**Ministère de l'Enseignement Supérieur**
**et de la Recherche Scientifique**
**Université de Carthage**

وزارة التعليم العالي و البحث العلمي
جامعة قرطاج

IPEIN
المعهد التحضيري للدراسات الهندسية بنابل
Institut Préparatoire aux Etudes d'Ingénieurs de Nabeul

# Informatique

Concours Blanc

25.04.2017

2 heures

***SM, SP, ST***

6 pages

*Corrigé*

## Exercice 1. (3 points)

**Solution:** (1 point)

$$\forall n \in \mathbb{N}^*, A_n(f) = \frac{b-a}{n} \sum_{k=0}^{n-1} f(\frac{a_k + a_{k+1}}{2})$$

**Solution:**

```
#Q2 (1 point)
def ptmilieu (f, a, b, n):
    int_ptmilieu = 0
    h = (b-a)/n
    t = (a+h)/2
    for k in range(n):
        int_rect += h*f(t)
        t += h
    return int_rect

#Q3 (1 point)
from math import sin
f = lambda t : sin (t)
ptmilieu(f, 0, pi, 100)
```

## Problème 1. (7 points)

**Solution:**

```
#=================
# Partie 1
#=================

#Q1 (1 point)
import numpy as np
def saisie_mat(n,m):
    LL=[]
    for i in range(n):
        L=[]
        for j in range(m):
            while True:
                try:
                    x=float(input("coefficient d'indice {},{} ?".format(i,j)))
                    break
                except:
                    continue
            L.append(x)
        LL.append(L)
    return np.array(LL)

n = 4; A = saisie_mat(n,n)

#Q2 (0.5 points)
L = np.eye(n)
#L = np.ones((n,n))

#Q3 (0.5 points)
U = np.zeros((n,n))

#Q4 (0.5 points)
def somme (L,U, p, i, j):
        s=0
        for k in range(1,p):
                s+=L[i-1,k-1] * U[k-1,j-1]
        return s

#Q5 (Décomposition LU) (1 point)
for i in range(1,n):
        for j in range(i,n+1):
                U[i-1,j-1] = A[i-1,j-1] - somme(L,U,i,i,j)
        for j in range(i+1,n+1):
                L[j-1,i-1] =(1/U[i-1,i-1]) * (A[j-1,i-1] - somme(L,U,i,j,i))
```

```
44
45  U[n-1,n-1] = A[n-1,n-1] - somme(L,U,n,n,n)
46
47  #Q6 (0.5 points)
48  print(A, L, U, np.dot(L,U))
49
50  #=================
51  # Partie 2
52  #=================
53  #Q1 (0.5 points)
54  def sommeU (U,X,i,n):
55          s = 0
56          for j in range(i+1,n+1):
57                  s += U[i-1,j-1] * X[j-1,0]
58          return s
59
60  #Q2 (0.5 points)
61  def sommeL (L,Y,i):
62          s = 0
63          for j in range(1,i):
64                  s += L[i-1,j-1] * Y[j-1,0]
65          return s
66
67  #Q3 (0.25 points)
68  B = saisie_mat(n,1)
69
70  #Q4 (0.5 points)
71  X = np.zeros((n,1))
72  Y = np.zeros((n,1))
73
74  #Q5 (0.5 points)
75  #Résolution LY=B
76  Y[0,0] = B[0,0]/L[0,0]
77  for i in range(2,n+1):
78          Y[i-1,0] = (1 / L[i-1,i-1]) * (B[i-1,0] - sommeL(L,Y,i))
79
80  #Q6 (0.5 points)
81  #Résolution UX=Y
82  X[n-1,0] = Y[n-1,0]/U[n-1,n-1]
83  for i in range(n-1,0,-1):
84          X[i-1,0] = (1 / U[i-1,i-1]) * (Y[i-1,0] - sommeU(U,X,i,n))
85
86  #Q7 (0.25 points)
87  print("Matrice A = "); print(A)
88  print("Matrice B = "); print(B)
89  print("Matrice X = "); print(X)
```

