Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université de Sfax
Institut Préparatoire aux Etudes
d'Ingénieurs de Sfax

وزارة التعليم العالي والبحث العلمي
جامعة صفاقس
المعهد التحضيري للدراسات الهندسية
بصفاقس

Année universitaire : 2018-2019

Examen semestre1

Matière : informatique

Classe : MP2, PC2 et PT2 — Nombre de pages : 3

Date : décembre 2018 — Durée : 2h

Exercice 1 *(10 pts)*

1. Définir une classe nommée Complexe permettant d'instancier des objets représentant les nombres complexes et possédant les méthodes suivantes :

- Un constructeur permettant d'initialiser les attributs x et y d'instance. Ces attributs représentent respectivement la partie réelle et imaginaire d'un nombre complexe. (valeurs par défaut : 0),
- Une méthode *module* retournant le module du nombre complexe,
- La surcharge de la méthode spéciale *sub* qui permet de soustraire deux nombres complexes,
- Une méthode *distance* retournant la distance entre deux points. (Chacun des points est défini par un nombre complexe),
- La surcharge d'une méthode spéciale d'affichage dont l'invocation sur une instance d'attributs x= 1.2 et y=2.3, permet d'afficher : « Complexe (1.2, 2.3) ».

Dans la suite, un polygone sera défini par une instance de la classe Polygone. L'attribut points de cette instance est une liste d'objets de la classe Complexe. Cette liste représente les sommets du polygone.

2. Définir une classe Polygone avec les méthodes suivantes :

- Un constructeur permettant d'initialiser les sommets d'un polygone et un attribut nom='polygone'.
- Une méthode *est_ferme* permettant de retourner True si le contour du polygone est fermé, False sinon.
- Une méthode *ferme_polygone* permettant d'ajouter un complexe qui ferme le polygone.
- Une méthode *perimetre* permettant de fermer un polygone s'il n'est pas fermé et retourner le périmètre de ce polygone (somme des distances entre les points). Utiliser la méthode distance de la classe Complexe.
- Une méthode d'affichage qui permet d'afficher le nom du polygone et son périmètre.

3. Définir une classe Triangle qui hérite de la classe Polygone avec un constructeur permettant d'initialiser trois attributs représentant les sommets du triangle et qui surcharge l'attribut d'instance nom='triangle'.

On rappelle qu'un polygone régulier à n côtés inscrit dans un cercle (C), de centre O et de rayon r, à des sommets dont les coordonnées sont :

$$x_k = r\cos(\frac{2k\pi}{n}) \quad \text{et} \quad y_k = r\sin(\frac{2k\pi}{n}) \text{ pour } k = 0 \,..\, n$$

4. Ecrire une fonction liste_points qui retourne une liste contenant des objets de la classe Complexe définissant la liste des points d'un polynôme régulier à n côtés.

5. Afficher le périmètre d'un polygone régulier de cinq côtés et d'un triangle inscrits dans un cercle de centre O et de rayon 2.

Exercice 2 *(10 pts)*

L'objectif de cet exercice est la manipulation des polynômes creux à une seule variable.

Un polynôme creux est un polynôme dont certains coefficients sont nuls.

Un polynôme est construit à partir de monômes.

Un monôme est une expression de la forme ax^n où $a\,(a \neq 0)$ est le coefficient du monôme et $n\,(n \geq 0)$ son degré.

Un monôme est représenté par un dictionnaire à un élément dont la clé est le degré n et la valeur est le coefficient a.

Exemple :

Le monôme $8x^2$ est représenté par le dictionnaire $\{2:8\}$.

Un polynôme creux est alors défini comme une association de monômes de degrés différents.

Exemple :

Le polynôme $-x^4+8x^2-5x$ est représenté par le dictionnaire $\{2:8\ ,\ 1:-5\ ,\ 4:-1\}$.

Le dictionnaire $\{0:1\ ,\ 5:1\ ,\ 8:1\}$ représente le polynôme x^8+x^5+1.

On se propose de construire la classe PolynomeCreux à coefficients réels dont le squelette (à compléter) est défini par :

```
class PolynomeCreux :
    """ Manipulation des polynômes creux à une seule variable """
    def __init__(self) :
            self.data={}    # initialisation à un polynôme nul
    def ajout_monome(self,monome={}) :
            """ Cette méthode ajoute un monôme saisi au clavier si le paramètre monome
            est nul ou ajoute le monôme nommé monome sinon """
            if len(monome) == 0 :
                    # Réponse à la Question 1
                    ⋮
            else :
                    # Si monome est non vide
                    degre=list(monome.keys())[0]        # extraction du degré
                    coeff=list(monome.values())[0]      # extraction du coefficient
                    try :
                        assert degre >= 0
                        assert type(degre) == int
                        assert type(coeff) == int or type(coeff) ==float
                        assert len(monome) = 1
                        self.data.update(monome)  # ou self.data[degre]=coeff
                    except :
                        print (" Erreur d'ajout du monome")
    def degree(self) :
            # Réponse à la Question 2
            ⋮
    def __call__(self,x0) :
            # Réponse à la Question 3
            ⋮
    def __add__(self,other) :          # other est un polynôme creux
            # Réponse à la Question 4
            ⋮
```

```
def __mul__(self,other) :                # other est un polynôme creux
        # Réponse à la Question 5
        ⋮

def __str__(self) :
        # Réponse à la Question 6
        ⋮

def  primitive(self) :
        # Réponse à la Question 7
        ⋮
```

Question 1

Compléter le script de la méthode **ajout_monome**. On rappelle que si le paramètre monome est nul cette méthode ajoute un monôme saisi au clavier (en faisant les contrôles nécessaires : degré positif et coefficient différent de zéro) sinon on ajoute le monôme nommé monome.

Question 2

Ecrire le script de la méthode, nommée **degree**, qui retourne le degré du polynôme.

Question 3

Ecrire le script de la méthode, nommée **__call__**, qui retourne la valeur du polynôme pour un réel ***x0*** donné.

Question 4

Ecrire le script de la méthode, nommée **__add__**, qui retourne le polynôme somme de deux polynômes.

Remarque : aucun monôme nul ne doit apparaître dans le polynôme résultat.

Question 5

Ecrire le script de la méthode, nommée **__mul__**, qui retourne le polynôme produit de deux polynômes.

Remarque : aucun monôme nul ne doit apparaître dans le polynôme résultat.

Question 6

Ecrire le script de la méthode, nommée **__str__**, qui retourne la chaîne représentant l'expression du polynôme ordonné par ordre décroissant.

Pour le polynôme représenté par $\{4:4\ ,\ 0:4\ ,12:6,9:1,7:-1\}$, la chaîne retournée est :

" 6*x**12 + x**9 - x**7 + 4*x**4 + 4 "

Question 7

Ecrire le script de la méthode, nommée **primitive**, qui retourne le polynôme représentant la primitive. On suppose que la constante d'intégration est nulle.

Question 8

On définit, l'intégrale d'un polynôme creux P en x entre les bornes a et b, par : $S = \int_a^b P dx$

Ecrire le script de la fonction, nommée **integrale**, permettant de retourner la valeur de S à partir d'un polynôme P, de type PolynomeCreux, et des bornes d'intégration a et b réels.