Exercice 2

1. 1. Pour aller de la case (0,0) à la case (2,3) on fait 3 déplacements vers la droite et [1] vers le bas.

   2. Comme on fait des déplacements de 1 pas à chaque étape, il faut faire 2 + 3 = 5 déplacements. Chaque déplacement nous amène sur une nouvelle case. En n'oubliant pas d'inclure la case (0,0) il faut donc parcourir 2 + 3 + 1 = 6 cases.

2. On liste tous les chemins et les sommes associées :

Chemin	Somme
(0,0) → (0,1) → (0,2) → (0,3) → (1,3) → (2,3)	11
(0,0) → (0,1) → (0,2) → (1,2) → (1,3) → (2,3)	10
(0,0) → (0,1) → (0,2) → (1,2) → (2,2) → (2,3)	14
(0,0) → (0,1) → (1,1) → (1,2) → (1,3) → (2,3)	9
(0,0) → (0,1) → (1,1) → (1,2) → (2,2) → (2,3)	13
(0,0) → (0,1) → (1,1) → (2,1) → (2,2) → (2,3)	12
(0,0) → (1,0) → (1,1) → (1,2) → (1,3) → (2,3)	10
(0,0) → (1,0) → (1,1) → (1,2) → (2,2) → (2,3)	14
(0,0) → (1,0) → (1,1) → (2,1) → (2,2) → (2,3)	13
(0,0) → (1,0) → (2,0) → (2,1) → (2,2) → (2,3)	16

La somme maximale est donc de 16.

3. 1. Le tableau `T'` est le suivant :

4	5	6	9
6	6	8	10
9	10	15	16

   2. La valeur `T'[0][j]` où j est non nul correspond à la somme des cases `(0,0)` à `(0,j)`, c'est à dire des cases de la première ligne du tableau.

   Il n'y a qu'un seul chemin qui corresponde à cette somme et il passe obligatoirement par la case à gauche (d'indice j-1) de la case `(0,j)`.

   Donc pour calculer la somme T'[0][j] on ajoute simplement la valeur de la case `(0,j)` (c'est à dire `T[0][j]`) à la somme obtenue à la case précédente (c'est à dire `T'[0][j-1]`).
   On a donc bien `T'[0][j] = T[0][j]+T'[0][j-1]`.

4. Si i et j son non-nuls, il y a deux chemins amenant à la case `(i,j)`. Le premier provient de la case du dessus `(i - 1,j)`, le second de la case de gauche `(i,j - 1)`.

   La valeur de `T'[i][j]` s'obtient donc en ajoutant la valeur de `T[i][j]` au maximum des deux chemins menant à cette case : `max(T'[i-1][j],T'[i][j-1])`.

5. 1. Le cas de base est atteint lorsque l'on atteint une case de la première ligne (i vaut 0) ou de la première colonne (j vaut 0). Dans ce cas on calcule la somme en additionnant la valeur de la case en question avec le résultat de `somme_max` avec comme argument `T` et la case précédente (sur la ligne si `i=0` ou la colonne si `j=0`).

2. On a :

```
def somme_max(T, i,
j): if i == 0 and j
== 0: return T[0][0]
elif i == 0 :
    return T[0][j] + somme_max(T,0,j-
1) elif j == 0 :
    return T[i][0] + somme_max(T,i-
1,0) else :
    return T[i][j] + max(somme_max(T,i-1,j), somme_max(T,i,j-1))
```

3. On appelle `somme_max(T,2,3)`.