TD1 – Concepts de la programmation orientée objet (POO)

Objectifs : Rappels python, notions d'objet, classe, attribut, et méthode

1. Rappels python

1) Complétez le code python qui vous est donné ci-dessous pour réaliser l'opération simple de calcul de la distance entre 2 points du plan décrits par leur abscisse et leur ordonnée.

```
import math
def calculer_distance( ?????? ):
    return (math.sqrt((xa-xb)**2+(ya-yb)**2))
#programme principal
x1=2
y1=-4
x2=12
y2=10
dist=calculer_distance( ??????? )
print( ???? )
```

2) Répondez aux questions ci-dessous :
   a. A quoi sert la ligne « import math »
   b. Pourquoi le mot-clé « def » est-il utilisé ?
   c. Combien de variables sont déclarées dans ce code ?
   d. A quoi sert la variable dist ?
   e. Est il possible de ne pas utiliser cette varaible ? si oui comment ?
   f. Qu'est ce qu'on appelle la définition d'une fonction, et un appel d'une fonction ?
   g. Quelle valeur va s'afficher à l'exécution du programme?
3) Écrire la fonction `calculer_distance_origine(x,y)` qui calculer la distance à l'origine (0,0) du point de coordonnées x,y

2. Passage à l'objet

Description de l'exercice : Le code python ci-dessous est la transformation en python orienté objet de l'exercice précédent.

```
import math
class Point:
    def __init__(self,abs,ord):
        self.x=abs
        self.y=ord
    def calculer_distance(self,p):
        d = math.sqrt((self.x-p.x)**2+(self.y-p.y)**2)
        return d
#prog principal
p1=Point(2,-4)
p2=Point(12,10)
dist=p1.calculer_distance(p2)
print(dist)
```

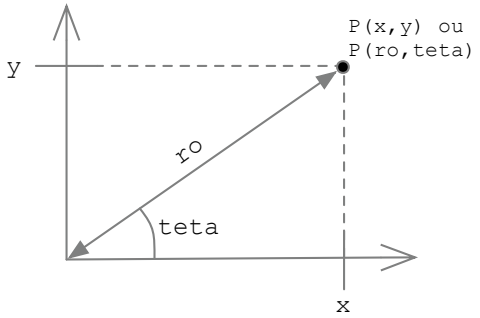
1) Répondez aux questions ci-dessous :
   a. Qu'est-ce qu'on appelle en OO le constructeur ?
   b. Quand la fonction `__init__` est-elle appelée ?
   c. Combien y a t'il d'attributs ? et de méthodes ? les nommer.
   d. Pourquoi faut-il toujours le mot-clé `self` devant les attributs ?
   e. Pourquoi faut-il toujours le mot-clé `self` en premier paramètre des méthodes ?

f. Quel est le type du paramètre passé en entrée lors de l'appel de `calculer_distance` ?
g. Pourquoi doit on écrire `p1.calculer_distance(p2)` et pas `calculer_distance(p1,p2)` ?

2) Rajoutez la méthode `reset` à la classe `Point`, remettant les attributs `x` et `y` à 0.
3) Rajoutez la méthode `distance_origine` à la classe Point permettant de retourner la distance du point à l'origine.
4) Modifiez le programme principal pour afficher la distance à l'origine de p1 et p2.

3. Développer une nouvelle classe Point

5) Description de l'exercice : Dans cet exercice vous devrez coder en python une classe `Point`, permettant d'utiliser les coordonnées cartésiennes ou polaire d'un point. Les calculs pour passer des coordonnées cartésiennes aux polaires sont rappelés ci-dessous.



$ro = \sqrt{x^2+y^2}$

$teta = \arccos(x/ro)$

Questions :

1) Écrire une classe `Point` possédant 4 attributs `x`,`y`, `ro` et `teta`, qui sont initialisés à zéro lors de la création d'un Point, et une méthode `calculer_distance` permettant de calculer la distance entre le point et un autre point passé en paramètre.
6) Écrire une méthode `calculer_ro_teta`() de la classe `Point` qui permet de mettre à jour la valeur de `ro` et `teta` du point, en fonction de son abscisse et de son ordonnée.
2) Écrire une méthode `setxy`(x,y) qui permet d'initialiser toutes les coordonnées d'un point dont on fait passer l'abscisse et l'ordonnée en paramètres d'entrée. Cette méthode fera appel à `calculer_ro_teta()`.
3) Écrire une méthode `calculer_xy()`, et `set_ro_teta()` permettant d'initialiser les coordonnées cartésiennes à partir des coordonnées polaires d'un point.
4) Écrire un programme principal qui crée 2 points, initialise l'un avec des coordonnées polaires, et l'autre avec des coordonnées cartésiennes, puis calcule et affiche la distance entre les 2.
5) Écrire une méthode de la classe `Point` qui affiche toutes les coordonnées du point.

4. Point et Triangle

Description de l'exercice : Vous aurez à créer une classe Triangle utilisant 3 points (de la classe de l'exercice précédent).

1) Écrire une classe `Triangle`, avec trois points passés en paramètres pour le constructeur. Ces trois points seront utilisés pour initialiser les 3 paramètres p1, p2, et p3 (les 3 points du triangle).
2) Écrire une méthode `calculer_perimetre` permettant de calculer le périmètre d'un triangle.
3) Écrire un programme principal qui déclare 3 `Points`, puis un `Triangle`, et appelle la méthode `calculer_perimetre`.