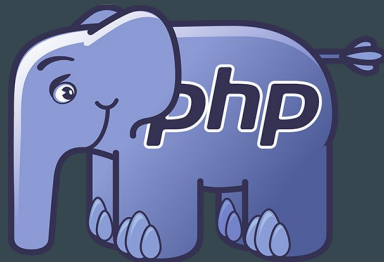


PHP

Partie 2 - Le langage

...

Mathieu Nibas
B3 - ECV 2023



Le programme de cette partie

La syntaxe de base

- Introduction
- Commentaires

Les instructions de base

- Déclarations / affectations
- Les opérateurs
- Les chaînes de caractères
- Les dates
- Les tableaux
- Les constantes du langage
- Les objets

Les structures

- Conditions
- Boucles

Les fonctions

- Appel de fonction
- Déclaration de fonction
- Passage d'arguments

Exercice



La syntaxe de base



La syntaxe de base

Introduction



Principes de base :

- Fichier “.php”
- Balises “<?php” et “?>”
- Mélangeable avec du HTML
- Toute instruction se termine par “;”
- **var_dump** sera votre meilleur ami !

La syntaxe de base

Les commentaires



```
/* Ceci est un commentaire  
qui peut se tenir sur plusieurs lignes  
*/
```

Un commentaire

- un bout de code ignoré par l'interpréteur.
- Une note à l'attention du développeur qui passe après.

En PHP, 2 types de commentaires :

- Commentaire sur une ligne
 - ↳ commence par “//”

```
// commentaire en ligne
```

- Commentaire sur plusieurs lignes
 - ↳ délimité par “/*” et “*/”

Les instructions de base



Les instructions de base

Déclarations / affectations



Affecter une variable \neq déclarer une variable

Déclarer une variable = créer une variable

Affecter une variable = lui donner une valeur

En PHP :

```
// Déclaration de variable
```

```
$variable;
```

```
// Affectation de variable
```

```
$variable = 3;
```

B. Les instructions de base

II. Les opérateurs

Opérateurs arithmétiques :

- + (addition)
- - (soustraction)
- * (multiplication)
- / (division)
- % (modulo)
- ** (puissance)

```
5 + 3; // 8
7 - 1; // 6
4 * 2; // 8
1 / 5; // 0.2
17 % 3; // 2
2 ** 10; // 1024
```



Les instructions base

Les opérateurs

```
1 < 2; // false
3 ≤ 4; // false
5 > 5; // false
6 ≥ 6; // true
4 == 8; // false
9 === 8; // false
2 <> 3; // true
1 ≠ 9; // true
7 ≇ '7'; // true
```

Opérateurs de comparaison :

- < (strictement inférieur)
 - <= (inférieur ou égal)
 - > (strictement supérieur)
 - >= (supérieur ou égal)
 - == (égal)
 - === (identique)
 - <> (non égal)
 - != (non égal)
 - !== (non identique)
-



Les instructions de base

Les opérateurs

Opérateurs logiques :

- **!** (non)
- **&&** (et)
- **AND** (et)
- **||** (ou)
- **OR** (ou)
- **XOR** (ou exclusif)



Tables de vérité des opérateurs logiques

AND / &&		
A	B	A && B
✓	✓	✓
✓	✗	✗
✗	✓	✗
✗	✗	✗

OR /		
A	B	A B
✓	✓	✓
✓	✗	✓
✗	✓	✓
✗	✗	✗

XOR		
A	B	A XOR B
✓	✓	✗
✓	✗	✓
✗	✓	✓
✗	✗	✗



Aparté sur le typage en PHP



Le typage en PHP

PHP est un langage faiblement typé

*Les variables ont un type intrinsèque
mais rien n'empêche un nombre de
devenir une chaîne de caractères.*

Le comportement par défaut est le
transtypage

*PHP va ajuster le type de la variable
pour des opérations intertypes
(addition, soustraction, comparaison,
...).*



**Attention lors des opérations faisant intervenir
des variables de types différents !**

Fin de l'aparté



Les instructions de base

Les chaînes de caractères



En PHP, une chaîne de caractères est délimitée par des guillemets simples ou doubles.

```
$str1 = 'Une chaîne de caractères';  
$str2 = "Une autre chaîne de caractères";
```

Une chaîne entre doubles guillemets interprète les variables.

```
$str3 = "Et encore $str1";  
  
/* $str3 = 'Et encore Une chaîne de caractères'  
*/
```

Les instructions de base

Les chaînes de caractères



On peut concaténer des chaînes de caractères entre elles avec l'opérateur de concaténation “.” :

```
$str4 = 'Et ' . 'encore ' . $str2;  
  
// $str4 = ???
```


Les instructions de base

Les chaînes de caractères



Il existe tout un tas de fonctions pour les chaînes de caractères :

→ *strlen*

→ *strtoupper*

```
$str5 = strtoupper('truc');  
// $str5 = ???  
  
$size = strlen($str4);  
// $size = ???
```

Aparté sur les fonctions



Les fonctions

Une fonction est une suite d'instructions que l'on nomme pour pouvoir le réutiliser à différents endroits de notre code.

