Python : les séquences (tuples et tableaux):

1. **Les séquences en Python:**
   Il est possible de "stocker" plusieurs grandeurs dans une même structure, ce type de structure est appelé une séquence. De façon plus précise, nous définirons une séquence comme un ensemble fini et ordonné d'éléments indicés de 0 à n-1 (si cette séquence comporte n éléments). Rassurez-vous, nous reviendrons ci-dessous sur cette définition. Nous allons étudier plus particulièrement 2 types de séquences : les tuples et les tableaux (il en existe d'autres que nous n'évoquerons pas ici).

2. **Les tuples en Python:**
   Comme déjà dit ci-dessus, un tuple est une séquence. Voici un exemple très simple :

```
mon_tuple = (5, 8, 6, 9)
```

   Dans le code ci-dessus, la variable "mon_tuple" référence un tuple, ce tuple est constitué des entiers 5, 8, 6 et 9. Comme indiqué dans la définition, chaque élément du tuple est indicé (il possède un indice):

   - le premier élément du tuple (l'entier 5) possède l'indice 0
   - le deuxième élément du tuple (l'entier 8) possède l'indice 1
   - le troisième élément du tuple (l'entier 6) possède l'indice 2
   - le quatrième élément du tuple (l'entier 9) possède l'indice 3

   Comment accéder à l'élément d'indice i dans un tuple ? Simplement en utilisant la "notation entre crochets" :

   Exercice 1: Testez le code suivant :

```
mon_tuple = (5, 8, 6, 9)
a = mon_tuple[2]
```

   Quelle est la valeur référencée par la variable a (utilisez la console pour répondre à cette question)? La variable mon_tuple référence le tuple (5, 8, 6, 9), la variable a référence l'entier 6 car cet entier 6 est bien le troisième élément du tuple, il possède donc l'indice 2.

   ATTENTION : dans les séquences les indices commencent toujours à 0 (le premier élément de la séquence a pour indice 0), oublier cette particularité est une source d'erreur "classique".

Exercice 2: Complétez le code ci-dessous (en remplaçant les ..) afin qu'après l'exécution de ce programme la variable a référence l'entier 8.

```
mon_tuple = (5, 8, 6, 9)
a = mon_tuple[..]
```

Un tuple ne contient pas forcément des nombres entiers, il peut aussi contenir des nombres décimaux, des chaînes de caractères, des booléens...

Exercice 3: Quel est le résultat attendu après l'exécution de ce programme ? Vérifiez votre hypothèse en testant ce programme.

```
mon_tuple = ("le", "monde", "bonjour")
print(mon_tuple[2] + " " + mon_tuple[0] + " " + mon_tuple[1] + "!")
```

Grâce au tuple, une fonction peut renvoyer plusieurs valeurs :

Exercice 4: Analysez puis testez le code suivant :

```
def add(a, b):
        c = a + b
        return (a, b, c)
mon_tuple = add(5, 8)
print(f"{mon_tuple[0]} + {mon_tuple[1]} = {mon_tuple[2]}")
```

Il faut bien comprendre dans l'exemple ci-dessus que la variable mon_tuple référence un tuple (puisque la fonction "add" renvoie un tuple), d'où la "notation entre crochets" utilisée avec mon_tuple (mon_tuple[1]...). La console permet d'afficher les éléments présents dans un tuple simplement en :

Exercice 5: Après avoir exécuté le programme ci-dessous, saisissez mon_tuple dans la console.

```
mon_tuple = (5, 8, 6, 9)
```

Il est possible d'assigner à des variables les valeurs contenues dans un tuple :

Exercice 6:

```
a, b, c, d = (5, 8, 6, 9)
```

Quelle est la valeur référencée par la variable a ? La variable b ? La variable c ? La variable d ? Vérifiez votre réponse à l'aide de la console Python.

3. **Les tableaux en Python:**
ATTENTION : Dans la suite nous allons employer le terme "tableau". Pour parler de ces "tableaux" les concepteurs de Python ont choisi d'utiliser le terme de "list" ("liste" en français). Pour éviter toute confusion, notamment par rapport à des notions qui seront abordées en terminale, le choix a été fait d'employer "tableau" à la place de "liste" (dans la documentation vous rencontrerez le terme "list", cela ne devra pas vous pertuber).

