3. Exemple pratique 2 : Approche inspirée de la vie réelle

Objectif :

Modéliser des voitures en utilisant la POO et simuler des interactions telles que l'accélération, le ralentissement, et une course entre deux voitures. Cela permet de comprendre comment gérer plusieurs objets et simuler des interactions comme dans la vie courante.

Étape 1 : Création d'une classe Voiture

Nous allons créer une classe `Voiture` qui modélise les caractéristiques et les comportements d'une voiture.

1. **Caractéristiques (Attributs) :**
    - **marque** : La marque de la voiture (ex : Tesla, BMW).
    - **modèle** : Le modèle de la voiture (ex : Model S, M3).
    - **vitesse_max** : La vitesse maximale que la voiture peut atteindre.
    - **vitesse_actuelle** : La vitesse actuelle de la voiture (initialement 0).
2. **Comportements (Méthodes) :**
    - **accelerer(valeur)** : Augmente la vitesse actuelle de la voiture de `valeur`, sans dépasser la vitesse maximale.
    - **ralentir(valeur)** : Diminue la vitesse actuelle de la voiture de `valeur`, sans descendre en dessous de 0.

Code de la classe Voiture :

```
class Voiture:
    def __init__(self, marque, modele, vitesse_max):
        # Attributs de la voiture
        self.marque = marque
        self.modele = modele
        self.vitesse_max = vitesse_max
        self.vitesse_actuelle = 0  # La voiture démarre à l'arrêt

    def accelerer(self, valeur):
        """Cette méthode permet d'accélérer la voiture."""
        self.vitesse_actuelle += valeur
        if self.vitesse_actuelle > self.vitesse_max:
            self.vitesse_actuelle = self.vitesse_max  # Limite la vitesse à la
vitesse max
        print(f"La voiture {self.marque} {self.modele} accélère à
{self.vitesse_actuelle} km/h.")

    def ralentir(self, valeur):
        """Cette méthode permet de ralentir la voiture."""
        self.vitesse_actuelle -= valeur
        if self.vitesse_actuelle < 0:
            self.vitesse_actuelle = 0  # La voiture ne peut pas aller en dessous
de 0
        print(f"La voiture {self.marque} {self.modele} ralentit à
{self.vitesse_actuelle} km/h.")
```

Explication du code :

- **Constructeur `__init__` :**
  - Il est utilisé pour initialiser les attributs de la voiture. Par exemple, on passe la **marque**, le **modèle** et la **vitesse maximale** en paramètres lors de la création de l'objet.
- **Méthode `accelerer()` :**
  - Elle prend une valeur comme argument et augmente la vitesse de la voiture. Si la vitesse dépasse la vitesse maximale, elle est limitée à la vitesse max.
- **Méthode `ralentir()` :**
  - Elle diminue la vitesse de la voiture en fonction de la valeur donnée. La vitesse ne peut pas être négative (elle est remise à 0 si elle descend en dessous).

Étape 2 : Création et manipulation de voitures

Maintenant, nous allons créer des objets à partir de la classe `Voiture` et interagir avec eux.

Code de la création des voitures :

```
# Création de voitures
voiture1 = Voiture("Tesla", "Model S", 250)  # Voiture 1 avec une vitesse max de
250 km/h
voiture2 = Voiture("BMW", "M3", 230)         # Voiture 2 avec une vitesse max de
230 km/h

# Manipulation des voitures
voiture1.accelerer(50)  # La voiture 1 accélère à 50 km/h
voiture2.accelerer(70)  # La voiture 2 accélère à 70 km/h

voiture1.ralentir(20)   # La voiture 1 ralentit à 30 km/h
voiture2.ralentir(50)   # La voiture 2 ralentit à 20 km/h
```

Explication du code :

- `Voiture("Tesla", "Model S", 250)` : Cette ligne crée un objet de la classe `Voiture` avec la marque **Tesla**, le modèle **Model S**, et une vitesse maximale de **250 km/h**.
- **Appel des méthodes** : Nous utilisons les méthodes `accelerer` et `ralentir` pour manipuler la vitesse des voitures.
  - La voiture 1 accélère à 50 km/h, puis ralentit à 30 km/h.
  - La voiture 2 accélère à 70 km/h, puis ralentit à 20 km/h.