Elles peuvent prendre des paramètres, qui lui sont alors donnés entre parenthèses. Ces paramètres servent à adapter le comportement de la fonction à une situation, à la valeur d'une variable, ...

Il existe des fonctions pour tout : écrire des emails, des fichiers, afficher un message, ...

La fonction strlen, par exemple, prend en paramètre... la chaîne de caractères dont on veut connaître la taille.



On verra en détail les fonctions en PHP un peu plus tard

Fin de l'aparté



Les instructions de base

Les dates

En PHP, les dates sont ou bien des *chaînes de caractères*, ou bien des *nombres*, ou bien des *objets*.

On interagit avec par l'intermédiaire de *fonctions*, notamment pour :

- Créer une date
- Faire des calculs de dates
- Formater une date



Les instructions de base

Les tableaux



Un tableau est une variable qui contient plusieurs valeurs, rangées dans des “cases”.

On peut les déclarer en les mettant entre crochets ou bien grâce à la fonction array.

```
<?php  
  
$arr = array(1, 2, 3);  
  
$arr2 = [4, 5, 6];  
  
?>
```

Les instructions de base

Les tableaux



On accède aux valeurs grâce à une clef, que l'on passe entre crochets.

Si on ne la précise pas, PHP les attribue automatiquement en comptant à **partir de zéro**.

```
<?php
    $arr = array(1, 2, 3);

    $arr[1];    // vaut 2 !

?>
```

Les instructions de base

Les tableaux



On peut préciser les clefs que l'on souhaite utiliser.

```
<?php
    $tableau_associatif = array(
        'clef' => 'valeur',
        'clef2' => 'valeur2',
        46 => 72,
        'truc' => 'muche',
        ...
    );

    // Vaut 'valeur2' !
    $tableau_associatif['clef2'];
?>
```


Les instructions de base

Les constantes

Il existe aussi des constantes. Comme leur nom l'indique, elles ne peuvent pas changer.

On déclare une constante avec la fonction **define** : `define('CONSTANTE', 42);`

PHP définit quelques constantes d'office. Par exemple :

- `__DIR__` : le répertoire courant
- `PHP_VERSION` : la version de PHP utilisée



Les instructions de base

Les objets



Un objet est un autre type de variable PHP qui peut contenir *plusieurs* choses.

On peut créer un objet avec le mot-clef **new** :

```
$objet = new Objet();
```

Un objet peut contenir des **variables**, des **fonctions**. On y accède grâce à une **flèche simple** :

```
$objet->variable;  
$objet->fonction();
```

Un objet peut aussi contenir des **constantes**. On y accède grâce à un **double deux-points** :

```
$objet::CONSTANTE;
```

Les structures



Les structures

Conditions



Les conditions servent à **conditionner** le code :

Si [truc] alors [machin]

Sinon si [autre truc] alors [autre machin]

Sinon [bidule]

```
if (truc) {  
    // machin  
} elseif (autre truc) {  
    // autre machin  
} else {  
    // bidule  
}
```

Les structures

Conditions



Il existe aussi les switch, qui permettent de tester la valeur d'une variable et d'exécuter du code selon la valeur.

```
switch ($piece) {  
    case 'pile':  
        echo 'Pierre a gagné.';  
        break;  
    case 'face':  
        echo 'Marie a gagné.';  
        break;  
    default:  
        'La pièce serait-elle tombée sur la tranche ? 0.o';  
}
```

C. Les structures

Boucles



4 types de boucles :

- **for** : *Pour* aller d'une condition de départ à une condition d'arrivée avec des étapes
 - **while** : *Tant que* la condition n'est pas remplie
 - **do while** : La même chose, mais en passant au moins une fois dans la boucle
 - **foreach** : Pour parcourir *chaque* élément d'un tableau
-

Les fonctions



Les fonctions

Appel de fonction



On **appelle** une fonction :

- En donnant son nom
- En donnant ses paramètres
 - ◆ Entre parenthèses
 - ◆ Séparés par des virgules
- En finissant l'instruction par un
“.”
;

Les fonctions

Déclaration de fonction



On **déclare** une fonction :

- En utilisant le mot-clef **function**
 - En donnant le **nom** de la fonction
 - En donnant ses **paramètres**
 - ◆ Entre parenthèses
 - ◆ Séparés par des virgules
 - En donnant le **corps** de la fonction
 - ◆ Entre accolades
 - ◆ En finissant par une instruction **return** (si on souhaite renvoyer quelque chose)
-

Les fonctions

Passage d'arguments



On peut **contraindre** les arguments des fonctions que l'on déclare :

- En leur donnant un **type** : `string`
`$nom`
- En leur donnant une **valeur par défaut** : `$param = une_valeur`

Il faut bien réfléchir à l'ordre des paramètres pour que les optionnels soient à la fin.

