Cours PHP

1. Introduction à PHP

- **Langage côté serveur** : génère du HTML envoyé au navigateur.
- Permet de créer des **pages dynamiques**.
- Nécessite : un **serveur web** (Apache, Nginx), **PHP**, et éventuellement une **base de données** (MySQL/MariaDB).
- Les fichiers PHP ont l'extension `.php`.

Rappel : le navigateur ne voit jamais le code PHP, seulement le HTML généré.

---

2. Installation et environnement

En local avec XAMPP

- XAMPP = Apache + MariaDB + PHP + Perl.
- Téléchargez sur : apachefriends.org
- Installation par défaut → Lancer **Apache** et **MySQL**.
- Placer vos fichiers PHP dans `htdocs`.

Exemple :

```
htdocs/
    monsite/
        index.php
```

Accès : `http://localhost/monsite/`

Remarque :

Les fichiers qui contiennent du PHP vont devoir être enregistrés avec l'extension `.php`. **Dans le cas où on intègre du code PHP dans un fichier HTML, il faudra également changer son extension en `.php`.**

---

3. Syntaxe de base

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

- Début : `<?php` / Fin : `?>`
- Les instructions se terminent par `;`
- PHP et HTML peuvent être mélangés.

`echo` (`print`) est utilisé pour l'affichage simple de texte ou de variables.

```php
$txt1 = "PHP";
$txt2 = "Informatique et réseaux";
$x = 5;
$y = 4;

echo "Affichage :";
echo "<h2>" . $txt1 . "</h2>";
echo "Etudier le PHP en " . $txt2 . "<br>";
echo $x + $y;
```

Résultat :

> Affichage :
>
> ## PHP
>
> Etudier le PHP en Informatique et réseaux
>
> 9

`var_dump()` est une fonction de débogage puissante qui affiche non seulement la valeur d'une variable, mais aussi son type et sa structure, notamment pour les tableaux et objets.

Par exemple, `var_dump($variable);` fournit des détails précis sur la variable, ce qui est utile lors du développement pour diagnostiquer des erreurs ou comprendre la structure des données.

```php
$nom = "Carnus";
$age = 20;
$rgb = array("rouge", "vert", "bleu");
$c = [1.5, 2.3, 5];
var_dump($nom);
echo "<br>";
var_dump($age);
echo "<br>";
echo var_dump($rgb) . "<br>";
var_dump($c);
```

Résultat :

> string(6) "Carnus"
>
> int(20)
>
> array(3) { [0]=> string(5) "rouge" [1]=> string(4) "vert" [2]=> string(4) "bleu" }
>
> array(3) { [0]=> float(1.5) [1]=> float(2.3) [2]=> int(5) }

4. Variables et types

Variables

- Commencent par `$`.
- Sensibles à la casse (`$age` ≠ `$Age`).
- Créées lors de la première affectation.

```php
$nom = "Carnus";
$age = 20;
echo "$nom a $age ans";
```

Types principaux

Type	Exemple
String	`$txt = "Bonjour";`
Integer	`$x = 10;`
Float	`$pi = 3.14;`
Boolean	`$ok = true;`
Array	`$tab = ["rouge","bleu"];`
Object	`new MaClasse();`
NULL	`$x = null;`

Opérateurs arithmétiques

Opérateur	Nom	Syntaxe
+	Addition	$x + $y
-	Subtraction	$x - $y
*	Multiplication	$x * $y
/	Division	$x / $y
%	Modulo	$x % $y
**	Exponentiel	$x ** $y

Fonctions mathématiques

```php
echo(pi()); // Sortie 3.1415926535898
echo(min(0, 150, 30, 20, -8, -200)); // Sortie -200
echo(max(0, 150, 30, 20, -8, -200)); // Sortie 150
echo(abs(-6.7)); // Sortie 6.7
```

```php
echo(sqrt(64)); // Sortie 8
echo(round(0.60)); // Sortie 1
echo(round(0.49)); // Sortie 0
echo(rand()); // Nombre aléatoire
echo(rand(10, 100)); // Nombre aléatoire entre 10 et 100
```

