
PROGRAMMATION OBJET

Exercice 0

1. Définir une classe **Personne** avec pour attributs le nom et le prénom de la personne.
2. Définir une classe **Amis** avec pour attributs deux personnes et une méthode **ami** qui prend une personne **p** en paramètre et retourne l'autre personne si **p** est l'un des deux amis, ou **None** sinon.
3. Définir une classe **Groupe** avec deux attributs, une liste personnes et une liste d'amis, et deux méthodes, l'une pour ajouter une personne et l'autre pour déclarer deux personnes amies. On s'assurera que les deux personnes amies font bien parties du groupe.
4. Ajouter une méthode qui permet de lister les amis d'une personne.

Exercice 1 On considère le programme suivant :

```
class Chat:
    def bonjour(self):
        print('miaou')

class Chien:
    def bonjour(self):
        print('ouaf')

c = Chat()
c.bonjour()
c = Chien()
c.bonjour()
```

1. Qu'affiche le programme suivant ? Comment expliquer ce résultat ?
2. Donner un autre exemple qui exploite cette propriété.

Exercice 2

1. Écrire les définitions complètes des classes **Point** et **Segment** données ci-dessous puis et tester chacune de leurs méthodes.

```
class Point:
    def __init__(self, coord_x = 0, coord_y = 0):
        self.x = coord_x
        self.y = coord_y

    def translater(self, v):
        self.x += v.x
        self.y += v.y

    def distance(self, p2):
        return math.sqrt((p2.x - self.x)**2 + (p2.y - self.y)**2)

class Segment:
    def __init__(self, point_1 , point_2):
        self.p1 = point_1
        self.p2 = point_2

    def translater(self, p):
        self.p1.translater(p)
        self.p2.translater(p)

    def milieu(self):
        return Point((self.p1.x + self.p2.x)/2, (self.p1.y + self.p2.y)/2)
```

La classe **Turtle** de la bibliothèque Python **turtle** contient la méthode **goto(x, y)** qui déplace la tortue à la position donnée et les méthodes **penup()** et **pendown()** pour lever et baisser le crayon.

2. Utiliser ces méthodes pour implémenter une classe **Dessin** avec deux méthodes, **croix** pour tracer une croix à une position donnée, et **trait** pour tracer un trait entre deux points.
3. Ajouter une méthode **dessiner** aux classes **Point** et **Segment** pour afficher par une croix et le segment par un trait.
4. Écrire un programme qui crée des points et des segments à des positions aléatoires (en utilisant la fonction **randint(min,max)** de la bibliothèque **random**) et les affiche.

Exercice 3 On utilise les classes **Point**, **Segment** et **Dessin** de l'exercice précédent. On veut créer une classe **Polygone** qui représente un polygone défini par un ensemble de sommets.

1. Définir et tester une implémentation de la classe **Polygone** contenant à la fois la séquence de points de ses sommets *et* la séquence de segments de ses côtés.
2. Dessiner le diagramme des objets représentant un polygone à trois côtés (un triangle). Que se passe-t-il si l'on modifie les coordonnées d'un point ?
3. Implémenter une méthode pour traduire tout le polygone et une méthode pour le dessiner. Quel est l'intérêt d'avoir à la fois une liste de points et une liste de segments ?

Exercice 4 Il est parfois nécessaire de savoir si un objet appartient à une certaine classe. La fonction Python **isinstance (objet, classe)** retourne **True** si objet est une instance de **classe**, **False** sinon.

1. Utiliser cette fonction pour définir des préconditions sur les méthodes des classes **Point** et **Segment** dont les paramètres sont des objets.
2. Dans l'exercice 1, on a défini les classes **Chien** et **Chat** et la méthode **bonjour** qui affiche « ouaf » ou « miaou » selon la classe de l'objet. On veut modifier cette méthode pour qu'elle prenne en paramètre l'animal à qui on dit bonjour :
 - un chat dit bonjour aux chats et ignore les autres animaux ;
 - un chien dit bonjour aux autres chiens et fait peur aux chats en aboyant « OUAF OUAF ».

Implémenter les classes **Chien** et **Chat** en utilisant la fonction **isinstance** pour obtenir ce comportement.