Langage Python A3 TD2

Partie 1: Python basics

Exo 1 : Conversion d'une chaîne en une liste d'entier

Soit la chaîne ch="12 13 10 8 20", **En une ligne et à l'aide de la compréhension de list**, créez une liste my_list, contenant la conversion en entier de tous les nombres présents dans la variable **ch** (my_list doit être égale à [12 13 10 8 20])

Recréez la liste mais avec la fonction map

Exo 2 : **map avec plusieurs inputs**

Vous avez à votre disposition trois listes de même taille **test**, **values1**, **values2**.
test = [1, 2, 3, 4, 6]
values1 = ['a', 'b', 'c', 'd', 'e']
values2 = ['A', 'B', 'C', 'D', 'E']

En une ligne, en utilisant map, créez un tuple **result** qui contient l'élément de **values1** si le correspondant dans **test** est impair, et l'élément de **values2** sinon. Le résultat attendu est ('a', 'B', 'c', 'D', 'E')

Exo 3 : **Compréhension : puissances de 2**

Vous avez à votre disposition un entier N.

En une ligne, créez une liste **my_list**, contenant les N premières puissances de 2 : 1,2,4,8.....**Exo 4** : **Compréhension : carré des nombres négatifs**

Vous avez à votre disposition une liste **numbers**=[-1,-2,-3,4,5,10,-5,3]

En une ligne, créez une liste **my_list**, qui contient les carrés de tous les nombres négatifs contenus dans **numbers**.

Exo 5 : **Compréhension : filtre**

Vous avez à votre disposition deux entiers **n** et **x**.

En une ligne, créez une liste **my_list**, contenant tous les entiers positifs, inférieurs à N, qui ne sont pas divisibles par X.

Exo 6 : **Compréhension : retournement de mots**

Lire **sentence** avec input()**.** En une ligne, créez une chaîne de caractères **revert** qui reprend l'ensemble des mots de sentence, mais en les inversant. (On considèrera que les mots sont séparés par des espaces)

Exemples : "Hello world" -> "olleH dlrow"
"Salut les gens" -> tulaS sel sneg"
Aide : regardez du coté des méthodes split et join

Exo 7 : **Compréhension : filtrage de mots**

Lire **sentence** avec input()**. En une ligne,** créez une chaîne de caractères words qui reprend l'ensemble des mots de sentence, en ne gardant que ceux composés de lettres minuscules. On considèrera que les mots sont séparés par des espaces.

Exemples :
"Hello world" -> "world"
"Salut les gens !" -> "les gens"
"la guerre de 14-18" -> "la guerre de"
Aide : regardez du coté des méthodes isalpha, islower, split et join

Exo 8: **Chaînes distinctes**

soit s1 et s2, composées de mots séparés par des espaces. Transformez s1 et s2 pour que les mots qui sont communs aux deux chaînes soient en majuscules, et les autres en minuscules.

Exemple :
pour "Hello World" et "Hello kitty", on aura "HELLO world" et "HELLO kitty" après traitement
pour "Mon ordi et ma TV sont cassés" et "Mon papa aime ma maman", on aura "MON ordi et MA tv sont cassés" et "MON papa aime MA maman"

Exo 9 : **Dico en compréhension : retournement**

Vous avez à votre disposition un dictionnaire **dico=**{'b':1, 'a':2, 'd':5}.

En une ligne, créez un dictionnaire **revert_dico**, contenant les items de dico, mais inversés. Résultat attendu : {1:'b', 2:'a', 5:'d'}

Exo 10 : **Compréhension : puissances de 2**

You have at your disposal a dictionary named **my_dict**, and a second dictionary called **correspondances**, which contains in keys all the values of **my_dict**. Create a dictionary called **results**, which contains the product of **my_dict** by **correspondances**.

