REPUBLIQUE TUNISIENNE

Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique

Concours Nationaux d'Entrée aux Cycles de Formation d'Ingénieurs Session 2017

الجمهورية التونسية

وزارة التعليم العالي والبحث العلمي

المناظرات الوطنية للدخول إلى مراحل تكوين المهندسين دورة 2017

Concours Mathématiques et Physique, Physique et Chimie et Technologie

Epreuve d'Informatique

Date : Vendredi 26 Mai 2017 Heure : 14 H Durée : 2 H Nombre de pages : 5

Barème : EXERCICE 1 : 8 points
EXERCICE 2 : 5 points
EXERCICE 3 : 7 points

DOCUMENTS NON AUTORISES
L'USAGE DES CALCULATRICES EST INTERDIT

EXERCICE 1

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
        :
```

Travail demandé :

Question 1

Compléter le script de la méthode **ajout_monome**. On rappelle que cette méthode ajoute un monôme saisi au clavier (en faisant les contrôles nécessaires) si le paramètre monome est nul ou ajoute le monôme nommé monome sinon.

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.

EXERCICE 2

Le schéma relationnel suivant permet de stocker des informations relatives à la gestion d'une base de données.

- **Utilisateur** (<u>IdU</u> , Nom , Prenom)

  La relation *Utilisateur* contient tous les utilisateurs de la base.
  - IdU : identifiant de l'utilisateur (entier), clé primaire.
  - Nom : nom de l'utilisateur (chaîne de caractères).
  - Prenom : prénom de l'utilisateur (chaîne de caractères).

- **Table** (<u>IdTable</u> , IdCreateur)

  La relation *Table* contient toutes les tables de la base.
  - IdTable : nom de la table (chaîne de caractères), clé primaire.
  - IdCreateur : identifiant du créateur de la table (entier).

- **Privilege** (IdTable , IdU , Droit)

  La relation *privilege* définit les droits de manipulation des tables identifiées par IdTable par les utilisateurs identifiés par IdU. modifier change insérer
  - IdTable , IdU: clé primaire.
  - Droit appartient à {'CREATE', 'DROP', 'ALTER', 'SELECT', 'INSERT', 'UPDATE', 'DELETE' , 'ALL', …}. autre

Travail demandé :

Question 1

En utilisant le module sqlite3 donner le script PYTHON permettant de créer la table **Utilisateur** dans la base "EXERCICE2.db" en exprimant toutes les contraintes d'intégrité mentionnées ci-dessus.

Dans la suite on suppose que les trois tables de la base "EXERCICE2.db" sont créées et remplies. Donner les requêtes SQL permettant de :

Question 2

Déterminer le nom et le prénom de tous les utilisateurs.

Question 3

Donner le nombre des utilisateurs de la base. phénomène

Question 4

Déterminer, pour chaque créateur, le nombre de tables créées.

Question 5

Déterminer les identifiants des utilisateurs ayant le droit de création de nouvelles tables.

Question 6

Classer par ordre alphabétique décroissant les noms des utilisateurs.

Question 7

Déterminer les noms et les prénoms des utilisateurs ayant le droit 'INSERT' associé à la table dont IdTable = 'Produit'.

Question 8

Donner en algèbre relationnelle l'équivalent de la requête écrite à la **Question 7**.

EXERCICE 3

L'objectif de l'exercice est l'utilisation de la méthode des moindres carrés pour approximer un polynôme à partir d'observations. La spécificité de cette méthode est de minimiser la somme des distances entre un polynôme g approximant et n points expérimentaux.

Etant donné une observation de n points distincts $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, on cherche à approcher cette observation, au sens des moindres carrés, par un polynôme $g(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_m x^m$ de degré m égal à $n-1$, donnant une meilleure approximation des valeurs y_i. Ceci revient à minimiser la distance qui sépare un point expérimental (x_i, y_i) du point approximant $(x_i, g(x_i))$.

Le polynôme donnant la meilleure approximation est celui qui minimise la somme S, des écarts entre les y_i et les $g(x_i)$, donnée par la formule suivante :

$$S(a_0, a_1, a_2, \dots, a_m) = \sum_{i=1}^{n} \left| y_i - \left(a_0 + a_1 x_i + a_2 x_i^2 + \dots + a_m x_i^m\right) \right|$$

Afin d'identifier la distance moyenne minimale, on cherche l'ensemble des valeurs des paramètres a_i minimisant cette somme.

Ce qui conduit à résoudre le système linéaire suivant :

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^0 & \sum_{i=1}^{n} x_i^1 & \cdots & \sum_{i=1}^{n} x_i^m \\ \sum_{i=1}^{n} x_i^1 & \sum_{i=1}^{n} x_i^2 & \cdots & \sum_{i=1}^{n} x_i^{m+1} \\ : & : & \cdots & : \\ \sum_{i=1}^{n} x_i^{m-1} & \sum_{i=1}^{n} x_i^m & \cdots & \sum_{i=1}^{n} x_i^{2m-1} \\ \sum_{i=1}^{n} x_i^m & \sum_{i=1}^{n} x_i^{m+1} & \cdots & \sum_{i=1}^{n} x_i^{2m} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ : \\ a_{m-1} \\ a_m \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i^0 y_i \\ \sum_{i=1}^{n} x_i^1 y_i \\ : \\ \sum_{i=1}^{n} x_i^{m-1} y_i \\ \sum_{i=1}^{n} x_i^m y_i \end{bmatrix}$$

En posant $U_k = \sum_{i=1}^{n} x_i^k$ et $v_k = \sum_{i=1}^{n} x_i^k y_i$, le système s'écrit alors :

$$\begin{bmatrix} U_0 & U_1 & \cdots & U_m \\ U_1 & U_2 & \cdots & U_{m+1} \\ : & : & \cdots & : \\ U_{m-1} & U_m & \cdots & U_{2m-1} \\ U_m & U_{m+1} & \cdots & U_{2m} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ : \\ a_{m-1} \\ a_m \end{bmatrix} = \begin{bmatrix} v_0 \\ v_1 \\ : \\ v_{m-1} \\ v_m \end{bmatrix} \Rightarrow U.a = v$$

Travail demandé :

Dans la suite, on suppose que les n points représentant une observation, sont stockés dans une liste de tuples Lp, où chaque tuple représente les coordonnées d'un point.

Le travail demandé consiste à déterminer les coefficients a_i du polynôme d'interpolation $g(x)$.

Question 1

Ecrire une fonction python, nommée **puiss**, qui, pour un réel x et un entier p, crée et retourne la liste $\left[x^0, x^1, \cdots, x^{2p}\right]$.

Question 2

Ecrire une fonction python, nommée **list_puiss**, qui, à partir d'une liste de réels L et d'un entier p, crée et retourne une liste de listes contenant, pour chaque réel r de L, une liste $\left[r^0, r^1, \cdots, r^{2p}\right]$.

Question 3

Ecrire une fonction python, nommée **calcul_mat**, qui, à partir de la liste de points Lp, crée et retourne la matrice U.

Question 4

Ecrire une fonction python, nommée **calcul_vect**, qui à partir de la liste de points Lp, crée et retourne le vecteur v.

Question 5

Ecrire un script python qui détermine les coefficients du polynôme d'interpolation en faisant les importations adéquates.