2021 - 2022

Algorithmique et programmation en Python

M. Lazhar Zouari

Problème 1

Un CODEC est un logiciel compresseur décompresseur de fichiers. En effet, les suites de bits composant un fichier comportent des similitudes comme 10000111. Plutôt que de stocker la totalité de cet octet, on gagne de la place en écrivant 14031 (qui se lit un quatre zéros trois un). Cet octet (huit bits) retrouva ensuite son format original à la décompression.
Il s'agit alors de saisir une chaîne de huit chiffres formés uniquement 0 et 1 pour désigner un octet puis la compresser suivant le principe de compression du CODEC et enfin l'afficher.
Exemple : Si octet = "10010111" Alors l'octet compressé est : "1201031"

Solution

```
def binaire (ch):
    i , l = 0, len(ch)
    while ((i < l) and (ch[i] in ["0", "1"])):
        i += 1
    return i == l

def saisir():
    ch = ""
    while ((len(ch) !=8) or (not binaire(ch))):
        ch = input ("Tapez une chaîne binaire : ")
    return ch

def compresser (ch):
    i , l, ch1 = 0 , len(ch), ""
    while ( i != l):
        occ, ok = 1, True
        while (ok and i != l-1):
            if ch[i] == ch[i+1] :
                occ += 1
                i += 1
            else:
                ok = False
        if occ == 1 :
            ch1+= ch[i]
        else:
            ch1 += str(occ) + ch[i]
        i += 1
    return ch1

octet = saisir()
octetCom = compresser(octet)
print ("Si octet = ",octet, "Alors l'octet compressé est : ", octetCom)
```

Problème 2

Écrire un programme qui permet de saisir une chaîne de chiffres **CH**, de chercher la combinaison maximale **CMAX** et la combinaison minimale **CMIN** qu'on peut obtenir à partir des chiffres de **CH**, et enfin de les afficher. Pour déduire **CMAX** à partir de **CH**, on vous propose les étapes suivantes : Chercher le plus grand chiffre dans CH Le placer dans la chaîne **CMAX** et **CMIN** Remplacer le chiffre qui était le plus grand par le caractère "*".

Solution

```
def saisir ():
    ch = ""
    while (not ch.isnumeric()): #Ou not ch.isdigit()
        ch = input("Tapez une chaîne de chiffres ")
    return ch

def PosPlusGrand(ch):
    j, l = 0, len(ch)
    for i in range (1, l):
        if ch[i] > ch[j] :
            j = i
    return j

def minMax(ch):
    ch1, ch2, l = "", "", len(ch)
    for i in range(l):
        p = PosPlusGrand(ch)
        ch1 = ch[p] + ch1
        ch2 = ch2 + ch[p]
        ch = ch[:p] + ch[p+1:]
    return ch2, ch1

ch = saisir()
cmax, cmin = minMax (ch)
print("La combinaison minimale ",cmin)
print("La combinaison maximale ",cmax)
```

Problème 3

Écrire un programme Python qui permet de remplir un tableau **T** par **n** caractère **(6 ≤ n ≤ 30)**.
Et de répartir ces n caractères sur trois tableaux et les afficher :
TL : Tableau de lettres
TC : Tableau de chiffres
TS : Tableau de symboles
Exemple :
Soit n = 10

T	H	4	!	K	}	2	R	$	8	d

On doit obtenir les tableaux suivants :

TL	H	K	R	d

TC	4	2	6

TS	!	}	$

Solution

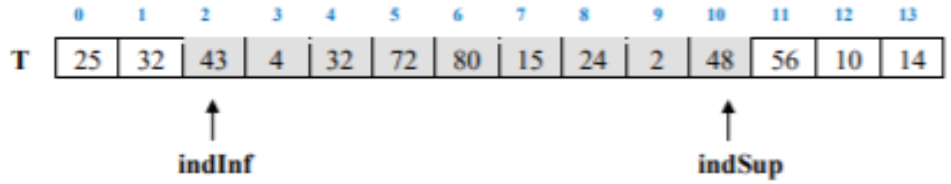
```
from numpy import array
def saisir():
    x = 0
    while not(x  in range(6, 31)) :
        x = int(input("Nombre des éléments : "))
    return x

def repartir(v, m):
    x , y , z = 0, 0, 0
    vl = array([str()]*m)
    vc = array([str()]*m)
    vs = array([str()]*m)
    for i in range(m):
        if "A" <= v[i].upper()<= "Z" :
            vl[x] = v[i]
            x = x + 1
        elif "0" <= v[i] <= "9" :
            vc[y] = v[i]
            y = y + 1
        else :
            vs[z] = v[i]
            z = z + 1
    return vl,vc,vs,x, y, z
```

```
def remplir(x):
    vect = array([str()]*x)
    for i in range(x):
        while len(vect[i]) != 1 :
            vect[i] = input ("Valeur de l'élément N ° " + str(i+1) + " : ")
    return vect

def afficher (v, x):
    for i in range(x):
        print (v[i], end = '  ')

n = saisir ()
t = remplir(n)
tl, tc, ts, n1, n2, n3 = repartir(t,n)
print ("Les lettres : ")
afficher (tl, n1)
print ("\nLes chiffres : ")
afficher (tc, n2)
print ("\nLes symboles : ")
afficher (ts, n3)
```