For example, if: **my_dict** contains "x": "y" and **correspondances** contain "y": "z" Then **results** will contain "x": "z"

Exemple:
1- my_dict = {"a":"b", "b":"c", "c":"a"} correspondances = {"a":"A", "b":"B", "c":"C"} then results contains : {'a': 'B', 'b': 'C', 'c': 'A'}
2- my_dict = {"a":"b", "b":"c", "c":"X"} correspondances = {"a":"A", "b":"D", "c":"C"} then results contains : {'a': 'D', 'b': 'C'}

Exo 11 : **sorted et list.sort**

a-Soit la liste l=[5,3,10,4,8,20,2], triez cette liste par ordre croissant

b- Soit la liste l=["bonjour","tout","le","monde"], triez cette liste selon la taille des mots ['le', 'tout', 'monde', 'bonjour']

c- Soit la liste de liste l=[[1,0,**5**],[1,2,**3**], [**8**,5]] , triez cette liste dans l'ordre décroissant en fonction de leur plus grand élément (le résultat est [[8, 5], [1, 0, 5], [1, 2, 3]]

d- Soit la liste l=[('Luc',22),('Lea',18),('Alice',23),('Luca',21), ('Max',20)], une liste de tuple représentant chacun le prénom et l'age de plusieurs personnes.

Triez cette liste par ordre alphabétique, affichez là puis triez-la selon l'âge et affichez-la de nouveau.

Exo 12 : **Générateurs (yield) : boomerang**

Rappel:Python generators are a simple way of creating iterators. If a function contains at least one yield statement, it becomes a generator function. The difference is that while a return statement terminates a function entirely, yield statement pauses the function saving all its states and later continues from there on successive calls.

Écrire un générateur **boomerang**, qui prend en paramètre un itérable fini, et génère les éléments itérables, puis les génère encore, mais en sens inverse.

On devra pouvoir l'utiliser de la façon suivante :
l = [1,2,3]
for x in boomerang(l):
  print(x) //affiche "1" puis "2", puis "3", puis "3", puis "2", puis "1"

Partie 2: Création d'une classe Point3D

Avant de créer la classe Point3D, analysez l'implémentation et l'utilisation de la classe CompteBancaire ci-dessous:

```
class CompteBancaire:
    """ une classe pour gérer des comptes bancaires"""
    cb_count=0
    def __init__(self, nomClient, iban, solde =0):
        self.nomClient=nomClient
        self.solde=solde
        self.iban=iban
        CompteBancaire.cb_count+=1
    def Affichage(self):
        print("Compte appartenant à", self.nomClient,"IBAN :  ", self.iban, "solde :", self.solde)
    def __eq__(self, autrecompte):
        return self.iban==autrecompte.iban
    def __str__(self):
        return f"Compte appartenant à {self.nomClient} IBAN :  {self.iban} solde : {self.solde}"
```

Et voici un exemple d'utilisation :

```
In [2]: c1=CompteBancaire("Dupont","FR20029999",1000)

In [3]: c2=CompteBancaire("Martin","FR20020123",100)

In [4]: c3=CompteBancaire("Martin","FR20020123")

In [5]: c1.Affichage()
Compte appartenant à Dupont IBAN :   FR20029999 solde : 1000

In [6]: print(c2)
Compte appartenant à Martin IBAN :  FR20020123 solde : 100

In [7]: print(c1==c2)
False

In [8]: print(c1==c3)
False

In [9]: c1.cb_count
Out[9]: 3

In [10]: c2.cb_count
Out[10]: 3

In [11]: c3.cb_count
Out[11]: 3

In [12]: CompteBancaire.cb_count
Out[12]: 3
```

Quel est le constructeur? Quelles sont les variables d'instances? Quelles sont les variables de classe? Qu'est ce qui a permis l'utilisation de print(c2)? Qu'est ce qui a permis le test == ?

Pour la liste des méthodes spéciales et le surcharge des opérateurs consultez:
https://docs.python.org/3/reference/datamodel.html#special-method-names

Créez une classe **Point3D** qui permet:

- Initialiser un point 3D ayant comme coordonnées le vecteur (x,y,z)
- Afficher les coordonnées d'un vecteur
- Calculer le module d'un vecteur
- surcharger les fonctions spéciales permettant:
    - addition,
    - soustraction
    - l'opérateur ==
    - multiplication scalaire entre 2 vecteurs multiplication par un scalaire

- produit vectoriel
- set item et get item

Créez une classe **Mass3D** qui permet de représenter une masse dans l'espace 3D. Une masse **est** un point3D qui a une valeur **m**. Créez les attributs et méthodes nécessaires.

Partie 3: nowledgeable https://nowledgeable.com