Spécialité NSI

LYCÉE de l'ÉLORN
le choix de votre réussite

Cours -TP
Les Listes et les Tuples en Python

I- Les listes

1.1 Définition

Une liste est une structure de données qui contient une série de valeurs. Python autorise la construction de listes contenant des valeurs de types différents (par exemple entier et chaîne de caractères, listes, tuples…). Elle est déclarée par une série de valeurs séparées par des virgules, et le tout encadré par des crochets.

Exemples :

```
>>> mes_évaluations=["Très bien", "Moyen", "Passable", "Excellent travail"]
>>> mes_notes=[10, 5, 12.5, 16, 5.5, 13]
>>>
>>> print(mes_notes)
  [10, 5, 12.5, 16, 5.5, 13]
>>> type(mes_notes)
<class 'list'>
```

1.2 Utilisation, opérations courantes

Les listes sont

C'est un objet **itérable** capable de renvoyer ses membres un par un, ce qui lui permet d'être itéré dans une boucle for.

```
>>> liste = [10,11,12,'a','b','c']
>>> for v in  liste:
        print(v)


10
11
12
a
b
c
```

```
>>> liste = [10,11,12,'a','b','c']
>>> for i,v in enumerate(liste, start=2):
        print(i,v)


2 10
3 11
4 12
5 a
6 b
7 c
```

On peut appeler les éléments d'une liste par leur position. Ce numéro est appelé indice (ou index) de la liste. Le 1er élément a l'indice 0.

```
>>> len(mes_notes)
6
>>> mes_notes[0]
10
>>> mes_évaluations[2]
'Passable'
```

L'indexage peut être positif ou négatif.
L'indice -1 pointe le dernier élément.

```
>>> mes_notes=[10, 5, 12.5, 16, 5.5, 13]
>>> mes_notes[-1]
13
>>> mes_notes[-2]
5.5
```

Comme les chaînes de caractères, les listes supportent l'opérateur + de concaténation, ainsi que l'opérateur * pour la duplication.

```
>>> mes_notes=[10, 5, 12.5, 16, 5.5, 13]
>>> mes_nouvelles_notes=[10, 15, 8, 6.5, 14, 18]
>>> toutes_mes_notes = mes_notes + mes_nouvelles_notes
>>> toutes_mes_notes
[10, 5, 12.5, 16, 5.5, 13, 10, 15, 8, 6.5, 14, 18]
>>> mes_notes*2
[10, 5, 12.5, 16, 5.5, 13, 10, 5, 12.5, 16, 5.5, 13]
```

L'opérateur + peut donc ajouter une valeur à une liste mais on utilise généralement la méthode .append() pour ajouter un seul élément à la fin d'une liste.

```
>>> mes_notes=[10, 5, 12.5, 16, 5.5, 13]
>>> mes_notes = mes_notes + [20]
>>> mes_notes
[10, 5, 12.5, 16, 5.5, 13, 20]
>>> mes_notes.append(18.5)
>>> mes_notes
[10, 5, 12.5, 16, 5.5, 13, 20, 18.5]
```

Voici une liste de listes qui peut générer un tableau en deux dimensions de valeurs :

```
>>> l=[["a","b","c"],["d","e","f"],["g","h","i"]]
>>> l[0][0]
'a'
>>> l[1][0]
'd'
>>> l[2][2]
'i'
>>> l[2][2]="Elorn"
>>> l
[['a', 'b', 'c'], ['d', 'e', 'f'], ['g', 'h', 'Elorn']]
```

	0	*1*	*2*
0	"a"	"b"	"c"
1	"d"	"e"	"f"
2	"g"	"h"	"i"

