php


```
 *
 * @var boolean
 */
define('PSI_INTERNAL_XML', false)

if (version_compare("5.2", PHP_V
    die("PHP 5.2 or greater is r

}
if (!extension_loaded("pcre"))
    die("phpSysInfo requires th
        properly.");

}

require_once APP_ROOT.'/includes/autoloader.inc.php';

// Load configuration
require_once APP_ROOT.'/config.php';

if (!defined('PSI_CONFIG_FILE') || !defined('PSI_DEBUG')) {
    $tpl = new Template("/templates/html/error_config.html");
    echo $tpl->fetch();
    die();
                                    t javascript
                                    l . strtolower(
```

Messmer Lucie

Dans ce cours :

- Introduction à PHP
- Déclaration et utilisation des variables
- Les types
- Chaînes de caractères
- Tableaux
- Opérateurs

Exercice PHP_1

Introduction à PHP

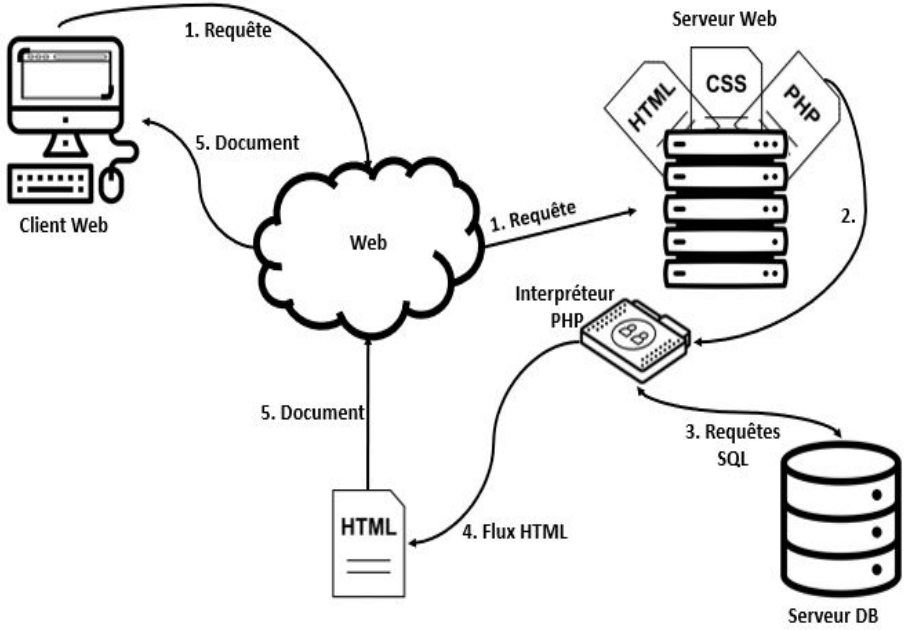
Qu'est-ce que le PHP ?

Le PHP, ou "Hypertext Preprocessor", est un langage de programmation libre principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale.
PHP est un langage impératif (effectue des opérations en séquences d'instructions) orienté objet, similaire à C++.

Pourquoi le PHP ?

Le PHP permet de créer des sites web dynamiques, contrairement aux sites web statiques, créé uniquement en HTML, CSS.
Qu'est-ce qu'un site web dit dynamique ? C'est tout simplement un site qui peut se modifier, se personnaliser, sans l'intervention du webmaster.

PHP



1. Le client, généralement un navigateur web, envoie une requête HTTP au travers d'une URL vers un serveur.
2. Le serveur identifie la page à renvoyer :
   - si c'est un document pouvant être envoyé immédiatement, il l'envoie
   - si c'est un document qui nécessite une interprétation, PHP par exemple, il le traite au préalable.
3. Le document PHP est interprété et au besoin fait appel à une base de données externes.
4. L'interprétation de la page PHP donne lieu à une page HTML, fourni au serveur.
5. Le serveur Web renvoie ce document, cette page obtenue à l'étape 4, au client pour affichage.

Syntaxe de Base

Les balises PHP

Le PHP doit être délimité par les balises <?php ?>.
Ces balises permettent à PHP de traiter seulement leur contenu, et ainsi d'ignorer le reste.

exemple :

Paragraphe lambda ignoré de PHP , mais affiché dans le navigateur
<?php echo "Paragraphe interprété/analysé par PHP. \n ";?>

La syntaxe de PHP est très proche de celle du langage C. L'usage est de placer une instruction par ligne, et chaque instruction doit être clôturée par ; (point-virgule).

