Formation

PHP

m2iformation.fr

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

PHP

Sommaire

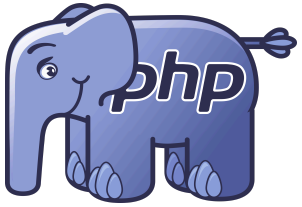
1. Introduction
2. Installation
3. Syntaxe de base
4. Les opérateurs
5. Les structures conditionnelles
6. Les structures itératives
7. Les fonctions
8. Les tableaux
9. Les super globales
10. Programmation orientée objet
11. Héritage
12. Interfaces
13. Traits
14. Méthodes magiques
15. Les exceptions
16. Les namespaces

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Introduction

Qu'est-ce-que PHP ?

- L'acronyme PHP signifie : *PHP Hypertext Preprocessor*
- C'est un langage de script **impératif orienté objet** qui est open source
- Le langage a été mis au point par Rasmus Lerdorf en 1995 basé sur le langage C
- Sa mascotte est un éléphant bleu nommée ElePHPant



Utilisation de PHP

- PHP est le langage côté serveur le plus utilisé sur le web, il représente 78% des parts de marché en 2022
- De nombreux CMS utilisent PHP : Wordpress, Drupal, Magento, Prestashop ...
- Plusieurs frameworks ont vu le jour sur PHP : Laravel, Symfony, Spiral

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Installation

Installation de PHP

- Windows (alternatives: Laragon, WAMP, XAMPP)

```
> choco install php
```

- MacOS (alternatives: XAMPP)

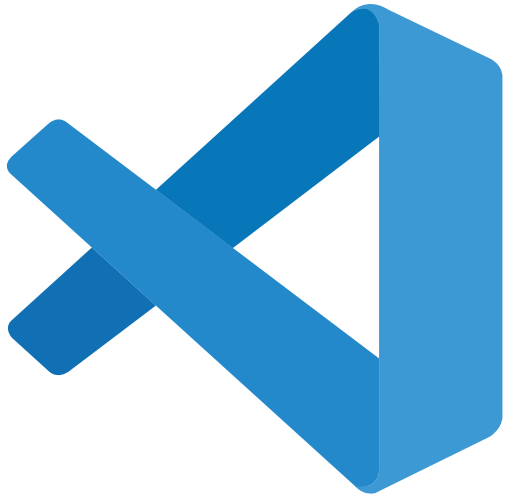
```
> brew install php
```

- Linux (alternatives: XAMPP)

Éditeurs

PHPStorm

VsCode



Vérifier l'installation de PHP

- Créer un fichier hello.php et y ajouter le code suivant:

```php
<?php

echo 'Hello world!';
```

- Ouvrir un terminal dans le repertoire du fichier et taper la commande suivante : `php hello.php`
- ⚠️ PHP doit être configuré dans les variables d'environnement pour pouvoir exécuter cette commande

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Syntaxe de base

Balise PHP

- Un fichier php commence toujours par la balise `<?php` et termine par `?>`
- La balise de fermeture est facultative dans le cas où le fichier ne contient que du code PHP
- Un raccourci de echo permet d'échapper une ou plusieurs de chaînes de manière plus succincte `<?= 'mon texte' ?>`

Échappement des balises HTML

- Tout ce qui se trouve en dehors d'une paire de balises ouvrantes/fermantes est ignoré par l'analyseur PHP, ce qui permet d'avoir des fichiers PHP mixant les contenus
- Ceci permet à PHP d'être contenu dans des documents HTML, pour créer par exemple des templates

```
<p>Ceci sera ignoré par PHP et affiché au navigateur.</p>
<?php echo 'Alors que ceci sera analysé par PHP.'; ?>
<p>Ceci sera aussi ignoré par PHP et affiché au navigateur.</p>
```

Les commentaires

- PHP permet d'écrire des commentaires de 3 manières
  1. `//` : commentaire sur une ligne (C, C++)
  2. `/* */` : commentaire multi-lignes
  3. `#` : commentaire sur une ligne (bash)

Les types

- PHP est un langage typé dynamiquement, le type d'une variable est déterminé au moment de l'exécution

1. **null** : absence de valeur
2. **bool** : booléen
3. **int** : entier
4. **float** : nombre à virgule flottante
5. **string** : chaîne de caractères
6. **array** : tableau
7. **object** : objet
8. **callable** : callback
9. **resource** : variable spéciale

Chaînes de caractères

- Les chaînes de caractères s'écrivent de plusieurs manières dans PHP

1. Guillemets simples `'ma chaîne'` : les séquences d'échappement pour les caractères spéciaux ne sont pas interprétées
2. Guillemets doubles `"ma chaîne"` : utilisation des caractères spéciaux supportés (ex: \n, \t, \" ...)

