PROGRAMMATION PYTHON

TP2

Dans les exercices qui suivent, nous allons voir

- Les listes
- Les tuples
- Les dictionnaires
- Les ensembles
- Les fonctions

Exercice 1 :

Définir la liste : liste =[17, 38, 10, 25, 72], puis effectuez les actions suivantes :

- Triez et affichez la liste ;
- Ajoutez l'élément 12 à la liste et affichez la liste ;
- Renversez et affichez la liste ;
- Affichez l'indice de l'élément 17 ;
- Enlevez l'élément 38 et affichez la liste ;
- Affichez la sous-liste du 2^e au 3^e élément ;
- Affichez la sous-liste du début au 2^e élément ;
- Affichez la sous-liste du 3^e élément à la fin de la liste ;
- Affichez la sous-liste complète de la liste ;
- Affichez le dernier élément en utilisant un indexage négatif

Exercice 2 :

Initialisez L1 comme une liste vide, et L2 comme une liste de cinq flottants nuls.
Affichez ces listes.
Utilisez la fonction range() pour afficher :

- Les entiers de 0 à 3 ;
- Les entiers de 4 à 7 ;
- Les entiers de 2 à 8 par pas de 2.

Définir chose comme une liste des entiers de 0 à 5 et testez l'appartenance des éléments 3 et 6 à chose.

Exercice 3 :

- Utilisez une liste en compréhension pour ajouter 3 à chaque élément d'une liste d'entiers de 0 à 5.
- Utilisez une liste en compréhension pour ajouter 3 à chaque élément d'une liste d'entiers de 0 à 5, mais seulement si l'élément est supérieur ou égal à 2.
- Utilisez une liste en compréhension pour obtenir la liste ['ad', 'ae', 'bd', 'be', 'cd', 'ce'] à partir des chaînes "abc" et "de".

Indication : utilisez deux boucles for imbriquées.

- Utilisez une liste en compréhension pour calculer la somme d'une liste d'entiers de 0 à 9

Exercice 4 :

Ecrire un programme qui demande à l'utilisateur d'entrer des notes d'élèves.

Si l'utilisateur entre une valeur négative, le programme s'arrête. En revanche, pour chaque note saisie, le programme construit progressivement une liste. Après chaque entrée d'une nouvelle note (et donc à chaque itération de la boucle), il affiche le nombre de notes entrées, la note la plus élevée, la note la plus basse, la moyenne de toutes les notes.

Exercice 5 : Tuples

Les tuples sont des listes dont le contenu n'est pas modifiable (on dit 'immutable' en Python, par opposition aux listes dont le contenu est 'mutable').

Ainsi, tout ce qui a été dit sur les listes s'applique aux tuples, à l'exception des procédures et fonctions destinées à modifier la liste sur laquelle elles portent. On pourra ainsi utiliser les fonctions copy, count et index.
Les tuples sont créés et représentés avec des parenthèses au lieu de crochets.
Les opérations d'accès à un élément ou à une sous-partie d'un tuple utilisent les crochets comme pour les listes.
Leur nom est une généralisation des termes 'couple' (2 éléments), 'triplet' (3 éléments), 'quadruplet' (4 éléments) ...n-uplet ou t-uplet (nombre quelconque n ou t d'éléments).

1. Tester le code ci-dessous

```
monTuple = tuple(range(1,11)) # creation d'un tuple à partir d'un range
print(monTuple)
print( monTuple[0]) # accès à un element avec les crochets
print(monTuple[3:5]) # accès à une sous partie avec les crochets et le slice (:)
print(monTuple[-1]) # accès au dernier élément
print(len(monTuple)) # usage de la function len(gth)
monTuple[3] = 44 # tentative de modification d'un élément => ERREUR !
monTuple = (11,21,31) # si on le réaffecte, on perd le tuple initial
print( monTuple)
```

Exercice 6 : Tuples

En utilisant un tuple, réalisez une fonction ***nom_mois*** qui a un entier compris entre 1 et 12 associe le nom du mois correspondant.

Lorsque **t** est un tuple, l'expression ***x in t*** permet de tester l'appartenance de la valeur de ***x*** à ce tuple. La réponse est booléenne. Il est parfois intéressant d'avoir plus de précisions comme l'indice d'une occurrence de ***x*** dans ***t***, ou bien le nombre d'occurrences.

