EXAMEN DE FIN DE SEMESTRE1

Matière : *INFORMATIQUE*
Classes : *2ème Année* ***Durée :*** *2 h*
Préparation : MP, PC et PT

Exercice 1 (6 points)

On désire dans cet exercice trier une liste à l'aide des piles à capacités non bornées. Pour cela, on demande d'écrire une classe **Pile** telle que :

p=Pile () devra créer et retourner une pile **p** vide.
p.empiler (x) empilera **x** au sommet de la pile **p**.
p.depiler () dépilera et retournera l'élément situé au sommet de la pile **p**.
p.est_vide () renverra un booléen disant si la pile **p** est vide.
p.sommet() renverra le sommet de la pile **p**.
p.inserer (x) insérera la valeur **x** dans la pile **p** supposée triée dans l'ordre croissant de sorte que la pile restera toujours triée dans le même ordre.
p.trier () triera une pile **p** (utiliser les méthodes **empiler (x), depiler ()** et **inserer(x)**).

Dans le programme principal, on demande de créer **p**, empiler dans **p** les valeurs 4, 7, 17, 10,7 et 2 puis trier **p**.

Exercice 2 (4 points)

Un point du plan est représenté par ses coordonnées. Ces derniers seront les attributs d'une classe **Point**. Un objet ***p*** de cette classe sera créé en indiquant ses coordonnées.

1. Définir la classe **Point** qui représente un point ***p*** telles que les commandes suivantes:

```
>>> p=Point(3,5)
>>>print(p)
```

donnent le résultat d'affichage suivant :

```
Le point a pour abscisse x= 3 et ordonné y= 5
```

2. Créer une classe **Droite**, pour représenter des droites du plan, non verticales. Ces droites sont assimilables aux équations de type $y=ax+b$.

Les attributs représentant un objet ***d*** de la classe **Droite** seront donc a et b.

Ajouter dans cette classe :

- une méthode **appartient(*p*)** qui permet de tester si un point ***p*** se trouve sur la droite ***d*** ou non.
- une méthode **position(*p*)** qui permet d'afficher sous forme d'un message (appartient, au-dessus ou au-dessous), la position d'un point ***p*** par rapport à la droite ***d***.

Problème (10 points)

On désire créer un module python permettant le contrôle des mouvements d'un robot.

Le script à réaliser sera composé de deux parties :

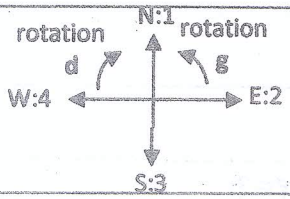
Partie A

Dans cette partie on définit une classe **CRobot** qui crée des robots, cette classe possède :

➢ Les caractéristiques suivantes :

Nom d'attribut	**Description**
type	Chaîne de caractères.
sn	*Entier qui désigne le numéro de série.*
orientation	Un entier qui désigne l'orientation du robot : 1 : NORD, 2 : EST, 3 : SUD, 4 : WEST
etat	*Booléen* qui désigne l'état du robot (initialement False)

➢ Les méthodes suivantes :

Nom de méthode	**Description**
Constructeur	lorsque l'on instanciera un nouvel objet, **CRobot()**, on pourra choisir le type et le Numéro de série, mais pas son Orientation (par défaut orientation=1) ni son État (par défaut etat=False).
getType ()	retourne le type de robot
getSN ()	retourne le numéro de série du robot
getOrientation ()	retourne l'orientation du robot
getStatus ()	retourne l'état du robot
setOrientation (nlle_orientation)	définit la nouvelle orientation du robot
setEtat(nl_etat)	définit le nouvel état du robot
tourner(sens) 	tourne le robot d'un 1/4 de tour, par défaut vers la gauche. La valeur **sens** ∈ {-1,1}, avec : - Si **sens=1** pour 1/4 de tour à gauche, - Si **sens=-1** pour 1/4 de tour à droite.
afficher()	affiche les informations du robot : l'état, l'orientation, le numéro de série et le type du robot.

