COURS

Programmation orientée objet : classes, attributs, méthodes

Tle

Programme de terminale générale, enseignement de spécialité — BO spécial n°8 du 25 juillet 2019

Relue le 14 novembre 2026

Un jeu vidéo manipule des personnages, une application bancaire des comptes : chacun a des caractéristiques et des comportements. La programmation orientée objet regroupe données et fonctions dans des objets, pour écrire un code plus clair et réutilisable.

1 Classe et objet

DÉFINITION

Modèle qui décrit un type d'objets : ses **attributs** (données) et ses **méthodes** (fonctions).

DÉFINITION

Exemplaire créé à partir d'une classe : `c = Compte("Léa", 100)`.

RÈGLE

Le **constructeur** `__init__(self, ...)` initialise les attributs ; `self` désigne l'objet en cours.

2 Écrire une classe

RÈGLE

La classe `Compte` commence par `class Compte:`. Son constructeur `def __init__(self, titulaire, solde):` affecte `self.titulaire = titulaire` et `self.solde = solde`. La méthode `def deposer(self, montant):` exécute `self.solde = self.solde + montant`.

RÈGLE

Appel : `c.deposer(50)` ; accès : `c.solde`.

3 Intérêt et vocabulaire

DÉFINITION

Principe qui consiste à manipuler un objet par ses méthodes plutôt que directement par ses attributs : on contrôle les modifications (un retrait ne rend pas le solde négatif).

RÈGLE

La POO facilite la **modularité** : chaque classe a une responsabilité claire. Elle sert à implémenter des structures de données (pile, file, arbre).

EXEMPLE

Ajoute une méthode retirer qui refuse un retrait supérieur au solde.

1. Définir `def retirer(self, montant):`.
2. Tester `if montant <= self.solde:` puis soustraire.
3. Sinon renvoyer `False` (ou lever une erreur) sans modifier le solde.

Une méthode contrôle l'état de l'objet : c'est l'intérêt de l'encapsulation.

LE PIÈGE QUI COÛTE LE PLUS DE POINTS

Oublier self dans la définition d'une méthode. — Python passe l'objet en premier argument : sans self, l'appel `c.deposer(50)` provoque une erreur.

Le réflexe : Écrire toujours self en premier paramètre.

CONCEVOIR UNE CLASSE

1. Identifier les données de l'objet (attributs).
2. Identifier ses actions (méthodes).
3. Écrire le constructeur **init**.
4. Tester en créant plusieurs instances.

À VÉRIFIER

- Ai-je défini le vocabulaire ?
- Mon code est-il testé sur un exemple ?
- Ai-je traité les cas limites ?
- Ai-je justifié la complexité ou le résultat ?

À RETENIR

- **Classe** (modèle) / **objet** (instance) ; attributs et méthodes.
- `__init__` et `self` ; appel `objet.methode()`.
- **Encapsulation**, modularité ; implémenter des structures de données.