Programmation Orientée Objet en Python

Classe :	Terminale NSI, classe entière
Capacités attendues :	-Écrire la définition d'une classe -Accéder aux attributs et aux méthodes d'une classe

Consigne : Créer un répertoire de travail nommé : POO. Dans ce répertoire, vous allez sauvegarder l'ensemble des documents relatifs à la partie POO : cours, énoncés de TPs et corrigés des exercices

Exercice 1 : Soit le script Python illustrant une classe **Cercle** :

```
from math import pi
class Cercle:
    nombre_cercles = 0
    def __init__(self, x, y, rayon):
        self.x = x
        self.y = y
        self.rayon = rayon
        Cercle.nombre_cercles += 1
        print("Après création, le nombre de cercles est: ", Cercle.nombre_cercles)

    def __del__(self):
        Cercle.nombre_cercles -= 1

    def afficher(self):
        print("Centre:({0},{1}), Rayon:{2}".format(self.x, self.y, self.rayon))

    def calculer_surface(self):
        return pi * self.rayon ** 2

#programme principal
cercle_1 = Cercle(10, 25, 2.5)
cercle_2 = Cercle(50, 30, 1)
cercle_3 = Cercle(5, 45, 3.25)
cercle_4 = cercle_3
cercle_1.afficher()
cercle_2.afficher()
cercle_3.afficher()
surface = cercle_1.calculer_surface()
print("La surface du cercle_1 est: ", surface)
surface = cercle_2.calculer_surface()
print("La surface du cercle_2 est: ", surface)
surface = cercle_3.calculer_surface()
print("La surface du cercle_3 est: ", surface)
del cercle_3
print("Après suppression, le nombre de cercles est: ", Cercle.nombre_cercles)
del cercle_2
print("Après suppression, le nombre de cercles est: ", Cercle.nombre_cercles)
```

1- Quels sont les attributs de la classe **Cercle** ? Préciser leur catégorie.

2- Quelles sont les méthodes d'instance ?

3- Quel message est affiché suite à la création de chaque objet ?

4- Combien d'objets ont été créés ? Donner une représentation mémoire après chaque instanciation ?

5- Combien d'objets restent-ils après la suppression de cercle_2 et cercle_3?

6- Dans votre éditeur Python, créer un fichier avec le nom : solution_exo1.py dans votre répertoire de travail, puis copier et coller le code de la classe Cercle. Manipuler la classe :

→ en créant et en supprimant des objets

→ en affichant la valeur des attributs

→ en appelant ses méthodes

7- Soit la classe **Point2D** ayant deux attributs (x, y) de type réel :

```
class Point2D:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def afficher_point(self):
        print("POINT2D(x=",self.x,", y=", self.y,")")
```

a- Dans votre fichier : solution_exo1.py, copier et coller (ou importer) la classe **Point2D**, ensuite modifier le constructeur de la classe **Cercle** afin que le centre du cercle soit représenté par un objet de type **Point2D .**

b- Créer de nouvelles instances de points et de cercles puis afficher les propriétés des objets créés ?

c- Modifier la méthode d'affichage de la classe Cercle afin d'obtenir l'affichage suivant :

```
mon_cercle.afficher() → Centre (x=  10 , y=  25 ), Rayon:  2.5
```

Exercice 2 :

On veut définir une classe nommée **Personne** représentant certains caractères d'une personne. La déclaration d'une personne revient à définir son **nom**, son **adresse**, son âge et à préciser comment on accède aux informations. Le nom et l'adresse sont des chaînes de caractères, l'âge est un entier. La classe **Personne** dont l'état est défini par trois attributs a :

→un constructeur qui permet de donner des valeurs aux attributs.

→la méthode **categorie_d_age()** qui retourne une des chaines de caractères suivantes : "Personne jeune", "Personne d'âge moyen" ou "Personne âgée" selon la valeur de l'attribut âge. La catégorie d'âge est définie comme suit : Si l'âge est > 60 alors "Personne âgée", si l'âge est < 20 alors "Personne jeune" et sinon "Personne d'âge moyen".

