Exercice 2 : *Ecriture d'une classe Cercle*

1. *Définir la classe Cercle, dont on donnera le centre par ses coordonnées et le rayon.*
2. *Écrire les accesseurs aux deux attributs.*
3. *Écrire les mutateurs aux deux attributs.*
4. *Écrire la méthode surface qui renvoie la valeur de la surface d'un cercle.*
5. *Écrire la méthode est_sur_le_cercle qui vérifie l'appartenance d'un point sur un cercle.*

Exercice 3 : *Ecriture d' une classe Fraction pour représenter un nombre rationnel.*

Cette classe possède deux attributs, numérateur et denominateur , qui sont entiers et désignent respectivement le numérateur et le dénominateur. On demande que le dénominateur soit un entier strictement positif.

1. *Ecrire le constructeur de cette classe. Le constructeur doit lever une ValueError si le dénominateur est négatif ou nul.* ***Explication :*** *une exception (par exemple ValueError) peut être levée lorsqu'un paramètre inadapté est donné à une fonction, voici la syntaxe : if condition :*

   raise ValueError("Texte expliquant l'erreur")
2. *Ajouter une méthode __str__ qui renvoie une chaîne de caractères de la forme "12/35", ou simplement "12" lorsque le dénominateur vaut 1.*
3. *Ajouter des méthodes __eq__ et __lt__ qui reçoivent une deuxième fraction en argument et renvoie True si la première fraction est égale (resp. strictement inférieure) à la seconde.*
4. *Ajouter des méthodes __add__ et __mul__ qui recoivent une deuxième fraction en argument et renvoie la somme (resp le produit) de ces deux fractions.*
5. *Tester et documenter ces méthodes.*

Exercice 4 : *Soient 2 classes Pièce et Appartement dont on a spécifié les méthodes*

1. *Écrire les constructeurs des deux classes.*
2. *Finaliser la classe Piece en écrivant les accesseurs et mutateurs.*
3. *Finaliser la classe Appartement.*
   (a) *Écrire la méthode ajouter qui permet d'ajouter une pièce à l'appartement.*
   (b) *Écrire la méthode nbPièces qui permet de retourner le nombre de pièces présentes dans l'appartement.*
   (c) *Écrire la méthode getSurfaceTotale, qui renvoie la surface totale de l'appartement.*
   (d) *Écrire la méthode getListePieces, qui renvoie la liste des pièces de l'appartement.*

```
class Piece:
    # nom est une string et surface est un float
    def __init__(self,nom,surface):
        """ chaque objet a pour attributs le nom de la pièce(string)
        et la surface de celle ci(float) en m2. On doit rentrer
        le couple nom de la pièce et la surface  pour chaque pièce."""

        # Accesseurs: retournent les attributs d'un objet de cette classe

    def getNom(self):
        ...
    def getSurface(self):
        ...
        # Mutateur: modifient les attributs, ici la surface d'une pièce
        # déjà renseignée
    def setSurface(self,s): # s est un float
        ...

class Appartement:
    # nom est une string
    def __init__(self,nom):
        """nomme l'appartement et une liste de pièces vide à remplir"""
        ...
    def ajouter(self,piece):
        """ajoute une pièce (instance(=objet) de la classe Piece)"""
        ...
    def nbPieces(self):
        """retourne le nombre de pièces de l'appartement"""
        ...
    def getSurfaceTotale(self):
        """retourne la surface totale de l'appartement (un float)"""
        ...
    def getListePieces(self):
        """retourne la liste des pièces"""
```

Le test sera le suivant :

```
# Dans l'éditeur PYTHON
a=Appartement('appt25')
p1=Piece("chambre", 11.1)
p2=Piece("sdbToilettes", 7)
p3=Piece("cuisine", 7)
p4=Piece("salon", 21.3)
print(p4.getNom(),p4.getSurface())
p1.setSurface(12.6)
a.ajouter(p1)
a.ajouter(p2)
a.ajouter(p3)
a.ajouter(p4)
print(a.getNom(),a.getListePieces())
print('nb pieces =', a.nbPieces(),', Surface totale =',a.SurfaceTotale())
```

et devra retourner

```
salon 21.3
appt25 [('chambre', 12.6), ('sdbToilettes', 7), ('cuisine', 7),
('salon', 21.3)]
nb pieces = 4 , Surface totale = 47.9
```