TD – LA PROGRAMMATION ORIENTEE OBJET

1ère partie

Introduction et objectifs

Ce premier TD consacré à la programmation orientée objet doit permettre de vous familiariser avec les concepts fondamentaux : notion de classe, d'attribut, de méthode (éventuellement spéciale), d'héritage et de méthode surchargée.

La classe `cplx`

Exercice N°1

Compléter la classe `cplx` vue en cours en ajoutant des méthodes permettant d'obtenir :

- Le conjugué.
- L'inverse.
- L'argument compris entre $-\pi$ et π.

Exercice N°2 – Surcharge d'opérateurs

Compléter la classe `cplx` en redéfinissant les méthodes spéciales `__add__`, `__sub__`, `__mul__` et `__truediv__` pour pouvoir additionner, soustraire, multiplier et diviser deux complexes `z1` et `z2` « naturellement », c'est-à-dire en écrivant : `z1 + z2`, `z1 – z2`, `z1 * z2` et `z1/z2`.

La classe `stack`

Exercice N°3

Définir une classe `stack` permettant de manipuler des piles en tant qu'objets Python.

La méthode spéciale `__init__` initialisera le seul attribut, `liste`, avec une liste vide :

```
self.liste = []
```

La classe `poly`

Exercice N°4

Construire la classe poly permettant de créer et manipuler des polynômes à coefficients réels.
Les attributs seront :

- `self.c` : liste des coefficients non nuls.
- `self.d` : liste des degrés associés (dans l'ordre décroissant).

Pour créer le polynôme : $-5x^{13}+24x^{8}-567$, on utilisera par exemple :

```
P1 = Poly([-5,24,-567],[13,8,0])
```

On écrira les méthodes :

- `__repr__`
- `eval` : renvoie la valeur prise par le polynôme pour un réel `x` passé en argument.
- `deriv` : renvoie le polynôme dérivé.
- `integrale` : renvoie l'intégrale de la fonction polynôme entre deux réels `a` et `b` passés en arguments.

Exercice N°5 – La classe Poly2d

Pour définir d'autres méthodes spécifiques aux polynômes du second degré, on crée la classe `Poly2d` héritant de la classe `Poly`.

On surchargera la méthode `__init__` de façon à simplifier la création du polynôme.
Pour créer le polynôme $5x^{2}-x+17$, on utilisera par exemple :

```
P1 = Poly2d(5,-1,17)
```

Dans cette même méthode, on aura bien sûr : `self.d = [2,1,0]`.

On pourra aussi (mais ce n'est pas obligatoire) surcharger la méthode `__repr__`.

On écrira les méthodes :

- `delta` : renvoie la valeur du discriminant associé.
- `sommet` : renvoie un tuple contenant l'abscisse et l'ordonnée du sommet de la parabole associée.
- `racine` : renvoie, selon le cas le booléen False (pas de racine réelle), un réel (racine double) ou deux réels (2 racines distinctes).
- `Ertan` : renvoie les coefficients de l'équation réduite de la tangente à la parabole associée au point d'abscisse `x` (ce réel étant passé en argument).