# NSI - première

Python 7 - Compréhension

qkzk

2021/04/27

Jusqu'ici nous avons crée les listes et les dictionnaires "à la main" ou en utilisant une boucle.

Il existe une autre manière de créer ces objets en Python, appelée "compréhension". Elle est puissante, pratique et surtout lisible.

## Listes par compréhension

### Créer des listes

On connait plusieurs méthodes pour créer des *listes* :

Directement

```
>>> liste = [1, 2, 3]
>>> liste
[1, 2, 3]
```

Avec `append`

```
>>> liste = []
>>> liste.append(1)
>>> liste.append(2)
>>> liste.append(3)
>>> liste
[1, 2, 3]
```

La première méthode est pratique quand on sait déjà ce que va contenir la liste... la seconde quand on la remplit au fur et à mesure, comme dans une boucle.

---

Souvenons nous de `range(debut, fin, pas)` : les nombres entre *début* et *fin* séparés de *pas*.

Donc `range(10, 30, 2)` :

```
10, 12, 14, 16, 18, 20, 22, 24, 26, 28
```

Pourquoi pas 30 ? Parce que le second paramètre est toujours exclu.

---

Testons :

#### Un exemple

Créer la liste des carrés des entiers pairs compris entre 10 et 31

```
carres_10_30 = []
for k in range(10, 30, 2):
    carres_10_30.append(k ** 2)
```

Qui contient :

```
[100, 144, 196, 256, 324, 400, 484, 576, 676, 784]
```

### Liste par compréhension

Le principe est le suivant :

- on crée une liste : `[ ... ]`
- des valeurs ... : `[ valeur ... ]`
- pour les éléments d'une collection : `[ valeur for element in collection ]`
- éventuellement on peut filtrer : `[ valeur for element in collection if condition]`

Pour la liste précédente :

```
carres_10_30 = [k ** 2 for k in range(10, 30, 2)]
```

Créeons la liste des longueurs des mots `person, Woman, man, camera, tv`

```
longueurs = [len(mot) for mot in ["person", "woman", "man", "camera", "tv"]]
```

va produire la liste :

```
[6, 6, 3, 6, 2]
```

recommençons mais cette fois, seulement si le mot ne contient pas la lettre "a"

```
longueurs = [len(mot) for mot in ["person", "woman", "man", "camera", "tv"] if "a" not in mot]
```

va produire la liste :

```
[6, 2]
```

On écrit souvent ces constructions sur plusieurs lignes pour améliorer la lisibilité :

```
longueurs = [len(mot)
             for mot in ["person", "woman", "man", "camera", "tv"]
             if "a" not in mot]
```

### Structures imbriquées

Il est possible d'inclure plusieurs sous structures, par exemple pour aplatir une liste ou créer des tableaux à deux dimensions :

```
>>> c = [(x, y) for x in range(3) for y in range(3)]
>>> c
[(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)]
>>> c = [ [x + y for x in range(3)] for y in range(3)]
>>> c
[[0, 1, 2], [1, 2, 3], [2, 3, 4]]
>>> plat = [d for ligne in c for d in ligne]
>>> plat
[0, 1, 2, 1, 2, 3, 2, 3, 4]
```

Vous rencontrerez parfois ces constructions mais les réaliser vous même n'est pas un objectif immédiat.

### Exercice 1

Créer la liste par compréhension des triples des entiers entre 10 inclus et 20 exclu.

Pour vous rassurer, on doit obtenir :

```
triples = [30, 33, 36, 39, 42, 45, 48, 51, 54, 57]
```

### Exercice 2

Rappel : l'expression `lettre in mot` renvoie `True` si la lettre est dans le mot et `False` sinon

**Consigne:** Construire *par compréhension* la liste des longueurs des mots contenant la lettre `e` parmi :

`liste_mots = ["tomate", "ballon", "salle", "bois"]`

Bien sûr, on doit trouver `[6, 5]`, car seuls `"tomate"` et `"salle"` contiennent un `e`

## Dictionnaires par compréhension

Le principe est exactement le même, on crée un dictionnaire en itérant sur une collection.

Par exemple, pour créer le dictionnaire des carrés des entiers de 1 à 10 et de leurs valeurs :

```
>>> d = {nombre: nombre ** 2 for nombre in range(1, 11)}
>>> d
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

On peut encore une fois filtrer, par exemple en évitant les nombres divisibles par trois :

```
>>> d = {nombre: nombre ** 2 for nombre in range(1, 11) if nombre % 3 != 0}
>>> d
{1: 1, 2: 4, 4: 16, 5: 25, 7: 49, 8: 64, 10: 100}
```

### Exercice 3

On considère la liste suivante :

```
liste_mots = ["tomate", "ballon", "salle", "bois"]
```

1. Créer par comprehénsion le dictionnaire ayant pour clés les mots et pour valeur leur dernière lettre.
2. Recommencer en filtrant les mots qui contiennent la lettre "e"