LES DONNEES DE TYPES CONSTRUITS

1. Les tuples en Python
2. Les listes en Python
3. Exercices sur les listes

Exercice1 : Retour sur le calcul d'une moyenne

Vous vous souvenez de cet exercice permettant de calculer une moyenne sans savoir au préalable le nombre de notes à saisir.
L'objectif de cet exercice est de reprendre ce programme et de créer la liste des notes qui ont été saisies.

Pour cela vous devrez utiliser quelques commandes sur les listes.
Tout d'abord, il faut créer la liste : « listenotes » qui au départ est vide.
Puis pour chaque note saisie, ajouter la note dans la liste créée ;
Et il faudra aussi penser à retirer la valeur d'arrêt de saisie « -1 » qui n'est pas une note !
Ci-dessous le programme que vous devez modifier
Pour obtenir :

```
>>> moyenne()
Entrer la premiere valeur: 5
valeur suivante ? -1 pour stopper
valeur 10
valeur suivante ? -1 pour stopper
valeur 15
valeur suivante ? -1 pour stopper
valeur 20
valeur suivante ? -1 pour stopper
valeur -1
[5, 10, 15, 20]
```

File Edit Format Run Options Window Help

```
def moyenne():
    S=0
    Stop=0
    n=1
    valeur=int(input("Entrer la premiere valeur: "))
    S=S+valeur
    while Stop!=-1:
        print("valeur suivante ? -1 pour stopper  ")
        valeur=int(input("valeur "))
        if valeur!=-1:
            n=n+1
            S=S+valeur
        else:
            Stop=-1
    M=S/n
    return print("la moyenne des ",n," valeurs est egale a  ",M)
```

D'après : https://pixees.fr/informatiquelycee/n_site/nsi_prem_pythonSequence.html

Exercice2 : liste de la somme des cubes

Reprendre l'exercice « somme_cube » de sorte qu'il retourne la liste des n premieres sommes des cubes ; vous renommerez ce programme « listesomme_cube »

Ci-dessous le programme initial pour obtenir :

```
>>> listesomme_cube(10)
[1, 9, 36, 100, 225, 441, 784, 1296, 2025, 3025]
>>>
```

```
File  Edit  Format  Run  Options  Window  Help
def somme_cube(n):
    s=0
    for k in range(1,n+1):
        s=s+k**3
    return s
```

Exercice3 : Travail sur les listes.

Voici des listes que vous allez saisir avant de travailler dessus.

listeCapitales1=["Berlin","Vienne","Bruxelles","Sofia","Nicosie","Zagreb", "Copenhague","Madrid","Talinn","Helsinki","Paris","Athène","Budapest","Dublin","Rome"]

listeCapitales2=["Riga","Vilnius","Luxembourg","La Valette","Amsterdam","Varsovie","Lisbonne","Prague","Bucarest","Londres","Bratislava","Ljubljana","Stockholm"]

listePays1=["Allemagne","Autriche","Belgique","Bulgarie","Chypre","Croatie","Dannemark", "Espagne","Estonie","Finfande","France","Grèce","Hongrie","Irlande","Italie"]

listePays2=["Lettonie","Lituanie","Luxembourg","Malte","Pays Bas","Pologne","Portugal","République-Tchèque","Roumanie","Royaume-Uni","Slovaquie","Slovénie","Suède"]

Les 2 premières listes sont les capitales respectives des pays qui sont dans les 2 listes suivantes.

D'après : https://pixees.fr/informatiquelycee/n_site/nsi_prem_pythonSequence.html

1. En utilisant la commande « nom_de_liste.expend(autre_liste) », créez une fonction « listeCapitales() » qui retourne une liste « listeCapitales » regroupant toutes les capitales.

2. Créez de même la fonction « listePays() » qui regroupe tous les paysdans une même liste nommée « listePays ».

3. En utilisant la commande « nom_de_liste.sort() », créez une fonction « listeCapitalesAlphabetique() » qui retourne la liste des capitales rangées par ordre alphabétique, nommée : « listeCapitalesAlphabetique ».

4. En utilisant la commande len(nom de liste), créez une fonction « nombreCapitales() » qui après avoir créé une liste regroupant toutes les capitales, indique le nombre de capitales.

