```
# ============================== #
# MP                             #
# DS d'informatique du 2/12/2017 #
# Corrigé de l'exercice 2        #
# ============================== #


## Importations

import numpy as np

## Question 1

def TriMat_gen(n):
    A = np.zeros((n,n))
    c = 1
    for i in range(n):
        for j in range(i,n):
            A[i,j] = c
            c += 1
    return(A)

## Question 2

def TriMat_test(T):
    (n,p) = T.shape
    # Le tableau T ne correspond pas à une matrice carrée
    if n != p:
        return(False)
    # Le tableau T correspond à une matrice carrée
    else:
        Sup, Inf = True, True
        for i in range(n):
            for j in range(n):
                coeff = (T[i,j] != 0)
                if coeff and i > j:
                    Sup = False
                if coeff and i < j:
                    Inf = False
                if not Sup and not Inf:
                    return(False)
        # Si on arrive ici, c'est que le tableau T correspond à une matrice
qui
        # est soit triangulaire inférieure, soit triangulaire supérieure.
        # Notons que si la matrice est diagonale (cf. la question 6), la
        # condition du "if" sera satisfaite et notre code retournera
"TriSup"
        if Sup:
            return("TriSup")
        else:
            return("TriInf")

## Question 3

def TriMat_det(T):
    n = T.shape[0]
    d = 1
    for i in range(n):
        d *= T[i,i]
    return(d)
```

```
## Question 4

def TriMat_transpose(T):
    n = T.shape[0]
    TT = np.zeros((n,n))
    # Le tableau T correspond à une matrice triangulaire supérieure
    if TriMat_test(T) == "TriSup":
        for i in range(n):
            for j in range(i,n):
                TT[j,i] = T[i,j]
    # Le tableau T correspond à une matrice triangulaire inférieure
    else:
        for i in range(n):
            for j in range(i+1):
                TT[j,i] = T[i,j]
    # Renvoi du résultat
    return(TT)

## Question 5

# Le cas d'une matrice triangulaire supérieure inversible nous est familmier
# et correspond à l'étape de remontée de l'lgorithme de Gauss pour la
résolution
# des systèmes linéaires...

def TriMat_inverse(T):
    # Le tableau T ne correspond pas à une matrice inversible
    if TriMat_det(T) == 0:
        return(False)
    # Le tableau T correspond à une matrice triangulaire supérieure
inversible
    elif TriMat_test(T) == "TriSup":
        n = T.shape[0]
        Idn = np.identity(n)
        Tinv = np.zeros((n,n))
        Tinv[n-1] = Idn[n-1] / T[n-1,n-1]
        for i in range(n-2,-1,-1):
            S = np.zeros((1,n))
            for j in range(i+1,n):
                S += T[i,j]*Tinv[j]
            Tinv[i] = (Idn[i] - S) / T[i,i]
        return(Tinv)
    # Le tableau T correspond à une matrice triangulaire inférieure
inversible
    # On renvoie la transposée de l'inverse de la transposée...
    else:
        return(TriMat_transpose(TriMat_inverse(TriMat_transpose(T))))

## Question 6

def Diag_exp(D,epsilon):
    if TriMat_test(D) != "TriSup" or TriMat_test(TriMat_transpose(D)) !=
"TriSup":
        raise ValueError("La matrice n'est pas diagonale !")
    else:
        from math import floor, log
        E = 2.718281828459045
        n = floor(log(E/epsilon)-1) + 1
        dim = D.shape[0]
```

```
        ExpD = np.zeros((dim,dim))
        x = 2**n
        for i in range(dim):
            ExpD[i,i] = (1 + D[i,i] / x)**x
        return(ExpD)

## Pour tester...

# A est triangulaire supérieure
A = TriMat_gen(4)
print("La matrice A :\n",A)
print("Le déterminant de A vaut :",TriMat_det(A))
Ainv = TriMat_inverse(A)
print("La matrice inverse de A :\n",Ainv)
print("Vérification. Produit de A et de son inverse :\n",np.dot(A,Ainv))
# B est triangulaire inférieure
B = TriMat_transpose(A)
print(B)
Binv = TriMat_inverse(B)
print(Binv)
print(np.dot(B,Binv))
# D1 est diagonale
D1 = np.array([[0.2,0,0],[0,0.35,0],[0,0,0.75]])
ED1 = Diag_exp(D1,0.0001)
print(ED1)
n = D1.shape[0]
for i in range(n):
    print(np.exp(D1[i,i])/ED1[i,i] - 1)
```