Les fonctions

Passage d'arguments



Par défaut une variable est **passée par valeur** => on ne peut pas modifier la variable dans la fonction.

On peut passer une variable **par référence** avec le symbole **&** => on peut modifier la variable dans la fonction.

```
<?php

function modification(&$variable) {

    // Notez le &

    $variable = 3;

}

$variable = 32;

modification($variable);

// $variable vaut à présent 3 !

?>
```

Les fonctions

Passage d'arguments



On peut donner une **infinité de paramètres** avec le *token* **...** => on reçoit un tableau contenant tous les paramètres sous le même nom.

```
<?php // Exemple tiré de la documentation officielle

function sum(...$numbers) {
    $acc = 0;

    foreach ($numbers as $n) {
        $acc += $n;
    }

    return $acc;
}

echo sum(1, 2, 3, 4); // Affiche 10

?>
```

Exercise



Exercice

Hello World !



Faite en sorte que la
fonction **HelloWorld**
retourne exactement la
valeur Hello World!



Exercice

Fonction avec argument



Créer une fonction from scratch qui s'appelle **jeRetourneMonArgument()**.

Exemple :

- Arg = "abc" ==> Return abc
- Arg = 123 ==> Return 123

Exercice

Fonction de concatenation



Créer une fonction from scratch qui s'appelle **concatenation()**. Elle prendra deux arguments de type string.

Elle devra retourner la concatenation des deux.

Exercice

Fonction de somme



Créer une fonction from scratch qui s'appelle **somme()**. Elle prendra deux arguments de type int.

Elle devra retourner la somme des deux.

Exemple :

argument 1 = 5

Argument 2 = 5 ;

Resultat : 10

Exercice

Fonction de soustraction



Créer une fonction from scratch qui s'appelle **soustraction()**. Elle prendra deux arguments de type int.

Elle devra retourner la soustraction des deux.

Exemple : argument 1 = 5
Argument 2 = 5 ; Resultat : 0

Exercice

Fonction de multiplication



Créer une fonction from scratch qui s'appelle **multiplication()**. Elle prendra deux arguments de type int.

Elle devra retourner la multiplication des deux.

Exemple : argument 1 = 5
Argument 2 = 5 ; Resultat :
25

Exercice

Fonction de majorité



Créer une fonction from scratch qui s'appelle **estIlMajeur()**. Elle prendra un argument de type int.

Elle devra retourner un boolean.

Si $\text{age} \geq 18$ elle doit retourner true

Si $\text{age} < 18$ elle doit retourner false

Exemple : $\text{age} = 5 \implies \text{false}$
 $\text{age} = 34 \implies \text{true}$

Exercice

Fonction plusGrand



Créer une fonction from scratch qui s'appelle **plusGrand()**. Elle prendra deux arguments de type int.

Elle devra retourner le plus grand des deux.

Exercice

Fonction plusPetit



Créer une fonction from scratch qui s'appelle **plusPetit()**. Elle prendra trois arguments de type int.

Elle devra retourner le plus petit des trois.



Exercice

Fonction premierElementTableau



Créer une fonction from scratch qui s'appelle **premierElementTableau()**. Elle prendra un argument de type array.

Elle devra retourner le premier élément du tableau.

Si l'array est vide, il faudra retourner `null`;

Exercice

Fonction `dernierElementTableau`



Créer une fonction from scratch qui s'appelle **`dernierElementTableau()`**. Elle prendra un argument de type array.

Elle devra retourner le dernier élément du tableau.

Si l'array est vide, il faudra retourner `null`;

Exercice

Fonction plusGrand



Créer une fonction from scratch qui s'appelle **plusGrand()**. Elle prendra un argument de type array.

Elle devra retourner le plus grand des éléments présent dans l'array. Si l'array est vide, il faudra retourner null;

Exercice

Fonction `verificationPassword`



Créer une fonction from scratch qui s'appelle **`verificationPassword()`**. Elle prendra un argument de type string.

