TP de Programmation avancée en Python : Série 3

Exercice 1 : Classe Person.

1) - Ecrire une **classe Python** nommée "Person" ayant comme attributs : **name** (nom), **age** (âge) et **gender** (genre). Implémentez ensuite les méthodes suivantes :

1. **Un constructeur __init__() :** qui prend les paramètres name, age et gender, et initialise les attributs correspondants de l'instance.
2. **Une méthode d'instance nommée introduce() :** qui affiche une présentation de la personne sous le format "Je m'appelle [name], j'ai [age] ans et je suis [gender]."

2) - Créez une autre **classe Python** nommée **"Student"** qui hérite de la **classe "Person"**. La **classe "Student"** aura un attribut supplémentaire : **section**. Implémentez les méthodes suivantes :

1. **Une méthode constructeur __init__ () :** qui prend comme paramètres name, age, gender et section, et initialise les attributs correspondants de l'instance en utilisant la méthode d'initialisation de la classe parent "Person".
2. **Une méthode affiche une présentation de l'étudiant nommée introduce_student() :** qui affiche une introduction du type "Je m'appelle [name], j'ai [age] ans et je suis [gender]. Je suis dans la section [section]."

Exercice 2 : Classe Vehicule

Imaginez que vous travailliez sur le **développement d'un logiciel de gestion de véhicules** pour une entreprise de **location de véhicules**. On suppose qu'il existe deux types **principaux de véhicules** : les **voitures** et les **motos**. Votre objectif est de créer un système permettant de suivre et de gérer les informations relatives à chaque véhicule, notamment **la marque, le modèle et l'année de fabrication**. En outre, vous souhaitez que chaque véhicule puisse être **mis en marche** et **arrêté** à la demande.
Pour atteindre cet objectif, vous êtes obligé de suivre les étapes suivantes :

1. **Établissez une classe de base nommée "Vehicule" :** avec des attributs tels que "marque", "modele" et "annee". Cette classe doit être dotée de deux méthodes essentielles : "demarrer()" et "arreter()".
2. **Créez deux sous-classes, "Voiture" et "Moto" :** qui héritent des caractéristiques de la classe "Vehicule". Dans ces sous-classes, ajoutez des attributs spécifiques supplémentaires tels que "nombre_de_portes" pour les voitures et "cylindree" pour les motos.
3. **Personnalisez la méthode "init()" :** dans les sous-classes pour initialiser les attributs spécifiques de chaque type de véhicule en plus des attributs hérités de la classe "Vehicule".
4. **Modifiez les méthodes "demarrer()" et "arreter()" :** dans les sous-classes pour qu'elles affichent des messages appropriés, par exemple, "La voiture démarre" ou "La moto s'arrête", en fonction du type de véhicule.
5. **Créez des instances de voitures et de motos :** en utilisant les sous-classes "Voiture" et "Moto". Appelez ensuite les méthodes de ces instances pour mettre les véhicules en marche et les arrêter, tout en affichant les détails spécifiques à chaque véhicule.

Exercice 3 : Classe Voiture

1) **classe Voiture**
Ecrire une **classe Python** nommée **Voiture** ayant les méthodes et attributs suivants :

1. **marque :** (type string ie chaîne de caractères)
2. **puissance_fiscale :** (type int ie un entier)

3. **carburant :** (type string ie une chaîne de caractères)
4. **Créer un constructeur __init__ :** constructeur de la classe.
5. **Créer une méthode afficher_infos() :** permettant d'afficher les informations de la voiture (marque, puissance fiscale, carburant).

2) héritage

Créez une **classe Renault héritant** de la classe **Voiture**.

Créer un attribut supplémentaire à la classe Renault : **modele** (type string ie une chaîne de caractères)
Créer un constructeur à la classe Renault : permattant d'appeller le constructeur de la **classe mère (Voiture)** et initialise l'**attribut modele.**
Ajoutez une méthode afficher_modele() : à la classe Renault qui affiche le modèle de la voiture.

3) Getters et Setters

Ajouter des méthodes **get_<attribut>** et **set_<attribut>** pour chaque attribut, permettant d'accéder et de modifier les attributs pour les **deux classes "Voiture"** et **"Renault".**

4) Tester l'Héritage

1. Créez un **objet** d'**instance** de la **classe Renault.**
2. Se servire des **méthodes** pour **afficher les informations** et le **modèle de la voiture**, puis modifiez quelques **attributs** en utilisant les **setters.**
3. Réaffichez les **informations** pour confirmer que les modifications ont été prises en compte.

