angular (Javascript) | Node.js (Javascript) |
| Vue JS | Ruby, Go, C#. etc |

# Le langage php

- PHP est l'un des langages de script côté serveur les plus populaires à l'heure actuelle.

- PHP peut être déployé sur la plupart des serveurs Web sur presque tous les systèmes d'exploitation et plates - formes , gratuitement.

- PHP signifiait à l'origine Personal Home Page, qui est maintenant officiellement connu sous le nom de Php Hypertext Preprocessor a été publié en l'année 1995 par Rasmus Lerdorf.

- PHP est un logiciel libre, Open Source publié sous la licence PHP.

- PHP est un langage interprété orienté Web. Syntaxiquement, c'est un mélange de C et de Perl. Les scripts PHP sont lus et interprétés par le moteur PHP

# Le langage php

- PHP est installé sur plus de 20 millions de sites web et sur plus d'1 million de serveurs web.

- Certaines des plus grandes marques en ligne, telles que Facebook, Digg, Flickr, Technorati, et Yahoo ! Le Monde (http ://lemonde..fr/) sont développées par PHP.

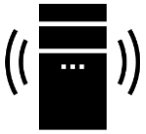
- Les CMS Content Management Systems les plus connus utilisent PHP, tels que:
  - Wordpress
  - Joomla
  - Drupal
  - Prestashop (e-commerce)
  - Magento (e-commerce)
- De grands Frameworks de développement Web aussi sont disponibles en PHP : Symfony, Zend, Laravel, Phalcon ,CakePHP, Yii, Slim

# Langage php

Navigateur

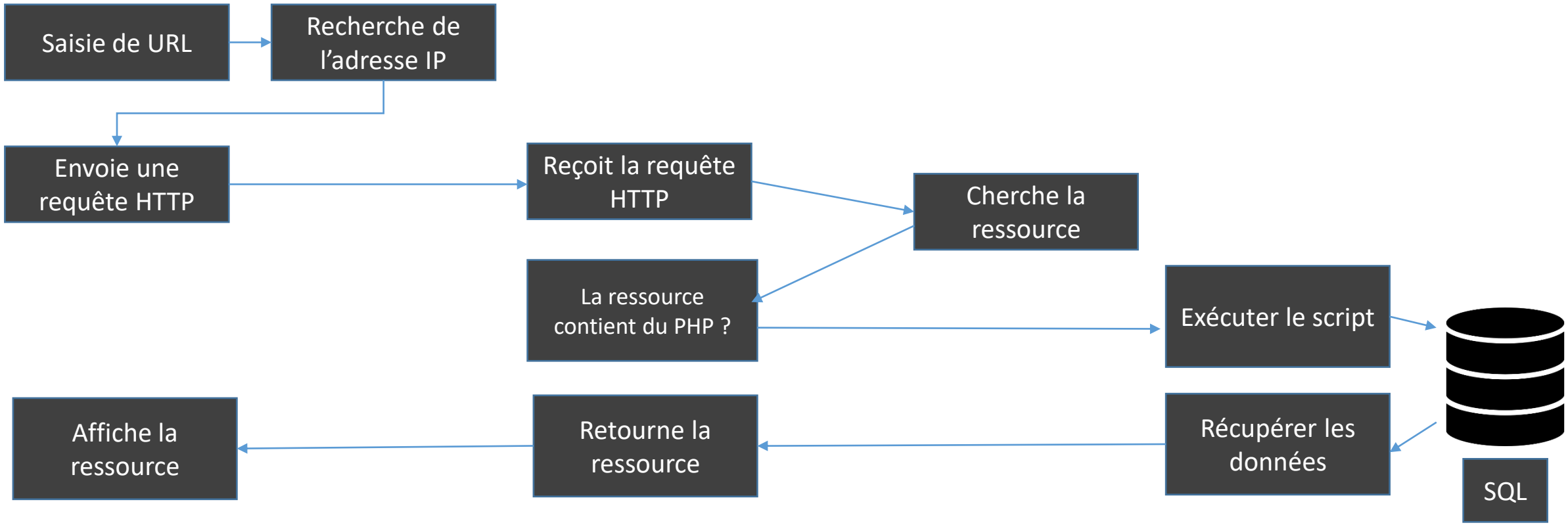

Internet


Serveur



Disque dur


Processeur PHP

# Le langage php

- Le PHP est indiqué dans la page par des balises d'ouverture et de fermeture **<?php et ?>**
- les instructions PHP se terminent par un point-virgule
- Toute ligne située à l'extérieur d'un bloc PHP ( entre <?php et ?>) est recopiée inchangée dans le flux de sortie comme un flux html.
- Chaque fichier PHP porte l'extension PHP
- Les commentaires sont soit entre /* et */ soit après //
- Sensible à la casse

```
<?php
//un script PHP
/* commentaire sur plusieurs
Lignes*/
?>
```

