TP 2 : Rappels sur les listes - Corrigé

1 BCPST 2 - TP Informatique

```
#%%
#Q1
def somme(L):
    x = L[0]
    for i in range(1,len(L)):
        x += L[i]
    return x

print(somme([1,2,3]))
print(somme(["vive"," ","les"," ","nouilles"]))

#%%
#Q2

def moyenne(L):
    return somme(L)/len(L)

print(moyenne([1,2,3]))
print(moyenne([1,15,68,61]))

#%%
#Q3
def maximum(L):
    m = L[0]
    for i in L:
        if i > m :
            m = i
    return m

print(maximum([1,2,9,8,2,3,6]))

#%%
#Q4
def minimum(L):
    m = L[0]
    for i in L:
        if i < m :
            m = i
    return m

print(minimum([1,2,0,8,2,3,6]))

#%%
#Q5
def indice_maximum(L):
    m = L[0]; im= 0
    for i in range(len(L)):
        if L[i] > m :
            m = L[i]
            im = i
    return im
```

```python
#Q6
def indice_maximum(L):
    m = L[0]; im= 0
    for i in range(len(L)):
        if L[i] > m :
            m = L[i]
            im = i
    return im

print(indice_maximum([1,2,9,8,2,3,6]))


#%%
#Q7

def second_maximum(L):
    L1 = [L[0], L[1]]
    m1 = maximum(L1); m2 = minimum(L1)
    for i in L :
        if i > m1 :
            m2 = m1
            m1 = i
        elif i > m2 :
            m2 = i
    return m2

print(second_maximum([1,2,9,8,2,3,6]))



#%%
#Q8

#Version 1
def separe_liste(L):
    n = len(L)//2
    return L[:n],L[n:]

print(separe_liste([1,2,3,4,5]))

#Version 2
def separe_liste(L):
    n = (len(L)+1)//2
    return L[:n],L[n:]

print(separe_liste([1,2,3,4,5]))



#%%
#Q9

def indice(L,x):
    l = []
    for i in range(len(L)):
        if L[i] == x:
            l += [i]
    return l

print(indice([1,2,1,2,1],1))
```

```
#%%
#Q10
def supprime_doublons(L):
    L1 = []
    for i in L :
        if i not in L1:
            L1.append(i)
    return L1

print( supprime_doublons([1,2,3,5,4,1,5,6,2,4,1,2,1]) )

#%%
#Q11
def envers(L):
    L1 = []
    for i in L :
        L1 = [i] + L1
    return L1

print(envers([1,2,3,4,5]))

#%%
#Q12

def alterne(L1,L2):
    L = []
    for i in range(len(L1)):
        L += [L1[i],L2[i]]

    return L

L1 = [1,1,1]; L2 = [2,3,4]
print(alterne(L1,L2))

#%%

#Q13
def taille(A):
    return len(A), len(A[1])

#%%
#Q14
def affiche(A):
    n,p = taille(A)
    for i in range(n):
        s = ""
        for j in range(p):
            s += str(A[i][j]) + " "
        print(s)
    print("")
```

```
#Meilleure version pour prendre en compte la taille des nombres dans l'
    affichage

def affiche(A):
    n,p = taille(A)
#La liste L va contenir le nombre maximal de caracteres des nombres dans
    chaque colonne
    L = []
    for j in range(p):
        m = 0
        for i in range(n):
            k = len( str(A[i][j]))
            if k > m:
                m = k
        L.append(m)

    for i in range(n):
        s = ""
        for j in range(p):
            k = len(str(A[i][j]))
            s += (L[j]-k+1)*" " + str(A[i][j])
        print(s)
    print("")

#%%
#Q15
def zeros(n,p):
    C = []
    for i in range(n):
        L=[]
        for j in range(p):
            L.append(0)
        C.append(L)
    return C
 #%%
#Q16
def somme(A,B):
    if taille(A) != taille(B) :
        print("erreur : somme impossible")
        return False
    n,p = taille(A)
    C = zeros(n,p)
    for i in range(n):
        for j in range(p):
            C[i][j] = A[i][j] + B[i][j]
    return C
#%%
#Q17
def identite(n):
