Affectations & Opérations (à la +=)

Complément - niveau intermédiaire

Il existe en python toute une famille d'opérateurs dérivés de l'affectation, qui permettent de faire en une fois une opération et une affectation. En voici quelques exemples.

Incrémentation

On peut facilement augmenter la valeur d'une variable numérique comme ceci

```
entier = 10

entier += 2
print 'entier', entier
```

Comme on le devine peut-être, ceci est équivalent à

```
entier = entier + 2
print 'entier', entier
```

Autres opérateurs courants

Cette forme, qui combine opération sur une variable, et réaffectation du résultat à la même variable, est disponible avec tous les opérateurs courants

```
entier -= 4
print 'après décrément', entier
entier *= 2
print 'après doublement', entier
entier /= 2
print 'mis à moitié', entier
```

Types non numériques

En réalité cette construction est disponible sur tous les types qui supportent l'opérateur en question. Par exemple les listes (que nous verrons bientôt) peuvent être additionnées entre elles:

```
liste = range(3)
print 'liste', liste
```

```
liste += ['a', 'b']
print 'aprés ajout', liste
```

Beaucoup de types supportent l'opérateur +, qui est sans doute de loin celui qui est le plus utilisé avec cette construction.

Opérateurs plus abscons

Signalons enfin qu'on trouve cette construction aussi avec d'autres opérateurs moins fréquents, par exemple

```
entier = 2
print 'entier:', entier
entier **= 10
print 'à la puissance dix:', entier
entier %= 5
print 'modulo 5:', entier
entier <<= 2
print 'double décalage gauche:', entier
```