Fonctions d'affichage

Le but initial de PHP est de permettre la création de page web dynamiques et donc de pouvoir produire des flux de texte HTML. Deux fonctions principales permettent de produire du texte en sortie :

- echo
- print()

Ils prennent en argument des chaînes de caractères.

Syntaxe :

```
echo "Bonjour";
print("Hello");
```

exemple :

Contenu du fichier PHP

```
<body>
	<?php
		echo '<p>Bonjour à tous</p>';
	?>
</body>
```

Résultat dans le navigateur

```
<html>...
	<body>
		<p>Bonjour a tous</p>
	</body>
 </html>
```

Le code PHP peut donc être implanté au sein du code HTML et inversement.
Ecrire du PHP uniquement là où il est necessaire rend la programmation plus simple et lisible.
Il est tout à fait possible d'écrire plusieurs scripts PHP à divers endroits de notre code HTML , y compris avant même la balise <head> ou <html>.

Les commentaires en PHP

Les commentaires permettent une meilleure lisibilité ainsi qu'une meilleure compréhension du code

Ils peuvent permettre de séparer notre code en bloc, ou encore d'apporter une clarification.

Attention tout de même à ne pas abuser des commentaires. Si vous avez besoin de sur-commenter car vous pensez que votre code n'est pas très clair, c'est que vous pouvez faire mieux.

Commentaire sur une ligne : // ou #

Commentaire sur plusieurs lignes : /* ... */

Raccourci sur VSCode : Ctrl + :

exemple :

```php
<?php

// Commentaire sur une ligne

# Commentaire sur une ligne

/* Commentaire sur
deux lignes */

?>
```

Les Variables

La déclaration d'une variable se fait avec :
Un nom commençant impérativement par $ et composé de caractères alphanumériques.
On assigne à cette variable une valeur (dont le type sera déterminé par la syntaxe).

L'association entre le nom et la valeur est faite par le signe = (égal) ; affectation de la valeur à la variable.

Syntaxe :

```
//variable lambda
$maVariableLambda = 12 ;
$ma_variable_lambda = 7 ;
```

Il existe des conventions de nommage pour les variables :

- Elles doivent donner du sens et être compréhensible. On doit savoir à quoi on à affaire rien qu'en lisant le nom.
- On n'utilise pas d'abréviation.
- Le début de chaque mot sauf le premier doit commencer par une majuscule (Camel Case).
- Généralement écrite en anglais.
- Pas d'accents, ni d'espaces.

Les types

PHP est dit non-typé, ou faiblement typé, c'est-à-dire qu'il se préoccupe peu des types. Ce langage fait donc parti des langages très flexibles, on peut utiliser n'importe quelle variable pour tout type de valeur.

Du coup, on ne définit généralement pas le type d'une variable ; il sera défini au moment de son exécution, suivant le contexte de son utilisation.

PHP supporte les types suivants : boolean, integer, float, string, array, object, resource et NULL.

Les 3 types principaux étant boolean, number et string.

Boolean ; valeur de vérité, true ou false.

Number ; nombre entier comme décimal

-> *integer* ; nombre entier

-> *float* ; nombre décimal

String ; chaîne de caractères (devant être entourée de guillemets)

Il existe deux fonctions à connaître concernant les types de variables :

var_dump() : affiche les infos structurées d'une variables, y compris son type et sa valeur

gettype() : retourne le type d'une variable

Dans un contexte de test, une valeur sera évaluée en booléen en appliquant les règles de conversion qui sont :

- toute valeur numérique non nulle (-1, 14, 0.57) sera true, zero (0) sera false
- toute chaîne de caractere **non** vide('random', '..') sera true, sinon ('') sera false
- la valeur NULL sera false
- toutes les autres valeurs seront true

Les différents ''casts'' existants sont :

- (int) ou (integer) : conversion en entier
- (bool) ou (boolean) : conversion en booleen
- (double), (float) : conversion en flottant au centième près.
- (string) : conversion en chaîne de caractères
- (array) : conversion en tableau
- (object) : conversion en objet
- (unset) : conversion en NULL

