Chapitre 2

Structures de données

Table des matières

1 Listes et tableaux **2**
1.1 Définitions . . . 2
1.2 Fonctions utiles . . . 2
1.2.1 Longueur . . . 2
1.2.2 Concaténation . . . 2
1.2.3 Changer ou insérer un élément . . . 2
1.2.4 Supprimer un élément . . . 3
1.2.5 Parcours d'une liste . . . 3
1.2.6 Construction d'une liste par compréhension . . . 4

2 Les chaînes de caractères **4**

3 Les dictionnaires **4**
3.1 Insertion d'un élément . . . 4
3.2 Méthodes et fonctions utiles . . . 5

1 Listes et tableaux

1.1 Définitions

Définition 1 :

Une liste est une collection ordonnée d'objets python, ces objets pouvant être de type différents. On identifie, en python, une liste et un tableau.

Exemple :

```
>>> ma_liste = [ 8,5.3,"bonjour", [8,1]]
>>> type(ma_liste)
<class 'list'>
```

Exercice 1 : Définir la liste suivante, et tester, puis analyser les appels suivants :

```
>>> ma_liste = [ 7,5.3,"bonjour", [8,1] , 15 , "coucou" , [12,0.5,81]]
>>> ma_liste[1]
>>> ma_liste[3][0]
>>> ma_liste[2:5]
>>> ma_liste[-1]
>>> ma_liste[-1][2]
>>> ma_liste[:2]
>>> ma_liste[4:]
```

1.2 Fonctions utiles

1.2.1 Longueur

La longueur d'une liste, c'est à dire le nombre d'élément qu'elle contient s'obtient grâce à la fonction `len` :

```
>>> len(ma_liste)
7
```

1.2.2 Concaténation

Concaténer deux listes revient à les mettre "bout à bout". L'opérateur simple en Python est le + :

```
>>> [1,5,6] + [8,9 ]
[1, 5, 6, 8, 9]
```

1.2.3 Changer ou insérer un élément

On peut changer un élément de la liste, ou insérer un élément dans une liste :

```
>>> ma_liste = [ 7,5.3,"bonjour", [8,1] , 15 , "coucou" , [12,0.5,81]]
>>> ma_liste[3]="rebonjour"
>>> ma_liste
```

```
[7, 5.3, 'bonjour', 'rebonjour', 15, 'coucou', [12, 0.5, 81]]
>>> ma_liste.append("au␣revoir")
>>> ma_liste
[7, 5.3, 'bonjour', 'rebonjour', 15, 'coucou', [12, 0.5, 81], 'au revoir']
```

1.2.4 Supprimer un élément

On peut supprimer un élément de la liste en pointant son index avec la fonction `del` :

```
>>> del ma_liste[2]
>>> ma_liste
[7, 5.3, 'rebonjour', 15, 'coucou', [12, 0.5, 81], 'au revoir']
```

Ou avec la méthode `pop`, en récupérant l'élément en fin de pile.

```
>>> ma_liste.pop()
'au revoir'
>>> ma_liste
[7, 5.3, 'rebonjour', 15, 'coucou', [12, 0.5, 81]]
```

ou encore en indiquant l'élément à retirer :

```
>>> ma_liste.remove("rebonjour")
>>> ma_liste
[7, 5.3, 15, 'coucou', [12, 0.5, 81]]
```

ou vider totalement la liste :

```
>>> ma_liste[:] = []
>>> ma_liste
[]
```

1.2.5 Parcours d'une liste

On utilise `in` pour vérifier l'appartenance d'un élément dans une liste, ou pour construire un parcours d'une liste :

```
>>> ma_liste = [ 8,5.3,"bonjour", [8,1] , 15 , "coucou" , [1,0.5,8]]
>>> 15 in ma_liste
True
```

Et le parcours :

```
>>> for element in ma_liste :
  print(element)


8
5.3
bonjour
[8, 1]
15
coucou
[1, 0.5, 8]
```

1.2.6 Construction d'une liste par compréhension

Il est facile de construire à partir d'une autre liste, en donnant la méthode de construction.

Exemple :

```
>>> [ 5 for i in range(7)]
[5, 5, 5, 5, 5, 5, 5]
>>> [ 2*i+1 for i in range(7)]
[1, 3, 5, 7, 9, 11, 13]
>>> [ i if i%2==0 else 5*i for i in range(10,20)]
[10, 55, 12, 65, 14, 75, 16, 85, 18, 95]
```

2 Les chaînes de caractères

On retrouve la majorité des opérations des listes :

Exercice 2 : Définir la chaine suivante, et tester les appels suivants :

```
>>> le_mot = "abracadabra"
>>> le_mot[3]
>>> le_mot[-1]
>>> le_mot[:3]
>>> le_mot[5:]
>>> le_mot[2:6]
>>> len(le_mot)
```

Remarque : Les chaines sont non mutables :

```
>>> le_mot[2] = "t"
TypeError: 'str' object does not support item assignment
```

3 Les dictionnaires

Un dictionnaire est une structure permettant de conserver certaines données définies par une clé :

Exemple :

```
>>> departements = {"01" : "Ain" , "02" : "Aisne" , "03" : "Allier" , "04" : "
Hautes-Alpes"}
>>> type(departements)
```

3.1 Insertion d'un élément

Il est possible de partir d'un dictionnaire vide, et d'insérer chaque éléments avec sa clé :

Exemple :

```
>>> ma_valise = {}
>>> ma_valise["chaussette"] = 5
>>> ma_valise["calecon"] = 6
>>> ma_valise["pull"] = 2
>>> ma_valise["pantalon"] = 3
>>> ma_valise["chemise"] = 2
```

Remarque : Si la clé n'existe pas, elle est ajoutée au dictionnaire, sinon l'ancienne valeur est remplacée par la nouvelle.

3.2 Méthodes et fonctions utiles

Tester les méthodes et les fonctions suivantes :

Exemple :

```
>>> ma_valise = {}
>>> ma_valise["chaussette"] = 5
>>> ma_valise["calecon"] = 6
>>> ma_valise["pull"] = 2
>>> ma_valise["pantalon"] = 3
>>> ma_valise["chemise"] = 2
>>> del ma_valise["chaussette"]
>>> ma_valise.pop("calecon")
>>> for i in ma_valise :
  print i
>>> for j in ma_valise.keys() :
  print(j)
>>> for k in ma_valise.values() :
  print(k)
>>> for k in ma_valise.items() :
  print(k)
```