Exercice 4 : Héritage multiple

Créez **deux classes de base**, **ClasseA** et **ClasseB**, en langage de programmation **Python**. Chacune de ces classes possède des méthodes distinctes : **methode_A()** pour **ClasseA**, affichant le message **"Hello From ClassA"**, et **methode_B()** pour **ClasseB**, affichant le message **"Hello From ClassB"**.
Définissez ensuite une **classe dérivée** nommée **ClasseC**, qui **hérite simultanément** des fonctionnalités de **ClasseA** et **ClasseB**.
Introduisez une méthode supplémentaire, **methode_C()**, propre à **ClasseC**, affichant le message **"Hello From ClasseC"**. Testez le fonctionnement de cette méthode pour assurer son bon déroulement.

Exercice 5 : Classe Héritage multiple 2

1. Développez deux **classes de base en langage Python**, à savoir **Machine** et **Camera**. Chacune de ces classes devrait comporter des **méthodes spécifiques** liées à **ses caractéristiques fonctionnelles**. Par exemple, la **classe Machine** pourrait être dotée de la méthode **allumer_machine()**, tandis que la **classe Camera** pourrait inclure la méthode **prendre_photo()**.
2. Élaborez une **classe Smartphone** exploitant le principe d'**héritage multiple** pour intégrer simultanément les fonctionnalités de **Machine** et de **Camera**. Au sein de la **classe Smartphone**, vous pouvez incorporer des **méthodes spécifiques**, telles que **appeler()** ou **utiliser_appareil_photo()**.
3. Employez la fonction **super()** pour invoquer de manière appropriée les **méthodes** des **classes parentes** au sein de la **classe Smartphone**. À titre d'exemple, si la **classe Machine** dispose d'une méthode **allumer_machine()**, vous pouvez la solliciter au moyen de **super().allumer_machine()**.
4. Effectuez l'**instanciation** d'un **objet** de la **classe Smartphone** et **procédez à l'appel de diverses méthodes** pour vérifier leur fonctionnement.

Exercice 6 :Poly Classe

Écrivez trois classes, chacune implémentant une méthode **show()** qui prend un argument, une chaîne de caractères. La méthode show doit afficher le nom de la classe ainsi que le message. Ensuite, créez une liste d'instances et appelez la méthode **show()** sur chaque objet de la liste

Exercice 7 :Poly Classe 2

1. Créer une classe **Forme** avec une méthode **aire().**
2. Créer 3 classes héritées :
   - **Cercle**
   - **Rectangle**
   - **Carré**
   - **Trapéze**
3. Dans chaque classe, réécrire la méthode **aire()**.
4. Créer une liste contenant plusieurs formes.
5. Afficher l'aire de chaque forme avec une boucle.

Exercice 8 :Poly Classe 3

Dans cet exercice, nous allons implémenter **3 classes** : une classe principale appelée **TravailleurEntreprise** et deux classes filles que nous appellerons respectivement **Directeur** et **Ingenieur**.
Chaque classe doit contenir les méthodes suivantes :

Classe TravailleurEntreprise :

- **__init__(self, ...)** : un constructeur qui permet d'initialiser trois attributs, à savoir le nom du travailleur, son salaire et son âge.
- **afficher _fonction(self)** : une méthode qui permet d'afficher la phrase : « Je suis un travailleur d'une entreprise ».
- **afficher _info(self)** : une méthode qui permet d'afficher les informations relatives au travailleur, comme son nom, son salaire et son âge.

Classe Directeur :

- **__init__(self, ...)** : un constructeur qui permet d'initialiser quatre attributs, à savoir le nom du directeur, son salaire, son âge et la prime qu'il a reçue.
- **afficher _fonction (self)** : une méthode qui permet d'afficher la phrase: « Je suis un directeur d'entreprise »
- **afficher _info (self)** : une méthode qui permet d'afficher les informations relatives au directeur, comme son nom, son salaire, son âge et la prime qu'il a reçue en euros.

Classe Ingenieur :

- **__init__(self, ...)** : un constructeur qui permet d'initialiser quatre attributs, à savoir le nom de l'ingénieur, son salaire, son âge et sa spécialité.
- **afficher _funcion(self)** : une méthode qui permet d'afficher la phrase : « Je suis un ingénieur ».
- **afficher _info (self)** : une méthode qui permet d'afficher les informations relatives à l'ingénieur, comme son nom, son salaire, son âge et sa spécialité. *(Voir la sortie dans la console)*

```
Je suis un directeur d'entreprise
Nom:  Benani Mohamed
Salaire:  35000
Age:  43
La prime annuelle perçue est 5000 DH.

Je suis ungénieur
Nom:  Iraki Adil
Salaire:  27000
Age:  32
Je suis ingénieur spécialisé en intelligence artificielle
```