# **PHP** : les fondamentaux


Zinedine CHEBOUBA
Imane LOUNISSA

# Plan

# Plan

# Plan

# PHP

## PHP

- Initialement pour **P**ersonal **H**ome **P**age ensuite pour **P**HP : **H**ypertext **P**reprocessor)
- langage de programmation open-source
  - orienté-objet et procédural
  - impératif
  - interprété
  - faiblement typé
- créé en 1994 par **Rasmus Lerdorf** dans le cadre d'un projet personnel pour gérer les visiteurs de son site web
- syntaxe très proche du C (procédural), C++ (procédural, orienté-objet) et Java (orienté-objet)

# PHP

## PHP, pourquoi ?

- Langage de haut niveau (pas de gestion de mémoire, pas d'allocation dynamique, pas de pointeur... comme en C et C++)
- Facile à apprendre et à utiliser
- Nombreuses documentations, supports vidéos, plusieurs exemples sur internet
- Énorme communauté : un des langages les plus utilisés dans le monde
- Permettant de développer rapidement des programmes portables : Windows, Mac OS, Linux...

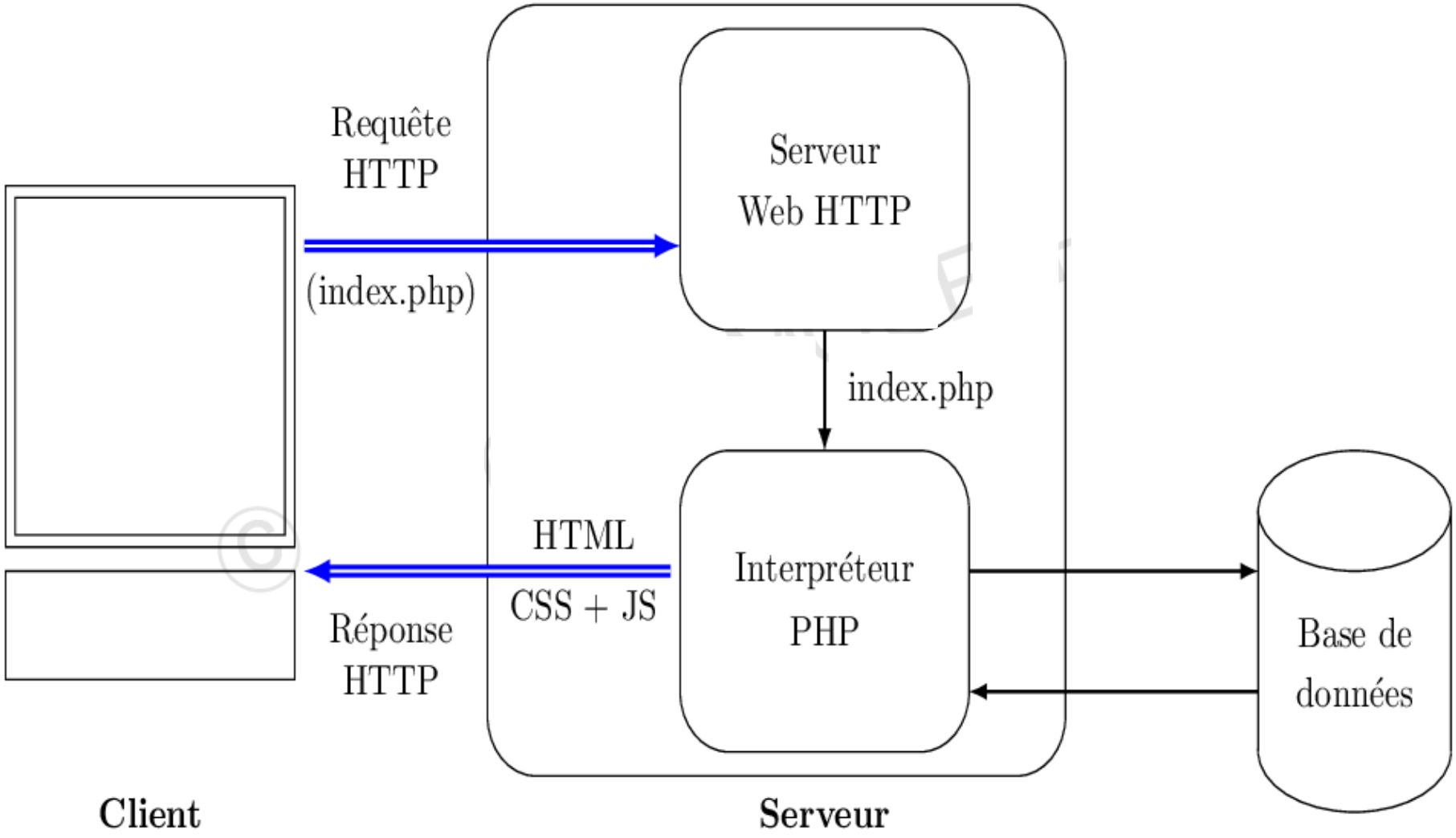
## Quel type d'application ?