## Problème 2. (10 points)

**Solution:** #Q1 (1 point)

```
CREATE TABLE `installer` (
        `id_poste`      INTEGER NOT NULL,
        `id_logiciel`   INTEGER NOT NULL,
        `date_installation`     DATE NOT NULL,
        `duree_licence` VARCHAR NOT NULL,
        PRIMARY KEY(id_poste,id_logiciel),
        FOREIGN KEY(`id_poste`) REFERENCES poste,
        FOREIGN KEY(`id_logiciel`) REFERENCES logiciel
);
```

```
#Q2 (1 point)
def executer(requete):
    import sqlite3
    connexion = sqlite3.connect('gerer_logiciels.db')
    cur = connexion.cursor()
    lt=cur.execute(requete).fetchall() #retourne une liste de tuples
    connexion.commit()
    connexion.close()
    return lt

#Q3 (1.5 points)
class Logiciel :
        def __init__ (self, nom, version, prix):
                self.nom = nom
                self.version = version
                self.prix = prix

        def enregistrer (self):
                """
                Permet d'enregistrer le logiciel p dans la table logiciel
                """
                requete = "INSERT INTO logiciel \
                VALUES (null,'"+self.nom+"','"+self.version+"','"+self.prix+"');"
                print (requete)
                executer(requete)

        def chercherId(self):
                """
                Permet de chercher un logiciel dans la table Logiciel
                par son nom et sa version et retourne son identifiant.
                """
                requete = "SELECT id FROM logiciel\
                WHERE nom = '"+ self.nom +"'\
```

```
                AND         version = '"+ self.version +"';"
                print (requete)
                lt=executer(requete)
                id=lt[0][0]
                return (id)

        def __str__(self):
                """
                visualiser un logiciel (nom, version, prix).
                """
                return "Logiciel("+self.nom+", "+self.version+", "+self.prix+")"

#Q4 (1 point)
def liste_salles():
        requete ="SELECT * FROM salle ;"
        liste_de_tuples=executer(requete)
        liste_de_salles=[]
        for tup in liste_de_tuples:
                nouv_salle = Salle(tup[1])
                liste_de_salles.append(nouv_salle)
        return liste_de_salles

#Q5 (0.5 points)
scilab = Logiciel("Scilab","2.3","infini")
scilab.enregistrer()


#Q6 (1 point)
toutes_les_salles = liste_salles()
for s in toutes_les_salles:
        les_postes = s.charger_liste_postes()
        for p in les_postes():
                p.installer(scilab,'04/04/2017', 'infini')

#Q7 (0.5 points)
#Définition du concept d'encapsulation en POO
"""
Le concept d'encapsulation est un concept très utile de la POO.
Il permet en particulier d'éviter une modification par erreur
des données d'un objet. En effet, il n'est alors pas possible
d'agir directement sur les données d'un objet ; il est nécessaire
de passer par ses méthodes qui jouent le rôle d'interface obligatoire.
"""
```

```
#Q8 (0.5 points)
#méthode __init__
self.__prix = prix

#méthode enregistrer
requete = "INSERT INTO logiciel \
VALUES (null,'"+self.nom+"','"+self.version+"','"+self.__prix+"');"

#méthode  __str__
return "Logiciel("+self.nom+", "+self.version+", "+self.__prix+")"

#Q9 (0.5 points)
def getPrix(self):
        return self.__prix

def enregistrer (self):
...
        VALUES (...+self.getPrix()+");"

def __str__(self):
        return "Logiciel ("+self.nom+", "+self.version+", "+self.getPrix()+")"

#Q10 (0.5 points)
def setPrix(self, nouveauPrix):
        assert type(nouveauPrix)==float and nouveauPrix > 0, raise ValueError
        self.__prix = nouveauPrix

#Q11 (1 point)
def nb_logiciels(self):
        requete =
        "SELECT COUNT (*) AS NB \
         FROM INSTALLER \
         WHERE id_poste = "+ self.id +";"
        lt=executer(requete)
        nb=lt[0][0] #Premier élément du premier tuple du résultat
        return (nb)

#Q12 (1 point)
```

```
SELECT * FROM poste
WHERE poste.id NOT IN (
                          SELECT installer.id_poste
                          FROM installer, logiciel
                          WHERE installer.id_logiciel = logiciel.id
                          AND logiciel.nom ='Python'
                        );
```