Université de Provence
Licence Math-Info
Première Année
V. Phan Luong

Algorithmique et Programmation en Python

FONCTIONS

Solutions du TD4 (*L'énoncé de Mr. J.L Paillet)*

1 Exercice 1

Ex 1.1 : croit(n) crée une list de 1 à n (range(n) crée une liste de 0 à n-1)

```
def croit(n) :
    l = []
    i = 1
    while i <= n :
        l.append(i)  # on peut écrire autrement l = l + [i]
        i = i + 1
    return l
```

Ex 1.2 : Créer une liste des multiples de m et inférieurs à n

```
def multiples(n, m):
    l = []
    i = 0
    while i*m <= n :
        l.append(i*m)
        i = i + 1
    return l
```

Ex 1.3 : Créer une liste de nombres de Fibonacci inférieurs à m

```
def F(m) :
    if m == 0: l = [1]
    elif m == 1: l = [1, 1]
    else:
        l = [1, 1]
        i = 2
        while i <= m :
            nouv = l[i-1] + l[i-2]
            l.append(nouv)
            i = i + 1
    return l
```

```
################### Script de Programmes de l'Exercice 1  #################
```

Ex 1.1

```
n = input('Entrer un nombre: ')
list = croit(n)
print list
```

Ex 1.2

```
n = input('Entrer n: ')
m = input('Entrer m: ')
list2 = multiples(n,m)
print list2
```

Ex 1.3

```
m = input('Entrer m: ')
list3 = F(m)
print list3
```

Ex 1.4

```
# Point 1


i = 0
while i <= m :
    if list3[i] % 2 == 0 :
        print list3[i], # la virgule pour ne pas descendre à la ligne
    i = i + 1
print ' '               # Pour descendre à la ligne


# Point 2:


i = 0
while i <= m :
    if list3[i] % 2 == 0 : i = i + 1
    else:
        del list3[i] # Supprimer i ème élément de la liste
        m = m - 1    # La longueur de la liste est réduite

print list3
```

Ex 1.5 :

```
# La fonction str() transforme son argument en chaine de caractères
# Pour les chaines, l'opérateur + fait la concaténation de chaines

list4 = []
i = 0
while i <= m :
    if list3[i] % 2 == 0 :
        list4.append('No '+str(i)+' = '+ str(list3[i]))
    i = i + 1
print list4
```

2 Exercice 2

Liste des nombres premiers inférieurs à n

Ex 2.1 : Pour chaque $i >= 3$ vérifier si i est premier en divisant i par les nombres premiers déjà retrouvés. Si i est premier alors ajouter i dans L.

```
def premiers(n) :
    L = [1, 2]
    i = 3
    while i < n :     # Verifier si i est premier: successivement divise i
        j = 1         # par les nb premiers trouves, commencer par L[1] = 2
        k = len(L)
        while j < k : # and L[j] <= i/2 pour terminer plus tot
            if i % L[j] == 0 : break  # i n est pas premier
            j = j + 1
        else :
        # ou j >= k or L[j] >= 2
        # terminer la boucle sans rencontrer break
            L = L + [i]
        i = i + 1
    return L
```

Ex 2.1 : Crible d'Eratosthene pour retrouver les nombre premiers $< n$

```
def crible(n) :
    L = range(n) # L = [0, 1, 2, 3, ..., n-1]
    i = 2        # L[i]==i, on commence par enlever les multiples de 2
    while i < n :
        j = i + i
        while j < n :
            L[j] = 0   # marquer les
            j = j + i  # multiples de i
        i = i + 1      # Chercher le nomb premier après i
        while i < n and L[i] == 0 :
            i = i + 1
        # revenir a la boucle while exterieur: enlever les multiples de i


    j = 0
    k = len(L)
    while j < k:      # Revisiter L pour enlever les éléments 0
        if L[j] == 0:
            del L[j]  # Supprimer un élément 0 de L, et cela reduit L
            k = k - 1
        else: j = j + 1
    return L

###################### Scripts de programme ######################
n = input('Entrer un entier : ')
print premiers(n)
print 'Et par method du Crible d Eratosthene: '
print crible(n)
```

3 Exercice 3

Ex 3.1 :

```
# produit scalaire de deux vecteurs u et v repprésentés sous forme de listes
def prodscal(u, v) :
    k = len(u)
    i , s = 0, 0   # i et s sont initialisées à 0
    while i < k :
        s = s + u[i] * v[i]
        i = i + 1
    return s   # le produit scalaire de u et v
```

Ex 3.2 :

```
def printmat_1(M) :
    i = 0
    l = len(M)
    while i < l :
        print(M[i])
        i = i + 1
```

Ex 3.4 :

```
# transposer une matrice M représentée sous forme de listes de lignes
def transpose(M) :
    Mt = []      # Mt: matrice transposee de M
    l = len(M)   # l : nombre de lignes de M
    k = len(M[0]) # k : nombre de colonnes de M
    j = 0    # i : indice de ligne, j : indice de colonne
    while j < k :
        Lt = []  # creer une ligne pour Mt
        i = 0
        while i < l :
            Lt = Lt + [M[i][j]]
            i = i + 1
        Mt = Mt + [Lt]
        j = j + 1
    return Mt
```

Ex 3.3 :

```
# Produit d'un vecteur avec une matrice
def app_vec_mat(V, M): # V: un vecteur, M: une matrice
    nlin = len(M)
    ncol = len(M[0])
    Mt = transpose(M)
    if nlin != len(V):
        print "Erreur de dimensions !"
        return -1
    else:
        R = []
        i = 0
        while i < ncol :
            R = R + [prodscal(V, Mt[i])]
            i = i + 1
        return R
```

Ex 3.5 : (*Complémenaire : Produit de deux matrices*)

```
# Produit de deux matrices
def prodmatrice(N, M) :
    nlinN = len(N)
    ncolN = len(N[0])
    nlinM = len(M)
    if ncolN != nlinM : print "Erreur de dimension !"
    else:
        i = 0
        R = []
        while i < nlinN :
            R = R + [app_vec_mat(N[i], M)]
            i = i + 1
        return R
```

Scripts de programme

```
M = []
ligne = list(input("une ligne : "))  # une ligne : 1,2,3
while ligne != [' '] :               # une ligne : ' '
    M = M + [ligne]
    print M
    ligne = list(input("une ligne : "))

print 'La matrice M: '
printmat_1(M)
print 'La transposee de M: '
Mt = transpose(M)
printmat_1(Mt)
print 'Nombre de lignes de M : ', len(M)
print 'Nombre de colonnes de M : ', len(M[0])
V = input("Un vecteur: ")
print 'Application de V a M : '
print app_vec_mat(V, M)
```

Ex 3.5 : (*Complémenaire : Produit de deux matrices*)

```
N = []
ligne = list(input("une ligne : "))  # une ligne : 1,2,3
while ligne != [' '] :               # une ligne : ' '
    N = N + [ligne]
    ligne = list(input("une ligne : "))
print 'La matrice N: '
printmat_1(N)

print 'Le produit de N et M : '
printmat_1(prodmatrice(N, M))
```