SÉRIE N° 22 :
PROGRAMMATION ORIENTÉE OBJET

Application

Exercice 1 :

1. Définissez la classe point dont les caractéristiques sont :
- les coordonnés cartésiennes x et y (dont x est l'abscisse et y est l'ordonnée)
- Les méthodes sont :
  - ✓ __init__(self, absc, ord) qui permet de créer une instance de classe point
  - ✓ Get_abscisse(self) qui retourne la valeur de x
  - ✓ Get_ordonne(self) qui retourne la valeur de y
  - ✓ Distance(self, autre) qui calcule la distance entre le point actuel et le point autre et retourne le résultat

2. Un triangle est défini par trois points différents :
Définissez la fonction est_isocèle (A, B, C) qui prend en paramètre 3 points et qui retourne True si le triangle ABC est isocèle ou False si non
Notion : on dit que un triangle est un isocèle ssi il possédait exactement deux côtés égaux

3. Définissez la fonction est_equilateral (A, B, C) qui prend en paramètre 3 points et qui retourne True si le triangle ABC est équilatéral ou False si non
Notion : on dit que un triangle ABC est équilatéral ssi ces côtés ont la même longueur

4. Définissez la fonction est_rectangle (A, B, C) qui prend en paramètre 3 points et qui retourne True si le triange ABC est triangle rectangle ou False si non
Notion : Un triangle rectangle est un triangle ayant un angle droit, c'est-à-dire de mesure 90°

Exercice 2 :

Un nombre complexe *z* se présente en général en coordonnées cartésiennes, comme une somme ***Re + img i***, où *Re* et *Img* sont des nombres réels quelconques et i (l'unité imaginaire) est un nombre particulier tel que $i^2 = -1$. Le réel *Re* est appelé partie réelle, le réel *Img* est sa partie imaginaire

1. Définissez la classe Complexe qui permet de gérer les nombres complexes
2. Créez la méthode **__init__(self, a, b)** qui permet de créer et d'initialiser un objet de la classe Complexe
3. Définissez les méthodes suivantes :
   - **set_Re(self, r)** qui permet de changer la partie réelle par le nombre r
   - **set_Img(self, im)** qui permet de changer la partie imaginaire par le nombre im
   - **get_Re(self)** qui retourne la partie réelle
   - **get_Img(self)** qui retourne la partie imaginaire
   - **afficher(self)** qui affiche un nombre complexe sous forme (Re (signe de Img) |Img| i)
     exemple :
     - si Re =3 et Img = 2 la méthode affiche : 3 + 2i
     - si Re =1 et Img =-5 la méthode affiche : 1 - 5i
     - si Re =4 et Img = 1 la méthode affiche : 4 + i
     - si Re =2 et Img =-1 la méthode affiche : 2 – i
   - **conjuguer(self)** qui calcule le conjugué d'objet actuel ensuite le retourne sous forme d'une instance de la classe Complexe
   - **module(self)** qui calcule et retourne le module d'objet actuel
   - **argument(self)** qui calcule et retourne le module d'objet actuel en degré
   - **triangulaire(self)** qui affiche l'objet actuel sous forme [module, argument]

4. définissez les méthodes magiques suivantes :

- **a. __add__(self, z)** qui représente l'opérateur + et qui calcule la somme d'objet actuel self et l'objet z ensuite retourne le résultat sous forme d'une instance de la classe Complexe
  exemple :
  ```
  >>> a=Complexe(1,2); b=Complexe(3,5)
  >>> c=a+b
  >>>c.afficher()
  4 + 7i
  ```
- **__sub__(self, z)** qui représente l'opérateur - et qui calcule la soustraction d'objet actuel self et l'objet z ensuite retourne le résultat sous forme d'une instance de la classe Complexe
  exemple :
  ```
  >>> a=Complexe(1,2); b=Complexe(3,5)
  >>> c=a-b
  >>>c.afficher()
  -2 - 3i
  ```
- **__mul__(self, z)** qui représente l'opérateur * et qui calcule la multiplication d'objet actuel self et l'objet z ensuite retourne le résultat sous forme d'une instance de la classe Complexe
  exemple :
  ```
  >>> a=Complexe(1,2); b=Complexe(3,5)
  >>> c=a*b
  >>>c.afficher()
  -7 + 11i
  ```
- **__truediv__(self, z)** qui représente l'opérateur / et qui calcule la division d'objet actuel self par l'objet z ensuite retourne le résultat sous forme d'une instance de la classe Complexe
  exemple :
  ```
  >>> a=Complexe(-7, 11); b=Complexe(3,5)
  >>> c=a/b
  >>>c.afficher()
  1 + 2i
  ```