TP 4

Exercice 1
On déinit dans cet exercice les classes *Segment* et *Point* pour représenter des segments de droites dans un plan définis par deux points du plan.

1. Érire la classe *Point*, avec son constructeurs et ajouter les accesseurs.
2. Érire la classe *Segment*. Chaque segment contient deux points nommés start et stop.
3. Érire dans la classe *Segment* une méthode $affficher$ permettant d'afficher les deux points d'un segment. On utilisera la méthode $affficher$ de *Point*.
4. Écrire la méthode qui renvoie le milieu d'un segment.
5. Écrire la méthode qui renvoie la longueur d'un segment.
6. Érire dans la classe Segment une méthode *translate* permettant de déplacer un segment dans le plan. On utilisera une méthode *translate* de *Point*.
7. Érire une méthode qui teste si deux segments ont une extrémité (un point) en commun. Dans quelle classe faut-il la mettre ? Quel est le type de retour de cette méthode ?
8. On peut également imaginer la classe *Triangle*. Codez cette classe, qui permettra notemment de déterminer si un triange est rectangle, isocèle ou équilatéral.
9. On pourra aussi calculer le périmètre du triangle.

Exercice 2
Notre but est de travailler avec les matrices de nombres entiers. Une matrice peut donc être vue comme un objet qui a comme paramètres un nombre de lignes, un nombre de colonnes et un tableau à une ou deux dimensions.
Ajouter à la classe Matrice les méthodes suivantes :

- *lit_matrice* : demande à l'utilisateur les dimensions d'une matrice ainsi que ses coéfficients, crée la matrice correspondante et la renvoie.
- $affiche\_matrice$: affiche une matrice ligne par ligne.
- *addition* : retourne la matrice résultat de l'addition de la matrice courante avec la matrice passée en argument.
- *produit* : retourne la matrice résultat du produit de la matrice courante avec la matrice passée en argument.
- *transpose* : retourne la transposée de la matrice courante.

Exercice 3
Nous souhaitons réaliser un programme qui simule l'embauche, le paiement et la débauche des employés d'une entreprise. Voici le code de la classe *Employe* à compléter :

```
class Employe:
  nb_Employes=0
  def __init__(self, nom, salaire):
    #a completer sans oublier d'incrementer le compteur
#methodes a ajouter comme par exemple les accesseurs, afficher...
```

Afin que ce code puisse fonctionner :

```
e1 = Employe("toto",1000)
e1.afficher()
print(e1.get_alaire())
nom = input("Entrez le nom de l'employe : ")
s = float(input("Entrez le salaire de l'employe : "))
e2 = Employe(nom,s)
```

```
e2.afficher()
print(e2.get_alaire())
printl("le nombre d'employes crees est : " + str(Employe.getNbEmployes()))
```

Un ouvrier est un employé travaillant dans un atelier portant un numéro. Coder la classe *Ouvrier* qui dérive de la classe *Employe*, et contient un membre de classe permettant de stocker le numéro de l'atelier dans lequel travaille l'ouvrier. Coder la classe Ouvrier et compléter le main précédemment écrit pour créer un Ouvrier ayant 1000 euros pour salaire et travaillant dans l'atelier 2.

Un technicien est "une sorte" d'employé travaillant dans un service portant un nom. Définir la classe Technicien dérivant de la classe Employe, qui contient un membre de classe permettant de stocker le nom du service auquel le technicien appartient ; un technicien percevra une prime annuelle d'un montant de 1000 euros.

Un cadre est un employé travaillant sur plusieurs projets portant chacun un nom. Définir la classe Cadre dérivant de la classe Employe, et qui contient des membres de classe permettant de stocker le nom de tous les projets auquel le cadre participe ; un cadre percevra une prime d'un montant de 3000 euros. De plus, vous devez disposer d'une méthode ajouterUnProjet qui ajoute le nom d'un projet à la liste des projets auquel le cadre participe, ainsi que d'une méthode afficherLesProjets qui affiche tous les projets auquel un cadre participe.

Un chef est un cadre dirigeant plusieurs employés. Définir la classe Chef qui contient des membres de classe permettant de stocker les employés sous la responsabilité de ce chef. De plus, vous devez disposer :

- d'une méthode *get_prime* renvoyant une prime dont le montant est fonction du nombre d'employés sous la responsabilité de ce chef ;
- d'une méthode *ajouter_employe* qui ajoute un employé à la liste des employés sous la responsabilité de ce chef ;
- d'une méthode $affficher\_les\_employes$ qui affiche tous les employés sous la responsabilité de ce chef.