TP1 : Introduction à la POO

Les notions abordées

Dans ce premier TP, vous allez découvrir les éléments essentiels de la programmation orientée objets avec Python. Le sujet vous propose de programmer une classe Voiture. Pour cela, vous aurez à manipuler deux fichiers, le premier appelé Voiture.py et qui contient la classe Voiture et le second qui est le fichier main.py où l'on pourra manipuler l'objet Voiture.

Partie 1 : le constructeur

Le constructeur permet de créer les instances de l'objet définit par la classe. Il est donc une étape essentielle dans la programmation d'un objet. La déclaration du constructeur se fait avec la fonction `__init__()`. Le constructeur permet de spécifier les attributs de l'objet. Nous allons spécifier ici trois attributs à savoir la marque de la voiture, son modèle ainsi que sa puissance fiscale.

1. Créer le constructeur correspondant à la classe Voiture et créer une instance nommée mycar1 dans le fichier main.py.
2. Ajouter l'attribut couleur et modifier le constructeur en conséquence.
3. Préparer un affichage pour voir les attributs de l'objet mycar1

Il est possible de spécifier dans le constructeur des valeurs par défaut. Cette technique va nous permettre de créer le constructeur par défaut, c'est à dire sans aucune valeur donnée lors de la création d'une instance. La spécification d'une valeur par défaut se fait en ajoutant ='valeur par défaut' après chaque attribut.

4. Définir des valeurs par défaut pour les attributs de la classe Voiture et tester le fonctionnement du constructeur dans les cas suivants. Pour chaque cas vous faites l'affichage des attributs de l'objet en question :
    a. Créer l'instance mycar2 avec constructeur vide.
    b. Créer l'instance mycar3 avec constructeur spécifiant trois valeurs pour les trois attributs.
    c. Créer l'instance mycar4 avec constructeur spécifiant une valeur uniquement pour l'attribut couleur.

Partie 2 : attributs publics et attributs privés

Les attributs d'une classe peuvent être soient publics ou privés. Dans le premier cas, il est possible d'accéder à un attribut en lecture/écriture simplement en faisant `objet.attribut`. C'est ce que vous avez fait dans la partie précédente. Ainsi lorsque nous avons créé l'objet mycar1, nous avons déclaré des attributs publics.

Dans le second cas il est impossible d'accéder directement aux attributs en dehors de l'objet.
Il est préférable dans la POO de déclarer les attributs privés et de coder des fonctions pour accéder soit en lecture ou en écriture à chaque attribut (accesseurs). Pour déclarer un attribut comme privé, il suffit de précéder son nom par un `__` (e.g. `self.__attribut`)

5. Transformer les attributs publics en privés et observer qu'il est impossible d'y accéder en dehors de la classe Voiture.

Partie 3 : les accesseurs

Maintenant que les attributs sont privés et qu'il n'est plus possible d'accéder directement à leurs valeurs, nous allons coder des accesseurs. Ils seront de deux types à savoir ceux en lecture et ceux en écriture.

Les accesseurs en lecture sont des fonctions qui renvoient simplement la valeur de l'attribut d'un objet de la classe et dont le nom commence généralement par get.
Á propos du nom des classes, variables, fonctions, il est fortement recommandé de suivre les conventions de nommage permettant la relecture du code. Pour cela, vous avez différentes pages notamment www.namingconvention.org/python/. Vous apporterez une attention particulière à bien respecter cette convention.

6. Implémenter l'ensemble des accesseurs en lecture pour chaque attribut de la classe Voiture.

L'accesseur en écriture est une fonction qui permet de modifier la valeur d'un attribut d'un objet de la classe et dont le nom commence généralement par set.

7. Implémenter l'ensemble des accesseurs en écriture pour chaque attribut de la classe Voiture.
8. Utiliser vos accesseurs (en lecture et en écriture) pour afficher et modifier les attributs de vos objets mycar1 et mycar2

Partie 4 : Affichage de l'instance d'un objet

On souhaiterait maintenant pouvoir afficher la valeur des attributs d'une instance de l'objet voiture. La fonction à utiliser serait donc la fonction print().

9. Qu'affiche la fonction print() quand on lui passe en argument une instance de la classe voiture ?

La redéfinition de la fonction `__str__(self)` dans la classe voiture permet de personnaliser l'affichage de l'instance de l'objet lorsqu'il est affiché avec la fonction print().

10. Définir la fonction __str__(self) afin d'afficher les attributs de la classe voiture et tester-la en utilisant la fonction print(). Votre affichage doit ressembler à :

```
Voici les caractéristiques de cette voiture:
- Marque : Mercedes
- Modele : Classe A
- Couleur : rouge
- puissance : 7
```

Partie 5 : un attribut de plus

On souhaite maintenant pouvoir ajouter des options à une voiture. Comme une voiture peut avoir plusieurs options, celles-ci seront stockées dans une liste. L'objectif sera de gérer les options d'une voiture à savoir la possibilité d'ajouter, de supprimer et de rechercher une option.

11. Modifier le constructeur de la classe Voiture afin de définir une liste d'options de taille nulle lors de l'instanciation d'une voiture.

12. Modifier la fonction __str__(self) afin d'afficher également les options

13. Ajouter les accesseurs (lecture et écriture) pour l'attribut option.

14. Coder la méthode `ajouter_option(self, opts)` qui permet d'ajouter une option à la voiture.
15. Coder la méthode `supprimer_option(self, opt)` qui supprime de la liste l'option recherchée
16. Coder la méthode `is_option_present(self, opt)` qui cherche dans la liste si une option est déjà présente ou non.
17. Au niveau de votre main.py faite une série de tests qui utilisent les méthodes que vous avez définis précédemment.

Référence :
TP1 de S.Bindel, IUT de Colmar