Devoir Surveillé 2

20 octobre 2022

Programmation orientée objet

T NSI

Exercice 1 — 7 points

Une agence immobilière développe un programme pour gérer les biens immobiliers qu'elle propose à la vente.

Dans ce programme, pour modéliser les données de biens immobiliers, on définit une classe Bim avec les attributs suivants :

— nt de type str représente la nature du bien (appartement, maison, bureau, commerce,...) ;

— sf de type float est la surface du bien ;

— pm de type float est le prix moyen par m^2 du bien qui dépend de son emplacement.

La classe Bim possède une méthode estime_prix qui renvoie une estimation du prix du bien. Le code (incomplet) de la classe Bim est donné ci-dessous :

```
1  class Bim:
2      def __init__(self,nature, surface, prix_moyen):
3          ...
4
5      def estime_prix(self):
6          return self.sf * self.pm
```

1) Recopier et compléter le code du constructeur de la classe Bim.

2) On exécute l'instruction suivante : b1 = Bim('maison',70.0, 2000.0)

Que renvoie l'instruction b1.estime_prix() ? Préciser le type de la valeur renvoyée.

3) On souhaite affiner l'estimation du prix d'un bien en prenant en compte sa nature :

— pour un bien dont l'attribut nt est 'maison' la nouvelle estimation du prix est le produit de sa surface par le prix moyen par m^2 multiplié par 1,1 ;

— pour un bien dont l'attribut nt est 'bureau' la nouvelle estimation du prix est le produit de sa surface par le prix moyen par m^2 multiplié par 0,8 ;

— pour les biens d'autres natures, l'estimation du prix ne change pas.

Modifier le code de la méthode estime_prix afin de prendre en compte ce changement de calcul.

4) Écrire le code de la méthode nb_maison(lst) qui prend en argument une liste Python de biens immobiliers de type Bim et qui renvoie le nombre d'objets de nature 'maison' contenus dans la liste lst.

Exercice 2 — 6 points

La classe **Personnage** permet de créer des personnages identifiés par un nom et munis d'une force. Le nom est une chaine de caractères et la force est un nombre entier. Un personnage peut perdre ou gagner des points de force. Il peut mener un combat avec un autre Personnage pour gagner ou perdre des points de force.

On donne le code Python de la classe **Personnage** ci-dessous :

```
class Personnage:
    def __init__(self, n, f):
        self.nom = n
        self.force = f

    def perd_force(self,f):
        self.force = self.force - f

    def gagne_force(self,f):
        self.force = self.force + f

    def combat(self,adversaire):
        if self.force < adversaire.force:
            adversaire.gagne_force(self.force // 2)
            self.perd_force(self.force // 2)
        else:
            adversaire.perd_force(adversaire.force // 2)

    def afficher(self):
        print(self.nom,"a une force de",self.force,"points." )
```

1) **a)** Quels sont les attributs d'un objet créé avec la classe Personnage ?

b) Quelles sont les méthodes de la classe Personnage ?

2) On saisit dans l'interpréteur Python la commande : **gandalf=Personnage("Gandalf" , 200)**.

a) Quel est le résultat de cette commande ?

b) Comment peut-on afficher ce personnage ? Quel est alors l'affichage ?

3) On donne le script suivant :

```
frodon=Personnage("Frodon",100)
gollum=Personnage("Gollum",80)
sam=Personnage("Hobbit joufflu",50)
gollum.combat(frodon)
sam.combat(gollum)

frodon.afficher()
sam.afficher()
gollum.afficher()
```

Quel est l'affichage à l'issu de ce script ?

Exercice 3 — 7 points

On veut créer une classe nommée Ville qui regroupe des informations sur des villes. Pour chacune des villes créées, les informations sont le nom de la ville, le nombre d'habitants et ses coordonnées GPS. Ces informations seront passés en argument dans le constructeur.

Les attributs de la classe Ville sont les suivants :

— L'attribut nom est de type str ;

— L'attribut nbh est de type int ;

— L'attribut gps est un tuple qui contient 2 valeurs de type float associées respectivement à la latitude et la longitude de la ville ;

Le tableau ci-dessous donne des informations sur trois villes normandes qui seront passées en argument lors de la construction de l'objet Ville.

Nom	**Nombre habitants**	**Coordonnées GPS (latitude, longitude)**
Caen	106 000	(49,186 53 ; −0,362 09)
Rouen	110 000	(49,448 79 ; 1,094 60)
Le Havre	172 000	(49,504 88 ; 0,129 64)

On précise que la première coordonnée GPS est la latitude indiquant la position nord-sud et la seconde coordonnée GPS est la longitude indiquant la position est-ouest.

1) Écrire en Python le constructeur de la classe Ville.

2) Quelle est l'instruction Python à écrire pour créer la ville v1 associée à la ville de Caen du tableau.

3) Écrire la méthode **afficher** qui renvoie une chaine de caractères affichant le nom de la ville et le nombre d'habitants de la ville. Par exemple, l'instruction v1.afficher() renvoie la chaine La ville de Caen a 106 000 habitants.

4) Une ville est située plus au nord qu'une autre ville si sa latitude est plus élevée. Par exemple, la ville du Havre a une latitude supérieure à la ville de Caen donc Le Havre est plus au nord que Caen.

La méthode **plus_au_nord_que** de la classe **Ville** prend en argument un objet ville et renvoie un booléen. La valeur renvoyée par la méthode est True si la ville à laquelle s'applique la méthode est plus au nord que la ville passée en argument.

a) Écrire en Python la méthode **plus_au_nord_que**.

b) Quelles instructions Python permettent de savoir si la ville de Rouen est plus au nord que Caen ?