Représentation des données - Types construits

Introduction

Comme nous l'avons vu, une donnée peut être représentée par :

- Un nombre (entier ou réel)
- Une chaîne de caractères
- Un booléen (Vrai ou Faux)

Ces représentations sont dites de types simples.

- Un entier –> type int (Integer)
- Un réel –> type float (Float)
- Un booléen –> type bool (boolean)
- Une chaîne de caractère –> type str (String)

Le type str est aussi de type simple mais il est particulier :
Chaque caractère d'une chaîne a un indice, et ce système d'indice permet d'accéder à une partie de la chaîne.
Par exemple : `ch="bonjour"`
`ch[0]` renvoie `'b'` et `ch[3:7])` renvoie `'jour'`

Avec ces types nous pouvons définir des variables qui représentent chacune une donnée. Cela est suffisant si nous n'utilisons qu'un petit nombre de données...
Mais si le nombre de données est important ou si nous avons besoin d'en regrouper certaines (comme les coordonnées d'un point)
Nous aurons alors besoin de définir des variables dont les valeurs sont des ensembles de valeurs.

Nous allons voir trois types d'objets : **Le type tuple (n-uplet)** , **le type list (liste)** et **le type dict (dictionnaire)**

Pour faire simple, un objet peut être simple ou construit (composé de plusieurs objets)
Un exemple :
Un objet 'train' peut être composé de deux objets 'trains' (accrochés entre eux) et chacun d'entre eux est composé d'une locomotive, de wagons,etc.
Chaque wagon étant composé d'objets...

Un premier modèle - le n-uplet ou tuple

On reprend les propriétés d'une chaîne de caractères :

- Les éléments sont ordonnés, chacun avec un indice
- On ne peut pas modifier les éléments par affectation

Et contrairement aux chaînes de caractères, les éléments d'un tuple peuvent être de n'importe quel type (entiers, booléens, chaînes de caractères, d'autres tuples, des listes ...)

Un tuple se définit avec des **parenthèses** : `mon_tuple=(1,2,3,'bonjour')` les éléments sont séparés par une virgule.

Un second modèle - Les listes

Les listes possèdent les propriétés suivantes :

- Les éléments sont ordonnés, chacun avec un indice
- **On autorise la modification d'un élément**
- Les éléments d'une liste peuvent être de tout type

L'utilisation de liste offre davantage de souplesse, du fait que la modification des éléments est possible

Une liste se définit avec des **crochets** : `ma_liste=[1,2,3,'bonjour']` les éléments sont séparés par une virgule.

Un troisième modèle - Les dictionnaires

Les deux modèles précédents ainsi que les objets de type str ont des éléments indexés par une suite d'entiers. **Ce sont des séquences**
Les dictionnaires ont les propriétés suivantes :

- Les éléments sont non ordonnés
- On autorise la modification d'un élément
- Les éléments d'un dictionnaire sont **indexés par une clé**, qui peut être autre que numérique. Un élément de dictionnaire est une paire : clé - valeur
- Les valeurs dans un dictionnaire peuvent être de tout type

Les dictionnaires sont très utiles pour manipuler des données du type :
félin : (lion, jaguar, tigre,..), reptile : (alligator, lézard, ...) etc.

Un dictionnaire se définit avec des **accolades** : `mon_dico={cle1 : val1,cle2:val2,cle3:val3}` les éléments sont séparés par une virgule.

Quel modèle choisir ?

Il n'y a pas de règles générales qui imposent tel ou tel modèle...
On peut :

- Choisir un tuple, car les éléments qu'il contient n'ont pas vocation à être modifiés
- Choisir une liste, pour la souplesse de son utilisation
- Choisir un dictionnaire car c'est la meilleure représentation des données à traiter

De plus, chacun de ces modèles peut contenir des éléments d'un autre modèle...

- Une liste de tuples...
- Un dictionnaire dont les valeur sont d'autres dictionnaires, des listes...
- À peu près n'importe quelle combinaison est envisageable...

Dans tous les cas le meilleur modèle est celui qui est le plus adapté au problème traité.