# Les listes et les tableaux

BENMANSOUR Asma

Avril 2018
1.0

# Table des matières

# Introduction

Le chapitre II a présenté les types de données simples, mais Python offre beaucoup plus : les conteneurs. De façon générale, un conteneur est un objet composite destiné à contenir d'autres objets. Dans ce chapitre nous allons étudier en détail les propriétés et les différents moyens (lecture, écriture et méthodes/fonctions) permettant la manipulation d'un type de donnée complexe appelé liste. Les listes sont aussi le moyen intégré en Python pour la définition de tableaux c'est-à-dire un cas particulier de liste ou les données stockés sont de même type. Les tableaux à une dimension (ex: vecteur) et les tableau à deux dimension seront tous les deux traités dans ce chapitre. *p.16*

# Objectifs spécifiques du chapitre

A l'issu de ce chapitre l'apprenant sera capable de :

- Manipuler (lire, écrire,..) des données de différents types en utilisant les listes. p.16
- Manipuler des données de même type en utilisant les tableaux
- Mettre en pratique les étapes permettant la Création d'un tableau à une dimension (vecteur) à savoir:

1. Lecture de la taille du vecteur
2. Initialisation du vecteur
3. Lecture de éléments du vecteur

- Mettre en pratique les étapes permettant la Création d'un tableau à deux dimensions (matrice)à savoir:

1. Lecture des dimensions : nombre de lignes et nombre de colonnes
2. Initialisation de la matrice
3. Lecture de éléments de la matrice

- Manipuler les tableaux à une et deux dimensions
- Mettre en pratique la boucle for pour parcourir un tableau existant afin de :

1. Ajouter des éléments
2. Supprimer des éléments
3. Afficher ses éléments

- Mémoriser et mettre en pratique quelques fonctions/méthodes sur les tableaux

# Les listes

II



## 1. Définition

Collection ordonnée et modifiable d'éléments éventuellement hétérogènes c'est-à-dire des éléments de différente nature ou différent type( entier, chaîne de caractère, réel)[3] p.16 .

### *Syntaxe*

Éléments séparés par des virgules, et entourés de crochets, si un élément appartenant à la liste est de type chaîne de caractère, il faut le mettre entre apostrophes[3]

### *Exemple : Exemples de listes*

- saison est une liste contenant que des chaînes de caractères.
- étudiant est une liste qui contient deux éléments de type chaine de caractère, un élément de type entier et un élément de type réel
- liste est une liste qui inclut deux élément de type <list>

```
>>> saisons=['hiver','printemps','été','automne']
>>> étudiant=['nom','prénom',18,1.70]
>>> liste=[saisons,étudiant]
>>> liste
[['hiver', 'printemps', 'été', 'automne'], ['nom', 'prénom', 18, 1.7]]

```

liste est une matrice de deux lignes et quatre colonnes incluant des éléments de différent type, l'indice de la première ligne est toujours à 0 et l'indice de la première colonne est toujours à 0. Pour accéder à un élément de liste : liste[indice de ligne][indice de colonne] comme suit :

```
>>> liste[0][3]
'automne'
>>> liste[1][2]
18
```

## 2. Création des listes

En Python les listes peuvent être crée en utilisant la syntaxe *list()* ou *[]* comme ceci:

```
>>> liste=list()
>>> type(liste)
<class 'list'>
>>> liste2=[]
>>> type(liste2)
<class 'list'>
>>> liste3,liste4=[],[3.14]*5
>>> print(liste3)
[]
>>> print(liste4)
[3.14, 3.14, 3.14, 3.14, 3.14]
>>> l1=list(range(4))
>>> print(l1)
[0, 1, 2, 3]
>>> l1=[range(4)]
>>> print(l1)
[range(0, 4)]
>>> l1=[i for i in range(4)]
>>> print(l1)
[0, 1, 2, 3]
```

### *Remarque : Utilisation équivalente de list ou []*

L'exemple ci dessus montre clairement que *l1=list(range(4))* et *l1=[i for i in range (4)]* sont équivalente la première utilise le mot clé list et la deuxième utilise []

## 3. Méthodes

Voici quelques méthodes qu'on peut appliquer sur les listes :

