Programmation Orientée Objet (POO)

Au programme :

Contenus	Capacités attendues	Commentaires
Paradigmes de programmation.	Distinguer sur des exemples les paradigmes impératif, fonctionnel et objet. Choisir le paradigme de programmation selon le champ d'application d'un programme.	Avec un même langage de programmation, on peut utiliser des paradigmes différents. Dans un même programme, on peut utiliser des paradigmes différents.
Vocabulaire de la programmation objet : classes, attributs, méthodes, objets.	Écrire la définition d'une classe. Accéder aux attributs et méthodes d'une classe.	On n'aborde pas ici tous les aspects de la programmation objet comme le polymorphisme et l'héritage.

« Paradigme » ?

- Un « paradigme » est une représentation du monde, une manière de voir les choses, un modèle de pensée qui repose sur une base définie.

Paradigme **impératif**

- La programmation impérative est un paradigme de programmation qui décrit les opérations en **séquences d'instructions exécutées par l'ordinateur pour modifier l'état du programme**.
- La programmation impérative se concentre sur la description du fonctionnement d’un programme : le **comment**.
- Ce type de programmation est **le plus répandu** parmi l'ensemble des langages de programmation existants : C, Pascal, commandes Unix, etc.

Paradigme **fonctionnel**

- La programmation fonctionnelle implique de **décomposer un problème en un ensemble de fonctions.**
- En programmation fonctionnelle **on décrit les résultats que l'on veut obtenir à partir des données** plutôt que la séquence d'instructions qui permettent d'obtenir les résultats.
- L'approche fonctionnelle considère le calcul en tant qu'évaluation de **fonctions** mathématiques. Vous donnez vos données en entrée aux fonctions, qui vous renvoient les valeurs calculées en sortie.
- En programmation fonctionnelle, **il n'y a pas d'état**, l'opération d'affectation est interdite, ce qui permet de s'affranchir des effets secondaires (ou effets de bord).

Paradigme **fonctionnel**

- Il existe des avantages théoriques et pratiques au style fonctionnel :
  - **preuves formelles** : Construire une preuve mathématique de l'exactitude d'un programme fonctionnel,
  - **modularité** : elle impose de décomposer le programme en petits morceaux,
  - **composabilité** : réutilisation des fonctions déjà écrites.
  - **facilité de débogage et de test** : Tester et déboguer un programme fonctionnelle est plus facile.

Paradigme **objet**

- La POO consiste en la définition et l'interaction de **briques logicielles appelées objets**; un objet représente un concept, une idée ou toute entité du monde physique.
- Les différents principes de la conception orientée objet aident à la **réutilisation** du code, au **masquage des données**.
- Mais comprendre toute la logique des objets et de leurs interactions est délicat et souvent difficile à tester en raison de ces interdépendances.

Programmation Orientée Objet (POO)

La programmation orientée objet (POO) est un **paradigme de programmation**, c'est-à-dire un ensemble de concepts et de modèles utilisés pour programmer avec des objets.

De nombreux langages de programmation sont orientés objets, par exemple : Ada, C++, C#, Eiffel, Java, JavaScript, Objective C, PHP, Python, Ruby, Smalltalk, VB.NET.

Programmation Orientée Objet (POO)

Une **classe** est une structure de données qui contient :

- des données (attributs) → toujours des noms
- des comportements (méthodes), qui sont les actions possibles sur cet objet → toujours des verbes