- Applications utilisables en ligne de commande (scripting)
- La partie serveur pour
  - applications web
  - applications mobiles
  - sites web dynamiques
- Services web

## Dates de sorties de quelques versions de PHP

- 8 juin 1995 : Version 1.0.
- 1er novembre 1997 : Version 2.0.
- 6 juin 1998 : Version 3.0.
- 22 mai 2000 : Version 4.0.
- 10 décembre 2001 : Version 4.1.
- 13 juillet 2004 : Version 5.0.
- 24 novembre 2005 : Version 5.1.
- 2 novembre 2006 : Version 5.2.
- 30 juin 2009 : Version 5.3.
- 1er mars 2012 : Version 5.4.
- 20 juin 2013 : Version 5.5.
- 28 août 2014 : Version 5.6.
- 3 décembre 2015 : Version 7.0.
- 1 décembre 2016 : Version 7.1.
- 30 novembre 2017 : Version 7.2.
- 6 décembre 2018 : Version 7.3.
- 21 novembre 2019 : Version 7.4.

# PHP


Requête
HTTP
(index.php)
Serveur
Web HTTP
index.php
HTML
CSS + JS
Interpréteur
PHP
Réponse
HTTP
Base de
données
Client
Serveur

# PHP

## De quoi on a besoin (le minimum) ?

- Un éditeur de texte (Bloc-notes, Notepad++, Sublime Text...)
- Un serveur web (**Apache**)
- Un interpréteur du langage **PHP**
- Et probablement un système de gestion de base de données (généralement **MySQL**)

# PHP

## De quoi on a besoin (le minimum) ?

- Un éditeur de texte (Bloc-notes, Notepad++, Sublime Text...)
- Un serveur web (**Apache**)
- Un interpréteur du langage **PHP**
- Et probablement un système de gestion de base de données (généralement **MySQL**)

## Sous Windows, on peut les trouver dans WAMP

`http://www.wampserver.com/`

# PHP

## Première utilisation de WAMP

- Démarrer WAMP
- Cliquer sur WAMP dans la barre de démarrage et choisir `Redémarrer les services`
- Si l'icône de WAMP n'est pas en vert, aller vérifier `http://forum.wampserver.com/read.php?1,88043`

# PHP

## Première utilisation de WAMP

- Démarrer WAMP
- Cliquer sur WAMP dans la barre de démarrage et choisir `Redémarrer les services`
- Si l'icône de WAMP n'est pas en vert, aller vérifier `http://forum.wampserver.com/read.php?1,88043`

## Quelques éléments dans le menu de démarrage de WAMP

- `localhost` : page de démarrage de WAMP
- `phpMyAdmin` : page web permettant la gestion des bases de données MySQL
- `Répertoire www` : emplacement des projets PHP sur le disque dur
- ...

# PHP

## On peut utiliser un IDE (Environnement de développement intégré)

- pour éviter d'utiliser la console et les commandes
- car un IDE intègre un compilateur lancé même pendant l'écriture du code
- pour profiter de la coloration syntaxique, l'auto-complétion, l'indentation automatique...
- pour avoir une bonne structuration du projet

# PHP

## Les règles de nommage en PHP

- Pour les classes : **Le Pascal case**
- Pour les méthodes, fonctions et variables : **Le snake case**
- Pour les noms de projets : **Le Kebab case**

| Convention de dénomination | Format |
|---|---|
| Camel Case (Lower Camel Case, Dromedary Case) | camelCase |
| Kebab Case (Dash Case, Lisp Case, Spinal Case) | kebab-case |
| Snake Case (Pothole Case) | snake_case |
| Pascal Case (Upper Camel Case, Studly Case) | PascalCase |

# PHP

## Les règles de nommage en PHP

- Pour les classes : **Le Pascal case**
- Pour les méthodes, fonctions et variables : **Le snake case**
- Pour les noms de projets : **Le Kebab case**

## Remarque

Certains anciens éléments PHP prédéfinis ne respectent pas ces règles.

