Le sujet comporte un total de 3 exercices indépendants
qui peuvent être traités dans l'ordre de votre choix.

Exercice N°1 – Une liste

Soit *n* un entier naturel non nul.

Ecrire une fonction `ListCreate` qui reçoit *n* en argument et renvoie la liste :

```
L = [0,1,0,1,2,0,1,2,3,…,0,1,2,…,n]
```

Ainsi, pour $n = 5$, la fonction `ListCreate` renverra la liste :

```
L = [0,1,0,1,2,0,1,2,3,0,1,2,3,4,0,1,2,3,4,5]
```

Exercice N°2 – Manipulation de liste

Soit une liste pouvant contenir exclusivement : des entiers, des flottants ou des chaînes de caractères. Par exemple :

```
L = [1,6.5,-34,'toto','bonjour',3.141592,'This is the end']
```

Ecrire une fonction `ListContent` qui reçoit en argument une liste telle que définie précédemment et :

- affiche un message du type :

```
Votre liste contient :
3 chaînes de caractères
2 entiers
2 flottants
```

- renvoie deux listes `LNUM` et `LSTRING`, sous-listes de `L`, contenant respectivement les éléments numériques et les chaînes de caractères de `L` (sans modification de l'ordre).
  Avec la liste ci-dessus, on obtiendra donc :

```
LNUM = [1,6.5,-34,3.141592]
LSTRING = ['toto','bonjour','This is the end']
```

Remarque : la liste `L`, passée en argument de la fonction, ne devra pas être modifiée.

Exercice N°3 – Un peu de géométrie…

Dans le plan rapporté à un repère orthonormé direct, on souhaite manipuler via des programmes Python des points et des vecteurs. Pour ce faire, on définit deux classes `point` et `vecteur` initialement de la façon suivante :

```
class point:
     def __init__(self,x,y,s) :
          self.name = s #de type str
          self.abs = x  #de type float
          self.ord = y  #de type float

class vecteur:
     def __init__(self,x,y,s) :
          self.name = s #de type str
          self.abs = x  #de type float
          self.ord = y  #de type float
```

1. Ecrire, pour la classe `point` :
   a. La méthode `__repr__`.
      Par exemple, si on a créé la variable `PA` comme suit : `PA=point(2,5,'A')`, la commande `print(PA)` devra renvoyer : `A(2,3)`.
   b. La méthode `milieu`.
      Elle recevra en argument `self` et `P` (de type `point`) et devra renvoyer un nouvel objet `point` correspondant au milieu du segment d'extrémités les points correspondant aux objets `self` et `P`.
   c. La méthode `translation`.
      Elle recevra en argument `self` et `v` (de type `vecteur`) et devra renvoyer un nouvel objet de type `point` correspondant à l'image du point correspondant à l'objet `self` par la translation dont le vecteur correspond à l'objet `v`.
   d. La méthode `rotation`.
      Elle recevra en argument `self`, `C` (de type `point`) et `alpha` (de type `float`) et devra renvoyer un nouvel objet de type `point` correspondant à l'image du point correspondant à l'objet `self` par la rotation de centre le point correspondant à l'objet `C` et d'angle le réel correspondant au flottant `alpha`.
2. Ecrire, pour la classe `vecteur` :
   a. La méthode `__repr__`.
      Par exemple, si on a créé la variable `Vect_u` comme suit : `Vect_u=vecteur(-4,7,'u')`, la commande `print(Vect_u)` devra renvoyer : `u(-4,7)`.
   b. La méthode `norme`.
3. Surcharger, dans la classe `vecteur` :
   a. L'opérateur `__neg__`.
   b. L'opérateur `__add__`.
   c. L'opérateur `__radd__`.