Syntaxe : `$var = (type)$var;`

La fonction **var_dump()** *permet d'afficher les informations d'une variable, y compris son type et sa valeur. Elle fonctionne egalement sur les tableaux et les objets et est très utile pour "debugger".*

exemple :

```
$var_1 = 4.56; // valeur numerique de type flottant
var_dump($var_1); // affiche : float(4.56)
```

Par exemple, dans un contexte de test, une valeur sera évaluée en booléen en appliquant les règles de conversion qui sont :

- toute valeur numérique non nulle (-1, 14, 0.57) sera true, zero (0) sera false
- toute chaîne de caractere **non** vide('random', '..') sera true, sinon ('') sera false
- la valeur NULL sera false
- toutes les autres valeurs seront true

Les ''casts'' autorises sont :

- (int) ou (integer) : conversion en entier
- (bool) ou (boolean) : conversion en booleen
- (double), (float) : conversion en flottant au centième près
- (string) : conversion en chaîne de caractères
- (array) : conversion en tableau
- (object) : conversion en objet
- (unset) : conversion en NULL

Syntaxe : `$var = (type)$var;`

Rappel : La fonction **var_dump()** *permet d'afficher les informations d'une variable, y compris son type et sa valeur. Elle fonctionne egalement sur les tableaux et les objets et est tres utile pour "debugger".*

exemple :

```
$var1 = 4.56; // valeur numérique de type flottant

var_dump($var1); // affiche : float(4.56)

$var1 = (string)$var1; // valeur désormais de type string

var_dump($var1); // affiche : string(4)"4.56"
```

Vérifier l'existence d'une variable, constante, fonction, etc

La fonction isset() permet de verifier si la variable passée en argument existe et est différente de NULL.
La fonction unset() détruit l'ensemble des variables passées en argument.
La fonction empty() détermine si une variable est vide.

NB : À la reception de données venant d'un formulaire ($_POST) ou de l'URL($_GET), il faudra systématiquement utiliser isset() pour être sûr que PHP recupère bien les les données du formulaire ou de l'URL.

Les constantes

À l'instar des variables, il est possible de déclarer des constantes en PHP, mais contrairement à ces dernières, ce sont des valeurs qui ne changeront pas au cours du temps.
NB : Une constante à valeur égale d'une variable prendra moins de place en mémoire.

Syntaxe :
define("NOM", value); permet d'affecter value à la constante nommée NOM.

defined("NOM"); permet de vérifier si la constante NOM est définie.

exemple :
Par convention, il est conseillé d'utiliser des noms en majuscules.

```
define('MON_NOM', 'Jeanne');
```

Les Chaînes de Caractères (string)

Déclarer une chaîne de caractères

Une déclaration de chaîne de caractères peut se faire de multiples façons :

- Syntaxe guillemets simple ' (simples quotes) : aucun contenu ne sera interprété à l'exception du texte.
- Syntaxe guillemets double " (doubles quotes) : les variables seront interprétées.
- Syntaxe Nowdoc <<<'SYMBOL' SYMBOL; → équivalent guillemets simples
- Syntaxe Heredoc <<<SYMBOL SYMBOL; → équivalent guillemets doubles

exemple :

```
$texte = 'tout le monde';
echo 'Bonjour $texte'; // affiche : Bonjour $texte, on recupère le nom de la variable
echo "Bonjour $texte"; // affiche : Bonjour a tous, on recupère la valeur de la variable
```

L'insertion des caractères spéciaux pourra se faire par échappement avec le symbole \ (anti-slash)

exemple :

```
echo 'Bonjour \$texte'; // affiche : Bonjour \$texte
echo "Bonjour \$texte"; // affiche : Bonjour $texte
echo 'Bonjour $texte'; // affiche : Bonjour $texte
```

Fonctions des chaînes de caractères

str_replace permet de remplacer toutes les occurences voulues par une autre chaîne de caractère.

```
str_replace(partie de chaîne à remplacer, remplacement, chaine)
```

exemple :

```
$chaine = 'Mon chien est un berger allemand';
echo str_replace('allemand', 'australien', $chaine);
// Résultat : Mon chien est un berger australien
```

strtr trouve les occurrences dans une chaîne de caractères et les remplace par les caractères voulus.

```
$chaine = 'Je suis développeur';
echo strtr($chaine, ' ', '_');
// Résultat : Je_suis_développeur
```