- Pour concaténer deux chaînes il suffit d'utiliser l'opérateur `.`, ex: `$a . $b`

Les variables

- Les variables php sont représentés par le signe `$` suivi du nom de la variable qui est sensible à la casse
- Un nom de variable commence toujours par une lettre ou le caractère `_` suivi de lettres ou chiffres

```php
$prenom = 'arthur';
$age = 25;
$estMajeur = true;
```

Les constantes

- Une constante est un identifiant qui ne peut pas être modifié et qui est sensible à la casse, on les écrit en majuscule

1. Définir une constante avec le mot clé `const`

```
const PRENOM = 'arthur';
```

2. Définir une constante avec la fonction `define()`

```
define("ANIMAL", "chat")
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Les opérateurs

Opérateurs arithmétiques

operateur	description	exemple
+	addition	1 + 2
-	soustraction	2 - 1
*	multiplication	2 * 2
/	division	2 / 2
%	reste d'une division euclidienne	10 % 2
**	puissance	2**4

Opérateurs d'affectation

- `$chat = 'felix'` permet d'affecter la valeur de l'expression de droite à la variable $chat grâce à l'opérateur `=`
- Il existe d'autres opérateurs d'affectations :

opérateur	équivalence	description
$a **+=** 2	$a = $a + 2	addition
$a **-=** 2	$a = $a - 2	soustraction
$a ***=** 2	$a = $a * 2	multiplication
...	...	...
$a **.=** "world"	$a = $a."world"	concaténation

Opérateurs de comparaison

opérateur	description	exemple
==	égalité après transtypage	1 == '1'
===	égalité stricte	1 === 1
!=	différent de après transtypage	1 != 2
<>	différent de après transtypage	1 <> 2
!==	différent de stricte	1 !== '2'
<	inférieur à	1 < 2
>	supérieur à	2 > 1
<=	inférieur ou égal	1 <= 2
>=	supérieur ou égal	2 >= 1

Les opérateurs logiques

Exemple	Nom	Résultat
$a **and** $b	And (Et)	**true** si $a ET $b valent true.
$a **or** $b	Or (Ou)	**true** si $a OU $b valent true.
$a **xor** $b	XOR	**true** si $a OU $b est true, mais pas les deux en même temps.
! $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.

Table de vérité

- cette table de vérité permet de montrer l'utilisation des opérateurs logiques selon les valeurs des expressions de A et de B

A	B	A OR B	A AND B	A XOR B	!A
false	false	false	false	false	true
false	true	true	false	true	true
true	false	true	false	true	false
true	true	true	true	false	false

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Les structures conditionnelles

Structure conditionnelle

- Une structure conditionnelle permet d'exécuter du code en fonction d'une ou plusieurs expressions logiques
- Elles sont utiles pour créer un programme dynamique et interactifs qui s'adapte à plusieurs situations
- Il existe 4 types de structure conditionnelles en PHP
  1. **if**
  2. **if** ... **else**
  3. **if** ... **elseif** ... **else**
  4. **switch**

Exemple d'utilisation

```php
if($a > $b) {
  echo "a est plus grand que b";
} else {
    echo "b est plus grand que a";
}

# syntaxe alternative
if($a):
    echo $a;
else:
    echo $c;
endif;
```

L'instruction switch

- L'instruction switch équivaut à une série d'instructions if .. elsif
- Elle permet de comparer (comparaison large `==`) une variable ou expression à un grand nombre de valeurs

