Structures de données

1. Généralités

En informatique on manipule essentiellement des données. Lorsqu'elles sont simples comme des nombres, des chaînes de caractères ou des booléens, on peut les stocker dans des variables qui seront typées en fonction de la nature de la donnée.
Par exemple, avec les affectations suivantes :

```
n = 4
p = -5
r = 0.7
phrase = "hello"
test = True
```

On obtient en retour les types des variables (Python est un langage où le typage est dynamique, c'est à dire que le type de la variable est automatiquement déterminé la première fois qu'elle est déclarée) :

```
>>> type(n)
<class 'int'>
>>> type(p)
<class 'int'>
>>> type(r)
<class 'float'>
>>> type(phrase)
<class 'str'>
>>> type(test)
<class 'bool'>
```

Lorsque l'on a besoin de manipuler un grand nombre de données, qu'elles soient de même type ou pas, on utilise des structures de données comme les tuples, les listes, les tableaux ou les dictionnaires.
Les chaînes de caractères, les tuples et les listes sont des séquences, c'est à dire des suites ordonnées d'éléments indexés par une suite d'entiers.

- Chaînes de caractères et tuples sont non mutables (on ne peut les modifier) ;
- Les listes sont mutables ;
- Un tableau est une liste de liste ;
- Tuples et listes peuvent contenir des éléments de n'importe quel type, y compris d'autres tuples, listes ou encore des dictionnaires...
- Chacun de ces types dits construits possède un certain nombre de méthodes qui permettent d'agir sur l'objet (ajout, suppression, tris...).

Dans un dictionnaire les éléments sont indexés par des clés et sont modifiables.

2. Exemple : moyenne d'un élève

L'exercice consiste à implémenter une fonction qui calcul la moyenne d'un élève.
Il faut stocker dans une seule variable, pour un élève les données suivantes :
Nom : Térieur
Prénom : Alain
Date de naissance : 01/01/2000
Programmation : 12
Algorithmique : 10
Projet : 15

2.1. Première solution : structure de données tuple (les données ne seront pas modifiables) :

```
eleve1 = ("Térieur", "Alain", "01/01/2000", 12, 10, 15)
```

La fonction **moyenne_tuple(eleve)** et le programme utilisant cette fonction peut être le suivant :

```
def moyenne_tuple(eleve):
    """calcule et retourne la moyenne des notes
    eleve : tuple
    moy : float
    """
    moy = (eleve[3] + eleve[4] + eleve[5])/3
    return moy

eleve1 = ("Térieur", "Alain", "01/01/2000", 12, 10, 15)
print(f'moyenne de {eleve1[1]} {eleve1[0]} : {moyenne_tuple(eleve1)}')
```

Le résultat obtenu est le suivant :

```
moyenne de Alain Térieur : 12.333333333333334
```

2.2. Deuxième solution : structure de données tuple contenant des listes (modifiables) :

```
eleve1 = (["Térieur", "Alain", "01/01/2000"], [12, 10, 15])
```

✍ Implémenter la fonction **moyenne_liste(eleve)** qui renvoie la moyenne de **eleve** avec le programme qui permet d'obtenir le résultat ci-dessous. La variable **eleve** a le format ci-dessus (tuple contenant des listes).

```
moyenne de Alain Térieur : 12.333333333333334
```

2.3. Troisième solution : structure de données dictionnaire (modifiable) :

```
eleve1 = {"Nom": "Térieur", "Prenom": "Alain", "Date": "01/01/2000", "Programmation": 12,
"Algorithmique": 10, "Projet": 15}
```

✍ Implémenter la fonction **moyenne_dict(eleve)** qui renvoie la moyenne de **eleve** avec le programme qui permet d'obtenir le résultat ci-dessous. La variable **eleve** a le format ci-dessus (dictionnaire).

```
moyenne de Alain Térieur : 12.333333333333334
```

2.4. Autre solution

Les données peuvent aussi écrite sous forme de liste, de liste de listes, de liste de tuples...

3. Abstraction et interface

Dans l'exemple ci-dessus, pour un même cahier des charges (calcul d'une moyenne), plusieurs implémentations ont été proposées.

Cet exemple montre le principe de l'abstraction des structures de données et des interfaces. Les modules proposés permettent de manipuler des notes sans avoir à connaître leur implémentation en interne. C'est l'abstraction des structures de données. La description des fonctions à la disposition des utilisateurs s'appelle l'interface. L'intérêt pour l'utilisateur, c'est qu'il n'est pas nécessaire de connaître les détails internes, ni le code des fonctions. Lorsque vous utilisez la tortue Python, vous n'avez pas besoin de savoir comment elle est représentée en mémoire ou comment est programmée la fonction **forward**. Vous avez juste besoin de savoir qu'elle existe et à quoi elle sert.

4. Moyenne de la classe et des matières

Soit une classe constituée des élèves suivants :

Nom : Térieur Prénom : Alain Date : 01/01/2000 Programmation : 12 Algorithmique :10 Projet : 15	Nom : Onette Prénom : Camille Date : 01/07/2004 Programmation : 7 Algorithmique :14 Projet : 11	Nom : Oma Prénom : Modeste Date : 01/11/2002 Programmation : 13 Algorithmique :8 Projet : 17

La classe est stockée dans une variable :
`classe1 = [eleve1, eleve2, ...]` (ou un tuple ou encore un dictionnaire...).

✍ À partir d'une structure de données choisies, implémenter un programme et les fonctions adéquates afin de :

- calculer et stocker les moyennes des élèves ;
- calculer et stocker la moyenne de chaque matière ;
- et d'obtenir le résultat ci-dessous.

Aide : il est possible de créer une variable pour les moyennes de chaque élève et pour les moyennes de chaque matière.

```
Moyenne de Alain Térieur : 12.3
Moyenne de Camille Onette : 10.7
Moyenne de Modeste Oma : 12.7

La moyenne en Programmation est de 10.7
La moyenne en Algorithmique est de 10.7
La moyenne en Projet est de 14.3
```

✍ Pour les plus rapides, refaire le programme et les fonctions en choisissant une autre structure de données.

Source : http://nsiterminale.janviercommelemois.fr/fichiers_pdf/feuille-interfaces.pdf
https://isn-icn-ljm.pagesperso-orange.fr/NSI-TLE/res/res_le_cours_rappels.pdf