- nom_liste.append(nombre) permet d'ajouter un nombre en fin de liste
- nom_liste.sort() permet de trier les éléments de la liste dans l'ordre croissant
- nom_liste.reverse() affiche l'image miroir de la liste initiale c'est à dire le dernier élément devient le premier, l'avant dernier élément devient le second et ainsi de suite.
- nom_liste.remove(nombre) permet de supprimer l'élément nombre de la liste.
- nom_liste.index(nombre) permet d'afficher l'indice de l'élément nombre qui se trouve dans la liste sachant que l'indice commence à 0.
- nom_liste[indice1 :indice2] permet d'afficher les éléments de la liste en partant de l'élément à l'indice1 jusqu'à l'élément à l'indice2-1.
- nom_liste.pop() retourne le dernier élément de la liste.
- nom_liste.count(nombre) retourne le nombre d'occurrence de nombre dans la liste.

```
>>> liste_nombres = [17, 38, 10, 25, 72,100,2,4.15,3.4]
>>> print(liste_nombres)
[17, 38, 10, 25, 72, 100, 2, 4.15, 3.4]
>>> liste_nombres.append(200)
>>> print(liste_nombres)
[17, 38, 10, 25, 72, 100, 2, 4.15, 3.4, 200]
>>> liste_nombres.sort()
```

```
>>> print(liste_nombres)
[2, 3.4, 4.15, 10, 17, 25, 38, 72, 100, 200]
>>> liste_nombres.reverse()
>>> print(liste_nombres)
[200, 100, 72, 38, 25, 17, 10, 4.15, 3.4, 2]
>>> liste_nombres.remove(200)
>>> print(liste_nombres)
[100, 72, 38, 25, 17, 10, 4.15, 3.4, 2]
>>> print(liste_nombres.index(2))
8
>>> liste_nombres[0]
100
>>> liste_nombres[0:5]
[100, 72, 38, 25, 17]
>>> print(liste_nombres.pop())
2
>>> print(liste_nombres.count(100))
1
```

# Les tableaux

III



Imaginons que dans un programme, nous ayons besoin simultanément de 10 valeurs (par exemple, des notes pour calculer une moyenne). Évidemment, la seule solution dont nous disposons à l'heure actuelle consiste à déclarer dix variables, appelées par exemple N1, N2, N3, etc. Après une succession de dix instructions ‘lire' distinctes on aura:

Moyenne=(N1+N2+N3+N4+N5+N6+N7+N8+N9+N10)/10

Imaginons maintenant que nous ayons besoin de quelques centaines ou quelques milliers de

valeurs!!!!

Un tableau est une liste contenant des valeurs de même type. Le tableau est un ensemble de valeurs de même type portant le même identifiant et repérées par un nombre appelé indice. L'indice du premier élément du tableau est toujours égale à 0 et donc si le tableau contient N éléments l'indice du dernier élément du tableau est toujours égale à N-1.

Pour désigner un élément du tableau on utilise son identifiant suivi de l'indice entre crochet, par exemple :Note [i] représente la $i^{ème}$ valeur du tableau Note

Les tableaux les plus utilisés sont:

1. les tableaux à une dimension (exemple: vecteurs)
2. les tableaux à deux dimensions (exemple: matrices)

## 1. Tableau à une dimension

### *Éléments fondamentaux d'un tableau à une dimension*

Quatre éléments fondamentaux définissent un tableau à une dimension appelé aussi vecteur:

1. Son nom: qui sera un identifiant choisi en respectant les règles usuelles de dénomination des variables(voir chapitre II). En tenant compte de l'exemple ci-dessous le nom du tableau est vec.

2. Sa dimension (1 dimension ou 2 dimensions). En tenant compte de l'exemple ci-dessous la dimension du tableau est 1.
3. Sa taille et les valeurs de ses indices (indice minimum, indice maximum). En tenant compte de l'exemple ci-dessous la taille du tableau est N le plus petit indice est 0 et le plus grand est N-1.
4. Le type de données qu'il contient. En tenant compte de l'exemple ci-dessous, le tableau vec contient des entiers.

