```
# 2023_16_01_robot.py

01|
02|
03| #%%Robot
04| #"Bottom up" : méthode ascendante
05|
06| def dico_asc(U):
07|     #Dictionnaire pour stocker les valeurs m(i,j) des
sommes maximales des utilités pour aller à la case (i,j)
08|     #c'est une fonction d ela matrice de utilités
09|     p=len(U)
10|     q=len(U[0])
11|     m={}
12|     m[(0,0)]=U[0][0]
13|     #On remplie les utilités maximales pour accéder
aux cases de la première ligne
14|     for j in range(1,q):
15|         m[(0,j)]=m[(0,j-1)]+U[0][j]
16|     #valeur du dictionnaire associée à la clé (0,j)
17|     for i in range(1,p):
18|         m[(i,0)]=m[(i-1,0)]+U[i][0]
19|     for i in range(1,p):
20|         for j in range(1,q):
21|             m[(i,j)]=max(m[(i-1,j)],m[(i,j-1)])+U[i ]
[j]
22|     return(m)
23|
24| #Méthode descendante, ou mémoïsation. On va construire
récurssivement le dictionnaire m mais en utilisant ce qui
a déjà été rempli.
25| def dico_desc(U):
26|     p=len(U)
27|     q=len(U[0])
28|     m={}
29|     def aux(i,j):
30|     #fonction auxiliare récurssive pour remplir le
dictionnaire
31|     # if m[(i,j)]!=None:
32|     # Ne marche pas si m n'est pas déjà complet :
m[(i,j)] donne la valeur de la clé (i,j). Si elle n'existe
pas il y a erreur
33|         if m.get((i,j))!=None :
34|   # m.get((i,j)) donne la valeur de la clé (i,j) si
elle existe et None si elle n'existe pas.
35|             return m[(i,j)]
```

```
36|         elif i==0 and j==0:
37|             m[(0,0)]=U[0][0]
38|             return m[(i,j)]
39|         elif i==0:
40|             m[(0,j)]=aux(0,j-1)+U[0][j]
41|             return m[(i,j)]
42|         elif j==0:
43|             m[(i,0)]=aux(i-1,0)+U[i][0]
44|             return m[(i,j)]
45|         else :
46|             m[(i,j)]=max(aux(i-1,j),aux(i,j-1))+U[i]
[j]
47|             return m[(i,j)]
48|     aux(p-1,q-1)
49|     return m
50|
51| #Reconstruction d'un chemin optimal jusqu'à la cas
(i,j) à partir du dico.
52|
53| def chemin(m,i,j):
54|     if i==0 and j==0 :
55|         return [(0,0)]
56|     elif i==0:
57|         return chemin(m,0,j-1)+[(i,j)]
58|     elif j==0:
59|         return chemin(m,i-1,0)+[(i,j)]
60|     elif m[(i,j-1)]>m[(i-1,j)] :
61|         return chemin(m,i,j-1)+[(i,j)]
62|     else :
63|         return chemin(m,i-1,j)+[(i,j)]
64|
65| def chemin_optimal(U):
66|     p=len(U)
67|     q=len(U[0])
68|     m=dico_asc(U)
69|     return chemin(m,p-1,q-1)
70|
71| U2=[[0,0,0,0,0,0,0,1,0,9],[2,0,0,0,1,0,8,0,0,7],
[2,0,0,0,2,0,0,0,0,0],[0,8,0,0,0,2,0,1,0,0],
[1,0,0,0,0,6,0,0,0,0]]
72|
73|
74|
75|
76|
```