Tri de listes : deuxième partie

Complément - niveau intermédiaire

Nous avons vu précédemment comment faire le tri simple d'une liste, en utilisant éventuellement le paramètre `reverse` de la méthode `sort` sur les listes. Maintenant que nous sommes familiers avec la notion de fonction, nous pouvons approfondir ce sujet.

Cas général

Dans le cas général, on est souvent amené à trier des objets selon un critère propre à l'application. Imaginons par exemple que l'on dispose d'une liste de tuples à deux éléments, dont le premier est la latitude et le second la longitude

```
coordonnees = [(43, 7), (46, -7), (46, 0)]
```

Il est possible d'utiliser la méthode `sort` pour faire cela, mais il va falloir l'aider un peu plus, et lui expliquer comment comparer deux éléments de la liste.

Voyons comment on pourrait procéder pour trier par longitude

```
def longitude(element):
    return element[1]

coordonnees.sort(key=longitude)
print "coordonnées triées par longitude", coordonnees
```

Comme on le devine, le procédé ici consiste à indiquer à `sort` comment calculer, à partir de chaque élément, une valeur numérique qui sert de base au tri.

Pour cela on passe à la méthode `sort` un argument `key` qui désigne une fonction, qui lorsqu'elle est appliquée à un élément de la liste, retourne la valeur qui doit servir de base au tri: dans notre exemple, la fonction `longitude`, qui renvoie le second élément du tuple.

Fonction de commodité : `sorted`

On a vu que `sort` réalise le tri de la liste "en place". Pour les cas ou une copie est nécessaire, python fournit également une fonction de commodité, qui permet précisément de renvoyer la copie triée d'une liste d'entrée. Cette fonction est baptisée `sorted`, elle s'utilise par exemple comme ceci, sachant que les arguments `reverse` et `key` peuvent être mentionnés comme avec `sort`

```
liste = [8, 7, 4, 3, 2, 9, 1, 5, 6]
```

```
triee = sorted(liste, reverse=True)
print 'liste triée tri', triee
print 'la liste initiale est intacte', liste
```

Nous avons qualifié `sorted` de fonction de commodité car il est très facile de s'en passer; en effet on aurait pu écrire à la place du fragment précédent:

```
liste = [8, 7, 4, 3, 2, 9, 1, 5, 6]
triee = liste[:]
triee.sort(reverse=True)
print 'liste triée tri', triee
print 'la liste initiale est intacte', liste
```

Pour en savoir plus

Pour avoir plus d'informations sur `sort` et `sorted` vous pouvez lire cette section de la documentation python sur le tri. (https://docs.python.org/2.7/howto/sorting.html)