Chapitre 2 : Listes en Python

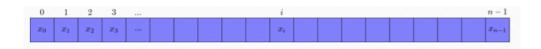
1 Qu'est-ce qu'un tableau?

D'un point de vue algorithmique, un tableau est une structure de données qui permet de stocker une **suite ordonnée** de valeurs $x_0, x_1, \ldots, x_i, \ldots, x_{n-1}$:

- chaque valeur est repérée par son indice i (commençant à 0),
- l'accès à une case à partir de son indice i se fait en **temps constant**, c'est-à-dire que ce temps ne dépend pas de i, ni de la longueur n du tableau : il est donc aussi rapide d'accéder à la première valeur qu'à la 200ème (par exemple).

Ceci est possible car dans un tableau :

- toutes les cases ont la même taille et occupent donc le même espace mémoire,
- les valeurs sont stockées dans des *cases contiguës* de la mémoire de l'ordinateur (appelée RAM : Random Acess Memory) du système informatique.



Stockage d'un tableau dans la mémoire

Le nombre de cases dans un tableau (non dynamique) est fixe.

2 Le type list en Python

Python utilise le type list pour implémenter les tableaux. Le langage permet aux tableaux d'adapter *dynamiquement* leur taille en fonction des insertions/suppressions d'éléments. Cependant, il est important de comprendre que ces mécanismes bien qu'automatiques représentent un coût temporel.

• Opérations de base sur les listes Python et complexité

Opération	Syntaxe	Complexité
Création d'une liste vide	[]	O(1)
Lecture d'un élément	x = L[i]	O(1)
Écriture d'un élément	L[i] = x	O(1)

Opération	Syntaxe	Complexité
Connaître la taille de la liste	len(L)	O(1)
Ajout d'un élément à la fin	L.append(x)	O(1)
Suppression du dernier élément	L.pop()	O(1)
Insertion d'un élément à l'indice i	L.insert(i,x)	O(n)
Suppression de l'élément d'indice i	L.pop(i)	O(n)
Copie d'une liste	L2 = L.copy()	O(n)

où L une variable de type list de taille *n*

• Exemple d'utilisation des opérations de base

```
L = [12, 23, 34, 45] # L vaut [12, 23, 34, 45]

valeur = L.pop() # L vaut [12, 23, 34] et valeur vaut 45

L.append(56) # L vaut [12, 23, 34, 56]

L.append(67) # L vaut [12, 23, 34, 56, 67]

L.insert(1, 13) # L vaut [12, 13, 23, 34, 56, 67]

valeur = L.pop(2) # L vaut [12, 13, 34, 56, 67] et valeur vaut 23

L2 = L.copy() # L vaut [12, 13, 34, 56, 67] et

# L2 est un autre tableau contenant [12, 13, 34, 56, 67]
```

• Autres opérations sur les listes Python

- L.reverse(): inverse les items d'une liste
- L.remove(x): supprime la première occurence de x (attention: génère une erreur si x n'est pas dans la liste).
- L.count(x): compte le nombre d'occurences d'une valeur x
- L.index(x): permet de connaître la position de l'item cherché x (attention: génère une erreur si x n'est pas dans la liste)
- L.sort(): trie la liste L
- x in L: vaut True si x est un élément de L, et False sinon.

• Exemple d'utilisation des autres opérations sur les listes

```
L = [1, -2, 0, 3, 0, -2, 0, 5]
L.reverse() # L vaut [5, 0, -2, 0, 3, 0, -2, 1]
nb0 = L.count(0) # nb0 vaut 3
val = -2
if val in L: # vaut True car -2 est dans la liste
   indice = L.index(val) # indice vaut 2
   L.remove(val) # L vaut [5, 0, 0, 3, 0, -2, 1]
L.sort() # L vaut [-2, 0, 0, 0, 1, 3, 5]
```