

Le but de ce TP est de revoir certaines opérations sur les polynômes en les programmant en Python.

Un polynôme sera représenté par sa liste de coefficients par degrés croissants.

Par exemple la liste `[2, 0, 1]` sera la représentation du polynôme $x^2 + 1$. La liste `[]` représentera le polynôme nul.

Pour créer un polynôme aléatoire de degré 100 prenant ces valeurs entre -99 et 100 (on importera le module **random**), on pourra utiliser la commande :

```
P = [random.randint(-99,100) for i in range(101)]
```

Pour déclarer une liste nulle de taille 100, on utilisera la commande :

```
[0 for i in range(100)]
```

Exercice 1 : Calcul de degrés

Écrire la fonction **degre** qui prend en argument un polynôme et renvoie le degré de celui-ci.

Par exemple `degre([2,3])` renverra 1, `degre([1])` renverra 0 et `degre([])` renverra -1

Exercice 2 : Évaluation d'un polynôme

(1) Écrire la fonction **evaluate** qui permet l'évaluation d'un polynôme par un nombre x_0 .

Par exemple : si $P(x) = x^2 + 2x + 3$, $P(2) = 11$ donc `evaluate([1,2,3],2)` retournera 11

(2) Pour évaluer de façon plus rapide un polynôme, on peut utiliser la méthode d'Hörner. Celle-ci consiste à évaluer le polynôme en x_0 comme :

$$((\dots((a_n x_0 + a_{n-1}) x_0 + a_{n-2}) x_0 + \dots) x_0 + a_1) x_0 + a_0.$$

Écrire la fonction **horner** qui utilise cette méthode pour évaluer les polynômes.

Par exemple pour calculer $P(2)$ dans l'exemple précédent, on utilisera les étapes :

```
resultat = 0
resultat = resultat * 2 + 1
resultat = resultat * 2 + 2
resultat = resultat * 2 + 3
```

On obtiendra bien 11

(3) On pourra comparer les deux derniers algorithmes sur des polynômes de très haut degrés (100).

Exercice 3 : Dériver un polynôme

(1)

Écrire la fonction **deriver** qui prend un polynôme en paramètre et renvoie le polynôme dérivé.

`deriver([1,2,1])` retournera la liste `[2,2]` (le polynôme dérivé de $x^2 + 2x + 1$ est $2x + 1$).

(2) Utiliser cette fonction et la fonction horner pour écrire la fonction **est_racine** qui prend en argument un polynôme et un nombre et affiche si ce nombre est une racine du polynôme en indiquant la multiplicité.

Exercice 4 : Opérations algébriques de base

(1) Écrire la fonction **somme** qui prend en arguments deux polynômes et renvoie la somme de ceux-ci.

`somme([1,2,3], [2,3,4])` renverra `[2,4,7]` `somme([1,2], [1,2,3])` retournera `[2,4,3]`

`somme([1,-1,2], [2,1,-2])` retournera `[3]` et non `[3,0,0]`.

(2) Écrire la fonction **produit** qui prend en arguments deux polynômes et renvoie le produit de ceux-ci.

`produit([1,2], [1,2,3])` retournera `[1, 4, 7, 6]`

(3) Écrire la fonction **compose** qui prend en arguments un polynôme P et renvoie le polynôme $P(X+1)$.

`compose([1,2,1])` retournera le polynôme `[4,4,1]`

Exercice 5 : Interpolation de polynômes

- (1) Écrire la fonction **interpolateur** qui prend en arguments une liste de n nombres et un nombre q et renvoie le q ème polynôme interpolateur de Lagrange.
- (2) Écrire la fonction **interpole** qui prend en arguments une liste de n nombres et renvoie l'unique polynôme de degré $n + 1$ qui interpole ces nombres.
- (3) Écrire une nouvelle fonction qui effectue le produit de deux polynômes en utilisant d'abord la fonction **horner** puis la fonction **interpole**