La fonction **mb_substr()** extrait le nombre de caractères en partant de la position debut dans la chaine.

```
mb_substr(chaîne, debut, nb, encodage)
```

La foncition **mb_strpos()** recherche la premiere occurrence de str rencontrée dans la chaîne en commençant la recherche en position de début et renvoi son index.

```
mb_strpos(chaîne, str, debut, encodage)
```

exemple :

```
$chaine = 'Ma voiture est rouge';
echo mb_substr ($chaine, 11, 3,'utf8');
// va afficher : est (on récupère les 3 caractères suivant la position 11)
$chaine = 'Ma voiture est rouge';
echo mb_strpos ($chaine, 'tur', 0,'utf8'); // va afficher : 6
```

D'autres fonctions pour le texte

`mb_strtolower(string, encodage)` : Renvoie la string en minuscule.

`mb_strtoupper(string, encodage)` : Renvoie la string en majuscule.

`trim(string)` : Supprime tous les espace, tabulations et fin de ligne dans la string.

exemple :

```
$chaine = ' Bonjour \n';
$test = trim($chaine);
echo mb_strlen($chaine, 'utf8'); // va afficher : 14
echo '<hr>';
echo mb_strlen($test, 'utf8'); // va afficher : 7
```

Concaténer des chaînes de caractères

Concaténer, ou mettre bout à bout des strings, est possible grâce à l'utilisation de . (popint). C'est l'opérateur de concaténation et est l'unique opérateur lié aux chaînes de caractères.

```
$var1 = 'Bonjour ';

$var2 = $varl . 'tout le monde'; // affiche : Bonjour tout le monde

echo '4'+'5'; // affiche : 9 ::: Addition

echo '4'.'5'; // affiche : 45 ::: Concaténation
```

Les Tableaux

Déclarer un tableau

Les tableaux sont des variables qui permettent de stocker plusieurs données.

NB : Les fonctions retournant plusieurs valeurs le font généralement sous la forme de tableaux.

Ils peuvent être une alternative aux bases de données si les informations à manipuler ne sont pas trop nombreuses et si le stockage à long terme de ces dernières ne nous importe pas.

Un tableau en PHP est comme une carte ordonnée. On associe des valeurs à des clefs. On les définit avec le mot-clef **array** ou **[ ]**.

Il existe plusieurs façons de déclarer un tableau :

```
// méthode classique, quelque peu obsolète :

$tableau = array(valeur1, valeur2, valeur3);

// méthode à privilégier :

$tableau = []
```

Les tableaux à indice (ou scalaires)

Le premier type de tableau et le plus simple est celui des tableaux à indice.
Chaque élément à un indice entier positif.

Attention ! Le premier élément porte l'indice zéro.

Un tableau peut se remplir en adressant chaque élément ou en fournissant les valeurs à la fonction array(), ou directement entre [].

exemple :

```
$tab[0] = 'P';
$tab[1] = 'H';
$tab[2] = 'P';

// sera l'équivalent du tableau :
$tab = array('P', 'H',    'P');

// ou :
$tab = ['P',    'H',    'P'];
```

Les tableaux associatifs

Un autre type de tableau est le tableau associatif. Chaque élément à une valeur associée qui sert de clef d'accès (key). On utilise la combinaison de symbole => pour assigner la correspondance entre la clef et sa valeur. Un tableau peut se remplir en adressant chaque élément ou en fournissant les valeurs à la fonction array(), ou directement entre [].

exemple :

```
$ventes['lundi'] = 1234;
$ventes['mardi'] = 5678;
$ventes['mercredi'] = 4444;
// est équivalent à :
$ventes = array('lundi'=>1234, 'mardi'=>5678, 'mercredi'=>4444);
// également équivalent à :
$ventes = ['lundi' =>1234, 'mardi' =>5678, 'mercredi' =>4444];
```

Les tableaux multi-dimensionnels

Il faut ici considérer que chaque élément d'un tableau peut lui-même être un tableau. Si l'on s'en tient à 2 dimensions, on pourrait donc imaginer que les indices du tableau principal sont les lignes et les indices des tableaux contenus sont les colonnes d'un tableau excel.
En combinant les indices (clefs) des differents tableaux, on peux facilement retrouver une donnée précise.

