Ce qu'il faut savoir sur les listes

Dans ce cours, on va voir les opérations élémentaires permettant de manipuler les listes en Python

Une aide en ligne est disponible ici : `https://docs.python.org/3.4/tutorial/datastructures.html`

1 Commandes de base

On a déjà vu quelques commandes ensemble :

— `L = [7,2,5,6]` permet de définir la liste L ;
— `L[2]` permet d'accéder au 3ème élément de la liste, ici 5 ;
— `len(L)` permet d'obtenir la taille de la liste, ici 4 ;
— `L[3] = -5` permet de redéfinir le 4ème élément et de remplacer 6 par -5.

On sait aussi que `L[4]` renvoyait une erreur, car la liste ne comportait que 4 éléments.

Pour afficher tous les éléments de la liste, on peut utiliser une boucle `for`. Il existe deux méthodes :

```
nbElements = len(L)
for i in range(nbElements):
    print(L[i])
```

```
for elem in L:
    print(elem)
```

2 Acceder à des éléments

2.1 Acceder à un élément

Pour accéder à un élément (par exemple le 3ème), on a vu qu'il suffisait d'utiliser la commande `L[2]`.

On remarque que `L[-1]` est équivalent à `L[len(L)-1]`

2.2 Accéder à une sous-liste

Avec une commande semblable, on peut accéder à une sous-liste.

— `L[a:b]` crée une liste contenant tous les éléments de la position a à $b-1$;
— `L[a:]` crée une liste contenant tous les éléments de la position a à la fin (équivalent à `L[a:len(L)]`) ;
— `L[:b]` crée une liste contenant tous les éléments de la positon 0 à $b-1$ (équivalent à `L[0:b]`) ;

— `L[:]` crée une copie de `L`.
La commande `L2 = L[1:3]` crée la nouvelle liste `L2` contentant tous les éléments `[2,5]`

3 Ajouter des éléments à la fin d'une liste

Avec la liste précédente, on sait que `L[4] = 3` renvoie une erreur. On peut quand même rallonger une liste.

3.1 Rajouter un élément

Une des méthodes est de concaténer deux listes pour rajouter 3, on fait alors :
`L += [3]` (où `L = L + [3]`)

On peut aussi rajouter un élément avec la commande **append** `L.append(3)`, on aura ainsi la liste `[7,2,5,6,3]`
Pour crée la liste contenant dix 0, on peut utiliser deux boucles différentes :

```
L = []
for i in range(10):
    L += [0]
```

```
L = []
for i in range(10):
    L.append(0)
```

3.2 Concaténer deux listes

En utilisant la même méthode que tout à l'heure, on peut rajouter une nouvelle liste à la suite. Par exemple l'opération `L += [7,2,5]` rajoute la liste `[7,2,5]` à la fin.
Remarque : avec la commande `L2 = L + [7,2,5]`, on crée une nouvelle liste `L2`.
On peut aussi modifier la liste avec la commande `extend`. Par exemple `L.extend([7,2,5])`.

4 Supprimer un élément : La commande `del`

— `del(L[a])` permet de supprimer l'élément à la position a ;
— `del(L[a:b])` permet de supprimer tous les éléments de la position a à $b-1$;
— `del(L[a:])` permet de supprimer tous les éléments de la position a à la fin ;
— `del(L[:b])` permet de supprimer tous les éléments jusqu'à $b-1$ (inclus) ;
— `del(L[:])` rend la liste vide.

5 Compréhension de liste

Cette section permet de manipuler facilement certaine liste. Elle n'est pas indispensable pour manipuler les listes, on peut se contenter des sections précédentes (une aide plus complète se trouve ici : `https://docs.python.org/3.4/tutorial/datastructures.html?highlight=del#list-comprehensions`) :
On a vu tout à l'heure comment crée une liste contenant 10 zéros. Avec les compréhensions de listes, il suffit d'utiliser la commande `[0 for i in range (10)]`

On peut aussi crée la liste des 10 premiers entiers avec la commande : `L = [i for i in range(10)]`,ce qui est équivalent à :

```
L = []
for i in range(10):
        L.append(i)
```

On peut approfondir en rajoutant plusieurs boucles :
`L = [(i,j) for i in range(3) for j in range(3)]` donnera la liste :
`[(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)]`.
On aurait pu la crée avec la boucle :

```
L = []
for i in range(3):
    for j in range(3):
        L.append((i,j))
```

On peut aussi rajouter des conditions par exemple, si on se donne la liste `L = [7,2,−3,−5, 4]`, la commande `L2 = [i for elem in L if elem>0]` permet de crée la liste `L` ne contenant que les nombres positifs.
Une commande équivalente serait :

```
L = [7,2,−2,−5,4]
L2 = []
for elem in L:
     if elem>0:
         L2.append(L)
```