Exercice 1 : Création de Classes et d'Objets

Objectif : Comprendre la création de classes, d'objets et l'utilisation des attributs et méthodes.

- Créez une classe Voiture avec les attributs suivants :
    - marque (string)
    - modele (string)
    - annee (int)
    - kilometrage (int, initialisé à 0)
- Ajoutez les méthodes suivantes :
    - demarrer : Affiche "La voiture démarre".
    - rouler : Accepte un nombre de kilomètres en paramètre et met à jour kilometrage.
- Créez un objet de la classe Voiture avec des valeurs pour marque et modele. Appelez ses méthodes.

Exercice 2 : Attributs de Classe et d'Instance

Objectif : Différencier les attributs de classe et d'instance.

1. Ajoutez un attribut de classe nombre_voitures à la classe Voiture pour compter le nombre total de voitures créées.
2. Modifiez le constructeur (__init__) pour incrémenter nombre_voitures à chaque création d'un objet.
3. Affichez le nombre total de voitures après avoir créé plusieurs objets.

Exercice 3 : Méthodes Spéciales

Objectif : Utiliser des dunder methods pour personnaliser le comportement des objets.

1. Ajoutez à la classe Voiture :
    - La méthode __str__ pour afficher les informations sur la voiture (ex. : "Voiture : Marque X, Modèle Y").
    - La méthode __eq__ pour comparer deux voitures selon leur marque et leur modele.
2. Testez ces méthodes en créant plusieurs objets Voiture.

Exercice 4 : Héritage

Objectif : Appliquer le concept d'héritage.

1. Créez une classe Camion qui hérite de la classe Voiture.
    - Ajoutez un nouvel attribut charge_max (int).
    - Ajoutez une méthode charger qui accepte un poids et vérifie s'il dépasse charge_max.
2. Créez un objet de type Camion et testez ses méthodes.

Exercice 5 : Polymorphisme

Objectif : Manipuler des méthodes partagées dans des classes différentes.

1. Créez une classe Moto avec les mêmes attributs que Voiture mais une méthode différente faire_wheelie qui affiche "La moto fait un wheelie !".
2. Créez une fonction générique afficher_infos qui accepte un objet de type Voiture ou Moto et affiche ses informations en appelant sa méthode __str__.

Exercice 6 : Encapsulation

Objectif : Utiliser des attributs privés pour restreindre l'accès.

1. Modifiez la classe Voiture :
    - Rendez kilometrage privé.
    - Ajoutez une méthode get_kilometrage pour accéder à sa valeur.
    - Ajoutez une méthode set_kilometrage pour modifier sa valeur, en autorisant uniquement des valeurs positives.
2. Testez ces changements avec des objets de la classe.

Exercice 7 : Méthodes Statiques et de Classe

Objectif : Différencier les méthodes statiques et les méthodes de classe.

1. Ajoutez à la classe Voiture :
    - Une méthode statique afficher_message qui affiche "Bienvenue dans notre système de gestion de voitures".
    - Une méthode de classe nombre_total qui affiche le nombre total de voitures.
2. Testez ces méthodes sans créer d'objet.

Exercice 8 : Projets Pratiques

Objectif : Mettre en œuvre plusieurs concepts de POO dans un projet complet.

1. **Gestion des étudiants** :
    - Créez une classe Etudiant avec les attributs nom, age, et moyenne.
    - Ajoutez des méthodes pour :
        - Vérifier si l'étudiant est admis (moyenne >= 10).
        - Afficher les détails de l'étudiant.
    - Créez une classe Groupe pour gérer une liste d'étudiants, avec des méthodes pour :
        - Ajouter ou supprimer un étudiant.
        - Calculer la moyenne générale du groupe.
2. **Système de calculatrice** :
    - Créez une classe Calculatrice avec des méthodes pour les opérations de base (+, -, *, /).
    - Ajoutez des méthodes pour enregistrer un historique des calculs effectués.
    - Créez une interface simple dans le terminal pour tester les fonctionnalités.

Exercice : Gestion d'une Bibliothèque

Vous allez développer un système de gestion pour une bibliothèque. Ce projet vous permettra d'explorer les classes, les objets, les méthodes, l'héritage, le polymorphisme, et l'encapsulation.

Description du système :

1. Une bibliothèque contient des **livres** et des **magazines**.
2. Chaque livre/magazine a des attributs communs, tels que :
    - titre (str)
    - auteur (str)
    - annee_publication (int)
3. Les livres ont des attributs spécifiques :
    - nombre_pages (int)
    - genre (str, par exemple : "Roman", "Science", etc.)
4. Les magazines ont un attribut spécifique :
    - frequence_publication (str, par exemple : "Mensuel", "Hebdomadaire").
5. La bibliothèque doit gérer une collection de livres et de magazines et permettre des actions comme :
    - Ajouter un livre ou un magazine.
    - Supprimer un livre ou un magazine.
    - Rechercher un élément par titre ou par auteur.
    - Afficher la liste des livres ou des magazines.

Étapes à suivre :

1. Classe ElementBibliotheque

Créez une classe parent ElementBibliotheque avec les attributs communs :

- titre
- auteur
- annee_publication

Ajoutez les méthodes suivantes :

- Un constructeur __init__ pour initialiser les attributs.
- Une méthode __str__ pour afficher une description d'un élément (par exemple : "Titre : X, Auteur : Y, Année : Z").

2. Classe Livre

Créez une classe Livre qui hérite de ElementBibliotheque :

- Ajoutez les attributs nombre_pages et genre.
- Surchargez la méthode __str__ pour inclure les informations spécifiques aux livres.

3. Classe Magazine

Créez une classe Magazine qui hérite de ElementBibliotheque :

- Ajoutez l'attribut frequence_publication.
- Surchargez la méthode __str__ pour inclure les informations spécifiques aux magazines.

4. Classe Bibliotheque

Créez une classe Bibliotheque pour gérer les livres et les magazines :

- Ajoutez un attribut collection qui est une liste contenant tous les éléments (livres et magazines).

Ajoutez les méthodes suivantes :

1. ajouter_element(self, element) :
    - Permet d'ajouter un livre ou un magazine à la collection.
2. supprimer_element(self, titre) :
    - Supprime un élément de la collection en fonction du titre.
3. rechercher_par_titre(self, titre) :
    - Recherche un élément par son titre et affiche ses informations.
4. rechercher_par_auteur(self, auteur) :
    - Affiche tous les éléments écrits par un auteur donné.
5. afficher_liste(self) :
    - Affiche tous les livres et magazines de la collection.