exemple :

```
$lundis = ['6', '12', '21'];
$mardis = ['8', '15', '19'];
$mercredis = ['7',    '21',    '20'];
$semaine = [$lundis, $mardis, $mercredis];
```

Consulter une valeur

tableau[n] : Renvoi la valeur présente à l'indice n. (tableau indice)
tableau[key] : Renvoi la valeur présente à la clef key. (tableau associatif)
tableau[key][n] : Renvoi la valeur présente à l'indice n du tableau contenu a la clef key. (tableau multi-dimensionnel)

Il est tout à fait possible de combiner des tableaux de plusieurs types.

exemple :

```
$tab1 = [1,2,3,4];
$tab2 = [5,6,7,8];
$tableau = ['lundi' => $tab1, 'mardi' => $tab2];
echo $tableau['mardi'][2]; // va afficher : 7
```

Ajouter/ modifier des donnees

```
tableau[] = value
```

Ajoute value **en fin de tableau**. L'indice est crée automatiquement.
Cette méthode sur un tableau associatif aura pour effet de créer un indice selon les même règles. Notre tableau aura donc au final des données accessibles par des indices et par des clefs.

```
$tab[0] = 1;
$tab[3] = 4;
$tab[] = 2;
print_r($tab); // va afficher : Array (    [0] => 1 [3] => 4 [4] => 2 )
```

Fonctionnement

Associer une valeur à la position indice. Si une valeur existe déjà à cet indice, elle est remplacée, sinon créée.

```
tableau[indice] = valeur
```

Associer une valeur à la clef key. Si une valeur existe deja à cette clef, elle est remplacée, sinon crée.

```
tableau[key] = valeur
```

exemple :

```
$ventes = ['lundi' =>1234, 'mardi' =>5678, 'mercredi' =>4444];
$ventes['mardi'] = 2222;
$ventes['jeudi'] = 6666;
print r($ventes);
//va afficher : Array ( [lundi] => 1234 [mardi] => 2222 [mercredi] => 4444 [jeudi] => 6666 )
```

```
tableau[key][indice] = valeur
```

exemple :

```
$tab1 = [1,2,3,4];
$tab2 = [5,6,7,8];
$tableau = ['lundi' => $tab1, 'mardi' => $tab2];
$tableau['mardi'][2] = 9;
```

array_push(tableau, valeur(s)) : Cette fonction ajoute une valeur à la fin du tableau, étant équivalent à tableau[] = valeur.

array_unshift(tableau, valeur(s)) : Cette fonction permet egalement d'insérer des valeurs mais en début de tableau. Cela aura pour effet de recalculer et décaler tous les indices !

array_pop(tableau) : Renvoi et supprime le dernier élément du tableau.
array_shift(tableau) : Renvoi et supprime le premier élément du tableau. Recalcul et décale les indices afin qu'ils commencent à zero.

```
$fruits = ['orange', 'banane', 'clementine', 'pomme'];
$sup_deb = array_shift($fruits);
echo $sup_deb; // va afficher : orange
print_r($fruits); // va afficher : Array ( [0] => banane [1] => clementine [2] => pomme )
$sup_fin = array_pop($fruits);
echo $sup_fin; // va afficher : pomme
print_r($fruits); // va afficher : Array ( [0] => banane [1] => clementine )
```

count(tableau) : Renvoi tout simplement le nombre d'éléments d'un tableau.

in_array(value, tableau) : Renvoi true si la valeur value est trouvée dans tableau, false dans le cas contraire.

array_search(value, tableau) : Renvoi la clef si la valeur value est trouvée dans tableau, false dans le cas contraire.