    C = []
    for i in range(n):
        L=[]
        for j in range(n):
            if i == j :
                L.append(1)
            else :
                L.append(0)
        C.append(L)
    return C
```

```
affiche(identite(3))

#%%

#Q18
def transposee(A):
    n,p = taille(A)
    C = zeros(p,n)
    for i in range(p):
        for j in range(n):
            C[i][j] += A[j][i]
    return C

#%%
#BONUS : trace d'une matrice
def trace(A):
    n,p = taille(A)
    s=0
    for i in range(n):
        s += A[i][i]
    return s

#Q19
def produit(A,B):
    n,p = taille(A)
    m,q = taille(B)

    if p != m :
        print("erreur : produit impossible")
        return False

    C = zeros(n,q)
    for i in range(n):
        for j in range(q):
            for k in range (p):
                C[i][j] += A[i][k]*B[k][j]
    return C
#%%
#Q20
def puissance(M,n):
    if n == 0:
        return identite(len(M))
    return produit(puissance(M,n-1),M)
```

```
#%%
#Q21

#1ere methode d'un coup :

def bloc(A,B,C,D):
    na, pa = taille(A)
    nb, pb = taille(B)
    nc, pc = taille(C)
    nd, pd = taille(D)

    if na + nc != nb + nd or pa + pb != pc + pd :
        print("dimension des blocs incompatible")
        return False
    N = na+nc ; P = pa+pb
    M = zeros(N,P)

    for i in range(N):
        for j in range(P):

            if i < na and j < pa :
                M[i][j] = A[i][j]
            elif i < na :
                M[i][j] = B[i][j-pa]
            elif j < pa :
                M[i][j] = C[i-na][j]
            else :
                M[i][j] = D[i-na][j-pa]
    return M


#2eme methode plus constructive :

# construit la matrice (A B)
def bloc_ligne(A,B):
    na, pa = taille(A)
    nb, pb = taille(B)
    if na != nb  :
        print("dimension des blocs incompatible")
        return False
    N = na ; P = pa+pb
    M = zeros(N,P)

    for i in range(N):
        for j in range(P):
            if  j < pa :
                M[i][j] = A[i][j]
            elif i < na :
                M[i][j] = B[i][j-pa]
    return M

# construit la matrice :
#( A )
#( B )
def bloc_colonne(A,B):
    return transposee(bloc_ligne(transposee(A),transposee(B)))

def bloc(A,B,C,D):
    return bloc_colonne(bloc_ligne(A,B), bloc_ligne(C,D))
```

```
M = [[1,2,3],[4,5,6],[7,8,9]]

O = zeros(3,2)
A = [[1,2],[3,4],[5,6]]
I = identite(3)
affiche(M); affiche(O); affiche(I); affiche(A)

M1 = bloc(M,O,I,A)
print(taille(M1))
affiche(M1)

#%%
#Q22
def anti_identite(n):
    M = zeros(n,n)
    for i in range(n):
        M[i][n-i-1] = 1
    return M

affiche(anti_identite(4))


def matrice1(n):
    M = zeros(n,n)
    if n%2 == 1:
        M[n//2][n//2] -= 1
    return somme(somme(M, identite(n)),anti_identite(n))

affiche(matrice1(11))

def matrice2(n):
    M = zeros(n,n)
    for i in range(n):
        for j in range(n):
            if i <= j:
                M[i][j] = 1
    return M

affiche(matrice2(11))

def matrice3(n):
    M = zeros(n,n)
    for i in range(n):
        for j in range(n):
            if (i+j) % 2 == 1:
                M[i][j] = 1
    return M

affiche(matrice3(11))
```

```
#%%
#Q23
def matrice4(n):
    M = zeros(n,n)
    for i in range(n):
        for j in range(n):
            if i % 2 == 0:
                M[i][j] = i*n+j+1
            else :
                M[i][j] = i*n+(n-j)
    return M


affiche(matrice4(7))
```