Exercices de POO intermédiaire en Python

Exercice 1 : Attributs de classe et encapsulation

Dans cet exercice vous aller créer une classe chargée de représenter un objet que vous connaissez tous : les Renault Clio.

Vous allez donc déclarer une classe clio dans laquelle nous allons intégrer quelques caractéristiques d'une vraie clio à savoir : le nombre de portes, la couleur de la carrosserie, et le prix neuf de la voiture.

L'objectif de cet exercice est de vous faire concevoir des objets plus complexes dont le comportement est plus proche de ceux observés dans le monde réel. Il n'y aura donc pas que de simples attributs et méthodes dans votre classe. Vous allez devoir déclarer des attributs de classe quand cela est nécessaire mais aussi peut-être des méthodes de classes ou encore encapsuler des attributs.

Ne vous inquiétez pas nous allons y aller étape par étape.

Pour vous aider, voici déjà quelques petites informations que vous devriez savoir :

- Une clio n'a que deux nombres de portes possibles : 3 et 5, c'est une spécificité du modèle, vous ne verrez jamais de clio avec 7 portes. Cela est donc propre au modèle Clio à quoi correspond la notion de modèle en POO ?
- Une clio n'a que 5 couleurs de carrosserie possibles (à vous de choisir), une couleur est définie par un nom « bleu_nuit » par exemple et une référence couleur telle que 213800058. Vous ne pouvez construire que des Clios avec les couleurs autorisées, là encore cet ensemble clés/valeurs appartient au modèle. Quand on détermine la couleur de la clio on doit également déterminer un autre attribut : le numéro de la couleur.
- Le prix neuf de la voiture est commun à toute les clios créées, si le prix neuf d'une voiture change alors celui de toutes les voitures change aussi.

Votre classe aura à minima les attributs suivants :

- number_doors : le nombre de porte de la voiture, s'assurer que ce soit 3 ou 5
- color_number : le numéro de couleur de la voiture (je vous laisse choisir les chiffres associés), ce numéro est assigné en même temps que la couleur
- color : le nom de la couleur de la voiture, s'assurer que ce soit une couleur autorisée

Concrètement vous allez stocker les valeurs autorisées dans des attributs de classe, créer des attributs d'objet mais assigner et récupérer leurs valeurs à l'aide de setters et de getters qui eux s'occuperont d'effectuer le travail de vérification nécessaire sur la base des attributs de classe.

Le prix neuf des voitures est quand à lui modifiable mais vous devez vous assurer qu'il soit compris entre 8000 et 30000 euros.

Exercice 2 : héritage

Etape 1 :

Créez deux classes sur des fichiers séparés « voiture » et « bus », ces classes permettent de gérer :
- L'immatriculation du véhicule

- La couleur du véhicule

Cependant chaque classe a ses spécificités, la classe voiture permet en plus de :
- gérer le nombre de portes de la voiture

La classe bus permet de :
- gérer le nombre d'étages du bus

Attention ces deux classes devront hériter d'une classe commune qui leur permettra de partager certaines caractéristiques.

Le nombre d'étages et de portes sont des valeurs qui ne changent jamais ! Une voiture ne peut pas avoir 1 porte par exemple, de même un bus à 3 étages n'existe pas.

Etape 2 :

Créez une instance de la classe bus et de la classe voiture, donnez leur l'immatriculation et la couleur de votre choix. Indiquez le nombre d'étages de votre choix pour le bus et le nombre de portes de votre choix pour la voiture.

Etape 3 :

Affichez les valeurs des propriétés de chaque véhicule à la suite.
Changez la couleur de la voiture et le nombre d'étages du bus.
Affichez à nouveau les valeurs des attributs

N'oubliez pas de vous assurer que les valeurs pour le nombre de portes et d'étages soient bien les valeurs attendues. Par exemple je ne peux pas créer un bus à trois étages. Cela vous permettra de réviser l'encapsulation, les getters et les setters.

Exercice 3 : Les méthodes magiques

Dans cet exercice nous allons simuler quelques comportements d'objets que nous pourrions retrouver dans la gestion d'un magasin.

Votre programme sera composé de deux classes principales : Customer et Employee. Toutes deux héritent d'une classe commune Person.

La classe personne définit les attributs nom, prénom et âge qui sont communs à la fois au client et à l'employé.

La classe client a comme attributs spécifiques un panier, vide par défaut, qui pourra contenir les produits que le client achète et un attribut avec le montant total à payer lors de son passage en caisse. Ce montant est mis à jour à chaque fois qu'un produit est ajouté au panier.

La classe employé possède quant à elle un attribut particulier, le statut du salarié. Par défaut celui-ci est employé. Les statuts possibles sont : employé, technicien, manager et cadre par ordre de hiérarchie.

Vous aurez également besoin de créer une classe Product qui représente les produits du magasin. Elle possède les attributs prix et nom du produit.

Vous pouvez maintenant instancier dans votre fichier main.py quelques produits factices, un client et un employé.

Cependant nous ne nous arrêtons pas en si bon chemin. Afin de faciliter l'utilisation de l'application, nous souhaitons modifier le comportement natif des classes Customer et Employee.

Pour la classe Customer, je dois pouvoir :

- Afficher une carte d'identité complète du client par un simple print. Autrement-dit, si j'ai une variable customer contenant une instance de la classe Customer je dois pouvoir écrire print(customer) et obtenir dans le terminal un petit texte affichant le nom, prénom et l'âge du client mais aussi le nom des produits qu'il a dans son panier et le montant à payer.

- Ajouter un produit au panier par une simple addition. Par exemple, si j'ai une variable customer représentant une instance de la classe Customer et une variable product représentant une instance de la classe Product, je dois pouvoir écrire customer + product. Le produit en question est alors ajouté au panier du client et le montant total à payer en caisse est mis à jour.

Pour la classe Employee, je dois pouvoir :

- Afficher une carte d'identité complète de l'employé par un simple print. Autrement-dit, si j'ai une variable employee contenant une instance de la classe Employee je dois pouvoir écrire print(employee) et obtenir dans le terminal un petit texte affichant le nom, prénom et l'âge de l'employé mais aussi sont statut.

- Vérifier que l'employé a le statut requis grâce à l'opérateur >=. Autrement-dit, si j'ai une variable employee contenant une instance de la classe Employee avec le statut 'employee' alors je dois pouvoir écrire print(employee >= 'manager') et cela affichera False dans la console. Inversement si j'écris print(employee >= 'employee') , cela doit afficher True dans la console.

Testez les comportements attendus dans le fichier main.py