Slices sur les séquences

L'utilisation des “slices” ou du “slicing” sur les listes, ou sur tout objet en séquence (tuple, chaîne de caractères, ...) permet de “découper” des sous-listes. Si la séquence s'appelle *sequence_name*, la syntaxe du slice est : sequence_name[start:stop:step] où start est l'indice du premier élément (par défaut 0), stop est l'indice du premier élément NON REPRIS (par défaut len(seq...)) et step le pas (par défaut step = 1). Les indices négatifs équivalent aux indices modulo len(seq...).

slices_01.py

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
"""
Utilisation des "slices" ou du "slicing" sur les listes,
ou sur tout objet en séquence (tuple, chaîne de caractères, ...)
Syntaxe : sequence_name[start:stop:step]
start est l'indice du premier élément, par défaut 0
stop est l'indice du premier élément NON REPRIS, par défaut len(seq...)
par défaut, step = 1
Les indices négatifs équivalent les indices modulo len(seq...)
"""
a=[1,3,5,7,11,13,15]
print "liste exemple : ",a
b=a[1:4:1]
print b  # [3, 5, 7]
print a[:5] # [1, 3, 5, 7, 11]
print a[3:] # [7, 11, 13, 15]
print a[-3:] # [11, 13, 15]
# un élément sur 2 :
print a[::2] # [1, 5, 11, 15]
# un truc pratique pour copier une liste (la nouvelle liste est
indépendante)
copie_a=a[:]
print a
# step=-1 permet de renverser la séquence
print a[::-1] # [15, 13, 11, 7, 5, 3, 1]
# slice et remplacement d'une sous-séquence par une séquence de
longueur indépendante
a[2:4]=[5,6,7]
print a # [1, 3, 5, 6, 7, 11, 13, 15]
# slice et insertion
a[5:5]=[8,9,10]
print a # [1, 3, 5, 6, 7, 8, 9, 10, 11, 13, 15]
```

Les slices s'appliquent aussi sur les “arrays” de la librairie NumPy.

Slices et objets python, comparaison avec .reverse()

La fonction id() permet de connaître l'identifiant unique d'un objet python en mémoire

Le slice “[::-1]” crée un nouvel objet qui est la liste dans l'ordre inversé des éléments de la liste d'origine, tandis que la fonctionnalité “.reverse()” appliquée à une liste effectue cette inversion sans créer une nouvelle liste. Les codes suivants permettent de le vérifier :

```
a = [1, 2, 3]
b = a
a = a[::-1]
print(a, b)

# Output: [3, 2, 1] [1, 2, 3]

print(id(a), id(b))
# → different identifiers
```

```
a = [1, 2, 3]
b = a
a.reverse()
print(a, b)

# Output: [3, 2, 1] [3, 2, 1]

print(id(a), id(b))
# → same identifier
```

Références

- Indexing vs Slicing in Python