```
switch ($i) {
    case 0:
        echo "i égal 0";
        break;
    case 1:
        echo "i égal 1";
        break;
}
```

L'instruction match

- L'instruction match est similaire à switch sauf que celle-ci renvoie une valeur et compare les valeurs de manière stricte `===`
- La valeur retournée n'a pas besoin d'être utilisée

```
$valeur = match ($aliment) {
    'pomme' => 'fruit',
    'chips' => 'apéritif',
    'courgette' => 'légume',
};
```

L'opérateur ternaire

- Cet opérateur est fréquemment utilisé comme raccourci pour la déclaration de `if ... else`
- La syntaxe : `condition ? exprSiVrai : exprSiFaux`

```php
$monResultat = $estMembre ? 'oui' : 'non';
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Les structures itératives

Les structures itératives en PHP

- Il existe 4 manières de créer des boucles dans le langage PHP :
  1. **while**
  2. **do while**
  3. **for**
  4. **foreach**

L'instruction while

- la boucle while permet de répéter une instruction tant que l'expression qui lui est passée est évaluée comme **true**
- Si l'expression du while est false avant la première itération, l'instruction ne sera jamais exécutée
- ⚠️ attention aux boucles infinies

```php
while ($i <= 10) {
    echo $i++; // Affiche i avant incrémentation
}
```

L'instruction for

- La boucle for a deux parties : une entête qui spécifie la manière de faire l'itération, et un corps qui est exécuté à chaque itération
- Dans cette forme de boucle, une variable prend des valeurs successives sur un intervalle
- `for (initialisation ; test ; itération) { opération };`

```
for($i = 0; i < 10; $i++) {
    echo $i
}
```

L'instruction foreach

- L'instruction foreach offre une manière simplifiée de parcourir les tableaux et objets

```
# 1ère syntaxe
foreach ($iterable as $value){
    //commandes
}

# 2ème syntaxes
foreach ($iterable as $key => $value){
    //commandes
}
```

Les instructions break et continue

- L'instruction `break` permet de sortir d'une structure for, foreach, while, do-while ou switch
- L'instruction `continue` permet de passer l'itération suivante sans exécuter le reste du code

```php
while($count < 5) {
    if ($count % 2 === 0) continue;
    if ($count === 3) break;
    echo $count;
}
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Les fonctions

Qu'est-ce-qu'une fonction ?

- Les fonctions font partie des briques fondamentales de PHP
- Une fonction est une procédure PHP, u**n ensemble d'instructions** effectuant **une tâche** ou **calculant une valeur**
- Afin d'utiliser une fonction, il est nécessaire de l'avoir auparavant définie au sein de la portée dans laquelle on souhaite l'appeler

Définition d'une fonction

Une définition de fonction est construite avec le mot-clé `function`, suivi par :

- Le **nom** de la fonction
- **Une liste d'arguments** à passer à la fonction, entre parenthèses et séparés par des virgules
- Les instructions définissant la fonction, entre accolades: **{ }**

```
function carre($nombre) {
    return $nombre * $nombre
}
```

Les arguments d'une fonction

- PHP permet de passer des arguments de 5 façons

1. Par valeur (comportement par défaut): `function hello($arg)`
2. Par référence: `function hello(&$arg)`
3. Par valeur avec valeur par défaut: `function hello($arg = 123)`
4. Par liste d'arguments: `function hello(...$args)`
5. Par argument nommé: `hello(nom: $valeur);`

Retour d'une fonction

- Une fonction peut retourner une valeur à l'aide de l'instruction `return`
- L'instruction `return` provoque termine l'exécution de la fonction et renvoie la valeur passée
- En cas d'absence de l'instruction `return`, la fonction renvoie la valuer `null`

Les fonctions fléchées

- Les fonctions fléchées sont une syntaxe plus concise des fonctions anonymes
- syntaxe: `fn (args) => expr`

```php
# Fonction fléchée
$maFonction = fn($x) => $x * 2;

# Fonction anonyme
$maSecondeFonction = function($x) {
    return $x * 2
}
```

Typage des fonctions

- PHP permet de typer les arguments ainsi que le type de retour des fonctions et méthodes
- Dans le cas où un type incorrecte est passé, l'exception TypeError est levée

```php
function maFonction(int $age, string $nom): bool
{
    return $age >= 18;
}
```

Types avancés

- **Union Type**: accepte des valeurs de types différents

```php
function myFunction(string|int|array $param)
```

- **Intersection Type**: la valeur donnée appartient à tous les types

```php
function myFunction(Stringable&Countable $param)
```

- **DNF Type**: supporte la combinaison des deux types précédents

```php
public function myFunction((Stringable&Countable)|null $param)
```

- **Null Type** : indique qu'un argument peut être null

```php
public function myFunction(?string $param)
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Les tableaux

Qu'est-ce-qu'un tableau ?