Opérateurs de comparaison

Opérateur	Nom	Exemple	Résultat
==	Egal	$x == $y	true si $x est égal à $y
===	Identique	$x === $y	true si $x est égal à $y, et sont du même type
!=	Différent	$x != $y	true si $x est différent de $y
<>	Différent	$x <> $y	true si $x est différent de $y
!==	Non identique	$x !== $y	true si $x est différent de $y, ou ne sont pas du même type
>	Supérieur	$x > $y	true si $x est supérieur à $y
<	Inférieur	$x < $y	true si $x est inférieur à $y
>=	Supérieur ou égal	$x >= $y	true if $x est supérieur ou égal à $y
<=	Inférieur ou égal	$x <= $y	true si $x est inférieur ou égal à $y
<=>	Spaceship	$x <=> $y	Retourne un entier inférieur, égal ou supérieur à zéro, selon que $x est inférieur, égal ou supérieur à $y.

Opérateurs logiques

Opérateur	Nom	Exemple	Résultat
and	And	$x and $y	true si $x et $y sont vrais (true)
or	Or	$x or $y	true si $x ou $y est vrai (true)
xor	Xor	$x xor $y	true si $x ou $y est vrai (true), mais pas les deux
&&	And	$x && $y	true si $x et $y sont vrais (true)
||	Or	$x || $y	true si $x ou $y est vrai (true)
!	Not	!$x	true si $x est faux (false)

Opérateurs de chaîne

Opérateur	Nom	Exemple	Résultat

Opérateur	Nom	Exemple	Résultat
.	Concaténation	`$txt1 . $txt2`	Concaténation de `$txt1` et `$txt2`
.=	Affectation avec concaténation	`$txt1 .= $txt2`	Ajoute `$txt2` à `$txt1`

5. Structures de contrôle

Conditions

```php
if ($x > 0) {
    echo "Positif";
} elseif ($x < 0) {
    echo "Négatif";
} else {
    echo "Zéro";
}

$x = 1;
switch($x){
    case 0:
        echo '$x = 0';
        break;
    case 1:
        echo '$x = 1';
        break;
    case 2:
        echo '$x = 2';
        break;
    default:
        echo '$x contient une autre valeur';
}

$d = 5;
switch ($d) {
  case 6:
    echo "Dimanche";
    break;
  case 5:
    echo "Samedi";
    break;
  default:
    echo "Jour de semaine";
}
```

Boucles

```php
for ($i = 0; $i < 5; $i++) {
    echo $i;
```

```
}

$couleurs = ["rouge", "vert"];
foreach ($couleurs as $c) {
    echo $c;
}

$x = 1;
while($x <= 4) {
    echo "Le nombre est : $x <br>";
    $x++;
}

$y = 1;
do {
    echo "Le nombre est : $y <br>";
    $y++;
} while ($y <= 4);
```

---

6. Fonctions

```
function addition($a, $b = 0) {
    return $a + $b;
}
echo addition(5, 3); // 8
```

- **Paramètres par défaut** : `$b = 0`
- **Retour** : `return`
- **Référence** : `function inc(&$x) { $x++; }`

---

7. Tableaux

```
$marques = ["HP", "Apple", "Dell"];
echo $marques[0]; // HP

$ages = ["Pierre" => 19, "Luc" => 20];
echo $ages["Luc"]; // 20
```

Boucle sur tableau associatif :

```
foreach ($ages as $nom => $age) {
    echo "$nom a $age ans";
}
```

8. Formulaires et validation – Méthodes GET et POST

Différences entre GET et POST

Méthode	Caractéristiques	Usage
GET	Les données sont envoyées dans l'URL (`?nom=Carnus&age=20`)	Liens de recherche, paramètres simples
POST	Les données sont envoyées dans le corps de la requête (invisibles dans l'URL)	Formulaires de connexion, données sensibles

Bonnes pratiques :

- Utiliser **POST** pour toute donnée sensible.
- Utiliser **GET** pour des paramètres qui peuvent être mis en favori ou partagés.

