# Les Listes en python

## 1- Définition d'une liste

Une liste est une collection ordonnée et modifiable d'éléments éventuellement hétérogènes.

```
>>> nombres = [5, 38, 10, 25]
>>> mots = ["jambon", "fromage", "confiture", "chocolat"]
>>> stuff = [5000, "Brigitte", 3.1416, ["Albert", "René", 1947]]
```

Dans le dernier exemple ci-dessus, nous avons rassemblé un entier, une chaîne, un réel et même une liste, pour vous rappeler que l'on peut combiner dans une liste des données de n'importe quel type, y compris des listes.

## 2- Accès aux éléments d'une liste

Pour accéder aux éléments d'une liste, on utilise les mêmes méthodes (index, découpage en tranches) que pour accéder aux caractères d'une chaîne :

```
>>> print(nombres[2])
10
>>> print(nombres[1:3])
[38, 10]
>>> print(nombres[2:3])
[10]
>>> print(nombres[2:])
[10, 25]
>>> print(nombres[:2])
[5, 38]
>>> print(nombres[-1])
25
>>> print(nombres[-2])
10
```

## 3- Les listes sont modifiables

Les listes sont des *séquences modifiables*. Cela nous permettra de construire plus tard des listes de grande taille, morceau par morceau, d'une manière dynamique (c'est-à-dire à l'aide d'un algorithme quelconque). Exemples :

```
>>> nombres[0] = 17
>>> nombres
[17, 38, 10, 25]
>>> stuff[3][1] = "Isabelle"
>>> stuff
[5000, 'Brigitte', 3.1415999999999999, ['Albert', 'Isabelle', 1947]]
```

## 4- Les listes sont des objets

Sous Python, les listes sont des objets pour les quels on peut appliquer un certain nombre de ***méthodes*** (fonctions) particulièrement efficaces. En voici quelques-unes :

```
>>> nombres = [17, 38, 10, 25, 72]
>>> nombres.sort()             # trier la liste
>>> nombres
[10, 17, 25, 38, 72]

>>> nombres.append(12)         # ajouter un élément à la fin
>>> nombres
[10, 17, 25, 38, 72, 12]

>>> nombres.reverse()          # inverser l'ordre des éléments
>>> nombres
[12, 72, 38, 25, 17, 10]

>>> nombres.index(17)          # retrouver l'index d'un élément
4

>>> nombres.remove(38)         # enlever (effacer) un élément
>>> nombres
[12, 72, 25, 17, 10]
```

En plus de ces méthodes, vous disposez encore de l'instruction intégrée **del**, qui vous permet d'effacer un ou plusieurs éléments à partir de leur(s) index :

```
>>> del nombres[2]
>>> nombres
[12, 72, 17, 10]
>>> del nombres[1:3]
>>> nombres
[12, 10]
```

### 5- Techniques de slicing avancé pour modifier une liste

Vous pouvez utiliser le « découpage en tranches » (*slicing*) pour ajouter ou supprimer des éléments dans une liste à l'aide du seul opérateur **[ ]**.

#### 5- a- Insertion d'un ou plusieurs éléments n'importe où dans une liste

```
>>> mots = ['jambon', 'fromage', 'confiture', 'chocolat']
>>> mots[2:2] =["miel"]             # équivalent à mots=mots[:2] + ["miel"] + mots[2:]
>>> mots
['jambon', 'fromage', 'miel', 'confiture', 'chocolat']

>>> mots[5:5] =['saucisson', 'ketchup']
>>> mots
['jambon', 'fromage', 'miel', 'confiture', 'chocolat', 'saucisson', 'ketchup']
```

#### 5- b- Suppression/remplacement d'éléments

```
>>> mots[2:5] = []          # [] désigne une liste vide
>>> mots
['jambon','fromage','saucisson', 'ketchup']

>>> mots[1:3] = ['salade']
>>> mots
['jambon', 'salade', 'ketchup']

>>> mots[1:] = ['mayonnaise', 'poulet', 'tomate']
>>> mots
['jambon', 'mayonnaise', 'poulet', 'tomate']
```

### 6- Création d'une liste de nombres à l'aide de la fonction range()

```
>>> L=list(range(10))
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> list(range(5,13))
[5, 6, 7, 8, 9, 10, 11, 12]
>>> list(range(3,16,3))
[3, 6, 9, 12, 15]
```

Les arguments négatifs sont autorisés :

```
>>> list(range(10, -10, -3))
[10, 7, 4, 1, -2, -5, -8]
```

### 7- Parcours d'une liste à l'aide de for, range() et len()

L'instruction **for** est l'instruction idéale pour parcourir une liste :

```
>>> prov = ['La','raison','du','plus','fort','est','toujours','la','meilleure']
>>> for mot in prov:
...     print(mot, end =' ')
...
La raison du plus fort est toujours la meilleure
```

Si vous voulez parcourir une gamme d'entiers, la fonction **range()** s'impose :

```
>>> for n in range(10, 18, 3):
...     print(n, n**2, n**3)
...
10 100 1000
13 169 2197
16 256 4096
```

Il est très pratique de combiner les fonctions **range()** et **len()** pour obtenir automatiquement tous les indices d'une séquence (liste ou chaîne). Exemple :

```
fable = ['Maître','Corbeau','sur','un','arbre','perché']
for index in range(len(fable)):
 print(index, fable[index])
```

```
0 Maître
1 Corbeau
2 sur
3 un
4 arbre
5 perché
```