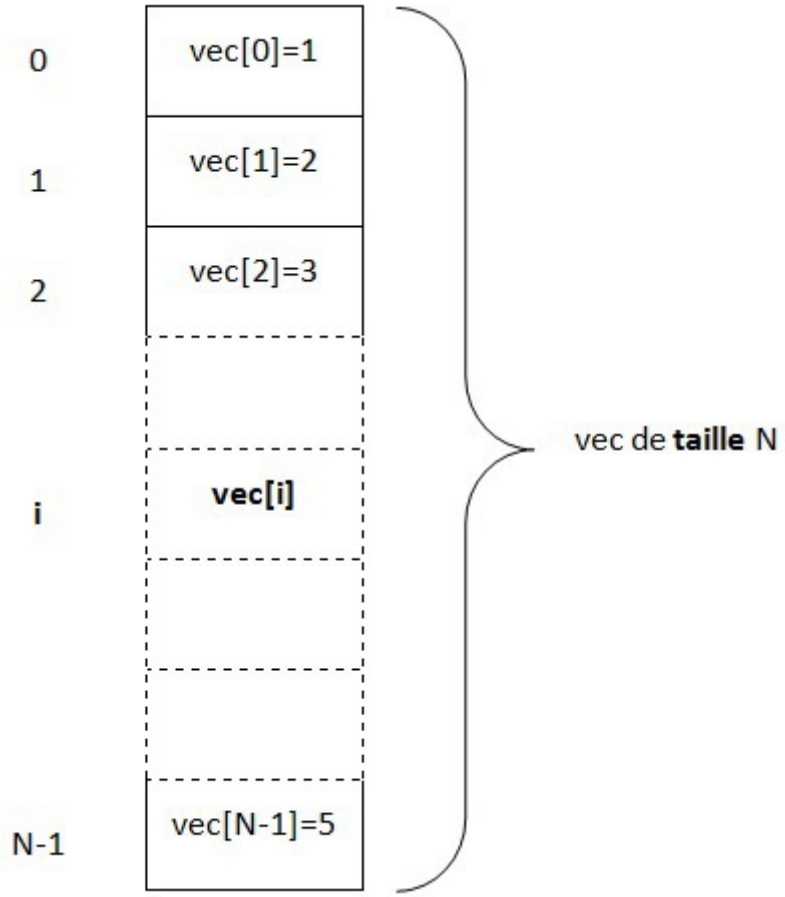


*Tableau à une dimension*

Les principales étapes permettant la création d'un tableau sont :

1. Lire la taille du tableau
2. initialiser le tableau en utilisant la syntaxe :
   [0 for i in range(0,taille)]
3. Lire les éléments d'un tableau pour désigner un élément du tableau on utilise la syntaxe :
   nom_tableau[indice]

### *Fondamental*

Nous ne pouvons pas lire les éléments d'un tableau sans l'avoir initialisé et nous ne pouvons pas initialiser un tableau sans connaître sa taille !

*Exemple : Lecture et écriture des éléments d'un tableau*

écrire le programme Python qui permet de saisir un tableau Vec de N nombres réel. Puis le programme doit afficher les éléments du tableau Vec ;

```
N = int(input('Entrer le nombre d'éléments N:'))
Vec = [0 for i in range(0,N)] #Initialisation des éléments du vecteur
print("Entrer éléments par élément les composantes du tableauVec(%d):"%N)
for i in range(0,N):
    print("Vec[%d]="%(i+1),end=" ")
    Vec[i]=float(input())
print("Les composantes du tableau Vec(%d) sont :"%N)
for i in range(0,N):
    print("Vec[%d]="%(i+1),Vec[i])

```

Voici un exemple d'exécution du programme ci dessus :

```
Entrer le nombre d'éléments N:4
Entrer éléments par élément les composantes du tableauVec(4):
Vec[1]= 1
Vec[2]= 2
Vec[3]= 3
Vec[4]= 4
Les composantes du tableau Vec(4) sont :
Vec[1]= 1.0
Vec[2]= 2.0
Vec[3]= 3.0
Vec[4]= 4.0
```

## 2. Exercice : Tableau de température

*[solution n°1 p.15]*

Écrire le programme python qui permet de saisir un tableau t de n nombres (réels) compris entre -40 et 50, qui représentent la plage de valeurs d'une température observable.

t=[0 for i in range(n)]

t[i]=float(input("la température doit etre >=-40 et <=50, saisissez la à nouveau"))

n=int(input("entrer la taille du tableau"))

while (t[i]<-40 or t[i]>50):

for i in range(n):

t[i]=float(input("entrer une température comprise entre -40° et 50°"))

## 3. Tableau à deux dimensions

1. Quatre éléments fondamentaux définissent un *tableau à deux dimensions* appelé *matrice*:
2. Son nom: qui sera un identifiant choisi en respectant les règles usuelles de dénomination des variables(voir chapitre II). En tenant compte de l'exemple ci-dessous le nom de la matrice est mat.
3. Sa dimension (1 dimension ou 2 dimensions). En tenant compte de l'exemple ci-dessous la dimension du tableau est 2.
4. Sa taille et les valeurs de ses indices (indice minimum, indice maximum). Le tableau à deux dimensions possède un certain nombre de lignes et un certain nombre de colonnes. Ainsi il nous faut deux indices pour repérer un élément de la matrice, l'indice de la ligne ou est situé l'élément et l'indice de colonne ou est situé l'élément. Que ce soit pour l'indice de ligne ou l'indice de colonne le plus indice est 0 et le plus grand indice est le nombre de ligne-1 et le nombre de colonne-1 respectivement pour les lignes et pour les colonnes. En tenant compte de l'exemple ci-dessous la taille du tableau est N x M. Le plus petit indice de ligne est 0 et le plus grand indice de colonne est N-1. Le plus petit indice de colonne est 0 et le plus grand indice de colonne est M-1.
5. Le type de données qu'il contient. En tenant compte de l'exemple ci-dessous, la matrice mat contient des réels.

