TP1 : MANIPULATION D'OBJETS

1. Git

Nous vous recommandons d'utiliser `etulab.univ-amu.fr`, sur lequel vous pouvez vous connecter avec vos identifiants AMU. D'autres alternatives sont possibles, comme `gitlab` ou `github`, qui sont largement utilisés, et sur lesquels vous pouvez vous créer un compte.
Il vous suffira ensuite de créer un projet, lui donner un nom (vous pouvez laisser les autres paramètres par défaut), et de le cloner dans un terminal en copie/collant l'url qui vous est donnée dans l'onglet *clone*. Si vous utilisez un IDE, vous pouvez importer un projet depuis l'url.

2. La classe Vector

(1) Créer un programme `vect.py` et recopier les commentaires suivants :
- # Imports éventuels
- # Déclaration des classes et fonctions
- # Programme principal

Déclarez une classe `Vector`, dont le constructeur affiche un message ; instanciez cette classe dans le programme principal, enfin exécutez le programme et vérifiez que le message s'affiche bien.

Dans la suite, placez toujours votre code dans la partie adéquate.

(2) Modifier la classe `Vector`, afin que le constructeur reçoive en paramètre une liste de coefficients `coeffs` (ou un itérable quelconque). Le constructeur stocke ensuite les coefficients en attribut dans une liste `list_coeffs`. Demandez-vous si cette liste serait mieux en public ou en privé.

(3) Ajouter une méthode `__str__()` qui renvoie les coefficients du `Vector` sous forme de chaîne de caractère sur une seule ligne, sous la forme : `[ coeff1 ; coeff2 ; ... ]`

Dans le programme principal, instanciez un `Vector` en lui passant par exemple [10, 20], ou (5, -2, 1.5), ou encore range(4,8).

(4) Rajouter une méthode `dimension()` qui renvoie la dimension du `Vector`.

Note : prenez l'habitude de tester systématiquement plusieurs cas de figure dans le programme principal pour chaque question.

(5) Rajouter une méthode `get(i)` qui prend en paramètre un entier `i` puis renvoie le coefficient d'indice `i` de la liste `list_coeffs`.

(6) Rajouter une méthode `calculate_sum(vec2)` qui reçoit en paramètre un `Vecteur vec2`. La méthode vérifie que l'instance courante et `vec2` ont la même dimension, sinon elle émet une exception `ValueError` avec un message d'erreur. La méthode calcule ensuite la somme des deux `Vectors` (l'instance courante et `vec2`) en se servant de leur méthode `get()`, et stocke le résultat dans une liste. Enfin, la méthode instancie un nouveau `Vector` en lui passant la liste résultat, puis renvoie le nouveau `Vector`.

3. Héritage : la classe Polynomial

(1) Créer le fichier `poly.py` à partir d'une copie de `vect.py`.

Déclarez à la suite de la classe `Vector`, une classe `Polynomial` qui dérive de `Vector` (autrement dit, la classe `Polynomial` sera une classe fille de la classe mère `Vector`). Le constructeur reçoit en paramètre une liste de coefficients coeffs (ou un itérable quelconque) puis les transmet au constructeur de la classe mère.

(2) Rajouter une méthode `degree()` qui renvoie la dimension du `Vector` -1, en se servant de la méthode `dimension()`.

(3) Surcharger la méthode `__str__()` qui renverra le `Polynomial` sous forme de chaîne de caractère sur une seule ligne, sous la forme : `coeff0 x^0 + coeff1 x^1 + ...`

(4) Rajouter une méthode `evaluate(x)` qui calcule la valeur du `Polynomial` pour la valeur `x`, puis renvoie la valeur calculée.