

## Cours de Python – 5 : les listes (tableaux)

### Les listes (ou tableaux) :

Une **liste** est une variable dans laquelle on peut mettre plusieurs variables.

Une liste est une variable qui peut stocker une suite (une collection) de nombres ou de caractères.

*Ex :* liste = [1,2,3]  
liste → [1, 2, 3]

### Taille d'une liste : Len

Une liste possède une taille définie par le nombre de données qu'elle contient.

→ le **1<sup>er</sup> élément** de la liste est par convention de **rang** égal à **0**

*Ex :* liste[0] → 1 , ... , liste[2] → 3 liste[4] → error

*Ex :* len(liste) → 3

### Ajouter un élément dans une liste : Append

*Ex :* liste.append(25) → ajoute 25 à la suite de la liste

liste → [1, 2, 3, 25]

### Modifier un élément dans une liste : On peut redéfinir une case du tableau

*Ex :* liste[3] = 10  
liste → [1, 2, 3, 10]

### Supprimer un élément dans une liste à partir de son index : Del

*Ex :* del liste[1]  
liste → [1, 3, 10]

### Supprimer un élément dans une liste à partir de sa valeur : remove

*Ex :* liste.remove(3)  
liste → [1, 10]

### Inverser les valeurs d'une liste : reverse

*Ex :* liste.reverse()  
liste → [10, 1]

### *Réinitialisation de la liste pour les fonctionnalités suivantes :*

*Ex :* liste = [1, 1, 2, 2, 3, 3, 3]

liste → [1, 1, 2, 2, 3, 3, 3]

### Compter le nombre d'occurrences d'une valeur : count

*Ex :* liste.count(3) → 3

### Trouver la première position d'un index : index

*Ex :* liste.index(3) → 4

Afficher les 3 premières occurrences :	liste[:3]	→ [1, 1, 2]
Afficher la dernière occurrence :	liste[-1]	→ 3
Afficher les 4 dernières occurrences :	liste[-4:]	→ [2, 2, 3, 3]
Afficher la 4 <sup>e</sup> occurrence en partant de la fin	liste[-4]	→ 2
Afficher toutes les occurrences :	liste[:]	→ [1, 1, 2, 2, 3, 3, 3]
Afficher de la 2 <sup>ème</sup> à la 5 <sup>ème</sup> occurrence	liste[1:5]	→ [1, 2, 2, 3]
Vider la liste	liste[:] = []	

On peut **additionner deux listes** :

```
>>> x = [1, 2, 3]
>>> y = [4, 5, 6]
>>> x + y
[1, 2, 3, 4, 5, 6]
z = x + y
z
```

On peut **multiplier deux listes** :

```
>>> x = [1, 2]
>>> x*5
[1, 2, 1, 2, 1, 2, 1, 2, 1, 2]
```

Cela peut être utile pour **initialiser une liste** :

```
>>> [0] * 5
[0, 0, 0, 0, 0]
```

On peut **boucler sur une liste** pour récupérer **chaque occurrence** :

```
>>> liste = ['a', 'd', 'm']
>>> for lettre in liste:
...     print lettre
...
a
d
m
```

On peut **boucler sur une liste** pour récupérer **chaque occurrence et son index** : **enumerate**

```
for lettre in enumerate(liste):
    print(lettre)
→ (0, 'a')
→ (1, 'd')
→ (2, 'm')
```

**La fonction range** génère une liste composée d'une simple **suite arithmétique**.

```
Ex : range(10)    → [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
     range(1,6)   → [1, 2, 3, 4, 5]
```

On peut **mettre à bout deux listes** : **extend**

```
x = [1, 2, 3, 4]
y = [4, 5, 1, 0]
x + y
→ [1, 2, 3, 4, 4, 5, 1, 0]
print x
→ [1, 2, 3, 4]
x.extend(y)
print x
→ [1, 2, 3, 4, 4, 5, 1, 0]
```

**Pour savoir si un élément est dans une liste** : **in**

```
liste = [1, 2, 3, 5, 10]
3 in liste    → True
11 in liste   → False
liste = ['a', 'e', 'i', 'o', 'u']
'o' in liste  → True
'y' in liste  → False
```

**Pour copier une liste dans une autre liste qui soit indépendante de la première !!!**

Méthode qui ne marche pas :

```
x = [1, 2, 3]
y = x
y[0] = 4
x
→ [4, 2, 3]
```

*En fait cette syntaxe permet de travailler sur une même élément nommé différemment*

Méthode qui marche :

```
x = [1, 2, 3]
y = x[:]
y[0] = 9
x
→ [1, 2, 3]
y
→ [9, 2, 3]
```