# Le langage PHP- Premier exemple

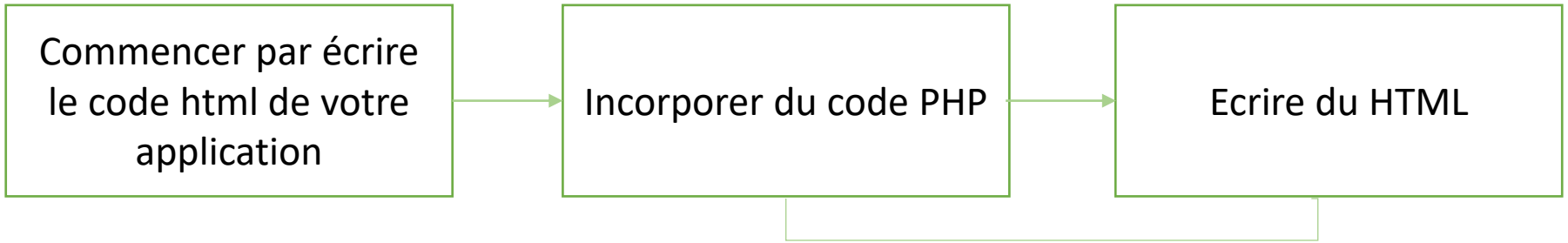
```php
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title> Exemple de script PHP </title>
</head>
<body>
<?php echo 'Bonjour généré dynamiquement en PHP !'; ?>
</body>
</html>
```

Exemple de script en PHP

```html
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title> Exemple de script PHP </title>
</head>
<body>
Bonjour généré dynamiquement en PHP !
</body>
</html>
```

Le PHP traduit en HTML

# Comment ça fonctionne




```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title> Exemple</title>
</head>
<body>
<?php echo 'Exemple de script PHP !'; ?>
</body>
</html>
```

HTML

PHP

HTML

# Instructions d'affichage

En utilisant echo: **echo Expression;**

- echo "Bonjour";
- echo (1+3)*15;
- echo $var;

- En utilisant print: **print(expression);**
  - print("Bonjour");
  - print((1+3)*15);
  - print($var);

- La fonction printf : **printf (chaîne formatée);**
  - printf ("La surface du cervle est %f",$Surface);
  - printf("Mon nom est %s", $nom);

# Instructions d'affichage

On peut inclure des caractères spéciaux pour contrôler le flux affiché :

- \n saut de ligne
- \r fin de ligne
- \t tabulation

Pour afficher le caractère " , on l'insère à l'aide du caractère d'échappement \

```
echo " Exemple d'insertion d'un caractère d'échappement: \" \n";
```

# Les variables

Le symbole « dollar » ($) : précède toujours le nom d'une variable.

La variable doit commencer par une lettre (après le $), ou le caractère _

Les variables n'ont pas besoin d'être déclarées

```php
<?php
   $mycounter = 1;
   $mystring  = "Hello";
   $myarray   = array("One", "Two", "Three");
?>
```

**Affectation par valeur ou par référence**

Affectation par valeur : $b=$a

Affectation par (référence) variable : $c = &$a

# Les variables- les types

En PHP, ce n'est pas nécessaire d'affecter un type à une variable avant de l'utiliser (comme python)

**boolean** : valeurs false et true, insensibles à la casse

**integer** : nombres

**double** : valeurs réelles

**string** : chaines de caractères

**array :** type d'une variable représentant un tableau

**object :** type d'une variable représentant un objet.

# Les variables- les types

Une même variable peut changer de type en cours de script

```
<?
$x=15; // x est une variable integer
$x="Maroc"; // x est maintenant une variable string
?>
```

