Les tuples

4. Ce qu'il faut retenir

Les listes

Une **liste** est une structure de données dans laquelle on peut pouvez stocker des éléments de types différents. Il peut être écrit sous forme de liste de valeurs (éléments) séparées par des **virgules** et placées entre **crochets** .

```
matieres = ['Physiques', 'Chimie', 'Maths', 4, 2.1, 8]
```

Les tuples

Les tuples (ou uplets ou, p-uplets ou n-uplets) sont des séquences, tout comme les listes. Les tuples utilisent des **parenthèses .**

```
matieres = ('Physiques', 'Chimie', 'Maths', 4, 2.1, 8)
```

Construction des tuples

Si le tuple n'est constitué que d'un seul element, on rajoute **une virgule (,)** à la fin de l'élément.

```
matieres = ('Physiques',)
```

Pour créer un tuple vide (0 élément), on met les parenthèses mais pas de virgule.

```
matieres = ()
```

Il est possible de créer des tuples par :

- par **concaténation** avec l'opérateur **+**

```
item = (5449000000996, "Coca-Cola 33cl", 0.70)
seul_item = ("Fanta", )
item += seul_item
```

- par **répétition** avec l'opérateur *****

```
nouvel_item = 4* item
```

- par **Compréhension** avec la boucle **for** ... **in**

```
mon_tuple = tuple(i*i for i in range(10))
```

- avec la fonction **tuple()**

```
mon_tuple = tuple ('Timoleon')
```

Longueur d'un tuple

Pour connaitre la longueur d'un tuple on utilise la fonction **len**

```
item = (5449000000996, "Coca-Cola 33cl", 0.70)

# Taille d'un tuple
taille_item = len(item)
```

Indexation

Comme les listes, chaque élément d'un tuple est identifiable par un **Index ou indice**. Le premier élément porte l'**indice [0].**

```
# Lecture d’un élément  à partir de son index

item = (5449000000996, "Coca-Cola 33cl", 0.70)
print('Les valeurs de chaque élément du tuple : ')
print(item[0])  # index 0
print(item[1])  # index 1
print(item[2])  # index 2
```

Appartenance

Nous pouvons tester si un élément existe dans un tuple ou non, en utilisant les mot-clés :

- **in** pour vérifier l'appartenance d'un élément à un tuple
- **not in** pour vérifier la non-appartenance de l'élément

```
item = (5449000000996, "Coca-Cola 33cl", 0.70, "Fanta",("Orangina","Sprit"))

# Sortie: True
print('Fanta' in item)

# Sortie: False
print('Oasis' in item)

# Not in tuple
# Sortie: True
print('g' not in item)
```

Parcours complet d’un tuple

Avec la notation indicielle, on peut effectuer des traitements sur chacun des éléments d’un tuple à l’aide d’une boucle non conditionnelle.

```
item = (5449000000996, "Coca-Cola 33cl", 0.70)
for i in range (len (item)):
        print (item[i])
```

Méthodes de tuple

Les méthodes qui ajoutent ou suppriment des éléments ne sont pas disponibles avec les tuples. Seules les deux méthodes suivantes sont disponibles.

- **count(x)** : Renvoie le nombre d'éléments x
- **index(x)** : Renvoie l'index du premier élément égal à x

```
item = (5449000000996, "Coca-Cola 33cl", 0.70, "Fanta",("Orangina","Sprit","Coca-Col
a 33cl"))

print(item.count("Fanta"))                                  # Sortie: 2
print(item.index(("Orangina","Sprit","Coca-Cola 33cl")))  # Sortie: 4
```

Les tuples nommés

Lorsqu'on a un tuple hétérogène dont chaque élément représente un **attribut** différent, on utilise un **tuple nommé**.

Un tuple nommé est créé avec la librairie **namedtuple** en déclarant un nom et une liste **d'attributs**.

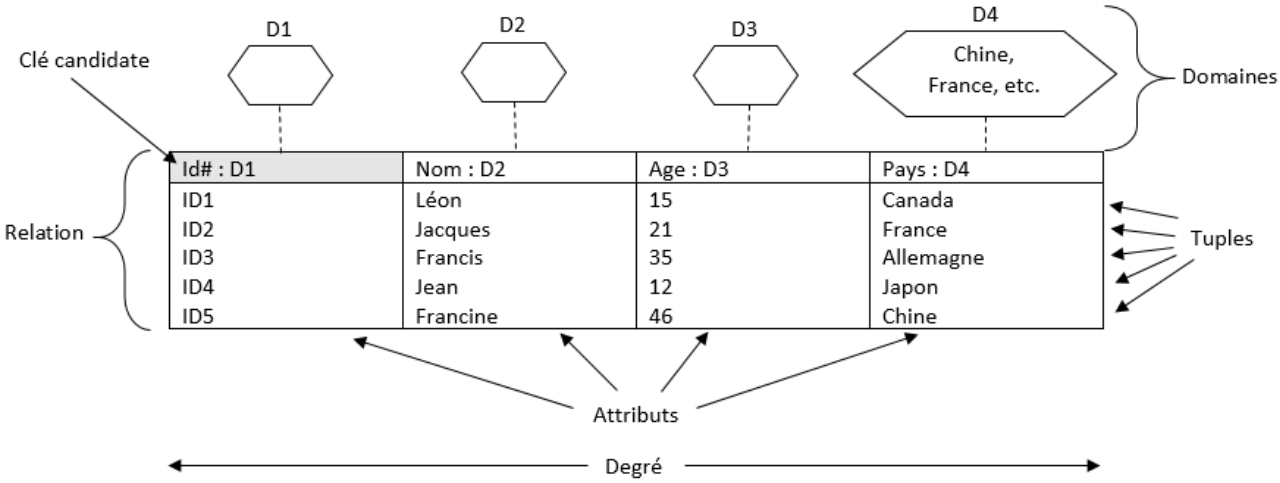
```
from collections import namedtuple

Item = namedtuple('Item', ['code_barre', 'description', 'prix'])

coca = Item(5449000000996, "Coca-Cola 33cl", 0.70)
```

Avantages et applications

Étant donné que les tuples sont assez similaires aux listes, les deux sont également utilisés dans des situations similaires.
Cependant, il existe certains avantages à implémenter un tuple sur une liste. Voici quelques-uns des principaux avantages :

- Étant donné que les tuples sont **immuables**, l'itération via tuple est plus rapide qu'avec la liste. Il y a donc une légère amélioration des performances.
- Si vous avez des données qui ne changent pas, leur implémentation en tant que tuple garantira qu'elles restent **protégées** en écriture.
- Les tuples qui contiennent des éléments **immuables** peuvent être utilisés comme **clé** pour un dictionnaire. Avec des listes, ce n'est pas possible.

Dans un modèle relationnel d'une base de données, un **tuple** est un enregistrement (une ligne d'une table).



Source : Wikipédia