| | 0 | 1 | | | | M-1 |
|---|---|---|---|---|---|---|
| 0 | mat[0][0]=1.4 | mat[0][1]=2.6 | | | | mat[0][M-1]=0.5 |
| 1 | mat[1][0]=1.5 | mat[1][1]=4.5 | | | | |
| | | | mat[i][j] | | | |
| | | | | | | |
| N-1 | mat[N-1][0]=1.8 | | | | | mat[N-1][M-1]=0.6 |

*Tableau à deux dimensions*

### *Fondamental*

Pour désigner un élément de la matrice on utilise son identifiant suivi de l'indice de ligne entre crochet suivi de l'indice de colonne entre crochet par exemple :

mat[i][j]représente la valeur $i^{ème}$ ligne et à la $j^{ème}$ colonne de la matrice mat.

## *Étapes de création d'une matrice*

Les principales étapes permettant la création d'une matrice sont :

1. Lire dimensions de la matrice c'est à dire lire nombre de lignes et lire le nombre de colonnes.
2. initialiser la matrice en utilisant la syntaxe :
   [[0 for j in range(0,nombre_colonnes)] for i in range(0,nombre_lignes)]
3. Lire les éléments d'un tableau, pour désigner une élément de la matrice on utilise la syntaxe :

nom_matrice[indice de ligne][indice de colonne]

### *Exemple : Lecture et écriture des éléments d'une matrice*

écrire le programme Python qui permet de saisir une matrice Tab de nombres réels et de dimension N x M . Puis le programme doit afficher les éléments de la matrice Tab.

```
N=int(input("Donnez le nombre de lignes de la matrice: "))
M=int(input("Donnez le nombre de colonnes de la matrice: "))
# Initialisation des éléments de la matrice
Tab = [[0 for j in range(0,M)] for i in range(0,N)]
print("Entrer ligne par ligne les éléments du tableau Tab(%d,%d):"%
(N,M))
for i in range(0,N):
    for j in range(0,M):
        print("Tab[%d,%d]="%(i+1,j+1),end=" ")
        Tab[i][j]=float(input())
print("Les coefficients du tableau Tab(%d,%d) sont :"%(N,M))
for i in range(0,N):
    for j in range(0,M):
        print("Tab[%d,%d]="%(i+1,j+1),Tab[i][j])

```

Voici un exemple d'exécution du programme ci dessus :

```
Donnez le nombre de lignes de la matrice: 3
Donnez le nombre de colonnes de la matrice: 4
Entrer ligne par ligne les éléments du tableau Tab(3,4):
Tab[1,1]= 1
Tab[1,2]= 2
Tab[1,3]= 3
Tab[1,4]= 4
Tab[2,1]= 5
Tab[2,2]= 6
Tab[2,3]= 7
Tab[2,4]= 8
Tab[3,1]= 9
Tab[3,2]= 10
Tab[3,3]= 11
Tab[3,4]= 12
Les coefficients du tableau Tab(3,4) sont :
Tab[1,1]= 1.0
Tab[1,2]= 2.0
Tab[1,3]= 3.0
Tab[1,4]= 4.0
Tab[2,1]= 5.0
Tab[2,2]= 6.0
Tab[2,3]= 7.0
Tab[2,4]= 8.0
Tab[3,1]= 9.0
Tab[3,2]= 10.0
Tab[3,3]= 11.0
Tab[3,4]= 12.0
```

Pour l'affichage des éléments de la matrice sous forme de matrice, on doit insérer un print avant chaque ligne de la matrice et on doit modifier le paramètre end="\n" du print à l'intérieur de la boucle comme ceci :

```
N=int(input("Donnez le nombre de lignes de la matrice: "))
M=int(input("Donnez le nombre de colonnes de la matrice: "))
# Initialisation des éléments de la matrice
Tab = [[0 for j in range(0,M)] for i in range(0,N)]
print("Entrer ligne par ligne les éléments du tableau Tab(%d,%d):"%
(N,M))
for i in range(0,N):
    for j in range(0,M):
        print("Tab[%d,%d]="%(i+1,j+1),end=" ")
        Tab[i][j]=float(input())
print("Les coefficients du tableau Tab(%d,%d) sont :"%(N,M))
for i in range(0,N):

    for j in range(0,M):
        print("Tab[%d,%d]="%(i+1,j+1),Tab[i][j])