Il est possible d'effectuer des conversions explicites (transtypage) avec la syntaxe :

Variable = (type) expression;

```
<?
$a=10.12; // x est une variable double
$b=(integer)$x; // b est une variable integer de
valeur 10
?>
```

# Les variables – locale vs globale

**Variable locale**

- Visible uniquement à l'intérieur d'un contexte d'utilisation

**Variable globale**

- Visible dans tout le script
- Utilisation de l'instruction **global()**
- Le script n'affichera rien à l'écran car l'instruction echo utilise la variable locale $a, et celle-ci n'a pas été assignée préalablement dans la fonction. Alors quelle est la solution?

```php
<?php
$a = 1; /* portée globale */

function test()
{
    echo $a; /* portée local
e */
}

test();
?>
```

# Les variables – locale vs globale

Par défaut, toutes les variables sont locales.

Pour rendre une variable globale accessible depuis une fonction, il faut expressément le spécifier à l'aide de l'instruction global suivie des noms des variables qui doivent être accessibles (séparés par des virgules).

```php
<?php
$a = 1;
$b = 2;
function somme() {
    global $a, $b;
    $b = $a + $b;
}
somme();
echo $b;
```

# Les variables – les fonctions isset() et unset()

La fonction **isset()** est une fonction intégrée en PHP qui vérifie si une variable est définie. Cette fonction renvoie TRUE si la variable existe et n'est pas NULL, sinon elle renvoie FALSE.

La fonction **unset()** permet de supprimer la variable ou les variables dont le nom a été passé en argument, et de désallouer la mémoire utilisée.

Pour détruire une variable globale: **unset($GLOBALS['NOM_VARIABLE']);**

```php
<?php
$a = "une variable en PHP";
if(isset($a)) echo "la variable existe";
unset($a);
echo "la variable a été supprimée ...";
?>
```

# Les variables – la fonction gettype()

- La fonction **gettype()** permet de connaître le type de la variable passée en argument.

  Elle renvoie une chaîne : ''string'' ou ''integer'' ou ''double'' ou ''array'' ou ''object''

```php
<?
$a= 10;
echo gettype($a) ; // => "integer"
$b= $a / 5;
echo gettype($a) ; // => "double"
?>
```

# Les variables – autres fonctions

- On peut également tester un type particulier à l'aide des fonctions **is_array, is_string, is_int, is_float, is_object** .

```php
<?
$a= 10;
echo is_int($a); // => (vrai)
echo is_double($a) // => (faux)
echo is_string($a) // => (faux)
$a += 0.5;
echo is_float($a) // => (vrai)
?>
```

Plusieurs autres fonctions prédéfinies à explorer :

- **Doubleval(), empty(), gettype(), intval(), is_array(), is_bool(), is_double(), is_float(), is_int(), is_integer, is_long(), is_object(), is_real(), is_numeric(), is_string(), Isset(), settype(), strval(), unset()**

# Les variables prédéfinis

En PHP, il y’a un grand nombre de variables prédéfinies. Ces variables sont superglobales, elles sont accessibles depuis n'importe où sans notion de portée.

Quelques exemples (la liste est longue!)

| Variable | Description |
|---|---|
| $_SERVER["SERVER_NAME"] | Le nom du serveur |
| $_SERVER["HTTP_HOST"] | Nom de domaine du serveur |
| $_SERVER["SERVER_ADDR"] | Adresse IP du serveur |
| $_SERVER["SERVER_PROTOCOL"] | Nom et version du protocole utilisé pour envoyer la requête au script PHP |
| $_SERVER["DATE_GMT"] | Date actuelle au format GMT |
| $_SERVER["DATE_LOCAL"] | Date actuelle au format local |
| $_SERVER["$DOCUMENT_ROOT"] | Racine des documents Web sur le serveur |

# Les constantes

Utiliser la fonction **define()** pour définir une constante. La fonction retourne la valeur true si la constante a été créé.

- **Define**("nom_constante", valeur_constante )
  - define (“pi”, 3.14) ;
  - define (“promotion”, 2024) ;

Le nom d’une constante ne doit pas commencer par $

Les constantes sont globales; elles peuvent être utilisées à n’importe quel endroit dans un script PHP.