- Un tableau est une structure de données représentant **une séquence finie** d'éléments auxquels on peut accéder efficacement par leur **position**, ou **indice**, dans la séquence
- Dans PHP tous les tableaux sont de type associatifs, c'est à dire que les valeurs sont indexés avec une clé

Syntaxe des tableaux

- Création d'un tableau : `array()`

```php
$associative = array('un' => 1, 'deux' => 2, 'trois' => 3);
```

- Syntaxe raccourcie : `[ ]`

```php
$associative = ['un' => 1, 'deux' => 2, 'trois' => 3];
```

- Accéder à un élément par son indice ou sa clé : `$tab[ind]`

```php
echo $associative['un']; // affiche 1
```

Suite syntaxe tableaux

- Déclaration d'un tableau simple

```
$tableau = [1, 2, 3]
```

- Ajouter un élément

```
$tableau[clé] = valeur;
$tableau[] = valeur;
```

- Connaître la taille d'un tableau

```
count($tableau)
```

Destructuration

- PHP permet de déstructurer un tableau pour en extraire les éléments et les stocker dans des variables

```php
[$a, $b, $c] = $tableau;

['a' => $a, 'b' => $b, 'c' => $c] = $tableau; // tableau associatif

[, , $c] = $array; // récupérer le 3ème élément uniquement
```

Opérateur Rest et Spread

- Un tableau peut être *spread* (répandu) dans une fonction

```
function maFonction($a, $b) {/* */};
maFonction(...$tableau);
```

- Les fonctions, elles, peuvent récupérer un nombre variables d'arguments (*rest parameters*)

```
function maFonction($a, $b, ...$args) {/* */};
```

- On peut également créer un tableau à partir d'autres tableaux l'aide de l'opérateur spread : `$result = [...$tab1, ...$tab2];`

Gérer les valeurs nulles

- PHP a introduit le mécanisme de `null coalescing` qui permet de gérer les cas où une valeurs peut être nulle : `??`

```php
$pseudo = $_GET['user'] ?? 'nobody';
// équivalent
$username = isset($_GET['user']) ? $_GET['user'] : 'nobody';
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Les supers globales

Qu'est-ce que les super globales ?

- Les Superglobales — Les variables internes qui sont toujours disponibles, quel que soit le contexte
- $GLOBALS
- $_SERVER
- $_GET
- $_POST
- $_FILES
- $_COOKIE
- $_SESSION
- $_REQUEST
- $_ENV

$GLOBALS

- $GLOBALS — Référence toutes les variables disponibles dans un contexte global
- C'est un tableau associatif contenant les références sur toutes les variables globales actuellement définies dans le contexte d'exécution global du script
- Les noms des variables sont les index du tableau

```php
$foo = "Exemple de contenu";
echo '$foo dans le contexte global : ' . $GLOBALS["foo"] . "\n";
```

$_SERVER

- $_SERVER — Variables de serveur et d'exécution
- `$_SERVER` est un tableau contenant des informations telles que les en-têtes, les chemins et les emplacements de script
- la plupart variables prises en compte dans la spécification » CGI/1.1 et sont susceptibles d'être définies

```php
echo $_SERVER['SERVER_NAME'];
```

$_GET

- Tableau associatif des valeurs passées au script courant via les paramètres d'URL
- Ce tableau n'est pas seulement rempli pour les requêtes GET, mais plutôt pour toutes les requêtes avec un query string

