Les tableaux

Création et manipulation de tableaux

En informatique on appelle un **tableau** une structure permettant de stocker un nombre donné de valeurs. Chaque valeur est accessible directement à partir de son **indice** (ou **index** en anglais) allant de 0 à $n-1$, où n est la taille du tableau. En Python, les tableaux sont représentés à l'aide de listes. Quand on parle de tableau, on n'utilise pas d'opération modifiant sa taille, comme `append`. Par contre, on pourra modifier les valeurs des éléments. Les opérations disponibles sont :

```
>>> tab1 = [4, 2, 1, 7, 2]       # définition explicite
>>> tab2 = [0]*5                 # création d'un tableau de taille 5 avec des 0
>>> len(tab1)                    # taille du tableau
5
>>> tab2[3] = tab1[2]            # modification et accès aux valeurs
>>> tab2
[0, 0, 0, 1, 0]
>>> 6 in tab1                    # test d'appartenance
False
>>> [2*i for i in range(2, 7)] # création par compréhension
[4, 6, 8, 10, 12]
```

Il est possible de parcourir un tableau par indice, par valeur ou les deux :

```
for i in range(len(tab)):
    print(tab[i])
```

```
for v in tab:
    print(v)
```

```
for i, v in enumerate(tab):
    print(tab[i], "=", v)
```

L'utilisation d'un tableau est très proche de celle d'un n-uplet. La différence c'est qu'il est possible de modifier les valeurs d'un tableau.
Il faut néanmoins faire attention au fait qu'un tableau est un objet mutable. Lorsqu'il est passé en paramètre à une fonction, toute modification du tableau dans la fonction perdurera même après l'exécution de la fonction.
En pratique si les valeurs ne doivent pas être modifiées, mieux vaut utiliser un n-uplet puisque lors de l'exécution d'un programme, la création et l'utilisation est plus rapide.

Tableaux de tableaux

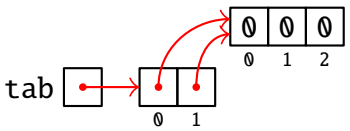
Les tableaux peuvent tout à fait contenir d'autres tableaux : `t2 = [[1, 2], [3, 4, 5]]`. Les éléments `t2[0]` et `t2[1]` sont des tableaux et on peut donc accéder directement aux valeurs avec l'expression `t2[i][j]`, où `j` est l'indice de l'élément dans le tableau d'indice `i`.

Exercice 1 : On considère le tableau `t2` ci-dessus.

1) Que renvoient `t2[0][1]` et `t2[1][0]` ?
2) Quelle expression renvoie la valeur `5` ?

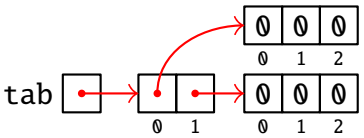
La plupart du temps, on utilise des tableaux contenant uniquement des tableaux de même taille. Cela permet de représenter des matrices, les pixels d'une image, les murs d'un labyrinthe, les case d'une grille de jeu… On dit qu'un tableau est de taille $m \times n$ lorsqu'il est composé de m tableaux de n éléments chacun. Pour créer un tableau de taille $m \times n$ on peut être tenté de faire `[[0]*n]*m`. Mais ce n'est pas une bonne idée.

```
>>> tab = [[0]*3]*2
>>> tab
[[0, 0, 0], [0, 0, 0]]
>>> tab[0][0] = 1
>>> tab
[[1, 0, 0], [1, 0, 0]]
```



On remarque que les deux sous-tableaux ont été modifiés. C'est parce que `[objet]*n` recopie `n` fois l'objet. Dans le cas d'un tableau (ou d'une liste), c'est le pointeur qui est dupliqué, pas les données. Pour remédier à ce problème, il faut recréer un sous-tableau à chaque fois. On peut utiliser la commande `[[0]*n for i in range(m)]`.

```
>>> tab = [[0]*3 for i in range(2)]
>>> tab[0][0] = 1
>>> tab
[[1, 0, 0], [0, 0, 0]]
```



On peut voir que les deux sous-tableaux sont bien différents. C'est parce qu'un nouveau tableau est créé à chaque itération, au lieu de copier le même à chaque fois.

Parcours d'un tableau de tableaux

Pour parcourir un tableau de tableaux, il faut utiliser des boucles imbriquées. En général on parcourt les lignes une par une et chacune des valeurs de chaque ligne. Mais on peut aussi parcourir les colonnes puis chaque élément de chaque colonne. On suppose que `tab` est un tableau de taille $m \times n$. On peut faire les parcours suivants :

```
for i in range(m):
    for j in range(n):
        print(tab[i][j])
```

```
for j in range(n):
    for i in range(m):
        print(tab[i][j])
```

```
for t in tab:
    for e in t:
        print(e)
```

Exercice 2 : On considère `tab = [[1, 2], [3, 4]]`. Que vont afficher chacune des boucles ?

Passage de 2 à 1 dimension

Dans un langage ne permettant pas de faire des tableaux de tableaux, ou à cause d'autres contraintes, il est tout à fait possible de n'utiliser qu'un tableau à une dimension où les lignes seraient les une après les autres. Ainsi les tableaux `[[1, 2], [3, 4]]` et `[1, 2, 3, 4]` représentent les mêmes données.

Exercice 3 : On considère le tableau `tab2` de taille 3×4 représenté ci-contre. On souhaite déterminer le tableau `tab1` de dimension 1 représentant les mêmes données.

1) Quelle est la taille de `tab1` ?
2) Quel est l'indice de 6 dans `tab1` ?
3) Réciproquement, quelle sera la valeur de l'élément d'indice 9 dans `tab1` ?

```
    0     1     2     3
0 [[4,    7,   10,    9],
1  [1,   -3,    6,    5],
2  [8,    0,    4,    2]]
```

Exercice 4 : On considère un tableau `tab2` de taille $m \times n$ et un autre tableau `tab1` de dimension 1 représentant les mêmes données.

1) Quelle est la taille de `tab1` ?
2) Quel est l'indice dans `tab1` correspondant à la valeur `tab2[i][j]` ?
3) Réciproquement, quelles valeurs prendre pour `i` et `j` pour que `tab2[i][j]` corresponde à `tab1[k]` ?