→la méthode **categorie_de_residence()** qui affiche "Habitant urbain", "Villageois" ou "Habitant rural" selon la valeur de l'attribut adresse. La catégorie d'adresse est définie comme suit : Si l'adresse contient le mot Ville alors "Habitant urbain", si elle contient le mot Village alors "Villageois" sinon "Habitant rural".

→la méthode **afficher_personne()** qui permet d'afficher les caractéristiques d'une personne : nom, âge et adresse

1- Créer un fichier Python avec le nom : solution_exo2.py dans votre répertoire de travail.

2- Créer le constructeur de la classe **Personne**.

3- Dans le programme principal, créer trois instances de personne avec les propriétés suivantes: ("Louisa", "Ville", 25), ("Emile", "Village", 72),("Charles", "Campagne", 43). Essayer d'afficher certaines propriétés des objets.

4- Ajouter les trois méthodes à la classe **Personne**.

5- Créer une liste de personnes et remplissez-la avec les instances déjà créées, puis appeler les méthodes **categorie_d_age()** et **categorie_de_residence()** pour chacune des personnes de la liste après les avoir affichées.

Exercice 3 :

Soit une classe nommée **Intervalle** qui représente une plage de valeurs entières comprises entre une borne inférieure et une borne supérieure. La classe **Intervalle** possède deux attributs de type entier (**valeur_min** et **valeur_max**) . On considère que les deux bornes sont incluses dans l'intervalle. La classe est dotée des méthodes et constructeur suivants :

→ Un constructeur prenant deux arguments entiers (bornes supérieure et inférieure pouvant être fixées dans n'importe quel ordre). Les valeurs par défaut de ces bornes est [0 ; 0].

→ une méthode **afficher()** qui affiche l'intervalle sous la forme : [min ; max].

→ une méthode **longueur()** qui retourne la longueur de l'intervalle.

→ une méthode **appartient(valeur)** qui retourne True si une valeur appartient à l'intervalle, et False sinon.

→ une méthode **est_inclus_dans(autre_intervalle)** qui retourne True si l'autre intervalle est inclus dans l'intervalle considéré.

1- Créer un fichier Python avec le nom : solution_exo3.py dans votre répertoire de travail.

2- Ajouter un constructeur de la classe **Intervalle** de telle manière de respecter les consignes expliquées ci-dessous.

3- Ajouter la méthode afficher() à la classe **Intervalle**.

4- Copier et coller les instructions ci-dessous permettant de créer trois instances d'intervalles. Vérifier que vous avez un affichage correct [min ; max] tel que min ≤ max.

```
#Programme principal
#Création d'instances
intervalle_1 = Intervalle()
intervalle_2 = Intervalle(200, 100)
intervalle_3 = Intervalle(50, 900)
intervalle_1.afficher() #affiche [ 0 ;  0 ]
intervalle_2.afficher() #affiche [ 100 ;  200 ]
intervalle_3.afficher() #affiche [ 5 ;  900 ]
```

4- Ajouter le reste des méthodes précédemment décrites à la classe **Intervalle**.

5- Tester le bon fonctionnement en ajoutant ces instructions de test à votre programme principal :

```
print("La longueur de I1 est: ", intervalle_1.longueur())
print("La longueur de I2 est: ", intervalle_2.longueur())
print("La longueur de I2 est: ", intervalle_3.longueur())
donnee = int(input("Entrer une valeur: "))
est_interne = intervalle_2.appartient(donnee)
print(donnee, "appartient à I2 --->", est_interne)
```

```
est_inclus = intervalle_2.est_inclus_dans(intervalle_3)
print("I3 est inclus dans I2 --->", est_inclus)
est_inclus = intervalle_3.est_inclus_dans(intervalle_2)
print("I2 est inclus dans I3 --->", est_inclus)
```