TP - Programmation Orientée Objet

M. Tellene

Organisation : dans votre dossier personnel, créer un dossier « programmation orientée objet » (ou « POO »), c'est ce dossier qui contiendra le travail fait lors de ce TP

1 Quelques rappels

- Une classe est introduite par le mot-clé `class`
- Le nom d'une classe commence toujours par une majuscule
- Une classe possède une construction : la méthode `__init__`
- Pour indiquer qu'une méthode ou un attribut appartient à une classe, il ne faut pas oublier le mot-clé `self` qui pour :
  - une méthode est le premier paramètre : `def ma_methode(self,...):`
  - un attribut est positionné avant celui-ci : `self.attribut = ...`

2 Un point c'est tout

Un point dans l'espace est caractérisé par un nom et 3 coordonnées : `x`, `y`, `z`

1. Écrire une classe `Point` avec le constructeur
2. Écrire une méthode d'affichage d'un point qui affiche comme suit :

```
nom_du_point : (valeur_x, valeur_y, valeur_z)
```

3. Écrire une méthode `inf_ou_egal`. `P.inf_ou_egal(Q)` renvoie `True` si `P.x` $<$ `Q.x`, en cas d'égalité à ce niveau, ce sont les coordonnées `y` qui décident, en cas d'égalité au niveau des `y` ce sont les coordonnées `z` qui décident.

   S'il y a toujours une égalité, la méthode renvoie `True`
4. Écrire une méthode `distance` qui renvoie la distance entre 2 points. Il est rappelé la formule de la distance entre 2 points ayant chacun 3 coordonnées :

$$d(P1, P2) = \sqrt{(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2}$$

5. Deux `P` et `Q` sont considérés comme étant identiques si `distance(P, Q)` < 0.0001

   Écrire la méthode `identique`

3 Stocker tous ces points

1. Écrire une classe `CollectionDePoints`. Cette classe admet 2 attributs : le nom de la collection et une liste de points (à la création, une liste est vide)
2. Écrire une méthode `ajouter` qui ajoute un point à la collection

3. Écrire une méthode d'affichage pour une collection qui affiche comme suit :

```
nom_de_la_collection : [nom_du_point : (valeur_x, valeur_y, valeur_z);
    nom_du_point : (valeur_x, valeur_y, valeur_z); ...]
```

4. Écrire une méthode `appartient` qui renvoie `True` si un point appartient à une collection. Pour ce faire, il faudra utiliser la méthode `identique` de la classe `CollectionDePoints`
5. Écrire une méthode `centre_gravite` qui renvoie le centre de gravité d'une collection de points

   Le centre de gravité sera point ayant pour valeur de :
   — `x`, la moyenne des coordonnées `x` des points de la collection
   — `y`, la moyenne des coordonnées `y` des points de la collection
   — `z`, la moyenne des coordonnées `z` des points de la collection
6. En utilisant la méthode de comparaison `inf_ou_egal`, écrire une méthode `tri` qui trie une collection de points