# Listes

September 27, 2019

## 1 Les listes

Deuxième type construit du Python, les listes sont certainement les plus utiles (et utilisées) avec comme particularité qu'une liste est mutable (contrairement aux tuples).

Les parenthèses **()** désignant les tuples, on utilise les crochets **[]** pour les listes.

Cette notation se retrouve dans beaucoup de langages. En javascript, on parle de tableaux (array). C'est de l'art...

### 1.1 Création d'une liste vide

Il y a plusieurs façons de construire une liste.

La plus simple des listes :

```
In [18]: a=[]
         type(a)

Out[18]: list

In [2]: a

Out[2]: []
```

a est donc une liste vide ... que l'on peut remplir.

### 1.2 Ajout d'un élément

On utilise la méthode **append** pour ajouter un élément à une liste.

On peut ajouter tout type de données.

```
In [19]: a.append("first")
         a

Out[19]: ['first']

In [20]: a.append(3.5)
         a

Out[20]: ['first', 3.5]
```

Pour connaître le nombre d'éléments d'une liste, on utilise aussi la méthode **len**.

```
In [8]: len(a)

Out[8]: 2
```

### 1.3 Ajout de plusieurs éléments

On peut utiliser une boucle pour construire une liste :

```
In [22]: b=[]
         for i in range(10):
             b.append('e'+str(i))
         b

Out[22]: ['e0', 'e1', 'e2', 'e3', 'e4', 'e5', 'e6', 'e7', 'e8', 'e9']
```

On peut effectuer des concaténations comme avec les chaînes de caractères ou les tuples.

```
In [23]: c=a+b
         c

Out[23]: ['first', 3.5, 'e0', 'e1', 'e2', 'e3', 'e4', 'e5', 'e6', 'e7', 'e8', 'e9']

In [24]: d=a*3
         d

Out[24]: ['first', 3.5, 'first', 3.5, 'first', 3.5]
```

Par cette dernière méthode, on peut créer rapidement des listes :

```
In [10]: d=[1]*12
         d

Out[10]: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

### 1.4 Liste par compréhension

Une méthode interessante est la création d'une liste par compréhension : on utilise un minimum de code dès la création de la liste.

Le code pour obtenir une liste équivalente à b :

```
In [2]: e=['e'+str(i) for i in range(10)]
        e

Out[2]: ['e0', 'e1', 'e2', 'e3', 'e4', 'e5', 'e6', 'e7', 'e8', 'e9']
```

On peut de plus utiliser des conditions :

```
In [27]: f=['e'+str(i) for i in range(10) if i%3==0]
         f

Out[27]: ['e0', 'e3', 'e6', 'e9']
```

### 1.5 Accés aux éléments d'une liste

Mêmes méthodes que pour les chaînes de caractères en accédant par leur indice ou position sachant que le premier élément a pour indice 0.

```
In [28]: e[0]

Out[28]: 'e0'
```

Les 3 premiers éléments :

```
In [29]: e[:3]

Out[29]: ['e0', 'e1', 'e2']
```

Le dernier élément :

```
In [30]: e[-1]

Out[30]: 'e9'
```

Du troisième élément au septième :

```
In [31]: e[2:7]

Out[31]: ['e2', 'e3', 'e4', 'e5', 'e6']
```

Les éléments à partir du troisième :

```
In [32]: e[2:]

Out[32]: ['e2', 'e3', 'e4', 'e5', 'e6', 'e7', 'e8', 'e9']
```

On peut aussi parcourir tous les éléments avec une boucle **for elt in liste**.

```
In [3]: for elt in e:
            print(elt,end=' ')

e0 e1 e2 e3 e4 e5 e6 e7 e8 e9
```

### 1.6 Mutabilité

Le gros plus des listes : la possibilité de modifier les éléments de la liste.

On effectue pour cela une simple affectation.

```
In [4]: e[1]='T'
        e

Out[4]: ['e0', 'T', 'e2', 'e3', 'e4', 'e5', 'e6', 'e7', 'e8', 'e9']
```

En cas de copie simple de listes, il faut prendre en compte qu'en cas de modification de l'une, **la copie sera aussi modifiée**.

```
In [5]: liste1=[1,2,3]
        a=liste1
        liste1[0]='Paf'
        a

Out[5]: ['Paf', 2, 3]
```

Pour éviter ce genre de problème, on peut créer une copie à l'aide d'une boucle :

```
In [8]: liste1=[1,2,3]
        a=[]
        for elt in liste1:
            a.append(elt)
        liste1[0]='Paf'
        a

Out[8]: [1, 2, 3]
```

### 1.7 Cas particulier : liste de listes

En intégrant une liste dans une liste, on peut créer un tableau pouvant être associé à une grille, un plateau de jeu, une feuille de tableur ...

Par exemple, pour un jeu de morpion, on a un tableau 3 sur 3

```
In [12]: jeu=[['_','_','_'],['_','_','_'],['_','_','_']]
         jeu

Out[12]: [['_', '_', '_'], ['_', '_', '_'], ['_', '_', '_']]
```

Pour placer un pion dans la partie supérieure droite :

```
In [13]: jeu[0][2]='X'
         jeu

Out[13]: [['_', '_', 'X'], ['_', '_', '_'], ['_', '_', '_']]
```

Donc jeu[i][j] permet d'accéder au j+1 éme élément de la liste située à la i+1 éme place de la liste principale.