INF8007 – Languages de scripts
Compréhensions

Antoine Lefebvre-Brossard

Hiver 2018

Ce que c'est

- Moyen de créer des structures de données de façon plus concise
- **Exemple :** Plutôt que de créer une liste avec les premiers nombres carrés avec la boucle :

```
squares = []
for x in range(10):
    squares.append(x**2)
```

il est possible de l'écrire avec :

```
squares = [x**2 for x in range(10)]
```

Avantages

- Plus concis et clair lorsque la fonction est simple
- Souvent plus rapide que la boucle équivalente (surtout si une fontion est appelé dans la boucle)
- Des trucs pour améliorer la performance peuvent être trouvé à *https://wiki.python.org/moin/PythonSpeed/PerformanceTips#Loops*
- Mais il faut faire attention à seulement optimiser les vrais bouchons (les compréhensions sont toutefois à privilégier dès que possible)

Compréhensions de list

- La structure pour une compréhension de liste est :
  ```
  [`f`(x) for x in `collection`]
  ```
- **Exemple :**
  ```
  squares = [x**2 for x in range(10)]
  ```
- Il est aussi possible de mettre des conditions
- **Exemple :** Le carré des nombres pairs et le cube des nombres impairs
  ```
  weird_list = [x**2 if x % 2 == 0
                else x**3
                for x in range(10)]
  ```

Exemples


- ```
  [100 * i + 10 * j + 1 * k
   for k in range(10)
   for j in range(10)
   for i in range(10)]
  ```

  a) [1, 2, 3, ..., 999, 1000]
  b) [0, 100, 101, ..., 998, 999]
  c) [0, 1, 2, ..., 998, 999]
  d) [0, 100, 110, ..., 998, 999]

- ```
  text = "Il a dit 'Mais non!' et la réponse a été 'Mais oui...'."
  " ".join(["".join(["\"" if char == "'" else char for char in word])
            for word in text.split()])
  ```

  a) "Il a dit 'Mais non !' et la réponse a été 'Mais oui...'."
  b) "Il a dit \"Mais non !' et la réponse a été \"Mais oui...'."
  c) "Il a dit \"Mais non !\" et la réponse a été \"Mais oui...\"."
  d) 'Il a dit "Mais non !" et la réponse a été "Mais oui...".'

Compréhensions de set et dict

- De la même façon, il est possible d'avoir des compréhensions d'ensembles ou de dictionnaires
- **Exemple :**

```
vocab = {word for word in text.split()}
```

- **Exemple :**

```
word2index = {w: i for i, w in enumerate(vocab)}
```

- Les mêmes boucles imbriquées et conditions y sont possible

Fonctions utiles

enumerate Retourne un générateur où chaque élément est un tuple avec l'index en premier et l'élément de la collection originale en second

zip Retourne un générateur où chaque élément est un tuple de chaque élément suivant des collections données en paramètre

‘dict‘.items() Retourne un générateur où chaque élément est un tuple d'une clé et de sa valeur correspondante

set().union(*[‘list of sets‘)] Permet de faire l'union de plusieurs ensembles (utile pour créer un vocabulaire à partir de plusieurs textes)

Exemples


- ```
  {w: i for i, w in enumerate(["chat", "chien", "perroquet"])}
  >>> {"chat": 0, "chien": 1, "perroquet": 2}
  ```
- ```
  [(v, p)
   for v, p
   in zip(["aller", "voir", "faire"], ["allé", "vu", "fait"])]
  >>> [("aller", "allé"), ("voir", "vu"), ("faire", "fait")]
  ```