Tableaux à une dimension

Ce chapitre a pour objectif de présenter les tableaux à une dimension ainsi que leur utilisation dans l'environnement Python.

I. Principe

Un tableau a pour rôle de regrouper un ensemble de données au sein d'une même entité afin d'en simplifier l'exploitation.

Supposons, par exemple, que l'on souhaite calculer la moyenne des notes d'une classe pour un DS donné.

Une première méthode pour effectuer cette tâche, serait de créer n variables (n est le nombre des élèves de la classe), de les saisir et de faire le calcul en sommant les notes et en divisant par n. Ainsi, avec Python, le code pourrait être le suivant:

```
note1=float(input("Note 1:"))
..........
noten=float(input("Note n:"))
moyenne=(note1+note2+....+noten)/n
print('La moyenne est de ',moyenne)
```

Il est aisé de voir que la solution présente plusieurs défauts:

- Si la classe est importante (comme dans certaines formations post-bac), le nombre de variables devient prohibitif,
- Si le nombre d'élèves varie, il faut modifier le programme en conséquence (ajouter ou supprimer des variables et réécrire la ligne de calcul de la moyenne),
- L'écriture du calcul de la moyenne est laborieuse.

Afin de résoudre ces difficultés, la plupart des langages informatiques intègrent des tableaux. Le principe est de "grouper" un ensemble de données dans une même entité. Chaque élément peut être atteint à l'aide de son numéro que l'on appelle indice. On peut se représenter un tableau de cette façon:

Données	12	16	4	13	9
Indice	0	1	2	3	4

Dans le cas de l'exemple ci-dessus, le tableau se compose de 5 éléments qui sont des entiers. Chaque valeur est accessible par son indice (compris entre 0 et 4). Admettons que l'on souhaite modifier le 4 en 5,5 et que le nom du tableau soit notes, il suffirait d'écrire notes[2]=5,5. Comme on peut le voir, il est très simple de manipuler une valeur particulière du tableau à l'aide de son indice.

A l'aide d'un tableau, nous voyons qu'il serait simple de résoudre notre problème de calcul de moyenne.

Ainsi, en pseudo-code, la solution serait la suivante:

```
Programme Calcul_Moyenne
Variables
      n, i              : entiers
      moyenne, total    : flottants
      notes             : tableau[0..99] de flottants
Début
      Afficher ("Veuillez saisir le nombre de notes :")
      Saisir (n)
      Pour i allant de 0 à n-1 faire
            Saisir(notes[i])
      Fin pour
      total=0
      Pour i allant de 0 à n-1 faire
            total=total+notes[i]
      Fin pour
      moyenne=total/n
      Afficher ("La moyenne est de", moyenne)
Fin Programme
```

Cette solution répond aux difficultés mentionnées précédemment:

- Le programme peut gérer un nombre important d'élèves à l'aide d'une seule variable sans que cela n'impacte son écriture,
- Le nombre de notes effectives est directement saisi par l'utilisateur sans modification du code.
- Le calcul de la moyenne est fortement simplifié à l'aide d'une boucle qui calcule le total.

II. Implémentation de tableaux avec Python

Le langage Python permet d'implémenter des tableaux à l'aide de listes. Ces listes sont très puissantes et permettent des stockages complexes mais, dans le cadre de ce chapitre, elles seront uniquement utilisées pour stocker des tableaux à une dimension et homogènes (toutes les cases contiennent le même type de données).
Une façon très simple de créer et stocker un tableau est d'utiliser l'instruction suivante:

nom_tableau=[donnée 1, donnée 2, …. , donnée n]

Prenons comme exemple un tableau de 6 éléments contenant des entiers:

```
>>>tab=[3, 5, 3, 9, 12, 17]
>>>tab
[3, 5, 3, 9, 12, 17]
```

La longueur d'une liste peut être obtenue à l'aide de l'instruction len. Ainsi, on obtiendra le résultat suivant:

```
>>>len(tab)
6
```

S'il l'on souhaite restituer ou modifier une valeur, on peut utiliser son indice (compris entre 0 et len(tab)-1). Dans le cas de notre exemple, l'indice est compris entre 0 et 5 (inclus). Il suffit d'indiquer le nom du tableau suivi de l'indice entouré d'une paire de crochets.

Ainsi, les manipulations suivantes sont possibles:

```
>>>tab[3]                    Simple affichage
9
>>>tab[6]
      ERREUR !!!
>>>tab[3]=4                  Modification de la donnée d'indice 3
>>>tab
[3, 5, 3, 4, 12, 17]         Le tableau se trouve impacté
```

III. Ajout / Suppression de données dans un tableau

Un programme informatique est souvent amené à ajouter ou supprimer des éléments d'un tableau. L'ajout d'un élément peut se faire très simplement à l'aide d'une concaténation:

nom_tableau=nom_tableau+liste additionnelle.

Par exemple, en utilisant le tableau précédemment défini, on peut effectuer l'opération suivante:

```
>>>tab=tab+[39, 22]
>>>tab
[3, 5, 3, 4, 12, 17, 39, 22]
```

Il est aussi possible d'utiliser l'opérateur append. La syntaxe est la suivante:

nom_tableau.append(nouvelle_valeur).

Ainsi, il est possible d'ajouter 49 au tableau précédent en utilisant l'instruction suivante:

```
>>>tab.append(49)
>>>tab
[3, 5, 3, 4, 12, 17, 39, 22, 49]
```

La suppression d'un élément du tableau est très simple: il suffit d'exploiter l'opérateur del. La syntaxe est la suivante:

del nom_tableau[début:fin+1]

Par exemple, si on souhaite supprimer le sixième élément (qui porte le numéro 5 puisque l'on commence à 0), on saisie l'instruction suivante:

```
>>>del tab[5:6]
>>>tab
[3, 5, 3, 4, 12, 39, 22, 49]
```

Pour supprimer les 4 derniers éléments (numéro 4 à 7), on saisit l'instruction suivante:

```
>>>del tab[4:8]
>>>tab
[3, 5, 3, 4]
```

IV. Copie de tableaux

Cette section a pour objectif de vous mettre en garde contre une erreur fréquente et difficile à diagnostiquer.

Lorsque l'on exploite un nombre entier et que l'on souhaite le copier, la séquence suivante est parfaitement fonctionnelle:

```
>>>a=5                        Initialisation
>>>b=a                        Copie
>>>a=a+1                      Incrémentation de a
>>>a
6                             a a été modifiée
>>>b
5                             b n'a pas changé
```

Par contre, la séquence suivante pose problème:

```
>>>tab=[3, 4, 5, 6]                Initialisation d'un tableau d'entier
>>>tab2=tab                       On tente la copie
>>>tab.append(7)                  On ajoute 7 au tableau de départ
>>>tab
[3, 4, 5, 6, 7]                         La donnée à bien été ajoutée
>>tab2
[3, 4, 5, 6, 7]                         tab2 a aussi été modifiée !
```

Cela tient au fait que les deux tableaux utilisent les mêmes emplacements mémoire. Toute modification sur l'un entraine la même modification sur l'autre. Nous verrons, en séance de TP, comment copier un tableau et éviter ces effets de bord entre l'original et la copie.