# PHP

## Le code PHP

- Les fichiers contenant un code PHP doivent avoir l'extension `.php`
- Un bloc de code PHP est situé entre les deux balises suivantes `<?php ... ?>`
- Une page PHP est une page HTML qui contient une ou plusieurs balises PHP

# PHP

## Les instructions

- Chaque instruction se termine par `;`
- Il est possible d'écrire plusieurs instructions sur une même ligne (**mais** ce n'est pas une bonne pratique)

# PHP

## Exemple

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Première page PHP</title>
</head>
<body>
<?php
    echo "Hello world";
?>
</body>
</html>
```

# PHP

## Fonctions d'affichage : `echo` et `print`

- utilisables avec et sans parenthèses.
- `echo` n'a pas de valeur de retour.
- `echo` sans parenthèse peut accepter plusieurs paramètres séparés par des virgules.
- `print` retourne toujours 1, donc elle peut être utilisée dans des expressions.
- `print` n'accepte qu'un seul paramètre.
- `echo` est légèrement plus rapide que `print`.

# PHP

## Les commentaires

Trois types de commentaires

# PHP

## Les commentaires

Trois types de commentaires

### Commentaire mono-ligne

```
// ceci est un commentaire sur une seule ligne
```

# PHP

## Les commentaires

Trois types de commentaires

### Commentaire mono-ligne

```
// ceci est un commentaire sur une seule ligne
```

### Commentaire sur plusieurs lignes

```
/* ceci est un
commentaire
sur trois lignes */
```

# PHP

## Les commentaires

Trois types de commentaires

### Commentaire mono-ligne

```
// ceci est un commentaire sur une seule ligne
```

### Commentaire sur plusieurs lignes

```
/* ceci est un
commentaire
sur trois lignes */
```

# PHP

## Exemple 1

```
$x = 5;
```

# PHP

## Exemple 1

```
$x = 5;
```

## Exemple 2

```
$_x = 5;
```

# PHP

## Exemple 1

```
$x = 5;
```

## Exemple 2

```
$_x = 5;
```

## Exemple 3

```
$ma_variable = 5;
```

# PHP

## Quatre types primitifs (scalaires) de variable selon la valeur affectée

- `int`
- `string`
- `boolean`
- `float` (ou `double`)

# PHP

## Exemple avec `integer`

```php
$var = 5;
echo "$var ", gettype($var), "<br>";
/* affiche 5 integer */
```

# PHP

## Exemple avec `integer`

```
$var = 5;
echo "$var ", gettype($var), "<br>";
/* affiche 5 integer */
```

## Exemple avec `double`

```
$var = 5.5;
echo "$var ", gettype($var), "<br>";
/* affiche 5.5 double */
```

# PHP

**Exemple avec** `integer`

```
$var = 5;
echo "$var ", gettype($var), "<br>";
/* affiche 5 integer */
```

**Exemple avec** `double`

```
$var = 5.5;
echo "$var ", gettype($var), "<br>";
/* affiche 5.5 double */
```

**Exemple avec** `boolean`

```
$var = TRUE;
echo "$var ", gettype($var), "<br>";
/* affiche 1 boolean */
```

# PHP

**Exemple avec** `string`

```php
$var = "hello";
echo "$var ", gettype($var), "<br>";
/* affiche hello string */
```

# PHP

**Exemple avec** `string`

```
$var = "hello";
echo "$var ", gettype($var), "<br>";
/* affiche hello string */
```

**Un seul caractère est aussi de type** `string`

```
$var = "h";
echo "$var ", gettype($var), "<br>";
/* affiche h string */
```

# PHP

**Exemple avec** `string`

```
$var = "hello";
echo "$var ", gettype($var), "<br>";
/* affiche hello string */
```

**Un seul caractère est aussi de type** `string`

```
$var = "h";
echo "$var ", gettype($var), "<br>";
/* affiche h string */
```

**Pour afficher plusieurs informations sur une variable, on utilise** `var_dump()`

