

Les compréhensions de liste python

Il existe des tonnes d'astuces **python** qui permettent d' « **optimiser le code** » .

Une de ces astuces est la **compréhension de liste** (ou **liste en compréhension** ou **list comprehension**).

L'idée est de simplifier le code pour le rendre plus lisible et donc plus rapide à écrire et plus simple à maintenir. En Python, l'obsession semble être d'avoir le code le plus bref possible. Mais attention, en terme d'opérations élémentaires, ça revient au même et n'économise rien ! que vous travaillez en compréhension ou « à l'ancienne », les instructions exécutées par le processeur in fine seront les même !

Syntaxe

```
new_list = [function(item) for item in list if condition(item)] // item, ça veut dire « élément »
```

Exemple : Filter une liste

Prenons un exemple d'une liste:

```
>>> a = [1,4,2,7,1,9,0,3,4,6,6,6,8,3]
```

Nous voulons filtrer les valeurs de cette liste et ne garder que ceux dont la valeur est supérieure à 5:

```
>>> b = []
>>> for x in a:
...     if x > 5:
...         b.append(x)
... 
```

```
>>> b
```

```
[7, 9, 6, 6, 8]
```

Il est possible de faire exactement ce que fait ce bloc de code en une seule ligne:

```
>>> b = [x for x in a if x > 5]
```

Sortie : b prendra pour état [7, 9, 6, 6, 8]

Exécuter une fonction sur chaque item d'une liste

Prenons l'exemple d'une conversion de string en integer de plusieurs items:

```
>>> items = ["5", "10", "15"]
```

```
>>> items = [int(x) for x in items]
```

```
>>> print(items)
```

```
[5, 10, 15]
```

Autres exemples :

valeur de l'expression [bidule for bidule in range(3)] : **[0, 1, 2]**

valeur de l'expression [[1,2,3] for bidule in range(3)] : **[[1, 2, 3], [1, 2, 3], [1, 2, 3]]**