# Les opérateurs en PHP

- `$a == $b` **Egal** TRUE si $a est égal à $b
- `$a === $b` **Identique** TRUE si $a est égal à $b et qu'ils sont du même type
- `$a != $b` **Différent** TRUE si $a est différent de $b
- `$a <> $b` **Différent** TRUE si $a est différent de $b
- `$a !== $b` **Différent** TRUE si $a est différent de $b ou s'ils ne sont pas du même type.
- `$a < $b` **inférieur** TRUE si $a est strictement inférieur à $b
- `$a > $b` **supérieur** TRUE si $a est strictement supérieur à $b
- `$a <= $b` **inférieur ou égal** TRUE si $a est inférieur ou égal à $b

# Les opérateurs en PHP

- PHP supporte les opérateurs de pre- et post- incrémentation et décrémentation,

| | |
|---|---|
| **++$a** | **Pre-incrémente** incrémente $a de 1, puis retourne $a; |
| **$a++** | **Post-incrémente** retourne $a, puis l'incrémente de 1 |
| **--$a** | **Pré-décrémente** décrémente $a de 1, puis retourne $a |
| **$a--** | **Post-décrémente** retourne $a, puis décrémente $a de 1 |

# Les opérateurs sur les chaines de caractères

- Il y a deux opérateurs de chaînes de caractères
  - Le premier est l'opérateur de concaténation ('.'), qui retourne la concaténation de ses deux arguments.
  - Le second est l'opérateur d'assignation concaténant (.=)

```
<? Php
        $a = "Bonjour";
        $b = $a . "le monde";
        $a = "Bonjour";
        $a .= "le monde";
?>
```

# Les opérateurs sur les chaines de caractères

Guillemets simple

- $i= ' PHP' ;
- $j=‘ Bienvenue dans le cours $i';
- echo $j
- **Résultat:**
- Affichage => Bienvenue dans le cours $i

Guillemets double

- $i=" PHP";
- $j= "Bienvenue dans le cours $i";
- echo $j
- **Résultat:**
- Affichage => Bienvenue dans le cours PHP

# Fonctions chaines

- strtolower($str) : conversion en minuscules;
- strtoupper($str) : conversion en majuscules;
- strlen($str) : retourne le nombre de caractères d'une chaîne;
- trim($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 $i;
- strnatcmp($str1,$str2) : comparaison de 2 chaînes ;

# Les structures de contrôle – if – elseif- else

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 (dernière condition réalisée) { série d'instructions }**

# Les structures de contrôle – Opérateur ternaire

Opérateur ternaire

**(condition) ? instruction si vrai : instruction si faux**

```php
<?php
$result = 18;
echo ($result >= 12) ? "Passed" : " Failed";
?>
```

# Les structures de contrôle – Exercice

**Ercire un script PHP qui :**

**1.** initialise 2 variables a et b

**2.** déclare une variable nbmin initialisée à 0

**3.** à l'aide d'un if, faire en sorte que la variable nbmin contienne la valeur minimale de a et b.

**4.** Refaire le même script en utilisant l'opérateur ternaire

# Les structures de contrôle- switch

L'instruction switch permet de comparer une variable à plusieurs valeurs.
C'est beaucoup plus compact et lisible qu'un code de type: if-elseif-elseif...

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

Il ne faut pas oublier le break à la fin de chaque case, sans quoi le reste du switch est exécuté.

Default permet de définir des instructions à effectuer par défaut, c'est à dire si aucune condition n'est vérifiée

# L'instruction switch

```php
<?php
if ($note == 0)
{
    echo "Tu es vraiment un gros nul !!!";
}

elseif ($note == 5)
{
    echo "Tu es très mauvais";
}

elseif ($note == 7)
{
    echo "Tu es mauvais";
}

elseif ($note == 10)
{
    echo "Tu as pile poil la moyenne, c'est un peu juste…";
}

elseif ($note == 12)
{
    echo "Tu es assez bon";
}

elseif ($note == 16)
{
    echo "Tu te débrouilles très bien !";
}

elseif ($note == 20)
{
    echo "Excellent travail, c'est parfait !";
}

else
{
    echo "Désolé, je n'ai pas de message à afficher pour cette note";
}
?>
```

```php
<?php
$note = 10;

switch ($note) // on indique sur quelle variable on travaille
{
    case 0: // dans le cas où $note vaut 0
        echo "Tu es vraiment un gros nul !!!";
    break;

    case 5: // dans le cas où $note vaut 5
        echo "Tu es très mauvais";
    break;

    case 7: // dans le cas où $note vaut 7
        echo "Tu es mauvais";
    break;

    case 10: // etc. etc.
        echo "Tu as pile poil la moyenne, c'est un peu juste…";
    break;

    case 12:
        echo "Tu es assez bon";
    break;

    case 16:
        echo "Tu te débrouilles très bien !";
    break;

    case 20:
        echo "Excellent travail, c'est parfait !";
    break;

    default:
        echo "Désolé, je n'ai pas de message à afficher pour cette note";
}
?>
```

