CHAPITRE

6 Le type liste de Python

Table des matières

1 Une structure bien utile !

Les variables de type `int`, `float` ou `str` servent à stocker de l'information.
Souvent, on souhaite conserver en mémoire la valeur de plusieurs variables. On va pour cela utiliser un nouveau type : les ***listes***.

Une liste permet de stocker plusieurs informations.

2 Le type `liste`

2.1 Déclaration d'une liste.

Alors que les chaînes de caractères se déclarent avec des doubles quotes `""`, on utilise des ***crochets*** pour déclarer une liste en Python.

```
1 l_1 = []                    #liste vide
2 l_2 = [2, 4, 5, 84, -1, 0] #liste de 6 éléments
```

Les variables `l_1` et `l_2` sont donc de ***types*** listes !

Les éléments d'une liste sont séparés par des virgules.

Souvent, on a recours à des listes de grandes tailles sans qu'on en connaisse le contenu. Le code suivant permet la génération d'une telle liste.

```
1 from random import randint
2 L = [randint(0, 500) for i in range(100)] #liste de 100 entiers choisis entre 0 et
  ↪  500
```

2.2 Nombre d'éléments d'une liste

Une liste `L` étant déclarée, l'instruction `len(L)` permet de retourner le nombre d'éléments de la liste.

Exercice n°1
Dans un script python, saisissez les trois listes `l_1`,`l_2` et `L` précédentes puis faites affichez dans la console leur nombre d'éléments en utilisant l'instruction `len`.

2.3 Accès aux éléments

On accède aux éléments d'une liste à l'aide de son ***indice***(ou son rang...) : le premier élément est à l'indice `0`, le second à l'indice `1`,....

L'indice du ***dernier*** élément d'une liste `L` est toujours `len(L) - 1` !

Pour accéder en ***lecture*** comme en ***écriture*** à un élément d'une liste `L` situé à l'***indice*** `i`, on utilise l'instruction `L[i]`.
Quelques manipulations classiques :

```
ma_super_liste = [5, 0, 2, 10, 8, 9, 4]   #variable de type liste
print(ma_super_liste[0])           #permet l'affichage du premier élément (indice 0)
print(ma_super_liste[1])           #permet l'affichage du deuxième élément (indice 1)
taille = len(ma_super_liste)
print(ma_super_liste[taille - 1])  #permet l'affichage du dernier élément (indice taille - 1)
```

Les opérations précédentes sont des opérations de ***lecture*** : on peut aussi ***écrire*** sur les listes :

```
print(ma_super_liste)    #affichage de la liste dans la console
ma_super_liste[0] = 1    #modification du premier élément de la liste
print(ma_super_liste)    #reaffichage pour constater la modification
```

Une liste est un objet ***mutable*** : une fois déclarée, on peut la modifier !

Exercice n°2

1. Recopier la liste `ma_super_liste` ci-dessus dans un script Python
2. Écrire le code qui permet d'afficher l'élément à l'indice 3.
3. Que se passe t-il si vous tapez l'instruction `ma_super_liste[10]` ?
4. Donnez l'instruction qui permet de changer le dernier élément de la liste en lui affectant la valeur 0.

3 Parcours d'une liste :

Comme une chaîne de caractères en Python, une liste est un objet ***itérable***, que l'on peut parcourir à l'aide d'une boucle `for`.Les deux programmes suivants sont presque équivalents :

```
#L est une liste
for i in range(len(L)):
    print(L[i])
```

```
#L est une liste
for elt in L:
    print(elt)
```

La première boucle permet de ***parcourir*** les ***indices*** de la listes(`L[i]` est l'élément de la liste situé au rang `i`) alors que la seconde permet de ***parcourir*** les ***éléments*** de la liste.

Exercice n°3
On considère une liste non vide `ma_liste`.