```
$var = "hello";
echo  var_dump($var) ,"<br>";
/* affiche C:\wamp64\www\premier-cours-php\index.php:17:string
  'hello' (length=5)*/
```

# PHP

## Remarques

- Pour vérifier un type, on peut utiliser la fonction `is_type($nom_variable)` : type à remplacer par le type que l'on cherche à vérifier
- Pour les booléens, il faut utiliser `is_bool($nom_variable)`.
- Ces fonctions retournent `1` si le type de la valeur de la variable passée en paramètre est vérifié, elles ne retournent rien sinon.

# PHP

## Remarques

- Pour vérifier un type, on peut utiliser la fonction `is_type($nom_variable)` : type à remplacer par le type que l'on cherche à vérifier
- Pour les booléens, il faut utiliser `is_bool($nom_variable)`.
- Ces fonctions retournent `1` si le type de la valeur de la variable passée en paramètre est vérifié, elles ne retournent rien sinon.

**Exemple avec** `is_string()`

```
$var = "hello";
echo is_string($var) ,"<br>";
/* affiche 1 */

echo is_bool($var) ,"<br>";
/* n'affiche rien */
```

# PHP

## Explication

**PHP** convertit une valeur booléenne `TRUE` en la chaîne ”1” et `FALSE` en ”” (la chaîne vide), par conséquence

- `echo true` affiche `1`
- `echo false` n’affiche rien

# PHP

**La fonction `is_numeric` permet de vérifier si le contenu d'une variable est numérique**

```php
echo(is_numeric(2));
/* affiche 1 */

echo(is_numeric(2.5));
/* affiche 1 */

echo(is_numeric("2"));
/* affiche 1 */

echo(is_numeric("-2.5"));
/* affiche 1 */

echo(is_numeric(true));
/* n'affiche rien */

echo(is_numeric("2a"));
/* n'affiche rien */

echo(is_numeric("a"));
/* n'affiche rien */
```

# PHP

## Autres fonctions `is_*`

- `is_float()`
- `is_real()` (ou son alias `is_float()`)
- `is_object()`
- `is_array()`
- `is_scalar()` pour tester si une variable est de type entier, nombre décimal, chaîne de caractères ou booléen.

# PHP

**Utiliser une variable non-déclarée et non-initialisée ⇒ Undefined variable**

```
$x = $y + 1;

echo  $x;
```

# PHP

**La constante `NULL` peut-être utilisée pour créer une variable sans l'initialiser. Cette dernière sera de type `null`**

```php
$var = NULL;

echo $var;
/* n'affiche rien */

echo gettype($var);
/* affiche NULL */
```

# PHP

**La fonction `isset` peut-être utilisée pour vérifier si une variable est déclarée et est différente de `NULL`**

```php
$var = NULL;

echo isset($var);
/* n'affiche rien */

$var = 2;
echo isset($var);
/* affiche 1 */
```

# PHP

**La fonction `isset` peut-être utilisée pour vérifier si une variable est déclarée et est différente de `NULL`**

```php
$var = NULL;

echo isset($var);
/* n'affiche rien */

$var = 2;
echo isset($var);
/* affiche 1 */
```

`empty()` **vs** `isset()` **vs** `is_null()`

| | "" | "wick" | NULL | false | true | 0 |
|---|---|---|---|---|---|---|
| empty() | true | false | true | true | false | true |
| isset() | true | true | false | true | true | true |
| is_null() | false | false | true | false | false | false |

# PHP

**La fonction `unset` peut-être utilisée pour détruire la variable passée en paramètre**

```php
$var = 2;
echo isset($var);
/* affiche 1 */

unset($var);
echo isset($var);
/* n'affiche rien */
```

# PHP

**Différence entre ”guillemets” et ’apostrophes’ avec** `echo`

```php
$var = 2;
echo "Contenu de ma variable : $var"
/* affiche Contenu de ma variable : 2 */

echo 'Contenu de ma variable : $var'
/* affiche Contenu de ma variable : $var */
```

# PHP

## Opérateurs arithmétiques

- $=$ : affectation
- $+$ : addition
- $-$ : soustraction
- $*$ : multiplication
- $/$ : division
- $\%$ : reste de la division
- $**$ : exponentiel

# PHP

## Quelques exemples avec l'addition

```
$x = 1;
$y = 3;
$z = '8';
$t = "2";
$n = 2.5;
echo($x + $y ."<br>"); /* 4 */
echo($x + $z ."<br>"); /* 9 */
echo($x + $t ."<br>"); /* 3 */
echo($x + $y + $z ."<br>"); /* 12 */
echo($z + $y + $x ."<br>"); /* 12 */
echo($x + $n ."<br>"); /* 3.5 */
```

# PHP

## Quelques raccourcis

- `$i++;` ≡ `$i = $i + 1;`
- `$i--;` ≡ `$i = $i - 1;`
- `$i += 2;` ≡ `$i = $i + 2;`
- `$i -= 3;` ≡ `$i = $i - 3;`
- `$i *= 2;` ≡ `$i = $i * 2;`
- `$i /= 3;` ≡ `$i = $i / 3;`
- `$i %= 5;` ≡ `$i = $i % 5;`

# PHP

## Exemple de post-incrémentation

