NSI 1ère – Mr Fleury – Mars 2020

Représentation des données : Types construits

Cette fiche de révision appartient au chapitre « Représentation des données ». Les notions suivantes sont abordées : les objets de types Tuple, Liste et Dictionnaire

Un objet de type construit est un objet composé de plusieurs objets comme par exemple :

- le type **tuple** (n-uplets) :
  - Les éléments sont ordonnés, chacun avec un indice
  - On ne peut pas modifier les éléments par affectation
- le type **list** (liste) :
  - Les éléments sont ordonnés, chacun avec un indice
  - On autorise la modification d'un éléments
- le type **dict** (dictionnaire) :
  - Les éléments sont indexés par des 'clés' et modifiables
  - On autorise la modification des éléments

Un objet de type tuple, un p-uplet, est une suite ordonnée d'éléments qui peuvent être chacun de n'importe quel type. Exemple de tuple :

Lycee = (1, "60", "Beauvais")

Les parenthèses ne sont pas obligatoires mais sont mises par convention.

Lycee[0] retournera 1 et ***Lycee[1]*** retournera "60"

Le tuple permet de renvoyer plusieurs valeurs lors du retour d'une fonction.

Généralement un tuple contient des valeurs qui ne sont pas susceptibles d'être modifiées.

L'opération de concaténation est possible sur les tuples.

Un objet de type liste, est une suite ordonnée d'éléments qui sont séparés par des virgules et entourés par des crochets. Dans certain langages on parle de **tableaux** à la place de listes.

Exemple de liste :

- une liste vide: ***titi = []***
- un tableau d'entiers: t1 ***= [1, 2, 3, 4, 5]***
- un tableau de caractères: ***couleur = ['bleu', 'rouge','vert']***
- un tableau de tableau: t2 = ***[[0, 2, 4, 6, 8],[1,3,5,7,9]]***
- un tableau de tuples: t3 = ***[(1, "60", "Beauvais"),(2, "80", "Amiens")]***

Les éléments d'une liste sont indexés par les entiers 0, 1, 2,. . . , n – 1.

l'appel de ***t1[2]*** retournera 3, l'appel de ***t2[1][2]*** retournera 5

Une liste est une variable mutable dans le sens où il est possible de modifier son contenu.

ATTENTION :Deux tableaux qui sont liés par une relation d'égalité pointent vers la même adresse mémoire. La modification de l'un entraîne la modification de l'autre.

la **méthode append** (qui signifie "ajouter" en anglais): permet d'ajouter un élément à la fin d'une liste:après une instruction ***t1.append(10)*** l'appel de t1 retournera ***[1, 2, 3, 4, 5,10]***

t1.remove[10] supprimera l'élément qui a pour valeur 10 dans la liste.

Pour connaître le nombre d'éléments d'une liste on utilise la méthode ***len.***

Pour connaître le nombre d'occurrence d'une valeur d'une liste on utilise la méthode ***count.***

On peut utiliser les listes pour facilement construire les ***Files*** ou ***Piles*** par exemple en Python ou

bien encore les ***matrices*** (liste à 2 dimensions) pour stocker des matrices de graphe.

<u>Un objet de type dictionnaire</u>, les indices sont remplacés par des objets du type str, float, tuple. On les appelle des clés et à chaque **clé** correspond une **valeur**. Ces clés ne sont pas ordonnées.

Exemple de dictionnaire : ***dico = {'Bleu': 0, 'Blanc': 1, 'Rouge': 2}***

Pour accéder aux éléments Clés et Valeur, nous avons les méthodes ***keys, values*** *et* ***item***

- l'appelle de ***dico.keys()*** *retournera* ***dict_keys(['Bleu'], ['Blanc'], ['Rouge'])***
- l'appelle de ***dico.values()*** *retournera* ***dict_values([0, 1, 2])***
- l'appelle de ***dico.items()*** *retournera* ***dict_items([('Bleu'], 0), ('Blanc'], 0), ('Rouge'], 0)])***

On en déduit trois manières de parcourir un dictionnaire. Il est possible d'utiliser les clés, les valeurs, ou les couples clés-valeurs.

Pour parcourir le dictionnaire avec les clés en utilisant une boucle :

```
for keys in dico.keys() :
    print(key)
```

Pour parcourir le dictionnaire avec les clés et valeurs en utilisant une boucle :

```
for keys,values in dico.items() :
    print(key, values)
```

Pour copier un dictionnaire, les comportements sont similaires à ceux rencontrés avec les listes en particulier si les valeurs sont des listes. Il est donc conseillé d'utiliser la fonction ***deepcopy*** du module ***copy*** pour être certain d'obtenir une "vraie" copie.

CE QU'IL FAUT RETENIR

De nombreuses fonctions sont communes aux 3 types construits, comme par exemple la fonction ***sorted***, il en existe beaucoup d'autres.

Pour choisir le type construit adéquat cela dépendra du type de données que l'on veut stocker.

Si nous avons besoin de garder en mémoire un grand nombre de valeurs comme dans le cas d'un traitement de données statistiques, il est préférable d'utiliser les types construits.

Il faut faire attention à la copie avec les types construits et utiliser les méthodes existantes pour l'effectuer.