Correction du DS 2

Informatique de tronc commun, première année

Julien Reichert

Exercice 1

```
def somme_totale(L):
    rep = 0
    for sousliste in L:
        for element in sousliste:
            rep += element
    return rep
```

Exercice 2

```
def reordonne(l, i):
    l2, l3 = [], []
    for indice in range(len(l)):
        if l[indice][0] == l[i][0] and indice != i:
            l2.append(l[indice])
        elif indice != i:
            l3.append(l[indice])
    return [l[i]] + l2 + l3
```

Exercice 3

```
def imprime_forme(n):
    if n == 1:
        print("*")
    else:
        print("*"*n)
        imprime_forme(n-1)
        print("*"*n)
```

Exercice 4

```
def est_present(l, x):
    ind_deb, ind_fin = 0, len(l)-1
    while ind_deb <= ind_fin:
        ind_mil = (ind_deb + ind_fin) // 2
        if l[ind_mil] == x:
            return True
        elif l[ind_mil] < x:
            ind_deb = ind_mil + 1
        else:
            ind_fin = ind_mil - 1
    return False
```

Exercice 5

On fait comme dans le TP 4 une dichotomie pour trouver le premier indice où se trouve l'élément, idem pour le dernier indice, et on en déduit le nombre d'indices dans l'intervalle. La fonction suivante trouve à la fois le premier et le dernier indice, pour éviter le copier-coller. Et donc, pour l'exercice 4 avec cette fonction, on détermine si le résultat est 0.

```
def nombre_occurrences_dicho(liste, element):
    n = len(liste)
    ind_min, ind_max = 0, n-1
    while ind_min < ind_max:
        ind_mil = (ind_min + ind_max) // 2
        if liste[ind_mil] > element:
            ind_max = ind_mil-1
        elif liste[ind_mil] < element:
            ind_min = ind_mil+1
        else:
            ind_g, ind_d = ind_mil, ind_mil
            while ind_min < ind_g:
                ind_mil_g = (ind_min + ind_g) // 2
                if liste[ind_mil_g] < element:
                    ind_min = ind_mil_g+1
                else:
                    ind_g = ind_mil_g-1
            while ind_max > ind_d:
                ind_mil_d = (ind_max + ind_d) // 2
                if liste[ind_mil_d] > element:
                    ind_max = ind_mil_d-1
                else:
                    ind_d = ind_mil_d+1
            if liste[ind_min] < element:
                ind_min += 1
            if liste[ind_max] > element:
                ind_max -= 1
            return ind_max - ind_min + 1
    return 0
```

Exercice 6

En cas d'égalité, on fait comme on veut.

```
def somme_gloutonne(L):
    s = L[0][0]
    i = 0
    j = 0
    nl = len(L)
    nc = len(L[0])
    while i < nl-1 or j < nc-1:
        if j == nc-1:
            i += 1
        elif i == nl-1 or L[i][j+1] > L[i+1][j]:
            j += 1
        else:
            i += 1 # surprenant, mais rassembler les cas serait moche
        s += L[i][j]
    return s
```

Exercice 7

La somme ainsi calculée n'est pas optimale, par exemple pour la liste `[[0, 0, 5], [2, 0, 0]]` l'algorithme va commencer par descendre et trouver une somme finale de `2` alors qu'il est possible d'avoir `5`.

Exercice 8

Cette fonction détermine dans une liste à quels indices figure un élément qui est strictement supérieur à tous ses prédecesseurs.