Langage Python A3 TD2

Partie 1: Python basics

Exo 1: Télécharger le fichier tableau.dat depuis DVO, ouvrez le avec un éditeur de texte pour avoir une idée sur le contenu (puis fermer le). chargez le contenu dans une liste d'entiers (par code python).

Triez les données (numériques) obtenues.

Exo 2: Vous avez à votre disposition un fichier texte dont chaque ligne est la représentation d'une valeur numérique de type réel (mais sans exposants). Créez le avec n'importe quel éditeur de texte sinon Par exemple :

14.896
7894.6
123.278
etc.

Écrivez un script qui recopie ces valeurs dans un autre fichier en les arrondissant en nombres entiers.

Exo 32 : Congés Annuels

a) Construire le dictionnaire « Semestre2 » des 6 premiers mois de l'année civile avec comme valeurs le nombre de jours respectif.
b) Afficher la liste des mois et celle des jours.
c) Faire un test si l'année est bissectile et mettre à jour le nombre de jours pour le mois de février.
d) Construire le dictionnaire « Semestre1 » qui commence du 4 septembre pour une durée de 6 mois avec comme valeurs le nombre de jours respectif.
e) Ajouter dans un dictionnaire « AnneeScolaire » les éléments du « Semestre1 » et ceux du "Semestre 2"
f) Créer le dictionnaire « VacanceScolaire » qui associe à chaque mois le nombre de jours de vacances selon le calendrier des vacances scolaires de l'année 2020-2021
g) Créer le dictionnaire « weekend » qui associe à chaque mois le nombre de jours de weekend (ne vous cassez pas la tete pour faire un algo, faites manuellement et au pif)
h) Créer le dictionnaire « JoursTravail » qui associe à chaque mois le nb de jours de travail effectif.

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.