1. Que fait l'instruction `L[0] = L[0] + 1` ?
2. Utiliser une boucle pour ajouter 1 à tous les éléments de cette liste.

4 Génération de listes :

En introduction de ce cours, on définit une liste de 100 nombres aléatoires. Détaillons la méthode utilisée

4.1 Construction de listes par compréhension

La méthode par compréhension consiste à déterminer les éléments d'une liste à l'aide d'une boucle intégrée entre les crochets. Quelques exemples :

```
from random import randint
L = [randint(1, 6) for i in range(50)] #liste de 50 entiers choisis entre 1 et 6
```

À chaque valeur de `i`, on choisit un entier entre 1 et 6.

```
L = [2*i for i in range(10)]
```

Cela donne la liste `L = [0, 2, 4, 6, 8,..., 16, 18]`(la dernière valeur de `i` est 9 !)

```
ma_liste = [i**2 for i in range(5)]
ma_seconde_liste = [elt - 2 for elt in ma_liste]
```

La première liste est `[0, 1, 4, 9, 16]` et la deuxième est `[-2, -1, 2, 7, 14]` : la variable `elt` parcourt la liste `ma_liste`.

4.2 La méthode append() :

Supposons le programme qui consiste à interroger plusieurs fois l'utilisateur puis à stocker ces diverses réponses dans une liste. Le programme serait alors :

```
liste_proposition = []                          #liste vide au départ
for i in range(10):
    info = int(input("Proposez un nombre"))     #on interroge l'utilisateur
    liste_proposition.append(info)              #on stocke la valeur à la fin de la
    ↪ liste
```

La méthode `append` appliquée à la liste `liste_proposition`, consiste à ajouter à la fin de la liste, la valeur de la variable `info`.

Pour utiliser la méthode `append` sur une liste, il faut que cette liste existe !

5 Autres méthodes...

L'interface des listes est riche. Si la première partie propose l'essentielle des méthodes à connaître, le paragraphe suivant donne d'autres méthodes moins utilisées.

5.1 Tri

Pour une liste `L`, l'instruction `sorted(L)` retourne la liste triée.

```
from random import randint
L = [randint(1, 200) for i in rang20)]
print(L)
sorted(L)
print(L)
```

5.2 Éjection

Pour une liste `L`, l'instruction `L.pop()` retourne le dernier élément de la liste et l'enlève de la liste.

```
L = [10, 5, 2, 14, 54]
print(L)
dernier = L.pop()
print(dernier)
print(L)
```

5.3 Appartenance

L'opérateur `in` permet de savoir si un élément appartient à une liste.

```
from random import randint
L = [randint(1, 6) for i in range(5)] # 5 lancers d'un dé
if 6 in L:
    print("Tu as obtenu au moins une fois 6")
else:
    print("Tu n'as pas obtenu un 6")
```

6 Le coin des exercices

Exercice n°4

On considère la liste `L = [2, 5, 6, 1, 8, 7, 3, 11, 12, 4]`.

1. Que donnerait l'instruction `print(L[2])` ?
2. Que donnerait l'instruction `print(L[-1])` ?
3. Que donnerait l'instruction `print(L[12])` ?
4. Que donnerait l'instruction `print(L[L[3]])` ?
5. Quelle instruction Python permet de donner le nombre d'éléments de la liste `L` ?
6. Donnez les instructions Python qui permettent de multiplier tous les éléments de la liste par 3.
7. On veut construire la liste `P` qui contient tous les éléments de `L` qui sont pairs. Donnez les instructions Python qui permettent de le faire.On utilisera la méthode `append`.

Exercice n°5

On considère la liste ci-contre : `L = [10 - i for i in range(10)]`.

1. Que donnerait l'instruction `print(L[1])` ?
2. Que donnerait l'instruction `print(L[4])` ?
3. Construire la liste `M = [1, 2, 4, 8, 16, 32, 64, 128]` par compréhension.

Exercice n°6

1. Construire par compréhension la liste de toutes les minuscules.
2. Construire par compréhension la liste de toutes les majuscules.

Exercice n°7

Liste de listes...

On suppose la liste suivante : `L = [[2, 4, 5], [1, 7, 3], [9, 8, 6]]`.

1. Que vaut `len(L` ?
2. Que donnerait l'instruction `print(L[1])` et `print(L[2][0])` ?
3. Quelle instruction python permet de modifier le `1` en `0` ?