Problème 4

Écrire un programme Python qui permet de déterminer et d'afficher tous les diviseurs suivis de tous les multiples d'un entier **p** donné, dans une partie d'un tableau **T** de **n** entiers donnés. Cette partie est délimitée par deux indices **indInf** et **indSup**. **(0 ≤ indInf < indSup < n ≤ 15).**

Exemple :



Pour **n** = 14, **p** = 8, **ind_inf** = 2 et **ind_sup** = 10, le programme affichera :
Les diviseurs de 8 sont : 4 2
Les multiples de 8 sont : 32 72 80 24 48

N.B : La solution doit comporter au moins deux sous programmes.

Solution

```
def saisir():
    x = 0
    while not(x  in range(2, 16)) :
        x = int(input("Nombre des éléments : "))
    return x

def remplir(x):
    from numpy import array
    vect = array([int()]*x)
    for i in range(x):
            vect[i] = int(input("Valeur de l'élément N ° " + str(i+1) + " : "))
    return vect

def indices (x):
    i , j  =  int(), int()
    while (i < 0) or (i >= j) or (j >= n) :
        i = int(input("Indice inférieur : "))
        j = int(input("Indice supérieur : "))
    return i, j

def determiner (x, v, p1, p2):
    ch1, ch2 = "", ""
    for i in range(p1, p2+1):
        if (v[i] >= x) and (v[i] % x == 0):
            ch2 = ch2 + str(v[i]) + " "
        if (v[i] <= x) and (x % v[i]==0):
            ch1 = ch1 + str(v[i]) + " "
     return ch1, ch2

n = saisir ()
t = remplir(n)
indInf, indSup = indices(n)
p = int(input("Tapez un entier"))
diviseurs, multiples = determiner (p, t, indInf, indSup)
print ("Les diviseurs de ", p, " sont : ", diviseurs)
print ("Les multiples de ", p, " sont : ", multiples)
```

Problème 5

L'algorithme suivant est celui d'une fonction permettant de calculer la somme d'une partie d'éléments d'un tableau **T** de **n** entiers, délimité par les indices **p1** et **p2**.

```
Fonction somme (T : TAB ; p1, p2 : entier) : entier
DEBUT
        s ← 0
        Pour i de p1 à p2 faire
                s ← s + T[i]
        Fin Pour
        Retourner s
FIN
```

En exploitant la fonction dont l'algorithme est ci-dessus, Écrire un programme Python qui permet de :

- Saisir un tableau **V** de **N** entiers strictement positifs (**5 ≤ n ≤ 20**).
- Afficher l'indice (**ind**) de l'élément du tableau dont l'écart entre la somme (**s1**) des éléments qui le précédent et celle des éléments qui le succèdent (**s2**) soit minimal
- Afficher les sommes **s1** et **s2** correspondantes

Exemple

Pour N = 12 et le tableau V suivant :

	0	1	2	3	4	5	6	7	8	9	10	11
V	11	3	9	24	30	7	4	14	16	21	13	16

Le programme affiche : S1 = 84, S2 = 80 et ind = 6

Solution

```
def saisir ():
    x = int()
    while x < 5 or x > 20 :
        x = int(input("Nombre des éléments : "))
    return x

def remplir (x):
    from numpy import array
    t = array([-1] * x)
    for i in range (x) :
        while t[i] < 0 :
            t[i] = int (input ("la valeur de l'élément N° " + str (i + 1)))
    return t

def somme (t, p1, p2):
    s = 0
    for i in range (p1,p2 + 1) :
        s = s + t[i]
    return s

def ecartMin (t, x):
    e1, e2, j = -1, 0, 1
    while ((e2 > e1) and (j != x - 1)):
        j = j + 1
        e1 = somme (t, 0, j - 1)
        e2 = somme (t, j + 1, x - 1)
    return j, e1, e2

n = saisir ()
v = remplir (n)
ind, s1, s2 = ecartMin (v, n)
print ('S1 = ', s1,' , S2 = ', s2,' et ind = ', ind)
```