# Les structures de contrôle- while

Elle permet d'exécuter un bloc d'instructions tant que la condition de test est vraie

**While(condition) {bloc d'instructions ;}**

On peut aussi utiliser la syntaxe alternative :

**While (condition) :Instruction1 ;Instruction2 ; …. endwhile ;**

```php
<?php
$num = 1;
while ($num <= 10 ) {
echo $num . "<br />";
$num++;
}
?>
```

# Les structures de contrôle- do while

Elle permet aussi d'exécuter un bloc d'instructions tant que la condition de test est vraie.

**Do {bloc d'instructions ;}while(condition) ;**

Le script entre { } est exécutée au moins une fois, car l'expression conditionnelle est testée en fin de boucle

```php
<?php
$num = 1;
Do {
echo $num . "<br />";
$num++;
}while($num <= 10);
?>
```

# Les structures répétitives - for

Elle permet aussi d'exécuter un bloc d'instructions tant que la condition de test est vraie.

```
for (instructionInit; condition; instructionIter) {
instruction1;
instruction2;
…
}
```

```php
<?php
for ($num = 0; $num <= 10; $num++) {
   echo $num.'<br>';
}
?>
```

L'instruction break permet de sortir d'une boucle à tout moment.
L'instruction continue permet de revenir au début de la boucle

# Les structures répétitives - foreach

Elle permet aussi d'exécuter un bloc d'instructions tant que la condition de test est vraie.

**L'instruction foreach pour les tableaux :**

```
foreach ($t as $valeur) {
...
}
foreach ($t as $cle=>$valeur) {
...
}
```

```php
<?php
$jours=array("Lu"=>"Lundi","Ma"=>"Mardi",
"Me"=>"Mercredi","Je"=>"Jeudi","Ve"=>"Vendredi",
"Sa"=>"Samedi","Di"=>"Dimanche" );
foreach($jours as $key=>$val) echo $key." ".$val.
"<br>\n";
?>
```

# Les fonctions en PHP

Déclaration et appel d'une fonction

```
function fonc($arg1, $arg2, $arg3, ..., $argn) {
 déclaration des variables ;
 bloc d'instructions ;
return $resultat ;
}
```

Une fonction n'a pas besoin d'être déclarée avant d'être utilisée

# Les fonctions en PHP

```
<?php
$var = true;
bar();
/* Impossible d'appeler foo() ici, car cette fonction n'existe pas. Mais nous pouvons utiliser bar() */
if ($var) {
 function foo () {
 echo "Je n'existe pas tant que le programme n'est pas passé ici.\n";
 }
}
/* à ce niveau foo() peut être appelée car $var est maintenant vrai */
if ($var) foo();
function bar() {
 echo "J'existe dès le début du programme.\n";
?>
```

# Les fonctions en PHP

```
<?php
function foo() {
 function bar() {
 echo "Je n'existe pas tant que foo() n'est pas appelé.\n";
}
}
/* Impossible d'appeler bar() ici car il n'existe pas. */
foo();
/* Maintenant, nous pouvons appeler bar(), car l'utilisation de foo() l'a rendu accessible. */
bar();

?>
```

# Les fonctions en PHP

Il est possible de passer des paramètres par défaut.

Attention! Tous les paramètres qui ont une valeur par défaut doivent figurer en dernier lors de la définition.

```php
function ttc($prix, $tva=0.2) {
return $prix-=$prix*$tva;
}
echo "Prix ttc = " . ttc(2000,0.5);
echo "Prix ttc = " . ttc(2000);
$montant=2000;
echo "Prix ttc = " . ttc($montant);
```

# Les fonctions en PHP

Par défaut, les paramètres sont passés par valeur.