Il n'est pas possible de modifier un tuple après sa création (on parle d'objet "immutable"), si vous essayez de modifier un tuple existant, l'interpréteur Python vous renverra une erreur. Les tableaux sont,comme les tuples, des séquences, mais à la différence des tuples, ils sont modifiables (on parle d'objets "mutables").

Pour créer un tableau, il existe différentes méthodes : une de ces méthodes ressemble beaucoup à la création d'un tuple :

```
mon_tab = [5, 8, 6, 9]
```

Notez la présence des crochets à la place des parenthèses. Un tableau est une séquence, il est donc possible de "récupérer" un élément d'un tableau à l'aide de son indice (de la même manière que pour un tuple).

Exercice 7: Quelle est la valeur référencée par la variable ma_variable après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse).

```
mon_tab = [5, 8, 6, 9]
ma_variable = mon_tab[2]
```

N.B. Il est possible de saisir directement mon_tab[2] dans la console sans passer par l'intermédiaire de la variable ma_variable. Il est possible de modifier un tableau à l'aide de la "notation entre crochets" :

Exercice 8: Quel est le contenu du tableau référencé par la variable mon_tab après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse).

```
mon_tab = [5, 8, 6, 9]
mon_tab[2] = 15
```

Comme vous pouvez le constater avec l'exemple ci-dessus, l'élément d'indice 2 (le nombre entier 6) a bien été remplacé par le nombre entier 15. Il est aussi possible d'ajouter un élément en fin de tableau à l'aide de la méthode "append" :

Exercice 9: Quel est le contenu du tableau référencé par la variable mon_tab après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
mon_tab = [5, 8, 6, 9]
mon_tab.append(15)
```

L'instruction "del" permet de supprimer un élément d'un tableau en utilisant son index :

Exercice 10: Quel est le contenu du tableau référencé par la variable mon_tab après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
mon_tab = [5, 8, 6, 9]
del mon_tab[1]
```

La fonction "len" permet de connaitre le nombre d?éléments présents dans une séquence (tableau et tuple)

Exercice 11: Quelle est la valeur référencée par la variable nb_len après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
mon_tab = [5, 8, 6, 9]
nb_ele = len(mon_tab)
```

Une petite parenthèse : on pourrait s'interroger sur l'intérêt d'utiliser un tuple puisque le tableau permet plus de choses ! La réponse est simple : les opérations sur les tuples sont plus "rapides". Quand vous savez que votre tableau ne sera pas modifié, il est préférable d'utiliser un tuple à la place d'un tableau.

4. **la boucle "for" : parcourir les éléments d'un tableau:**
La boucle for... in permet de parcourir chacun des éléments d?une séquence (tableau ou tuple) :

Exercice 12: Analysez puis testez le code suivant :

```
mon_tab = [5, 8, 6, 9]
for element in mon_tab:
        print(element)
```

Quelques explications : comme son nom l'indique, la boucle "for" est une boucle ! Nous "sortirons" de la boucle une fois que tous les éléments du tableau mon_tab auront été parcourus. element est une variable qui va :

- au premier tour de boucle, référencer le premier élément du tableau (l'entier 5)
- au deuxième tour de boucle, référencer le deuxième élément du tableau (l'entier 8)
- au troisième tour de boucle, référencer le troisième élément du tableau (l'entier 6)
- au quatrième tour de boucle, référencer le quatrième élément de le tableau (l'entier 9)

Une chose importante à bien comprendre : le choix du nom de la variable qui va référencer les éléments du tableau les uns après les autres (element) est totalement arbitraire, il est possible de choisir un autre nom sans aucun problème, le code suivant aurait donné exactement le même résultat :

```
mon_tab = [5, 8, 6, 9]
for toto in mon_tab:
        print (toto)
```

Dans la boucle for... in il est possible d?utiliser la fonction prédéfinie range à la place d?un tableau d?entiers :

Exercice 13: Analysez puis testez le code suivant :

```
for element in range(0, 5):
        print (element)
```

Comme vous pouvez le constater, "range(0,5)" est, au niveau de la boucle "for..in", équivalent au tableau [0,1,2,3,4], le code ci-dessous donnerait le même résultat que le programme vu dans le "À faire vous-même 12" :