1.3 Le Découpage ou Slicing

Le slicing (saucissonnage) permet d'obtenir une sous-liste depuis une liste. La syntaxe est la suivante :

liste[debut:fin:pas]
liste[debut:fin]
liste[debut:]
liste[:fin]
liste[::pas]
liste[debut::pas]
liste[:fin:pas]

début : indice du 1er élément à sélectionner (par defaut 0)
fin : indice du dernier élément à sélectionner
pas : par défaut 1

```
>>> caractères=["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m"]
>>> caractères[2:5]
['c', 'd', 'e']
>>> caractères[5:]
['f', 'g', 'h', 'i', 'j', 'k', 'l', 'm']
>>> caractères[::2]
['a', 'c', 'e', 'g', 'i', 'k', 'm']
>>> caractères[2::2]
['c', 'e', 'g', 'i', 'k', 'm']
>>> caractères[:5:2]
['a', 'c', 'e']
```

1.4 Tests sur le contenu d'une liste

```
>>> mes_notes=[10, 5, 12.5, 16, 5.5, 13, 20]
>>> 16 in mes_notes
True
>>> 18 in mes_notes
False
>>> if 20 in mes_notes : print("Champagne !")

 Champagne !
```

1.5 Fonctions sur les listes

liste.append(x) : ajoute un élément x.
liste.extend(liste2) : ajoute la liste liste2 à la fin.
liste.insert(i, x) : insert l'élément i à la position x, et décale le reste (comme l[i:i] = x).
liste.remove(x) : enlève le premier élément identique à x (erreur si non trouvé).
liste.pop() : enlève le dernier élément et le renvoie.
liste.pop(0) : enlève le premier élément et le renvoie.
liste.pop(i) : enlève l'élément à la position i et le renvoie.
liste.index(x) : renvoie l'index du 1er élément identique à x (erreur si non trouvé).
liste.count(x) : compte combien de fois l'élément x est dans la liste.
liste.sort() : trie en place les éléments de la liste.
liste.reverse() : renverse la liste en place.

```
>>> mes_notes=[10, 5, 15, 12.5, 16, 5.5, 13, 15, 6.5, 18]
>>> mes_notes.sort()
>>> mes_notes
[5, 5.5, 6.5, 10, 12.5, 13, 15, 15, 16, 18]
>>> mes_notes.count(15)
2
```

Combinaison des fonctions *range()* et *list()*

```
>>> list(range(1,10))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(10,0,-1))
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

II- Les Tuples

1.1 Définition

Les tuples (p-uplet) ressemblent aux listes, mais on ne peut pas les modifier une fois qu'ils ont été créés. On dit qu'un tuple ***n'est pas mutable***. On le définit avec des parenthèses.

```
>>> mes_notes=(10, 5, 15, 12.5, 16, 5.5, 13, 15, 6.5, 18)
>>> type(mes_notes)
<class 'tuple'>
>>> mes_notes[5]
5.5
>>> mes_notes[5:]
(5.5, 13, 15, 6.5, 18)
```

Parfois, les tuples ne sont pas entourés de parenthèses. Ainsi, on peut utiliser la notation suivante :

```
>>> a,b,c=(10,12,14)
>>> a,b,c
(10, 12, 14)
>>> a
10
>>> b
12
>>> c
14
>>> #cela revient à
>>> (a,b,c)=(10,12,14)
```

III- Travail proposé

a) Tester les exemples du cours sur une console Python.

A partir du script suivant :

```
mes_notes=[]    # création d'une liste vide
note=input("Entrer une note : ")
while note != "fin":
    mes_notes.append(note)
    note=input("Entrer une note : ")
print(mes_notes) # affiche la liste saisie
```

b) Réécrire 3 fonctions (sans utiliser les fonctions existantes !) : moyenne(), maxi() et mini() capable de définir la moyenne, la note minimale et la note maximale des éléments contenus dans une liste de nombres. Utiliser ces fonctions dans le script ci-dessus.

c) Créer un module sous le nom de fichier « gestion_des_notes.pi » contenant les trois fonctions précédentes. Importer ce module sur le script du programme principal.

d) Placer le fichier « gestion_des_notes.pi » dans un sous-dossier « mes_modules ». Faites une importation de ce module.

e) Créer un nouveau programme afin de gérer de la même façon les notes de 5 disciplines. Ajouter des modules capables de délivrer les 3 meilleures puis les 3 moins bonnes notes de chaque discipline et de calculer la moyenne générale.