Boucles imbriquées

Les fonctions autorisées pour réaliser les différents exercices sont :

- `append()`
- `min(),max()` pour l'exercice 2, fonction ordre3
- `sort()` pour le dernier exercice

Exercice n°1 : Premiers pas

- Écrire un algorithme permettant de calculer : $\sum_{\substack{i=1 \\ j=1}}^{100} \frac{i}{j}$

- Écrire une fonction `commun` qui, étant données deux listes de valeurs numériques `L1` et `L2`, renvoie une liste contenant les éléments communs aux deux listes.

Exercice n°2 : Trier des mots

- Écrire une fonction `ordre1` qui prend en argument une liste de mots et la modifie en les ordonnant par nombre de lettres croissants. La fonction ne renvoie rien.
- Écrire une fonction `voyelle` qui prend en argument une chaîne de caractères et renvoie le nombre de voyelles qu'elle comporte. (conseil : on pourra utiliser un dictionnaire dont les clés seront les différentes voyelles de l'alphabet en majuscule/minuscule et les valeurs, le nombre d'occurences dans la chaîne)
- Écrire une fonction `ordre2` qui prend en argument une liste de mots et la modifie en les ordonnant en fonction du nombre croissant de voyelles qu'ils comportent. La fonction ne renvoie rien.
- Écrire une fonction `ordre3` qui prend en argument une liste de mots et la modifie en les classant par ordre alphabétique. La fonction ne renvoie rien.

Exercice n°3 : un code secret dans Prison Break

Dans la série « Prison Break », le héros, Michael Scofield, transmet un lieu de rendez-vous au docteur Tancredi via un message codé à l'aide de chiffres. L'agent spécial Mahone qui intercepte le message pense tout d'abord à un numéro de téléphone puis à des coordonnées GPS. Il finit par comprendre que, pour coder son message, Scofield a utilisé la correspondance des chiffres et des lettres de l'alphabet sur un clavier de téléphone. Sachant qu'un même chiffre peut correspondre à plusieurs lettres (voir le document ci-après), l'agent Mahone demande à un

collaborateur spécialiste en informatique de lui sortir l'ensemble des mots pouvant être écrit à l'aide de la suite chiffrée du message codé et réussit à établir le lieu de rendez-vous.

Aujourd'hui, c'est à vous de décoder le message chiffré suivant : `59686`

Si vous y parvenez, vous découvrirez le nom de la ville du monde que j'ai préféré visiter.

Conseils :

- Appuyez-vous sur un dictionnaire ayant pour clés les chiffres et pour valeurs, les listes de lettres correspondantes
- Créez une fonction `decodage` prenant en argument un message `code` (au format `str`) de cinq chiffres et qui affiche l'ensemble des combinaisons de caractères possibles.

Exercice n°4 : Trier des points

On dispose de points dans le plan muni d'un repère orthonormé d'origine O. Chaque point a possède un couple de coordonnées (x,y) représenté par la liste `[x,y]`. Il s'agit de trier ces points par rapport à leur distance à O, de la plus petite à la plus grande.

- Écrire une fonction `distance2` qui prend en argument une liste de deux nombres nommée `point` représentant un point P du plan (point est la liste des coordonnées de P) et qui renvoie le carré de la distance euclidienne entre ce point et O.
- Écrire une fonction `compare` qui prend en paramètres deux listes `p1` et `p2` représentant deux points P_1 et P_2 et qui renvoie -1 si P_1 est plus proche de O que P_2, 1 si P_2 est plus proche de O que P_1, 0 si les deux points sont équidistants de O.

- Écrire une fonction `tri_points` qui, pour une liste composée de listes de deux nombres, trie cette liste suivant les distances entre chacun des points et O. (conseil : on pourra utiliser le tri à bulles)

Exercice n°5 : Tri à bulles vs Fonction sort()

L'objectif est de comparer les temps d'exécution du tri à bulles vis à vis de la fonction `sort()` du langage python. On utilisera le code du tri à bulles vu en cours et, pour mesurer le temps, la fonction `time` du module `time` dont le mode d'utilisation est :

```
From time import time
top=time()
#ici le programme à exécuter
print(time() - top)
```

- .Construire une liste `L1` de 100 nombres entiers pris au hasard entre 1 et 100000 , bornes comprises. Pour cela, on pourra utiliser la fonction `randint` du module `random`.

```
From random import randint
randint(a,b)#renvoie un entier aléatoire compris entre a et b
```

- Créer de la même manière deux autres listes `L2` et `L3` contenant respectivement 1000 et 10000 entiers pris au hasard entre 1 et , bornes comprises 100000 .
- Effectuer le tri de chaque liste à l'aide de l'algorithme du tri à bulles tout en affichant les temps d'exécution. Que peut-on conjecturer quant à la complexité de cet algorithme.
- Effectuer maintenant le tri de chaque liste à l'aide de la fonction `sort()` tout en affichant le temps d'exécution. Que dire de l'efficacité de cette fonction par rapport au tri à bulles ? Que peut-on conjecturer quant à sa complexité ?