5. Créez de même la fonction « nombrePays() » qui donne le nombre des pays contenus dans la liste « listePays »

6. En utilisant la commande « zip(liste1,liste2) », faire afficher la correspondance entre un pays et sa capitale.
   La syntaxe de cette commande est :

```
for i, j in zip(listel,liste2):
    print(i, " ... ", j)
```

```
>>> capitalesPays()
Allemagne  a pour capitale   Berlin
Autriche  a pour capitale   Vienne
Belgique  a pour capitale   Bruxelles
Bulgarie  a pour capitale   Sofia
Chypre  a pour capitale   Nicosie
Croatie  a pour capitale   Zagreb
Dannemark  a pour capitale   Copenhague
Espagne  a pour capitale   Madrid
Estonie  a pour capitale   Talinn
Finfande  a pour capitale   Helsinki
France  a pour capitale   Paris
```

7. En utilisant les boucles vues en python, créez une liste nommée « capitalePays qui donne des tuples qui correspondent au pays et sa capitale respective.

```
>>> capitalePays()
[('Autriche', 'Vienne'), ('Belgique', 'Bruxelles'), ('Bulgarie', 'Sofia'), ('Chy
pre', 'Nicosie'), ('Croatie', 'Zagreb'), ('Dannemark', 'Copenhague'), ('Espagne'
, 'Madrid'), ('Estonie', 'Talinn'), ('Finfande', 'Helsinki'), ('France', 'Paris'
), ('Grèce', 'Athène'), ('Hongrie', 'Budapest'), ('Irlande', 'Dublin'), ('Italie
', 'Rome'), ('Lettonie', 'Riga'), ('Lituanie', 'Vilnius'), ('Luxembourg', 'Luxem
bourg'), ('Malte', 'La Valette'), ('Pays Bas', 'Amsterdam'), ('Pologne', 'Varsov
ie'), ('Portugal', 'Lisbonne'), ('République Tchèque', 'Prague'), ('Roumanie', '
Bucarest'), ('Royaume-Uni', 'Londres'), ('Slovaquie', 'Bratislava'), ('Slovénie'
, 'Ljubljana'), ('Suède', 'Stockholm')]
>>>
```

D'après : https://pixees.fr/informatiquelycee/n_site/nsi_prem_pythonSequence.html

Exercice4 : Travail sur des grilles

Une grille peut être considérée comme une liste dans laquelle chaque élément est lui-même une liste.
Exemple : [[1,2,3],[4,5,6]] est une grille à 2 lignes et 3 colonnes.

- Créer un fichier python que vous renommerez « grille.py »

- Créer une fonction creeGrille(l,c) qui renvoie une grille carrée de taille lxc (l lignes, c colonnes) où tous les éléments sont des 0.

```
>>> creeGrille(4,3)
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

- Créer une fonction afficheGrille(grille) qui renvoie un affichage lisible de la grille.

```
>>> afficheGrille(creeGrille(4,3))
0 0 0
0 0 0
0 0 0
0 0 0
```

On considère la grille suivante : grille=[[1,2,3],[4,5,6],[7,8,9],[2,3,7]]

- Créer une fonction sommeLigne(grille,l) qui renvoie la somme de tous les éléments de la l-ième ligne d'une grille.

```
>>> afficheGrille(grille)
1 2 3
4 5 6
7 8 9
2 3 7
>>> sommeLigne(grille,1)
6
>>> sommeLigne(grille,2)
15
>>> sommeLigne(grille,3)
24
>>> sommeLigne(grille,4)
12
```

- Créer une fonction sommeColonne(grille,c) qui renvoie la somme de tous les éléments de la c-ième colonne d'une grille.

```
>>> afficheGrille(grille)
1 2 3
4 5 6
7 8 9
2 3 7
>>> sommeColonne(grille,1)
14
>>> sommeColonne(grille,2)
18
>>> sommeColonne(grille,3)
25
```

D'après : https://pixees.fr/informatiquelycee/n_site/nsi_prem_pythonSequence.html

On considère la grille carrée suivante :
grille1=[[4,2,3,5],[4,5,6,8],[7,8,9,4],[2,3,7,1]]

- Créer une fonction sommeDiagonale(grille) qui renvoie la somme de tous les éléments de la diagonale principale d'une grille carrée

```
>>> afficheGrille(grille1)
4 2 3 5
4 5 6 8
7 8 9 4
2 3 7 1
>>> sommeDiagonale(grille1)
19
```