Institut Supérieur du Numérique

(SUPNUM)

TP6: Programmation Orientée Objet en Python

Exercice 1

1. Créez une classe appelée Personne qui a les attributs suivants: nom, prénom, âge et adresse.
2. Créez un constructeur pour la classe Personne qui prendra les arguments nom, prénom, âge et adresse et initialisera les attributs de la classe.
3. Ajoutez une méthode appelée presentation qui affichera une phrase qui présente la personne en utilisant ses attributs nom, prénom, âge et adresse.
4. Créez deux objets Personne en utilisant votre classe et appelez la méthode presentation pour chaque objet.

Exercice 2

Une classe Voiture est conçue comme indiqué dans le diagramme de classe ci-dessous :

Voiture
-marque :string
-couleur :string
-Prix :int
-puissance :int
+Voiture()
+Voiture(marque)
+getCouleur() :string
+getPrix ():int

+ indique un mode d'accès public ; - indique un mode d'accès privé

1. Ecrire le code source de la classe Voiture
2. Ecrire un programme de test appelé TestVoiture (dans un autre fichier source appelé TestVoiture.py) qui utilise la classe Voiture comme suit :
   - Construire une instance de la classe Voiture appelée v1 dont les attributs sont inutilisés à leurs valeurs par defaut.
   - Construire une instance v2 en invoquant le constructeur, qui définit la marque, le prix et la couleur de la voiture à leurs valeurs passées comme paramètres.
3. Modifier votre constructeur pour construire une instance Voiture avec deux arguments - un entier pour le prix et un String pour la marque.
4. Modifier le programme de test TestVoiture pour construire une instance de voiture en utilisant ce constructeur.
5. Ajouter un getter pour la variable marque afin de récupérer la marque de cette instance.
6. Modifier le programme de test pour tester cette méthode.
7. Dans TestVoiture, pouvez-vous accéder directement à la variable d'instance puissance (par exemple, print(v1.puissance) ; ou attribuer une nouvelle valeur à la puissance (par exemple, v1.puissance=200) ? Faites un essai et expliquez les messages d'erreur.
8. Ajouter un contrôle pour n'accepter que des valeurs entières et positives pour la puissance de la voiture
9. Ajouter deux méthodes pour pouvoir modifier la marque et la couleur d'une instance voiture
10. Ecrire la méthode __str__() qui renvoie toutes les propriétés de la classe voiture séparées par un point-virgule
11. Essayer d'appeler la méthode __str__() dans la classe testVoiture d'une manière implicite.
12. Ecrire la méthode __eq__() : 2 voitures sont égales s'elles ont les mêmes propriétés.