```php
<?php
echo 'Bonjour ' . htmlspecialchars($_GET["name"]) . '!';
?>
```

$_POST

- tableau associatif des valeurs passées au script courant via le protocole HTTP et la méthode POST
- Les informations sont récupérées lors de l'utilisation de la chaîne `application/x-www-form-urlencoded` ou `multipart/form-data` comme en-tête HTTP Content-Type dans la requête

```php
<?php
echo 'Bonjour ' . htmlspecialchars($_POST["name"]) . '!';
?>
```

$_SESSION

- La session est un mécanisme qui permet de conserver des informations pour un utilisateur lorsqu'il navigue d'une page à une autre
- Pour utiliser les sessions, il faut d'abord appeler la fonction `session_start()` au début de chaque page
- on peut accéder au tableau `$_SESSION` et lui assigner des valeurs à sauvegarder temporairement

```php
$_SESSION['favcolor'] = 'green';
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Programmation Orientée Objet

Qu'est-ce-que la POO ?

- La programmation orientée objet est un paradigme de programmation qui consiste à modéliser un système comme un ensemble d'objets, où chaque objet représente un aspect donné du système
- Les objets contiennent des fonctions (ou méthodes) et des données
- Un objet fournit une interface publique pour le reste du code qui voudrait l'utiliser, mais maintient son propre état interne

Principes de la POO

- La programmation orientée objet est basée sur les principes suivant:
  - Encapsulation
  - Abstraction
  - Héritage
  - Polymorphisme

Classe

- Une classe en PHP est une structure qui définit les propriétés et les méthodes communes à un ensemble d'objets
- Une classe est comme un modèle ou un plan qui permet de créer des objets ayant les mêmes caractéristiques et comportements

```
class Chien {
    // Propriétés de la classe
    // Constructeur
    // Méthodes
}
```

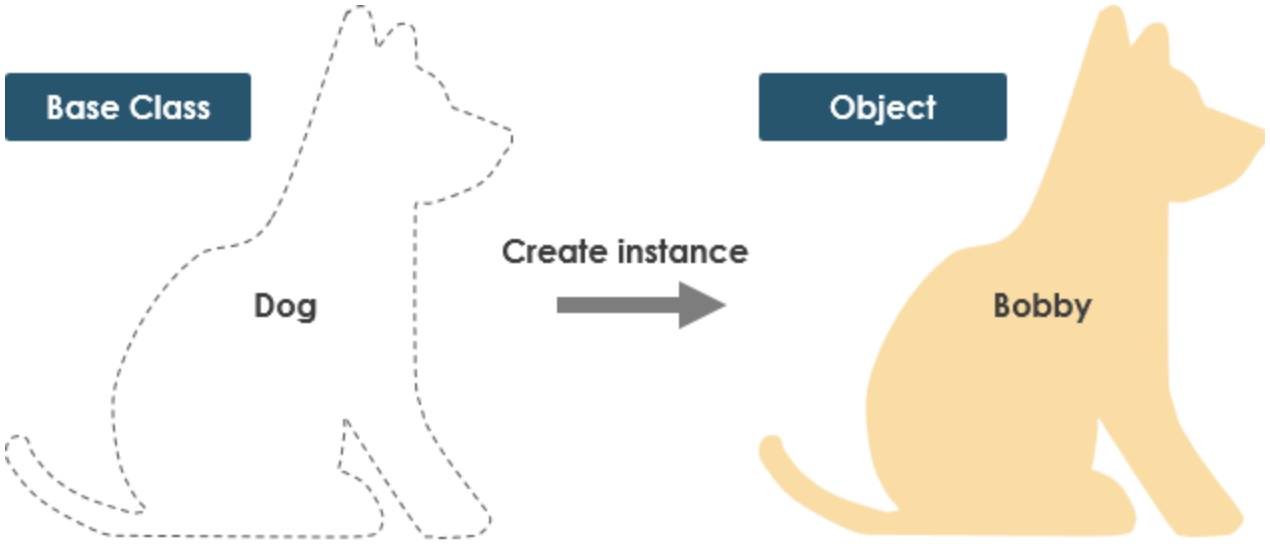
Objet

- Une instance de classe (ou objet) en PHP est une représentation particulière d'une classe
- Pour créer un objet à partir d'une classe, on utilise le mot-clé `new` suivi du nom de la classe

```php
$rex = new Chien();
```