1. Définir la classe **CRobot** contenant toutes les méthodes précitées.

2. Dans le programme principal, instanciez un objet **robot_omron** de type (type=LD60) et de numéro de série (sn=3703000003) de la classe **CRobot** et afficher ses caractéristiques.

Partie B

Dans cette deuxième partie on désire commander un robot mobile humanoïde (type= Asimo) et de numéro de série (sn=3001). Pour cela on définit une classe CRobotMobile qui :

- hérite de **CRobot** ;
- se caractérise, en plus, par les attributs entiers abscisse (x) et ordonné (y) qui définissent la position du robot mobile(par défaut x=0 et y=0) ;
- possède une méthode **avancer(nb_pas)** qui permet d'avancer le robot selon son orientation :

  si l'orientation du Robot est vers l'Est, l'abscisse x augmente de nb_pas
  si l'orientation du Robot est vers le West, l'abscisse x diminue de nb_pas
  si l'orientation du Robot est vers le Nord, l'ordonnée y augmente de nb_pas
  si l'orientation du Robot est vers le Sud, l'ordonnée y diminue de nb_pas

- possède une méthode **afficher_position()** qui affiche la position du robot mobile (x et y).

3. Définir la classe **CRobotMobile**. Avec ses méthodes : **constructeur**, **avancer(nb_pas)** et **afficher_position()**.

N.B. Pour appeler une méthode de la classe mère (CRobot) on écrit : **CRobot.nom_méthode(self)** ou **super().nom_méthode()**

4. Redéfinir la méthode **afficher()** tout en utilisant celle de la classe mère et la méthode **afficher_position()**.

5. Dans le programme principal on demande de:

5.1. Définir un dictionnaire **direction** dont les clés sont : NORD, EST, SUD, WEST et leurs valeurs respectives sont : 1, 2, 3 et 4.

5.2. Créer et tester un objet **robot** de la classe **CRobotMobile** selon les étapes suivantes:

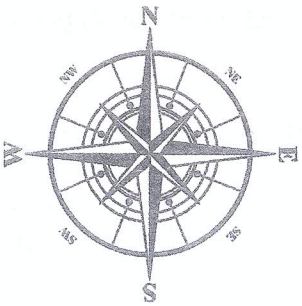
i. Orienter le robot vers l'Est

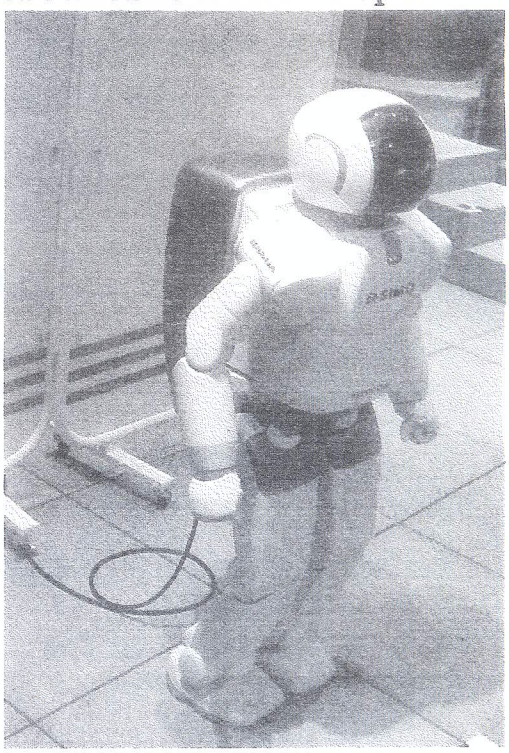
ii. Avancer de 4 pas

iii. Avancer de 6 pas vers le Nord

iv. Avancer de 14 pas vers l'Est

v. Reculer de 8 pas vers le Sud





Bonne chance