```
$i = 2;
$j = $i++;
echo($i); /* affiche 3 */
echo($j); /* affiche 2 */
```

# PHP

## Exemple de post-incrémentation

```
$i = 2;
$j = $i++;
echo($i); /* affiche 3 */
echo($j); /* affiche 2 */
```

## Exemple de pre-incrémentation

```
$i = 2;
$j = ++$i;
echo($i); /* affiche 3 */
echo($j); /* affiche 3 */
```

# PHP

**Pour permuter le contenu de deux variables, on peut utiliser la décomposition**

```
$a = 2;
$b = 0;
[$a, $b] = [$b, $a];
echo $a . " " . $b;
/* affiche 0 2 */
```

# PHP

**Pour permuter le contenu de deux variables, on peut utiliser la décomposition**

```
$a = 2;
$b = 0;
[$a, $b] = [$b, $a];
echo $a . " " . $b;
/* affiche 0 2 */
```

# PHP

## L'opérateur . pour concaténer deux chaînes de caractères

```
$string = "bon";
$string2 = "jour";
$str_concat = $string . $string2;
echo  $str_concat;
/* affiche bonjour */
```

# PHP

## L'opérateur . pour concaténer deux chaînes de caractères

```
$string = "bon";
$string2 = "jour";
$str_concat = $string . $string2;
echo  $str_concat;
/* affiche bonjour */
```

## L'opérateur .= permet de faire concaténation + affectation

```
$string = "bon";
$string2 = "jour";
$string .= $string2;
echo  $string;
/* affiche bonjour */
```

# PHP

## La fonction `strcmp()` pour comparer deux chaînes de caractères

```php
$string = "bon";
$string2 = "jour";

echo strcmp($string, $string2);
/* affiche -1 */

echo strcmp($string2, $string);
/* affiche 1 */

echo strcmp($string, $string);
/* affiche 0 */
```

# PHP

## La fonction `strcmp()` pour comparer deux chaînes de caractères

```
$string = "bon";
$string2 = "jour";

echo strcmp($string, $string2);
/* affiche -1 */

echo strcmp($string2, $string);
/* affiche 1 */

echo strcmp($string, $string);
/* affiche 0 */
```

## Pour comparer deux chaînes de caractères, on peut aussi utiliser l'opérateur ==

```
echo $string == $string;
/* affiche 1 */

echo $string == $string2;
/* n'affiche rien */
```

# PHP

## Fonctions utiles pour les chaînes de caractères

- `strlen()` : la longueur de la chaîne
- `strtoupper()` : pour convertir une chaîne de caractères en majuscule
- `strtolower()` : pour convertir une chaîne de caractères en minuscule
- `trim()` : pour supprimer les espaces au début et à la fin (autres variantes : `ltrim()` et `rtrim()`)
- `substr()` : pour extraire une sous-chaîne de caractères
- `strpos()` : pour retourner la position d'une sous-chaîne dans une chaîne, -1 sinon.
- `strcmp($str,$str2)` : pour comparer `str` à `str2`. Elle retourne `0` en cas d'égalité, `1` si le code ASCII du premier caractère différent de la première chaîne est supérieur à celui de la deuxième chaîne, `-1` sinon.
- `str_split()` : pour transformer une chaîne de caractères en tableau de sous-chaînes de caractères
- ...

# PHP

## Pour connaître la longueur d'une chaîne

```
$str = "bonjour";
echo(strlen($str));
/* affiche 7 */
```

# PHP

## Pour connaître la longueur d'une chaîne

```php
$str = "bonjour";
echo(strlen($str));
/* affiche 7 */
```

## Pour supprimer les espaces au début et à la fin de la chaîne

```php
$str = "  bon jour  ";
echo(strlen($str));
/* affiche 12 */

$sans_espace = trim($str);
echo(strlen($sans_espace));
/* affiche 8 */
```

# PHP

**Pour extraire une sous-chaîne à partir de l'indice 3 jusqu'à la fin**

```
$str = "bonjour";
echo(substr($str, 3));
/* affiche jour */
```

# PHP

**Pour extraire une sous-chaîne à partir de l'indice 3 jusqu'à la fin**