```

Voici un exemple d'exécution, vous allez voir la différence dans l'affichage de la matrice :

```
Donnez le nombre de lignes de la matrice: 3
Donnez le nombre de colonnes de la matrice: 4
Entrer ligne par ligne les éléments du tableau Tab(3,4):
Tab[1,1]= 1
Tab[1,2]= 2
Tab[1,3]= 3
Tab[1,4]= 4
Tab[2,1]= 5
Tab[2,2]= 6
Tab[2,3]= 7
Tab[2,4]= 8
Tab[3,1]= 9
Tab[3,2]= 10
Tab[3,3]= 11
Tab[3,4]= 12
Les coefficients du tableau Tab(3,4) sont :

Tab[1,1]= 1.0  Tab[1,2]= 2.0  Tab[1,3]= 3.0  Tab[1,4]= 4.0
Tab[2,1]= 5.0  Tab[2,2]= 6.0  Tab[2,3]= 7.0  Tab[2,4]= 8.0
Tab[3,1]= 9.0  Tab[3,2]= 10.0  Tab[3,3]= 11.0  Tab[3,4]= 12.0
```

## *Fondamental : Parcourir la matrice ligne par ligne ou colonne par colonne ?*

- Pour parcourir la matrice on utilise deux boucles imbriqués, si on veut parcourir la matrice ligne par ligne que ce soit pour la lecture, l'affichage des éléments ou autre on doit commencer la boucle associée à l'indice de ligne puis la boucle associée à l'indice de colonne comme dans les exemples ci dessus.
- Pour parcourir la matrice colonne par colonne on fait l'inverse, on commence par la boucle associée à l'indice de colonne puis la boucle associé à l'indice de ligne comme ceci :

```
>>> for j in range(0,M):
    print('')
    for i in range(0,N):
        print("Tab[%d,%d]="%(i+1,j+1),Tab[i][j])
```

Voici le résultat d'exécution avec les même valeurs pour les éléments de la matrice :

```
Tab[1,1]= 1.0
Tab[2,1]= 5.0
Tab[3,1]= 9.0

Tab[1,2]= 2.0
Tab[2,2]= 6.0
Tab[3,2]= 10.0

Tab[1,3]= 3.0
Tab[2,3]= 7.0
Tab[3,3]= 11.0

Tab[1,4]= 4.0
Tab[2,4]= 8.0
Tab[3,4]= 12.0
```

## 4. Exercice : Résultat d'affichage du programme sur une matrice

*[solution n°2 p.15]*

mat=[[1,1,1,1],[2,2,2,2],[3,3,3,3],[4,4,4,4]]

for i in range(0,2):

<tab>for j in range(0,4):

<tab><tab>s=s+mat[i][j]

print(s)

QueL est le résultat de ce programme ?

- 12
- 44

# Solutions des exercices

## > **Solution** n°1

Exercice p. 10

Écrire le programme python qui permet de saisir un tableau t de n nombres (réels) compris entre -40 et 50, qui représentent la plage de valeurs d'une température observable.

```
n=int(input("entrer la taille du tableau"))
t=[0 for i in range(n)]
for i in range(n):
t[i]=float(input("entrer une température comprise entre -40° et 50°"))
while (t[i]<-40 or t[i]>50):
t[i]=float(input("la température doit etre >=-40 et <=50, saisissez la à nouveau"))
```

## > **Solution** n°2

Exercice p. 14

```
mat=[[1,1,1,1],[2,2,2,2],[3,3,3,3],[4,4,4,4]]
for i in range(0,2):
<tab>for j in range(0,4):
<tab><tab>s=s+mat[i][j]
print(s)
```

QueL est le résultat de ce programme ?

- ⊙ 12
- ○ 44

# Bibliographie


[1] Swinnen Gérard, 02/02/2012, "Apprendre à programmer avec python 3", (3e édition), Eyrolles, 435 p, Noire

[2] Jachym, Marc, "Cours de Programmation avec le langage Python: Niveau débutant en programmation",Licence professionnelle – Métrologie dimensionnelle et qualité IUT de St Denis, Université Paris 13.

[3] Cordeau, Bob, Introduction à Python 3, version 2.71828.