- Dans ce cas on a instancié la classe Chien et obtenu une instance de cette classe

Différence entre classe et objet



Properties	Methods
Color	Sit
Eye Color	Lay Down
Height	Shake
Length	Come
Weight	

Property Values	Methods
Color: Yellow	Sit
Eye Color: Brown	Lay Down
Height: 17 in	Shake
Length: 35 in	Come
Weight: 24 pounds	

Propriétés

- Les variables au sein d'une classe sont appelées **propriétés** (membre, champ)
- Elles sont définies en utilisant au moins un **modificateur** (public, private, static, readonly, ...) ainsi qu'un **type**
- La déclaration d'une propriété peut comprendre une valeur initiale
- Pour accéder à la **propriété d'une instance** il faut utiliser la syntaxe: `$obj->prop`
- Pour accéder à une **propriété statique** la syntaxe est: `Class::prop`

La pseudo-variable $this

- La pseudo-variable `$this` est disponible au sein de n'importe quelle méthode
- Lorsque qu'une méthode est appelée au sein d'un objet, `$this` est la valeur de l'objet appelant

```php
class Chien {
    private string $nom;

    public function __construct($nom) {
        $this->nom = $nom; // $this permet d'assigner le nom à l'instance
    }
}
```

Le constructeur

- Le constructeur est une méthode spéciale qui est appelée automatiquement lors de la création d'un objet d'une classe
- Le constructeur permet d'initialiser les propriétés de l'objet avec des valeurs données en paramètres ou par défaut
- `public function __construct(...$params): void`

Promotion du constructeur

- PHP permet d'utiliser les paramètres du constructeur pour créer des propriétés de l'objet

```
class Point {
    public function __construct(protected int $x, protected int $y = 0) {
    }
}
```

- Lorsqu'un argument du constructeur inclus un modificateur de visibilité, PHP l’interprétera comme une propriété d'objet

Destructeur

- Le destructeur est une méthode spéciale qui est appelée automatiquement lorsque l'objet d'une classe est détruit
- Il permet de libérer les ressources utilisées par l'objet, comme la mémoire ou les fichiers ouverts
- `public function __destruct(...$params): void`

Visibilité

- La visibilité d'une propriété, d'une méthode ou d'une constante peut être définie en préfixant sa déclaration avec un mot-clé :
  - **public**: accessible partout
  - **protected**: accessible dans la classe et les classes dérivées
  - **private**: accessible uniquement dans la classe

Statique

- Une propriété ou une méthode statique appartient à la classe dans laquelle elle a été définie, et non à une instance de cette classe
- On accède à un élément statique avec l'opérateur de résolution de portée `::` (Paamayim Nekudotayim)
- Une propriété statique partage la même valeur pour toutes les instances de la classe

```php
public static $nombreInstance = 0;
public static function direBonjour() { /* code */ }
echo Classe::nombreInstance
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Héritage

Qu'est-ce que l'héritage ?

- L'héritage en PHP est un mécanisme qui permet à une classe de définir des propriétés et des méthodes communes à un ensemble de classes dérivées
- Une classe qui hérite d'une autre classe est appelée une sous-classe ou une classe enfant
- Une classe qui est héritée par d'autres classes est appelée classe parent ou super classe
- Pour déclarer qu'une classe hérite d'une autre classe, on utilise le mot-clé `extends` suivi du nom de la classe parente

Exemple d'application de l'héritage

- Dans cet exemple, la classe Chien hérite de la classe Animal et peut accéder à sa propriété nom et à son constructeur
- La classe Chien redéfinit également la méthode parler (surcharge)

```php
class Animal {
    public $nom;
    public function __construct($nom) {
        $this->nom = $nom;
    }
    public function parler() {
        echo "je suis un animal";
    }
}
```

```php
class Chien extends Animal {
    public function parler() {
        echo $this->nom;
    }
}
```

Parent et Self

- Lors de la surcharge d'une méthode, il est possible d'appeler la méthode d'origine à l'aide du mot clé `parent`

```
    protected function sayHello() {
        parent::sayHello();
        echo 'coucou';
    }
```

- `self` fait référence à la classe à l'intérieur de celle-ci, son fonctionnement est similaire à `$this`