Formulaire HTML - GET

```
<form method="get">
    Nom: <input type="text" name="nom">
    <input type="submit" value="Envoyer">
</form>
```

Traitement PHP

```
<?php
if (isset($_GET['nom'])) {
    echo "Bonjour " . htmlspecialchars($_GET['nom']);
}
?>
```

Formulaire HTML - POST

```
<form method="post">
    Nom: <input type="text" name="nom">
    <input type="submit" value="Envoyer">
</form>
```

Traitement PHP

```php
<?php
if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $nom = trim($_POST["nom"]);
    echo "Bonjour " . htmlspecialchars($nom);
}
?>
```

Validation email

```php
$email = filter_input(INPUT_POST, "email", FILTER_VALIDATE_EMAIL);
if (!$email) {
    echo "Email invalide";
}
```

Remarques

- Toujours vérifier si la variable existe (`isset()` ou `empty()`).
- Toujours nettoyer les données (`trim()`, `htmlspecialchars()`).
- Éviter de mélanger GET et POST pour le même formulaire.

---

9. Inclusion de fichiers

```php
include "header.php";
require "config.php";
```

- `include` → avertissement si absent.
- `require` → erreur bloquante si absent.

---

10. Bonnes pratiques (de sécurité)

- Toujours **valider** les entrées utilisateur.
- Séparer **HTML / PHP** au maximum.
- Utiliser `htmlspecialchars()` pour éviter les failles XSS.
- Activer `error_reporting(E_ALL)` en développement.

Exercices

1. Syntaxe de base

Écrire un script PHP qui affiche :

```
Bonjour Charles Carnus !
```

où "Charles Carnus" est stocké dans une variable.

---

2. Variables et opérateurs

Créer un script qui :

- Définit deux nombres
- Calcule leur somme, leur différence, leur produit et leur quotient
- Affiche chaque résultat avec un message clair

---

3. Conditions

Afficher :

- "Bonjour" si l'heure est < 18h
- "Bonsoir" sinon

Note : utiliser `date("H")`

---

4. Boucles

Afficher tous les nombres de 1 à 10 avec une boucle `for`.

---

5. Tableaux

Créer un tableau avec 5 prénoms et les afficher avec `foreach`.

---

6. Formulaires (GET)

Créer un formulaire demandant le prénom.

Lorsqu'on l'envoie, la page affiche :

```
Bonjour <prénom> !
```

Utiliser `htmlspecialchars()` pour sécuriser.

---

7. Validation (POST)

Créer un formulaire qui demande un email et affiche :

- "Email valide" si c'est un email correct

- "Email invalide" sinon

Utiliser `filter_var()` avec `FILTER_VALIDATE_EMAIL`.

TP – Carnet de contacts

Objectif

Créer une application simple en PHP pour gérer un petit carnet de contacts.

Étape 1 : Page d'accueil

- Affiche un message de bienvenue
- Lien vers "Ajouter un contact" et "Voir la liste"

Étape 2 : Formulaire d'ajout (POST)

- Nom
- Email
- Téléphone

Étape 3 : Validation

- Nom obligatoire
- Email valide
- Téléphone composé uniquement de chiffres

Étape 4 : Stockage

- Utiliser un tableau PHP pour stocker les contacts (en mémoire, pas de base de données pour l'instant)

Étape 5 : Liste des contacts

- Afficher tous les contacts ajoutés
- Afficher un message si aucun contact n'est présent

Étape 6 (optionnelle) : Organisation du code

- Séparer l'en-tête et le pied de page avec `include` / `require`

Remarques :

- Commencer par des scripts simples et tester chaque étape.
- Bien lire les messages d'erreur de PHP.
- Ne pas hésiter à utiliser `var_dump()` pour comprendre ce que contient une variable.
