Avec ce **Concept #5 : Encapsulation**, tu entres dans un aspect plus **propre** et **sécurisé** de la POO : **cacher et contrôler l’accès aux données** 🔐

**🧠 Concept #5 — Encapsulation (Protection des données)**

**🔹 1. Qu’est-ce que l’encapsulation ?**

L’**encapsulation**, c’est le fait de **protéger les attributs internes d’un objet**, pour éviter qu’ils soient **modifiés de manière non contrôlée**.

Tu **masques les détails internes**, et tu **exposes seulement ce qui est nécessaire**.

🗣️ En clair : "Tu peux utiliser mon objet, mais pas toucher à ce qu’il y a sous le capot, sauf si je t’y autorise."

**🧩 Niveau de visibilité en Python :**

| **Syntaxe** | **Nom** | **Accès** | **Usage** |
| --- | --- | --- | --- |
| public | self.nom | Libre | Attribut classique |
| protected | self.\_nom | Conventionnel | Usage interne uniquement |
| private | self.\_\_nom | Caché/renommé | Ne peut pas être accédé directement |

⚠️ En Python, **rien n’est vraiment privé** (on peut tout forcer), mais on suit **des conventions**.

**📘 Exemple d’encapsulation :**

python

CopyEdit

class Compte:

def \_\_init\_\_(self, titulaire, solde):

self.\_\_titulaire = titulaire # Attribut privé

self.\_\_solde = solde # Attribut privé

def afficher\_solde(self):

print(f"Solde de {self.\_\_titulaire} : {self.\_\_solde} €")

💡 Ici, \_\_titulaire et \_\_solde ne peuvent pas être modifiés directement.