```
mon_tab = [0, 1, 2, 3, 4]
for element in mon_tab:
        print (element)
```

ATTENTION : si vous avez dans un programme "range(a,b)", a est la borne inférieure et b a borne supérieure. Vous ne devez surtout pas perdre de vu que la borne inférieure est incluse, mais que la borne supérieure est exclue.

Il est possible d'utiliser la méthode "range" pour "remplir" un tableau :

Exercice 14: Quel est le contenu du tableau référencé par la variable mon_tab après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
mon_tab = []
for element in range(0, 5):
        mon_tab.append(element)
```

5. **Créer un tableau par compréhension:**
   Nous avons vu qu'il était possible de "remplir" un tableau en renseignant les éléments du tableau les uns après les autres :

```
mon_tab = [5, 8, 6, 9]
```

ou encore à l'aide de la méthode "append". Il est aussi possible d'obtenir exactement le même résultat qu'à l'exercice 13 en une seule ligne grâce à la compréhension de tableau :

Exercice 15: Quel est le contenu du tableau référencée par la variable mon_tab après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
mon_tab = [p for p in range(0, 5)]
```

Les compréhensions de tableau permettent de rajouter une condition (if) :

Exercice 16: Quel est le contenu du tableau référencé par la variable mon_tab après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
l = [1, 7, 9, 15, 5, 20, 10, 8]
mon_tab = [p for p in l if p > 10]
```

autre possibilité, utiliser des composants "arithmétiques" :

Exercice 17: Quel est le contenu du tableau référencé par la variable mon_tab après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
l = [1, 7, 9, 15, 5, 20, 10, 8]
mon_tab = [p**2 for p in l if p < 10]
```

Rappel : p**2 permet d?obtenir la valeur de p élevée au carrée

Comme vous pouvez le remarquer, nous obtenons un tableau (mon_tab) qui contient tous les éléments du tableau l élevés au carré à condition que ces éléments de l soient inférieurs à 10. Comme vous pouvez le constater, la compréhension de tableau permet d'obtenir des combinaisons relativement complexes.

6. **Travailler sur des "tableaux de tableaux":**
   Chaque élément d'un tableau peut être un tableau, on parle de tableau de tableau. Voici un exemple de tableau de tableau :

```
m = [[1, 3, 4], [5 ,6 ,8], [2, 1, 3], [7, 8, 15]]
```

Le premier élément du tableau ci-dessus est bien un tableau ([1, 3, 4]), le deuxième élément est aussi un tableau ([5, 6, 8])... Il est souvent plus pratique de présenter ces "tableaux de tableaux" comme suit :

```
m = [[1, 3, 4],
     [5, 6, 8],
     [2, 1, 3],
     [7, 8, 15]]
```

Nous obtenons ainsi quelque chose qui ressemble beaucoup à un "objet mathématique" très utilisé : une matrice

Il est évidemment possible d'utiliser les indices de position avec ces "tableaux de tableaux". Pour cela nous allons considérer notre tableau de tableaux comme une matrice, c'est à dire en utilisant les notions de "ligne" et de "colonne". Dans la matrice ci-dessus :

Pour cibler un élément particulier de la matrice, on utilise la notation avec "doubles crochets" : m[ligne][colonne] (sans perdre de vu que la première ligne et la première colonne ont pour indice 0)

Exercice 18: Quelle est la valeur référencée par la variable a après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
m = [[1, 3, 4],
     [5, 6, 8],
     [2, 1, 3],
     [7, 8, 15]]
a = m[1][2]
```

Comme vous pouvez le constater, la variable a référence bien l'entier situé à la 2e ligne (indice 1) et à la 3e colonne (indice 2), c'est-à-dire 8.

Exercice 19: Quel est le contenu du tableau référencé par la variable mm après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
m = [1, 2, 3]
mm = [m, m, m]
m[0] = 100
```

Comme vous pouvez le constater, la modification du tableau référencé par la variable m entraine la modification du tableau référencé par la variable mm (alors que nous n'avons pas directement modifié le tableau référencé par mm). Il faut donc être très prudent lors de ce genre de manipulation afin d'éviter des modifications non désirées.

Il est possible de parcourir l'ensemble des éléments d'une matrice à l'aide d'une "double boucle for" :

Exercice 20: Analysez puis testez le code suivant :

```
m = [[1, 3, 4],
     [5, 6, 8],
     [2, 1, 3],
     [7, 8, 15]]
nb_colonne = 3
nb_ligne = 4
for i in range(0, nb_ligne):
        for j in range(0, nb_colonne):
                a = m[i][j]
                print(a)
```