Les tuples

Les tuples, c'est quasiment la même chose que les listes. La différence principale, c'est qu'on ne peut ni ajouter ni enlever d'éléments à un tuple.

Pourquoi utiliser un tuple plutôt qu'une liste alors ?

Pour des questions de rapidité, principalement !

Les tuples, étant plus restreints en termes de fonctionnalités que les listes, prennent ainsi moins de place dans la mémoire de votre ordinateur.

Si vous savez que la taille de vos listes auront un nombre fixe d'éléments, et que vous allez devoir gérer des millions de listes, il peut être préférable de passer par des tuples pour alléger votre programme.

Pour définir un tuple, la syntaxe est similaire aux listes sauf qu'on utilise les parenthèses au lieu des crochets :

```
1. mon_tuple = (1, 2, 3)
```

Comme les listes, un tuple peut contenir des éléments de différents types :

```
1. mon_tuple = (250, "Python", True)
```

Toutes les méthodes que l'on va voir dans la prochaine partie pour ajouter et enlever des éléments à une liste ne seront donc cependant pas applicable sur un tuple.

Heureusement, il est possible de convertir un tuple en liste et vice-versa grâce aux fonctions `list` et `tuple`:

```
1. >>> mon_tuple = (1, 2, 3)
2. >>> liste = list(mon_tuple)
3. [1, 2, 3]
4. >>> mon_tuple = tuple(liste)
5. (1, 2, 3)
```

Donc pas de panique, vous pouvez tout de même modifier un tuple, en le transformant en liste puis en le transformant de nouveau en tuple. Ce n'est pas très optimal, mais c'est possible.

Pour plus d'informations et d'exemples sur les tuples, je vous conseille d'aller lire le glossaire associé sur **Docstring** :