Algorithmique 5 : listes et matrices

Exercice 1 :

En Python, une liste est une collection d'éléments divers séparés par des virgules, l'ensemble étant enfermé entre crochets. Commenter chacune des instructions suivantes :

```
mois=['janvier','février','mars','avril','mai','juin','juillet','août',
'septembre','octobre','novembre','décembre']
nbjours=['31','28','31','30','31','30','31','31','30','31','30','31']

print("**********************")
i=0
while i<len(mois):
    print(mois[i],end=' ')
    i=i+1

print("\n") #on passe à la ligne
print("**********************")
a,b=0,0
while a<12:
    b=a%12
    a=a+1
    print(a,mois[b])

print("**********************")
for i in range(13): #i prend successivement les valeurs 0,1,2,..,12
    print("le mois de",mois[i],"a",nbjours[i],"jours.")
```

Exercice 2 : liste de nombres au hasard

On crée une liste de n nombres entiers pris au hasard entre -100 et 100. Commenter les instructions suivantes :

```
from random import *
def liste_aleatoire(n):
        s = [0]*n #on crée une liste remplie de 0
        for i in range(n):
                s[i] = randint(-100,100)
                #on demande un entier au hasard entre -100 et 100
        return s

n=int(input("Donnez la longueur de la liste (inférieure à 200): "))
lst=liste_aleatoire(n)
print(lst)

lst.reverse()
print(lst)
lst.sort()
print(lst)
```

Exercice 3 : manipulations simples

Commenter chacune des manipulations suivantes :

```
nombres=[-12,4,-3,18,7,21,19,-14,2,4,-7]
#on pourra faire des essais avec les nombres que l'on veut et
#aussi faire un essai avec une liste comportant au moins deux nombres égaux
nombres.append(15)
print(nombres)
nombres.remove(-3)
print(nombres)
del nombres[7]
print(nombres)
print(nombres.index(4))
```

Exercice 4 : recherche d'un maximum

Créer une liste de vingt nombres entiers et écrire un algorithme qui fournit le plus grand nombre de la liste.

Exercice 5 : crible d'Eratosthène

Ecrire une suite d'instructions qui demande à l'utilisateur un entier non nul n, créee une liste de n nombres tous égaux à 1, et met à 0 les éléments de la liste dont l'indice n'est pas un nombre premier (c'est-à-dire qui est multiple d'un entier supérieur ou égal à 2), puis affiche tous les nombres premiers inférieurs ou égaux à n.

Exercice 6 : matrices

En Python, une matrice à n lignes et p colonnes peut être créée comme « liste de listes » en écrivant chaque ligne elle-même comme une liste :

```
m=[[1,2,3],[-1,0,1],[-2,1,1],[-1,-2,4]]
print(m)
print(m[0][2])
print(m[1])
for k in range(0,3):
    print(m[k][0])
print(len(m))
```

Exercice 7 : opérations matricielles simples

1. Rédiger un algorithme de multiplication d'une matrice A par un coefficient réel r (A et r étant demandés à l'utilisateur).
2. Rédiger un algorithme d'addition de deux matrice m_1 et m_2 de mêmes dimensions (m_1 et m_2 étant demandées à l'utilisateur).

Exercice 8 : produit matriciel

Rédiger un algorithme réalisant le produit $m_1 \times m_2$ de deux matrice m_1 et m_2 (m_1 et m_2 étant demandées à l'utilisateur, et l'algorithme retournant « Erreur » si les dimensions de m_1 et m_2 ne sont pas compatibles).

Exercice 9 : calculs matriciels avec Numpy

En Python, le module Numpy possède des fonctions permettant de réaliser certains calculs matriciels. Commenter les instructions suivantes :

```
from numpy import *
#instructions diverses calcul matriciel
x = matrix([[1,2,4],[-1,0,1],[2,1,1]]) # attention à la syntaxe pour déclarer une matrice !
y = matrix([[-1,1,3],[2,0,1],[0,4,-1]])
print(x*y)
z=x.I
print(z)
print(x*z)
print(x**2)
```

Exercice 10 : résolution de systèmes linéaires

En Python, le module Numpy permet de résoudre les systèmes linéaires. Commenter les instructions suivantes, puis compléter le programme pour vérifier que le produit ax est bien égal à la matrice unité.

```
from numpy import *
#resolution systemes lineaires
a = matrix(((4,8,12),(3,8,13),(2,9,18)))
print(a)
b=mat('[4 ; 5 ; 11]')
print(b)
x=linalg.solve(a,b)
print(x)
```