Pour passer une variable par référence, il faut que son nom soit précédé du symbole  & (exemple &$a)

```php
function échanger(&$a,&$b) {
$tmp = $a;
$a = $b;
$b = $tmp;
}
$x=1; $y=10;
échanger($x,$y);
echo "$x $y"; // affiche 10  1
?>
```

# Les fonctions en PHP

Il est possible de retourner plusieurs valeurs en utilisant un tableau

```php
<?php
function getArray() {
    return array(1, 2, 3);
}
$secondElement = getArray()[1];
?>
```

```php
<?php
    function func() {
        $array["a"] = "Foo";
        $array["b"] = "Bar";
        $array["c"] = "Baz";
        return $array;
    }

    $foo = func();
    echo $foo["b"] // cela affiche "Bar"
?>
```

# Les fonctions en PHP - exercice

Ecrire une fonction en php qui permet de vérifier si une chaine de caractères est palindrome. Un mot est dit palindrome si on peut le lire indifféremment de gauche à droite ou de droite à gauche en gardant le même sens (Exemple été, colloc, elle, ici, kayak, etc).

```php
function palindrome($chaine)
{
$chaine1="";
for ($i=strlen($chaine);$i>=0;$i--)
{
$chaine1=$chaine1.substr($chaine,$i,1);
}
if ($chaine==$chaine1)
return 1;
else
return 0;
}
```

```php
//appel de la fonction
$s="colloc";
if (palindrome($s))
echo $s." est un palindrome";
else
echo $s." n'est un palindrome";
```

# Les tableaux(1)

En php, un tableau est créé avec la fonction array() qui prend comme arguments des paires « key =>value » séparées par des virgules.

Les éléments d'un tableau peuvent pointer vers d'autres tableaux

Les éléments d'un tableau peuvent appartenir à des types distincts

L'indice d'un tableau en PHP commence de 0

Pas de limites supérieures pour les tableaux

La fonction **count()** renvoie le nombre d'éléments d'un tableau

# Les tableaux - exemple

**Exemple de déclaration d'un tableau d'entiers**

```
$var=array(10,15,17,23,9);
echo $var[0]; // 10
echo $var[2]; // 17
```

**Exemple de déclaration d'un tableau de chaines de caractères**

```
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi",
"Samedi");
$JJ = $jour[3]; // affecte "Mercredi" à $JJ
```

**L'utilisation de la fonction array n'est pas obligatoire et on peut déclarer un tableau à la volée.**

```
$tab2[0]=2;
$tab2[]=6; // equivaut $tab2[1]=6
$tab2['test']='Ma chaîne';
```

# Les tableaux – tableaux multidimensionnels

En php, il est possible de créer des tableaux multidimensionnels, c'est un tableau de tableaux

```php
<?php
echo " <br>matrice<br>";
$matrice = array();
 $matrice[0] = array('X','O','X');
 $matrice[1] = array('X','X','O');
 $matrice[2] = array('X','O','O');

for ($row = 0; $row < 3; $row++) {
  for ($col = 0; $col < 3; $col++) {
    echo "\r[".$matrice[$row][$col]."]\t";
  }
  echo "<br>";
}
```

[X] [O] [X]
[X] [X] [O]
[X] [O] [O]

# Les tableaux - foreach

Il existe une instruction très pratique (foreach) pour afficher le contenu d'un tableau, ou pour modifier les éléments :

```php
<?php
$tab=array(1=>'un',2=>'deux',3=>'trois');
foreach($tab as $valeur)
{ echo "$valeur \n";// affiche un deux trois
}
foreach($tab as $cle => $valeur) {
echo "$cle => $valeur\n"; // affiche 1 => un, 2 => deux,
3 => trois
?>
```

# Les tableaux - Exercice

Écrire un script php permettant d'initialiser puis de parcourir et afficher le contenu du tableau ci-dessous

| Nom | Stock | Vendu |
|---|---|---|
| Volvo | 22 | 18 |
| BMW | 15 | 13 |
| Saab | 5 | 2 |
| Land Rover | 17 | 15 |