Réalisez une fonction nommée ***indice*** qui renvoie l'indice de la première occurrence de son premier paramètre dans le tuple passé en second paramètre, si elle existe. Par exemple :

```
In [8]: indice (1, (3, 1, 4, 1, 5, 9, 2))
        # doit renvoyer 1

Out[8]: 1
```

Réalisez une fonction nommée ***nbre_occurrences*** qui renvoie le nombre d'occurrences du premier paramètre dans le tuple passé en second paramètre. Par exemple :

```
In [10]: nbre_occurrences (1, (3,1,4,1,5,9,2))
         # doit renvoyer 2

Out[10]: 2

In [11]: nbre_occurrences (8, (3,1,4,1,5,9,2))
         # doit renvoyer 0

Out[11]: 0
```

Exercice 7 : set

Définir deux ensembles (sets) : X = {a,b,c,d} et Y = {s,b,d}, puis afficher les résultats suivants :

- Les ensembles initiaux ;
- Le test d'appartenance de l'élément 'c' à X ;
- Le test d'appartenance de l'élément 'a' à Y
- Les ensembles X −Y et Y − X ;
- L'ensemble X ∪Y (union) ;
- L'ensemble X ∩Y (intersection).

Exercice 8 : Dictionnaires

Le type dictionnaire (ou tableau associatif) permet de représenter des tableaux structurés.
En effet, à chaque clé un dictionnaire associe une valeur, et cette valeur peut elle-même être une structure de donnée (liste, tuple ou un dictionnaire. . .).
Soit le tableau suivant représentant des informations physico-chimiques sur des éléments simples (température d'ébullition (Te) et de fusion (Tf), numéro (Z) et masse (M) atomique :

Au	T_e/T_f	2970	1063
	Z/A	79	196.967
Ga	T_e/T_f	2237	29.8
	Z/A	31	69.72

Affectez les données de ce tableau à un dictionnaire dico python de façon à pouvoir écrire par exemple :
print dico["Au"]["Z/A"][0] # affiche : 79

Exercice 9

Taper les codes ci-dessous et observer les résultats

1.

```
def function(entree=10,diviseur=2):
     while(entree>0):
         print(entree/diviseur, entree%diviseur)
         entree=entree-1
# Tapez encore une fois <Enter> si vous Ãates en ligne de commande
function()
function(4,2)
function(diviseur=4,entree=12)
```

2.

```
def multiplication(n,p):
    return n*p # pour retourner une valeur
# Tapez encore une fois <Enter> si vous Ãates en ligne de commande
a=multiplication(3,6)
print(a)
```

3.

```
# les valeurs des arguments saisis vont etre stockes dans un t-uple - cf. print(type(args))
def fonction_test(*args):
    print("type de args :",type(args))
    for arg in args:
        print("paramètre:", arg)
        print()
fonction_test()
fonction_test(1)
fonction_test(1,"a")
fonction_test(1,"a",3)
```

Un nombre heureux est un nombre entier qui, lorsqu'on ajoute les carrés de chacun de ses chiffres, puis les carrés des chiffres de ce résultat et ainsi de suite jusqu'à l'obtention d'un nombre à un seul chiffre égal à 1 (un).

Exemple :

N=7 est heureux, puisque :

- $7^2=49$
- $4^2+9^2=97$
- $9^2+7^2=130$
- $1^2+3^2+0^2=10$
- $1^2+0^2=1$

On est arrivé à un nombre d'un seul chiffre qui est égal à 1, donc N=7 est heureux

Travail demandé :

Ecrire une fonction **heureux(nb)** qui permet de déterminer si un nombre entier **nb** est heureux ou non.

Exercice 10

On veut crypter une chaine de caractères données CH dont la taille ne dépasse pas 50 caractères en une chaine résultat Res de la manière suivante : parcourir la chaine CH de gauche à droite en comptant le nombre d'occurrence successives de chaque caractère de la chaine CH, puis de ranger la chaine Res, ce nombre suivi du caractère e question.

Ecrire un programme Python permettant de saisir la chaine CH qui doit être non vide et forméeuniquement par des lettres alphabétiques, puis de former et d'afficher la chaine Res selon le principe décrit précédemment.

Exemple

- Si CH='aaaFyBsssssssssssazz'alors la chaine Res qui seraaffichée est '3a1F1y1B12s1a2z'