Affectation simultanée

Complément - niveau basique

Nous avons déjà parlé de l'affectation par ***sequence unpacking*** (en Semaine 2, séquence "Les tuples"), qui consiste à affecter à plusieurs variables des "morceaux" d'un objet, comme dans :

```
x, y = ['spam', 'egg']
```

Dans ce complément nous allons voir une autre forme de l'affectation, qui consiste à affecter **le même objet** à plusieurs variables. Commençons par un exemple simple :

```
a = b = 1
print 'a', a, 'b', b
```

La raison pour laquelle nous abordons cette construction maintenant est qu'elle a une forte relation avec les références partagées; pour bien le voir, nous allons utiliser une valeur mutable comme valeur à affecter :

```
# on affecte a et b au même objet liste vide
a = b = []
```

Dès lors nous sommes dans le cas typique d'une référence partagée; une modification de `a` va se répercuter sur `b` puisque ces deux variables désignent **le même objet** :

```
a.append(1)
print 'a', a, 'b', b
```

Ceci est à mettre en contraste avec plusieurs affectations séparées :

```
# si on utilise deux affectations différentes
a = []
b = []

# alors on peut changer a sans changer b
a.append(1)
print 'a', a, 'b', b
```

On voit que dans ce cas chaque affectation crée une liste vide différente, et les deux variables ne partagent plus de donnée.

D'une manière générale, utiliser l'affectation simultanée vers un objet mutable crée

mécaniquement des **références partagées**, aussi vérifiez bien dans ce cas que c'est votre intention.