La Programmation Orientée Objet
Exercices (corrigé)

1. Rectangle

Réaliser l'exercice sur papier.

✍ Écrire en Python une classe nommée **`Rectangle`**, admettant les attributs **`longueur`**, **`largeur`**.

✍ Ajouter les méthodes :

- **`get_longueur`** qui renvoie la longueur du rectangle.
- **`get_largeur`** qui renvoie la largeur du rectangle
- **`perimetre`** qui renvoie le périmètre du rectangle.
- **`aire`** qui renvoie l'aire du rectangle.
- **`homothetie`** qui réalise une homothétie k des dimensions (k × longueur, k × largeur).

```
class Rectangle:

    def __init__(self, longueur, largeur):
        self.longueur = longueur
        self.largeur = largeur

    def get_longueur(self):
        return self.longueur

    def get_largeur(self):
        return self.largeur

    def perimetre(self):
        return 2*(self.longueur + self.largeur)

    def aire(self):
        return self.longueur * self.largeur

    def homothetie(self, k):
        self.longueur = k * self.longueur
        self.largeur = k * self.largeur
```

✍ Ecrire le code Python qui instancie un objet rectangle, affiche sa longueur et sa largeur, puis affiche son périmètre puis son aire. Qui demande la valeur de l'homothétie, qui réalise le calcul et qui affiche les nouvelles valeurs du rectangle.

Le résultat obtenu dans la console :

```
La longueur du rectangle est 6 et sa largeur est 4
Son périmètre est de 20
Son aire est de 24
Quelle est la valeur de l'homothétie ? 3
La longueur du rectangle est 18 et sa largeur est 12
```

```
rect = Rectangle(6, 4)
print(f"La longueur du rectangle est {rect.get_longueur()} et sa largeur est {rect.get_largeur()}")
print(f"Son périmètre est de {rect.perimetre()}")
print(f"Son aire est de {rect.aire()}")
k = int(input("Quelle est la valeur de l'homothétie ? "))
rect.homothetie(k)
print(f"La longueur du rectangle est {rect.get_longueur()} et sa largeur est {rect.get_largeur()}")
```

2. Pièce et appartement

On suppose écrites les classes Piece et Appartement, dont on donne les en-têtes de méthodes :

```
# fichier piece.py

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
        . . .

    # retourne les attributs d'un objet de cette classe
    def get_surface(self):
        . . .

    # retourne les attributs d'un objet de cette classe
    def get_nom(self):
        . . .

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



# fichier appartement.py

from piece import*

class Appartement:
    # nom est une string
    def __init__(self, nom):
        #nomme l'appartement et une liste de pièces vide à remplir
        . . .
    # pour avoir le nom de l'appartement
    def get_nom(self):
        . . .

    # pour ajouter une pièce de classe Piece
    def ajouter(self, piece):
        . . .

    # pour avoir le nombre de pièces de l'appartement
    def nb_pieces(self):
        . . .

    # retourne la surface totale de l'appartement (un float)
    def get_surface_totale(self):
        . . .

    # retourne la liste des pièces avec les surfaces
    def get_liste_pieces(self): # sous forme d'une liste de tuples
        . . .
```

Les classes répondent au test ci-dessous réalisé dans la console :

```
>>> a = Appartement('appt25')
>>> p1 = Piece('chambre', 11.1)
>>> p2 = Piece('sdb_toilettes', 7)
>>> p3 = Piece('cuisine', 7)
>>> p4 = Piece('salon', 21.3)
>>> p4.get_nom()
'salon'
>>> p4.get_surface()
21.3
>>> p1.set_surface(12.6)
>>> a.ajouter(p1)
>>> a.ajouter(p2)
>>> a.ajouter(p3)
>>> a.ajouter(p4)
>>> a.get_liste_pieces()
[('chambre', 12.6), ('sdb_toilettes', 7), ('cuisine', 7), ('salon', 21.3)]
>>> a.nb_pieces()
4
>>> a.get_surface_totale()
47.900000000000006
```

✍ Écrire (sur feuille) un programme principal utilisant ces deux classes qui va :

- créer une pièce « chambre1 » , de surface 20 m^2et une pièce « chambre2 » », de surface 15 m^2 ;
- créer une pièce « séjour » », de surface 25 m^2et une pièce « sdb » », de surface 10 m^2 ;
- créer une pièce « cuisine » », de surface 12 m^2 ;
- créer un appartement « appart205 » qui contiendra toutes les pièces créées ;
- afficher la surface totale de l'appartement créé ;
- afficher la liste des pièces et surfaces de l'appartement créé.

```
from appartement import*

ch1 = Piece("chambre1", 20)
ch2 = Piece("chambre2", 15)
sej = Piece("séjour", 10)
cuis = Piece("cuisine", 12)
appart = Appartement("appart205")
appart.ajouter(ch1)
appart.ajouter(ch2)
appart.ajouter(sej)
appart.ajouter(cuis)
print(appart.get_surface_totale())
print(appart.get_liste_pieces())
```

✍ Compléter toutes les méthodes des classes Piece et Appartement.

```
# fichier piece.py

class Piece:
    def __init__(self, nom, surface):
        self.nom = nom
        self.surface = surface
```

```
    def get_surface(self):
        return self.surface

    def get_nom(self):
        return self.nom

    def set_surface(self, s):
        self.surface = s


# fichier appartement.py

from piece import*

class Appartement:
    def __init__(self, nom):
        self.liste_de_pieces = []
        self.nom = nom

    def get_nom(self):
        return self.nom

    def ajouter(self, piece):
        self.liste_de_pieces.append(piece)

    def nb_pieces(self):
        return len(self.liste_de_pieces)

    def get_surface_totale(self):
        total = 0
        for piece in self.liste_de_pieces:
            surf = piece.get_surface()
            total = total + surf
        return total

    def get_liste_pieces(self):
        l = []
        for piece in self.liste_de_pieces:
            surf = piece.get_surface()
            nom = piece.get_nom()
            l.append((nom, surf))
        return l
```

✍ Créer un tableau qui classe les méthodes de ces deux classes selon leur type : constructeur, accesseur, mutateur ou autre.

Constructeurs : les deux init.
Accesseurs : Piece.get_surface(), Piece.get_nom(), Appartement.get_nom(), Appartement.get_surface_totale(), Appartement.get_liste_pieces().
Mutateurs : Piece .set_surface(), Appartement.ajouter().
Autre : Appartement.nb_pieces() (nb_pieces n'est pas un attribut de la classe Appartement, cette valeur n'est stockée nulle part, elle est juste calculée par la méthode len de self.liste_de_pieces).

Sources : http://www.math93.com/images/pdf/NSI/terminale/NSI_Classes.pdf