Étape 3 : Ajout d'une interaction entre voitures

Dans cette étape, nous allons simuler une **course** entre deux voitures. Nous utiliserons une boucle pour que chaque voiture accélère à chaque tour de boucle jusqu'à ce qu'une des voitures atteigne sa vitesse maximale.

Code de la course :

```
# Fonction pour simuler une course entre deux voitures
```

```
def course(voiture1, voiture2):
    while voiture1.vitesse_actuelle < voiture1.vitesse_max and
voiture2.vitesse_actuelle < voiture2.vitesse_max:
        voiture1.accelerer(20)
        voiture2.accelerer(20)

    # Afficher le résultat final
    if voiture1.vitesse_actuelle == voiture1.vitesse_max:
        print(f"{voiture1.marque} {voiture1.modele} a atteint sa vitesse
maximale en premier !")
    elif voiture2.vitesse_actuelle == voiture2.vitesse_max:
        print(f"{voiture2.marque} {voiture2.modele} a atteint sa vitesse
maximale en premier !")

# Lancer une course entre les deux voitures
course(voiture1, voiture2)
```

Explication du code :

- `while voiture1.vitesse_actuelle < voiture1.vitesse_max` : Cette boucle continue tant qu'aucune des deux voitures n'a atteint sa vitesse maximale.
    - À chaque itération, les voitures accélèrent de 20 km/h.
- **Vérification finale** : Après la course, nous affichons quelle voiture a atteint la vitesse maximale en premier.

Améliorations et extensions possibles :

- **Ajouter des conducteurs :**
    - Nous pourrions ajouter une classe `Conducteur` pour simuler les conducteurs des voitures. Les conducteurs pourraient avoir des caractéristiques comme leur expérience ou leur style de conduite.

```
class Conducteur:
    def __init__(self, nom, age, experience):
        self.nom = nom
        self.age = age
        self.experience = experience  # Années d'expérience

    def conduire_voiture(self, voiture, acceleration):
        voiture.accelerer(acceleration)
        print(f"{self.nom} conduit la {voiture.marque} {voiture.modele} à
{voiture.vitesse_actuelle} km/h.")
```

- **Ajouter du carburant :**
    - Pour rendre la simulation plus réaliste, nous pourrions ajouter un attribut `carburant` qui diminuerait à chaque accélération. Si le carburant est à zéro, la voiture ne pourrait plus accélérer.

```
class Voiture:
    def __init__(self, marque, modele, vitesse_max, carburant):
        self.marque = marque
        self.modele = modele
        self.vitesse_max = vitesse_max
        self.vitesse_actuelle = 0
```

```
        self.carburant = carburant  # Quantité de carburant en litres

    def accelerer(self, valeur):
        if self.carburant > 0:
            self.vitesse_actuelle += valeur
            self.carburant -= 1  # Réduction du carburant à chaque accélération
            if self.vitesse_actuelle > self.vitesse_max:
                self.vitesse_actuelle = self.vitesse_max
            print(f"La voiture {self.marque} {self.modele} accélère à
{self.vitesse_actuelle} km/h avec {self.carburant} litres de carburant
restants.")
        else:
            print(f"La voiture {self.marque} {self.modele} n'a plus de carburant
pour accélérer.")
```

Objectif atteint :

En utilisant cette approche, nous avons pu modéliser des voitures, comprendre comment manipuler plusieurs objets, et simuler des interactions entre eux. Ces exemples vous offrent une base solide pour comprendre les principes fondamentaux de la POO tout en développant des solutions pratiques inspirées de la vie réelle.