**Une liste en mode compréhension :**

```
>>> L1=list(range(10))
>>> L1
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> S=[i**2 for i in range(4)]
>>> S
[0, 1, 4, 9]
```

```
>>> L2=[x for x in L1 if i%3==0]
>>> L2
[0, 3, 6, 9]
```

```
>>> Q=[i**2 for i in range(10) if i%3==0]
>>> Q
[0, 9, 36, 81]
```

### 8- Opérations sur les listes

On peut appliquer aux listes les opérateurs **+** (concaténation) et ***** (multiplication) :

```
>>> fruits = ['orange','citron']
>>> legumes = ['poireau','oignon','tomate']
>>> fruits + legumes
['orange', 'citron', 'poireau', 'oignon', 'tomate']
>>> fruits * 3
['orange', 'citron', 'orange', 'citron', 'orange', 'citron']
```

L'opérateur ***** est particulièrement utile pour créer une liste de n éléments identiques :

```
>>> T = [0]*7
>>> T
[0, 0, 0, 0, 0, 0, 0]
```

Supposons par exemple que vous voulez créer une liste **B** qui contienne le même nombre d'éléments qu'une autre liste **A**. Vous pouvez obtenir ce résultat de différentes manières, mais l'une des plus simples consistera à effectuer : **B = [0]*len(A).**

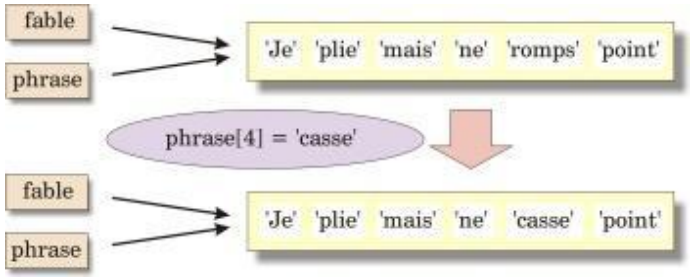
### 9- Copie d'une liste

Considérons que vous disposez d'une liste **fable** que vous souhaitez recopier dans une nouvelle variable que vous appellerez **phrase**. La première idée qui vous viendra à l'esprit sera certainement d'écrire une simple affectation telle que :

```
>>> fable = ['Je','plie','mais','ne','romps','point']
>>> phrase = fable
```

En procédant ainsi, sachez que *vous ne créez pas une véritable copie*. À la suite de cette instruction, il n'existe toujours qu'une seule liste dans la mémoire de l'ordinateur. Ce que vous avez créé est seulement *une nouvelle référence* vers cette liste. Essayez par exemple :

```
>>> fable = ['Je','plie','mais','ne','romps','point']
>>> phrase = fable
>>> fable[4] ='casse'
>>> phrase
['Je', 'plie', 'mais', 'ne', 'casse', 'point']
```



## Exercices
## Série 1

### Exercice 1 :

Soit la liste suivante : [19, 6, 15, 6, 33, 9]

Mettez en œuvre les opérations suivantes :

- Ajoutez en fin de liste l'élément 17 et affichez la liste ;
- ajoutez en début de liste l'élément 24 et affichez la liste ;
- triez, affichez la liste ;
- renversez et affichez la liste ;
- supprimez le dernier élément de la liste. Affichez l'élément et la liste ;
- supprimez le premier élément de la liste. Affichez l'élément et la liste;
- ajoutez à la fin de la liste la sous liste [27, 23, 4, 8] ;
- affichez la sous liste à partir du troisième élément ;
- affichez la sous liste composée du 2ième au 5ième élément ;
- affichez la liste composée des 2 derniers éléments en utilisant l'indexage négatif ;
- affichez à l'aide de la boucle for la somme du contenu de la liste.

### Exercice 2 :

Écrivez une fonction qui retourne la liste des carrés et des cubes des nombres de 20 à 40.

### Exercice 3 :

Écrivez une fonction qui retourne la liste des sinus des angles de 0° à 90°, par pas de 5°.
Attention : la fonction sin() du module math considère que les angles sont fournis en radians (360° = 2 π radians).

### Exercice 4 :

Écrivez une fonction qui affiche à l'écran les 15 premiers termes des tables de multiplication par 2, 3, 5, 7, 11, 13, 17, 19 (ces nombres seront placés au départ dans une liste) sous la forme d'une table similaire à la suivante :

```
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
3 6 9 12 15 18 21 24 27 30 33 36 39 42 45
5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
```

etc.

### Exercice 5 :

Vous disposez d'une liste de nombres entiers quelconques, certains d'entre eux étant présents en plusieurs exemplaires. Écrivez une fonction qui recopie cette liste dans une autre, en omettant les doublons. La liste finale devra être triée.

### Exercice 6 :

Écrivez une fonction permettant de trier une liste. Cette fonction ne pourra pas utiliser la méthode intégrée sort() de Python : vous devez donc définir vous-même l'algorithme de tri.

### Exercice 7 :

Soient les listes suivantes :

```
t1 = [31,28,31,30,31,30,31,31,30,31,30,31]
t2 = [' Janvier ' , ' Février ' , ' Mars ' , ' Avril ' , ' Mai ' , '
Juin ' ,' Juillet ' , ' Août ' , ' Septembre ' , ' Octobre ' , '
Novembre ' , ' Décembre ' ]
```

Écrivez un petit programme qui insère dans la seconde liste tous les éléments de la première, de telle sorte que chaque nom de mois soit suivi du nombre de jours correspondant :

```
['Janvier',31,'Février',28,'Mars',31, etc.].
```