exemple :

```
$fruits = ['orange', 'banane', 'clementine', 'pomme'];
$reponse = in_array('banane', $fruits);
var_dump($reponse); // va afficher : bool(true)
$reponse = array_search('clementine', $fruits);
var_dump($reponse); // va afficher : int(2)
```

explode(delimiteur, string) : Retourne un tableau de valeurs crées en découpant la chaîne de caractères suivant le paramètre délimiteur.
implode(delimiteur, array) : Retourne une chaîne de caractères constituée de toutes les valeurs de tableau separées par le paramètre délimiteur. (Ce dernier est optionnel)

exemple :

```
$fruits = ['orange', 'banane', 'clementine', 'pomme'];
$chaine = implode(',',$fruits);
print_r($chaine);
// va afficher : orange,banane,clementine,pomme
$tableau = explode(',', $chaine);
print_r($tableau);
// va afficher : Array (    [0] => orange [1] => banane [2] => clementine [3] => pomme )
```

sort(tableau, option)

Modifie l'ordre des éléments du tableau selon le paramètre option. Si ce dernier est omi, le tri se fait sur les valeurs, de la plus petite à la plus grande. (option SORT_REGULAR)

Quelques options de tri possible :

- SORT_NUMERIC compare les éléments numériquement
- SORT_STRING compare les éléments comme des chaînes
- SORT_NATURAL compare les éléments "naturellement"

exemple :

```
$fruits = ['orange', 'banane', 'clementine', 'pomme'];
sort($fruits); print r($fruits);
// va afficher : Array (    [0] => banane [1] => clementine [2] => orange [3] => pomme )
$nombres = ['img1.jpg', 'img3.jpg', 'img12.jpg', 'img2.jpg'];
sort($nombres, SORT_NATURAL); print r($nombres);
// va afficher : Array (0] => img1.jpg [1] => img2.jpg [2] => img3.jpg [3] => img12.jpg)
```

Les opérateurs

Les opérateurs arithmetiques

EXEMPLE	NOM	RESULTAT
-$a	Négation	Opposé de $a.
$a + $b	Addition	Somme de $a et $b.
$a - $b	Soustraction	Différence de $a et $b.
$a * $b	Multiplication	Produit de $a et $b.
$a / $b	Division	Quotient de $a par $b.
$a % $b	Modulus	Reste de la division entière de $a par $b.
$a ** $b	Exponentielle	Résultat de l'élévation de $a à la puissance $b.

Les opérateurs logiques

EXEMPLE	NOM	RESULTAT
! $a	Not (Non)	TRUE si $a n'est pas TRUE.
$a && $b	And (Et)	TRUE si $a ET $b sont TRUE.
$a || $b	Or (Ou)	TRUE si $a OU $b est TRUE.
$a and $b	And (Et)	TRUE si $a ET $b valent TRUE.
$a xor $b	XOR	TRUE si $a OU $b est TRUE, mais pas les deux en même temps.
$a or $b	Or (Ou)	TRUE si $a OU $b valent TRUE.

Les opérateur || et && sont prioritaires sur or et and, ce qui permet d'écrire :

```
$a = $b || $c and $d;
// est équivalent à :
$a = ($b || $c) && $d;
```

Les opérateurs de comparaison

EXEMPLE	NOM	RESULTAT
$a == $b	Égal	TRUE si $a est égal à $b après le transtypage.
$a === $b	Identique	TRUE si $a est égal à $b et qu'ils sont de même type.
$a != $b	Différent	TRUE si $a est différent de $b après le transtypage.
$a <> $b	Différent	TRUE si $a est différent de $b après le transtypage.
$a !== $b	Différent	TRUE si $a est différent de $b ou bien s'ils ne sont pas du même type.
$a < $b	Plus petit	TRUE si $a est strictement plus petit que $b.
$a > $b	Plus grand	TRUE si $a est strictement plus grand que $b.
$a <= $b	Inférieur ou égal	TRUE si $a est plus petit ou égal à $b.
$a >= $b	Supérieur ou égal	TRUE si $a est plus grand ou égal à $b.

Les opérateurs d'affectation

Opération	Nom	Description
=	affectation simple	$a = 3 donne la valeur 3 à $a
++	auto-incrementation	$a++ ajoute 1 à $a
--	auto-decrementation	$a-- enlève 1 à $a
+=	incrementation	$a += 3 ajoute 3 à $a
-=	decrementation	$a -= 3 enlève 3 à $a
/=	division	$a /= 3 divise la valeur de $a par 3
*=	multiplication	$a *= 3 multiplie la valeur de $a par 3
%=	modulo	$a %= 3 reste de la division entière par 3
. =	concaténation	$a .= 'bonjour' ajoute 'bonjour' en fin de chaîne