Points forts et points faibles des structures de données intégrées de Python

Le type **list** dispose de plusieurs méthodes. Les méthodes des listes rendent très facile leur utilisation comme des piles, où le dernier élément ajouté est le premier récupéré. Il est également possible d'utiliser une liste comme une file, où le premier élément ajouté est le premier récupéré

In [1]:

```
stack = [3, 4, 5]
stack.append(6)
stack.append(7)
stack
```

Out[1]:

```
[3, 4, 5, 6, 7]
```

Un **tuple** consiste en différentes valeurs séparées par des virgules. Il n'est pas possible d'affecter de valeur à un élément d'un tuple ; par contre, il est possible de créer des tuples contenant des objets mutables, comme des listes. Un problème spécifique est la construction de tuples ne contenant aucun ou un seul élément.

In [7]:

```
t = 12345, 54321, 'hello!'
t[0]
```

Out[7]:

```
12345
```

In [5]:

```
t
```

Out[5]:

```
(12345, 54321, 'hello!')
```

Python fournit également un type de donnée pour les ensembles. Des accolades, ou la fonction **set()** peuvent être utilisés pour créer des ensembles. Un ensemble est une collection non ordonnée sans élément dupliqué. Des utilisations basiques concernent par exemple des tests d'appartenance ou des suppressons de doublons. Les ensembles supportent également les opérations mathématiques comme les unions, intersections, différences et différences symétriques.

In [9]:

```
basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
fruit = set(basket)
fruit
```

Out[9]:

```
{'apple', 'banana', 'orange', 'pear'}
```

Un autre type de donnée très utile, natif dans Python, est le **dictionnaire**. À la différence des séquences, qui sont indexées par des nombres, les dictionnaires sont indexés par des clés, qui peuvent être de n'importe quel type immuable ; les chaînes de caractères et les nombres peuvent toujours être des clés. Le plus simple est de

considérer les dictionnaires comme des ensembles non ordonnés de paires clé: valeur, les clés devant être uniques (au sein d'un dictionnaire).

In [10]:

```
tel = {'jack': 4098, 'sape': 4139}
tel['guido'] = 4127
tel
```

Out[10]:

```
{'guido': 4127, 'jack': 4098, 'sape': 4139}
```

In [11]:

```
tel['jack']
```

Out[11]:

```
4098
```

In [12]:

```
del tel['sape']
tel['irv'] = 4127
tel
```

Out[12]:

```
{'guido': 4127, 'irv': 4127, 'jack': 4098}
```

In [13]:

```
tel.keys()
```

Out[13]:

```
dict_keys(['guido', 'irv', 'jack'])
```

In [14]:

```
'guido' in tel
```

Out[14]:

```
True
```

Ref : https://www.afpy.org/doc/python/2.7/tutorial/datastructures.html