Elle devra retourner un boolean qui vaut `true` si le password fait au moins 8 caractères et `false` si moins.

Exercice

Fonction capital



Créer une fonction from scratch qui s'appelle **capital()**. Elle prendra un argument de type string. Elle devra retourner le nom de la capitale des pays suivants :

France ==> Paris

Allemagne ==> Berlin

Italie ==> Rome

Maroc ==> Rabat

Espagne ==> Madrid

Portugal ==> Lisbonne

Angleterre ==> Londres

Tout autre pays ==> Inconnu

Il faudra utiliser la structure **SWITCH** pour faire cette exercice.

Exercice

Fonction listHTML



Créer une fonction from scratch qui s'appelle **listHTML()**. Elle prendra deux arguments :

- Un string représentant le nom de la liste
- Un array représentant les éléments de cette liste

Elle devra retourner une liste HTML. Chaque element de cette liste viendra du tableau passé en paramètre.

Exemple : Paramètre : Titre : **Capitale** Liste :

["Paris", "Berlin", "Moscou"] Résultat :

—<h3>Capitale</h3>ParisBerlinMoscou

Exercice

Fonction remplacement



Créer une fonction from scratch qui s'appelle **remplacerLesLettres()**. Elle prendra un argument de type string.

Elle devra retourner cette même string mais en remplaçant les e par des 3, les i par des 1 et les o par des 0

Exemple :

→ input : "Bonjour les amis" ==> Output :

`B0nj0ur l3s am1s`

→ input : "Les cours de programmation

Web sont trop cools" ==> Output :

`L3s c0urs d3 pr0grammat10n`

`W3b s0nt tr0ps c00ls`

Exercice

Fonction date



Créer une fonction from scratch
qui s'appelle **quelleAnnee()**.

Elle devra retourner un integer
representant l'année actuelle.

Exercice

Fonction date



Créer une fonction from scratch qui s'appelle **quelleDate()**. Elle devra retourner une string représentant la date actuelle sous le format suivant DD/MM/YYYY

Où *DD* représente le jour actuelle, *MM* le mois actuel et *YYYY* l'année actuelle. Les valeurs doivent être numérique et non au format String.

Exercice

La boucle For

En utilisant la boucle `for`, afficher
la table de multiplication du
chiffre 5.



Exercice

La boucle For

En utilisant deux boucles `for`,
écrire un script qui produit le
résultat ci-dessous.

1
22
333
4444
55555



Exercice

La boucle while

En utilisant la boucle `while`,
afficher tous les codes postaux
possibles pour le département 77.



Exercice

Déclarer une variable avec le nom de votre choix et avec la valeur 0.

Tant que cette variable n'atteint pas 20, il faut :

- l'afficher ;
- incrémenter sa valeur de 2 ;

Si la valeur de la variable est égale à 10, la mettre en valeur en gras avec la balise HTML appropriée.



Exercice

Les tableaux



Déclarer une variable de type array qui stocke les informations suivantes :

- France : Paris
- Allemagne : Berlin
- Italie : Rome

Afficher les valeurs de tous les éléments du tableau en utilisant la boucle `foreach`.

Exercice

Les tableaux



En utilisant la fonction `rand()`,
remplir un tableau avec **10**
nombres aléatoires.

Puis, tester si le chiffre **42** est
dans le tableau et afficher un
message en conséquence.

Enfin, afficher le contenu de
votre tableau avec `var_dump`.

Exercice

Les tableaux



En utilisant la fonction `rand()`, remplir un tableau avec 10 nombres aléatoires.

Puis, trier les valeurs dans deux tableaux distincts.

- Le premier contiendra les valeurs inférieures à 50
- Le second contiendra les valeurs supérieures ou égales à 50.

Enfin, afficher le contenu des deux tableaux.

Exercice

Les tableaux



En utilisant le tableau ci-dessous, afficher seulement les pays qui ont une population supérieure ou égale à 20 millions d'habitants.

```
<?php
    $pays_population = array(
        'France' => 67595000,
        'Suede' => 9998000,
        'Suisse' => 8417000,
        'Kosovo' => 1820631,
        'Malte' => 434403,
        'Mexique' => 122273500,
        'Allemagne' => 82800000,
    );
?>
```

Aparté sur les fonctions de tri



Les fonctions de tri

Tout le monde réfléchit différemment sur ce problème... et c'est **très bien** !

Il existe **plein** de tris différents, qui ne se comportent pas de la même manière.

Il est important de comprendre la **complexité** de chacun et ce qu'elle implique.

Tri rapide

Tri fusion

Tri à bulles

Tri par tas

Tri par insertion

...



Fin de l'aparté