```
echo self::$maVariableStatique;
```

Empêcher l'héritage et la surcharge

- Le mot-clé `final` empêche les classes enfants de redéfinir une méthode ou constante en préfixant la définition avec final
- Si la classe elle-même est définie comme finale, elle ne pourra pas être étendue

```
final class BaseClass {
   public function test() {
       echo "BaseClass::test() appelée\n";
   }
}
```

Classe abstraite

- Les classes abstraites ne peuvent pas être instanciées directement, elles servent de modèles pour les classes dérivées
- Les classes abstraites sont déclarées avec le mot-clé `abstract` et peuvent contenir des méthodes abstraites et des méthodes normales
- Une méthode abstraite est une méthode qui n'a pas de corps, mais seulement une signature

Exemple d'application de l'abstraction

- Dans cet exemple Animal est définit avec une méthode abstraite et une méthode normale
- Dans la classe Chien la méthode seDéplacer() doit être redéfinie

```php
abstract class Animal {
    // méthode abstraite
    abstract public function seDeplacer();

    // méthode normale
    public function manger() {
        echo "Je mange";
    }
}
```

```php
class Chien extends Animal {
    // définition de la méthode abstraite
    public function seDeplacer() {
        echo "Je cours";
    }
}
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Interfaces

Qu'est-ce qu'une interface ?

- Les interfaces permettent de définir un **contrat** entre une classe et son utilisation
- Une interface spécifie quelles **méthodes** une classe doit **implémenter**, sans définir comment ces méthodes fonctionnent
- Toutes les méthodes déclarées dans une interface doivent être **publiques**

Composition et héritage

- La composition consiste à inclure une instance d'une classe comme attribut d'une autre classe
- La différence entre l'héritage et la composition est que les classes et les objets dans un code d'héritage sont étroitement couplés, ce qui signifie qu'ils ne doivent pas être modifiés parce qu'en changeant le parent, on risque de modifier l'enfant, et donc de casser le code
- En général, il est recommandé de préférer la composition à l'héritage dans les cas où cela est possible

Implémenter une interface

```php
interface EtreVivant {
    public function manger(): void;
    public function dormir(): void;
}
```

```php
class Humain implements EtreVivant {
    public function manger(): void {
        echo "je mange\n";
    }

    public function dormir(): void {
        echo "ZzZzzZz\n";
    }
}
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Les traits

Qu'est-ce qu'un trait ?

- Un trait est un mécanisme de réutilisation de code
- Il permet de **réutiliser** un ensemble de méthodes dans plusieurs classes **indépendantes**
- La sémantique entre les classes et les traits **réduit la complexité** et évite les problèmes typiques de l'héritage multiple et des mixins
- La meilleure façon de comprendre ce que sont les traits et comment les utiliser est de les considérer pour ce qu'ils sont essentiellement : du copier-coller assisté par le langage

Détail sur les traits

- Un trait est semblable à une classe, mais il ne sert qu'à grouper des fonctionnalités d'une manière intéressante
- Il n'est pas possible d'instancier un Trait en lui-même
- C'est un ajout à l'héritage traditionnel, qui autorise la composition horizontale de comportements, c'est à dire l'utilisation de méthodes de classe sans besoin d'héritage

Exemple d'utilisation de trait

```php
trait Hello {
  public string $name;

  public function sayHello() {
    echo "Hello, " . $this->name . "!\n";
  }
}
```

```php
class Person {
  use Hello;
  public function __construct($name) {
    $this->name = $name;
  }
}
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Méthodes magiques

Qu'est-ce que les méthodes magiques ?