# Les tableaux – Solution exercice

```php
<?
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);
for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

# Tri d'un tableau

**sort() - trie les tableaux dans l'ordre croissant**

**rsort() - trie les tableaux dans l'ordre décroissant**

**asort() - Trie les tableaux associatifs dans l'ordre croissant, en fonction de la valeur.**

**ksort() - Trie les tableaux associatifs dans l'ordre croissant, en fonction de la clé**

**arsort() - Trie les tableaux associatifs par ordre décroissant, en fonction de la valeur**

**krsort() - Trie les tableaux associatifs par ordre décroissant, en fonction de la clé**

# Tri d'un tableau - exemple

```php
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);
for($i = 0; $i < count($numbers); $i++) {
  echo $numbers[$i];
  echo "<br>";
}
?>
```

Résultat

2
4
6
11
22

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);
for($i = 0; $i < count($cars); $i++) {
  echo $cars[$i];
  echo "<br>";
}
?>
```

Résultat

Volvo
Toyota
BMW

# La fonction array_filter()

array_filter($variable, "fonction") retourne un tableau contenant les enregistrements filtrés d'un tableau à partir d'une fonction.

```php
<?php
function impair($var)
{return ($var % 2 == 1);}

function pair($var)
{return ($var % 2 == 0);}

$array1 = array ("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);
$array2 = array (6, 7, 8, 9, 10, 11, 12);
echo "Impairs :\n";
print_r(array_filter($array1, "impair"));
echo "Pairs :\n";
print_r(array_filter($array2, "pair"));
?>
```

Exemple pour filtrer les nombres pairs & impairs

# Les fonctions de tableaux

PHP offre une grande liste de fonctions de manipulation de tableaux.

- count(): retourne le nombre d'éléments d'un tableau
- Unset(): supprime un élément d'un tableau
- List(): permet de récupérer les éléments d'un tableau dans des variables différentes:
  - *list ($var1, $var2, ..) = $tableau*
  - Le tableau doit obligatoirement être indexé: 0, 1, …
  - Elle peut être utilisée pour parcourir les éléments d'un tableau

# Les fonctions de tableaux- implode

- **implode():** permet de construire une chaîne de caractères constituée des éléments d'un tableau. Les éléments sont séparés par un symbole ou un séparateur qu'il faut préciser comme 1er paramètre de la fonction.

**$chaine = implode($separateur, $tableau);**

Exemple:

```
function printImplode($t)
{     $s = implode(“ , ”, $t);
      Print ($s);
}
```

# Les fonctions de tableaux- explode()

- **explode():** permet de faire le travail inverse de la fonction **implode()**. Ainsi on peut éclater une chaîne en un tableau d'éléments. Les éléments obtenus sont des sous-chaînes séparées dans la chaîne par un même séparateur qu'on précise comme 1$^{er}$ paramètre de la fonction.

**$tableau = explode($separateur, $chaine , [ max]);**

- Le 3$^{ème}$ paramètre est optionnel, utilisé pour limiter le nombre d'éléments à extraire. Si *max* est défini, le tableau contiendra *max* éléments. Et le dernier élément contiendra le reste de la chaîne.

# Les fonctions de tableaux- parcours d'un tableau

- Reset(): **$element = reset($tableau)**: pointer sur le premier élément et retourner sa valeur
- Next(): **$element = next($tableau)**: pointe sur l'élément suivant et retourne sa valeur
- Prev(): **$element = prev($tableau)**: pointe sur l'élément précédent et retourne sa valeur
- End(): **$element = end($tableau)**: pointe sur le dernier éléments et retourne sa valeur

# Les fonctions de tableaux- affichage

- Print_r(): obtenir une représentation texte d'un tableau.

```
Array ( [1] => elt1 [2] => 78 )
```

- La fonction var_dump() qui est disponible pour tous les types de données permet d'obtenir encore davantage d'informations.

```
array (size=2)
  1 => string 'elt1' (length=4)
  2 => int 78
```

# Les fonctions de tableaux- recherche

- ***Array_key_exists('clé', tableau)***:
  - Permet de vérifier si dans un tableau la clé passée en argument existe.
  - Elle retourne un Boolean.

```
$result = array_key_exists('key', $tableau);
```

- ***In_array(value,tableau)***:
  - Permet de vérifier si la valeur passée en argument existe dans le tableau.
  - Elle retourne un Boolean aussi.

```
$result= in_array(value, $tableau)
```

- ***Array_search(***): idem

```
$result = array_search(value, $tableau)
```