```python
# ========================= #
# E3A / Exercices types 2015 #
# Exercice 2                 #
# ========================= #

import numpy as np

# Fonctions voisins donnant LA LISTE DES VOISINS d'un sommet donné dont le
# numéro est passé en argument.
def voisins(i):
    global M
    n = M.shape[0]
    L = []
    for j in range(n):
        if M[i,j] != -1 and M[i,j] != 0:
            L.append(j)
    return(L)

# Fonctions degre donnant LE NOMBRE DE VOISINS d'un sommet donné dont le
# numéro est passé en argument.
def degre(i):
    global M
    n = M.shape[0]
    N = 0
    for j in range(n):
        if M[i,j] != -1 and M[i,j] != 0:
            N += 1
    return(N)

# Fonction longueur donnant la longueur d'une chaîne du graphe considéré.
# Si la chaîne est impossible (deux sommets consècutifs présents dans la liste
# ne sont pas adjacents), la fonction renvoie -1.
def longueur(L):
    global M
    longueur = 0
    chaine = True
    i = 0
    while chaine == True and i <= len(L)-2:
        p = M[L[i],L[i+1]]
        if p != -1:
            longueur += p
        else:
            chaine = False
        i += 1
    if chaine:
        return(longueur)
    else:
        return(-1)

# SCRIPT pour les tests
# =====================

# La matrice des distances
M = np.array([[0,9,3,-1,7],[9,0,1,8,-1],[3,1,0,4,2],[-1,8,4,0,-1],[7,-1,2,-1,0]])
print(M)
# Une chaîne possible
L_1 = [4,2,0,1,3,2,4]
long_1 = longueur(L_1)
if long_1 != -1:
    print("La longueur de la chaîne",L_1,"vaut",long_1)
else:
    print("La chaîne",L_1,"n'est pas valide !")
# Un chaîne impossible
L_2 = [4,2,0,3,1,2,4]
long_2 = longueur(L_2)
if long_2 != -1:
    print("La longueur de la chaîne",L_2,"vaut",long_1)
```

```
else:
    print("La chaîne",L_2,"n'est pas valide !")
```