- Les méthodes magiques sont des méthodes spéciales qui écrasent l'action par défaut de PHP quand certaines actions sont réalisées sur un objet
- Toutes les méthodes commençant par __ sont réservées par PHP
- Les méthodes **construct et **destruct sont des méthodes magiques

La méthode __toString()

- La méthode `__toString()` détermine comment l'objet doit réagir lorsqu'il est traité comme une chaîne de caractères
- Par exemple, ce que `echo $obj;` affichera

```php
public function __toString()
{
    return 'Mon objet';
}
```

La méthode __invoke()

- La méthode `__invoke()` est appelée lorsqu'un script tente d'appeler un objet comme une fonction

```php
class CallableClass
{
    public function __invoke(int $x)
    {
        var_dump($x);
    }
}
$obj = new CallableClass;
$obj(5);
```

__debugInfo()

- Cette méthode est appelée par `var_dump()` lors du traitement d'un objet pour récupérer les propriétés qui doivent être affichées
- Si la méthode n'est pas définie dans un objet, alors toutes les propriétés publiques, protégées et privées seront affichées

```php
class C {
    private $prop;

    public function __construct($val) {
        $this->prop = $val;
    }

    public function __debugInfo() {
        return [
            'propSquared' => $this->prop ** 2,
        ];
    }
}

var_dump(new C(42));
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Les exceptions

Définition d'une exception

- Une exception est un événement qui se produit lors de l'exécution d'un programme et qui **interrompt** le flux normal des instructions
- En PHP, une exception est un objet de la classe **Exception** ou d'une classe dérivée
- Lorsqu'une exception est lancée, le code qui suit l'instruction `throw` n'est pas exécuté, et PHP cherche le premier bloc `catch` qui peut gérer l'exception
- Si aucun bloc catch n'est trouvé, alors PHP affiche un **message d'erreur fatal** et termine le script

Utilisation des exceptions

- Une exception est lancée avec l'instruction `throw`
- Le code exécuté est généralement entouré d'un bloc `try`
- Chaque `try` doit avec un bloc `catch` ou `finally`
- Plusieurs blocs `catch` peuvent être utilisés pour attraper différentes classes d'exceptions
- Si une exception n'est pas attrapée, une erreur fatale issue de PHP sera envoyée avec un message *"Uncaught Exception ..."*

Lancer une exception

- Utilisation du mot clé `throw` pour lancer une exception

```php
function verifierNombre($nombre) {
  if ($nombre > 1) {
    // Lancer une exception si le nombre est supérieur à 1
    throw new Exception("Le nombre doit être inférieur ou égal à 1");
  }
  return true;
}
```

Capturer une Exception

- Gestion d'une exception dans un bloc Try Catch

```php
try {
    1 / 0;
} catch (Exception $e) {
    echo 'Exception reçue : ',  $e->getMessage(), "\n";
} finally {
    echo "Première fin.\n";
}
```

Création d'une Exception

```php
class FileNotFoundException extends Exception {
  private $filename;

  public function __construct($filename) {
    $this->filename = $filename;
    parent::__construct("Le fichier $filename n'existe pas");
  }

  public function getFilename() {
    return $this->filename;
  }
}
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Les namespaces

Définition d'un espace de nom

- Les namespaces sont des moyens de regrouper des classes, des interfaces, des fonctions et des constantes qui ont un rapport logique entre eux
- Ils permettent d'éviter les conflits de noms entre les différents éléments du code et de faciliter l'organisation et la maintenance du code
- Un namespace est déclaré avec le mot-clé `namespace` suivi du nom du namespace

Intérêt des namespaces

Les espaces de noms sont conçus pour résoudre deux problèmes :

1. **Collisions de noms** entre le code que vous créez, les classes, fonctions ou constantes internes de PHP, ou celle de bibliothèques tierces
2. La capacité de faire des **alias** ou de **raccourcir** des *Noms_Extrememement_Long* pour aider à la résolution du premier problème et améliorer la lisibilité du code

Utilisation des espaces de nom

Il existe 3 manières pour faire référence à un élément via son namespace

1. Un nom sans qualificatif

```
$a = new foo();
```

2. Un nom qualifié ou préfixé

```
$a = new sousespacedenoms\foo();
```

3. Un nom absolu ou préfixé avec un opérateur global

```
$a = new \espacedenomscourant\foo();
```

Importation et alias

- PHP permet d'importer des constantes, classes, interfaces traits et énumérations depuis un namespace

```php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

class ConceptController extends AbstractController
```

- Il est également possible de donner à un alias à l'aide du mot clé `as`

```php
namespace foo;
use My\Full\Classname as Another;
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Merci pour votre attention

Des questions ?