Tris de listes

Complément - niveau basique

python fournit une fonction standard pour trier une liste, qui s'appelle, sans grande surprise, `sort`.

Utilisation basique

Voyons comment se comporte `sort` sur un exemple simple.

```
liste = [8, 7, 4, 3, 2, 9, 1, 5, 6]
print 'avant tri', liste
liste.sort()
print 'apres tri', liste
```

On retrouve ici, avec l'instruction `liste.sort()` un cas d'appel de méthode (ici `sort`) sur un objet (ici `liste`), comme on l'avait vu dans la vidéo sur la notion d'objet.

La première chose à remarquer est que la liste d'entrée a été modifiée, on dit "en place", ou encore "par effet de bord".

On aurait pu imaginer que la liste d'entrée soit restée inchangée, et que la méthode de tri renvoie une copie triée de la liste, ce n'est pas le choix qui a été fait. De cette façon, c'est à l'utilisateur de la méthode de réaliser au préalable une copie de la liste initiale si c'est nécessaire dans son cas d'usage. On évite ainsi d'imposer une opération de copie, qui peut être coûteuse en mémoire, lorsque ce n'est pas utile.

Tri décroissant

On remarque aussi que la liste est triée par ordre croissant. Si vous souhaitez au contraire l'ordre décroissant, vous pouvez le faire comme ceci:

```
liste = [8, 7, 4, 3, 2, 9, 1, 5, 6]
print 'avant tri', liste
liste.sort(reverse=True)
print 'apres tri décroissant', liste
```

Nous n'avons pas encore vu à quoi correspond cette formule `reverse=True` dans l'appel à la méthode, ceci sera approfondi dans le chapitre sur les appels de fonction, mais dans l'immédiat vous pouvez utiliser cette technique telle quelle.

Chaînes de caractères

Cette technique fonctionne très bien sur tous les types numériques (enfin, à l'exception des complexes; en guise d'exercice: pourquoi ?), ainsi que sur les chaînes de caractères:

```
liste = ['spam', 'egg', 'bacon', 'beef']
liste.sort()
print 'après tri', liste
```

Comme on s'y attend, il s'agit cette fois d'un tri lexicographique, dérivé de l'ordre total sur les caractères. Il faut souligner toutefois, pour les personnes n'ayant jamais été exposées à l'informatique, que cet ordre, quoique déterministe, est arbitraire en dehors des lettres de l'alphabet.

Ainsi par exemple:

```
'a' < 'z'
```

bon; mais par contre

```
'Z' < 'a'
```

Ce qui explique ceci

```
liste = ['abc', 'Zoo']
liste.sort()
print liste
```

Et lorsque les chaines contiennent des espaces ou autres ponctuations, le résultat du tri peut paraître surprenant:

```
liste = [' zoo', 'ane']
liste.sort()
print liste
```

À suivre

Il est possible de définir sois-même le critère à utiliser pour trier une liste, et nous verrons cela très bientôt une fois que nous aurons introduit la notion de fonction.