```
$str = "bonjour";
echo(substr($str, 3));
/* affiche jour */
```

**On peut aussi préciser le nombre de caractère à extraire**

```
$str = "bonjour";
echo(substr($str, 3, 2));
/* affiche jo */
```

# PHP

**Pour extraire une sous-chaîne à partir de l'indice 3 jusqu'à la fin**

```
$str = "bonjour";
echo(substr($str, 3));
/* affiche jour */
```

**On peut aussi préciser le nombre de caractère à extraire**

```
$str = "bonjour";
echo(substr($str, 3, 2));
/* affiche jo */
```

**Pour extraire les trois derniers caractères, on utilise une valeur négative**

```
$str = "bonjour";
echo(substr($str, -3)); /* eq substr($str, 4) avec 4 = length - 3 */
/* affiche our */
```

# PHP

**Pour déterminer l'indice d'une sous-chaîne dans une chaîne de caractères**

```
$str = "Bonjour les bons jours";
echo(strpos($str, "bon"));
/* affiche 12 */
```

# PHP

**Pour déterminer l'indice d'une sous-chaîne dans une chaîne de caractères**

```
$str = "Bonjour les bons jours";
echo(strpos($str, "bon"));
/* affiche 12 */
```

**Pour une recherche insensible à la casse**

```
$str = "Bonjour les bons jours";
echo(stripos($str, "bon"));
/* affiche 0 */
```

# PHP

**Pour déterminer l'indice d'une sous-chaîne dans une chaîne de caractères**

```
$str = "Bonjour les bons jours";
echo(strpos($str, "bon"));
/* affiche 12 */
```

**Pour une recherche insensible à la casse**

```
$str = "Bonjour les bons jours";
echo(stripos($str, "bon"));
/* affiche 0 */
```

**S'il n'y a aucune occurrence, elle ne retourne rien**

```
$str = "Bonjour les bons jours";
echo(strpos($str, "soir"));
/* n'affiche rien */
```

# PHP

**Pour déterminer la dernière occurrence d'une sous-chaîne dans une chaîne de caractères**

```
$str = "Bonjour les bons jours";
echo(strrpos($str, "jour"));
/* affiche 17 */
```

# PHP

**Pour déterminer la dernière occurrence d'une sous-chaîne dans une chaîne de caractères**

```
$str = "Bonjour les bons jours";
echo(strrpos($str, "jour"));
/* affiche 17 */
```

**Pour déterminer la dernière occurrence d'une sous-chaîne dans une chaîne de caractères (insensible à la casse)**

```
$str = "Bonjour les bons jours";
echo(strripos($str, "bon"));
/* affiche 12 */
```

# PHP

**Pour accéder à un caractère d'indice `i` dans une chaîne de caractères**

```
// soit directement via l'indice
echo($str[i]);
```

# PHP

**Pour accéder à un caractère d'indice `i` dans une chaîne de caractères**

```
// soit directement via l'indice
echo($str[i]);
```

**Ou**

```
/* soit en faisant l'extraction d'une sous chaine de
   caractères */
echo(substr($str, i, 1));
```

# PHP

**Étant données les deux chaînes de caractères suivantes**

```
$ma_chaine = "Hello les holoulos";
$motif = "lo";
```

# PHP

**Étant données les deux chaînes de caractères suivantes**

```
$ma_chaine = "Hello les holoulos";
$motif = "lo";
```

## Exercice

En utilisant les fonctions sur les chaînes de caractères, écrire un script **PHP** qui permet de retourner la position de l'avant dernière occurrence de `$motif` dans `$ma_chaine`.

# PHP

## Correction

```
<?php
    $ma_chaine = "Hello les holoulos";
    $motif = 'lo';
    $sans_dernier_motif = substr($ma_chaine, 0,
       strrpos($ma_chaine, $motif));
    echo "$sans_dernier_motif <br>";
    echo strrpos ($sans_dernier_motif, $motif);
?>
```

# PHP

**Étant données les deux chaînes de caractères suivantes**

```
$ma_chaine = "Hello les holoulos";
$motif = "lo";
```

# PHP

**Étant données les deux chaînes de caractères suivantes**

```
$ma_chaine = "Hello les holoulos";
$motif = "lo";
```

## Exercice

En utilisant les fonctions sur les chaînes de caractères, écrire un script **PHP** qui permet de supprimer l'avant dernière occurrence de `$motif` dans `$ma_chaine`.