

# W2-S4-C1-listes

December 15, 2014

## 1 Méthodes spécifiques aux listes

### 1.1 Complément - niveau basique

Voici quelques unes des méthodes disponibles sur le type `list`.

#### 1.1.1 Trouver l'information

Pour commencer, rappelons comment retrouver la liste des méthodes définies sur le type `list`:

```
In []: help(list)
```

Si vous ignorez pour l'instant les méthodes dont le nom commence et termine par `--` (nous parlerons de ceci en semaine 5), vous trouvez les méthodes utiles listées entre **`append`** et **`sort`**.

Certaines de ces méthodes ont été vues dans la vidéo sur les séquences, c'est le cas notamment de **`count`** et **`index`**.

Nous allons à présent décrire les autres, partiellement et brièvement. Un autre complément décrit la méthode **`sort`**. Reportez-vous au lien donné en fin de notebook pour obtenir une information plus complète.

Donnons nous pour commencer une liste témoin

```
In []: liste = range(4)
      print 'liste', liste
```

**Avertissement** soyez bien attentifs au nombre de fois où vous exécutez les cellules de ce notebook; par exemple une liste renversée deux fois peut donner l'impression que **`reverse`** ne marche pas :) N'hésitez pas à utiliser le menu 'Cell -> Run All' pour réexécuter une seule fois le notebook entier.

#### 1.1.2 `append`

La méthode **`append`** permet d'ajouter un élément à la fin d'une liste:

```
In []: liste.append('ap')
      print 'liste', liste
```

#### 1.1.3 `extend`

La méthode **`extend`** réalise ma la même opération mais avec plusieurs éléments

```
In []: liste.extend(['ex1', 'ex2'])
      print 'liste', liste
```

Ces deux méthodes **`append`** et **`extend`** sont donc assez voisines; avant de voir d'autres méthodes de `list`, prenons un peu le temps de comparer leur comportement avec l'addition `+` de liste. L'élément clé ici, on l'a déjà vu dans la vidéo, est que la liste est un objet **mutable**. **`append`** et **`extend`** **modifient** la liste sur laquelle elles travaillent, alors que l'addition **crée un nouvel objet**

```
In []: l1 = range(5)
      l2 = l1 + ['ap', 'ex1', 'ex2']
      print 'l1', l1
      print 'l2', l2
```

Comme on le voit, après une addition, les deux termes de l'addition sont inchangés; c'est pour cela que l'addition est disponible sur tous les types séquences, mais que **append** et **extend** ne sont par exemple **pas disponibles** sur les chaînes de caractères, qui sont **immuables**.

#### 1.1.4 insert

La méthode **insert** permet, comme le nom le suggère, d'insérer un élément à une certaine position; comme toujours les indices commencent à zéro et donc

```
In []: liste.insert(2, 'remplace 2')
      print 'liste', liste
```

On peut remarquer qu'un résultat analogue peut être obtenu avec une affectation de slice; par exemple pour insérer au rang 5 (i.e. avant 'ap'), on pourrait aussi bien faire

```
In []: liste[5:5] = ['s1']
      print 'liste', liste
```

#### 1.1.5 remove

La méthode **remove** détruit la première occurrence d'un objet dans la liste.

```
In []: liste.remove(3)
      print 'liste', liste
```

#### 1.1.6 pop

La méthode **pop** prend en argument un indice; elle permet d'extraire l'élément à cet indice. En un seul appel on obtient la valeur de l'élément et on l'enlève de la liste:

```
In []: popped = liste.pop(0)
      print 'popped', popped, 'liste', liste
```

Si l'indice n'est pas précisé, c'est le dernier élément de la liste qui est visé

```
In []: popped = liste.pop()
      print 'popped', popped, 'liste', liste
```

#### 1.1.7 reverse

Enfin **reverse** renverse la liste, le premier élément devient le dernier:

```
In []: liste.reverse()
      print 'liste', liste
```

On peut remarquer ici que le résultat se rapproche de ce qu'on peut obtenir avec une opération de slicing comme ceci

```
In []: liste2 = liste[::-1]
      print 'liste2', liste2
```

à la différence toutefois qu'avec le slicing c'est une copie de la liste initiale qui est retournée, la liste de départ n'est quant à elle pas modifiée.

#### 1.1.8 Pour en savoir plus

<https://docs.python.org/2/tutorial/datastructures.html#more-on-lists>