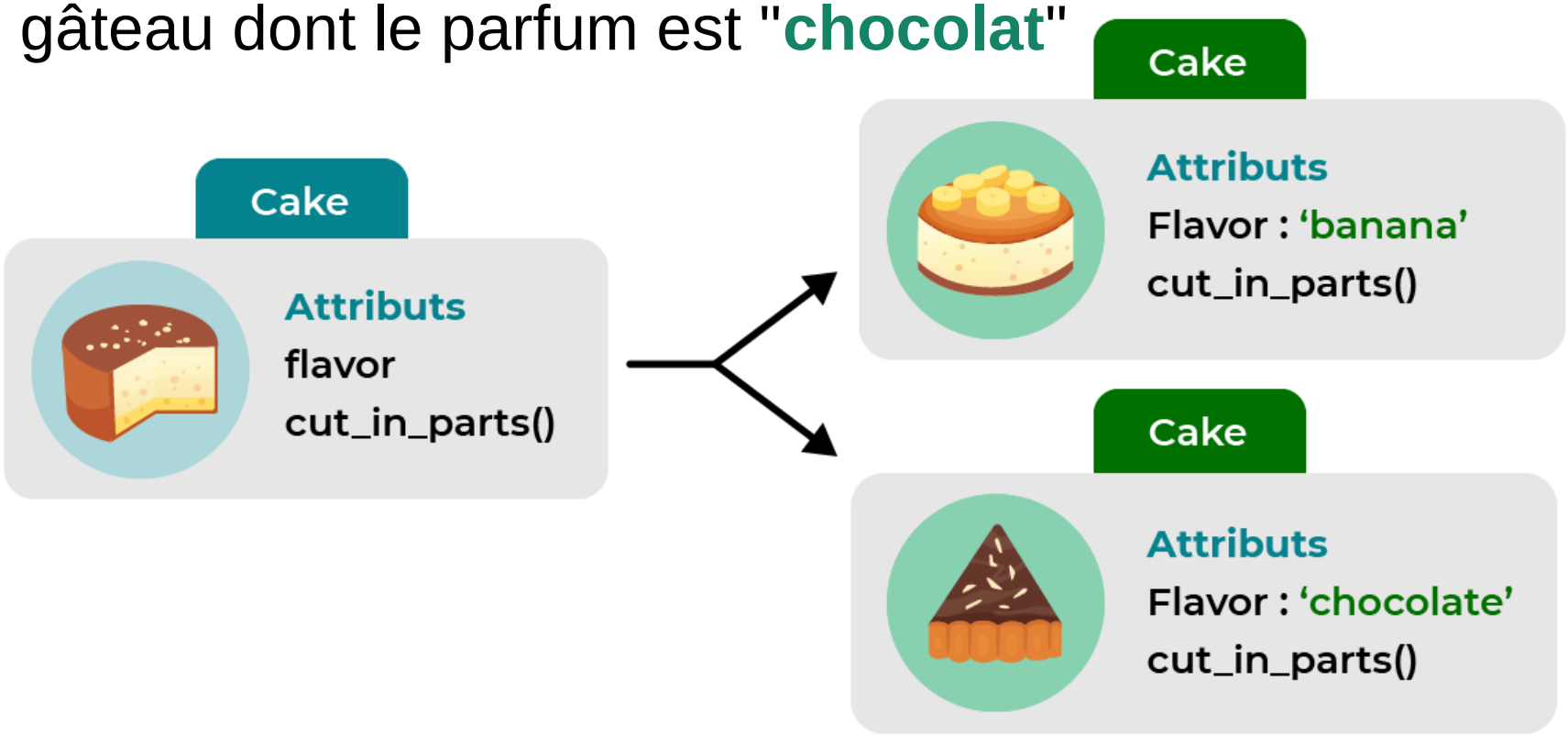
Chaque classe est une brique logicielle. On peut créer un (ou plusieurs) **objet** qui est une instance unique d'une classe avec ses valeurs propres.

Programmation Orientée Objet (POO)

Par exemple la classe "**Cake**" décrit un gâteau, avec comme donnée (attribut) son parfum (l'attribut "**flavour**") et comme méthode "**cut_in_parts**" (le couper en parts).

On a 2 instances de cette classe : 2 objets

- un gâteau dont le parfum est "**banana**"
- un gâteau dont le parfum est "**chocolat**"



Exemple : Classe Animal

```
class Animal:
    def __init__(self):
        self.__age = 0
        self.__nom = ''
    def get_age(self):
        return self.__age
    def get_nom(self):
        return self.__nom
    def vieillir(self):
        self.__age += 1
    def nommer(self, nom):
        self.__nom = nom
```

```
a1 = Animal()
print(a1.get_nom())
a1.nommer('Milou')
print(a1.get_nom())
```

Ressources

- Sur itsforeveryone.org/lycee :
  - Ce diaporama
  - **Cours POO : Lien vers cours et exercices (avec corrigés)**
    - **P1 : Structurer avec des classes**
    - **P2 : Paradigme Objet**
- Sur le site et sur papier :
  - **Paradigme de Programmation Orientée Objet (POO)**
    - Synthèse de la POO en Python
  - **Exercice 3 du sujet 0.B de l'épreuve écrite du bac NSI 2024**
    - Cet exercice porte sur la programmation Python (dictionnaire), la programmation orientée objet, les bases de données relationnelles et les requêtes SQL.