TP 12 - Polynômes

Un polynôme est caractérisé par la liste de ses coefficients. Ainsi, nous avons un moyen efficace de représenter un polynôme en PYTHON.
Dans cette fiche, une liste de nombres est donc associé à un polynôme et vice-versa.

$$P = a_0 + a_1X + a_2X^2 + \cdots a_nX^n \leftrightarrow [a_0, a_1, a_2, \ldots, a_n]$$

Ici, nous prenons la convention de ranger les coefficients par ordre croissant des degrés. Par exemples :
- Le polynôme $1 + X^2 - 2X^3$ est modélisé par `[1,0,1,-2]`.
- Inversement, la liste `L=[0,1,2,-1,4]` fait référence au polynôme $P = X + 2X^2 - X^3 + 4X^4$.
- Considérant la liste `L` et le polynôme associé P, alors le coefficient de degré k est donné par `L[k]`.

Exercice 1
Donner la liste associée aux polynômes suivants :
- $Q = 1 + X^2 + 3X^4 - X^5$
- $R = X + X^2 - X^3$
- $S = X^k$

Exercice 2 Compléter la fonction `coeff(P,i)` qui retourne le coefficient de X^i du polynôme P :

```
def coeff(P,i):
    if i<...:
        return ...
    else:
        return ...
```

Exercice 3 Écrire une fonction `degre(P)` qui retourne le degré de P en veillant à tester la non nullité du coefficient dominant.
Par ailleurs, la fonction enlèvera les éventuels coefficients nuls inutile suivant le coefficient dominant.
Par convention informatique, le degré du polynôme nul est -1.

Exercice 4 Donner une fonction `somme(P,Q)` qui retourne $P+Q$.
Tester votre fonction sur un exemple.

Exercice 5 Donner une fonction `produit(P,Q)` qui retourne PQ.
On rappelle que si $A = \sum_{i=0}^{p} a_iX^i$ et $B = \sum_{i=0}^{q} b_iX^i$ alors

$$AB = \sum_{i=0}^{p+q} c_iX^i \text{ avec } c_i = \sum_{j=0}^{i} a_jb_{i-j}$$

Tester votre fonction sur un exemple.

Exercice 6 Définir les deux fonctions suivantes :
1. Donner une fonction `puissance(P,n)` qui retourne le polynôme P^n.
2. En déduire, une fonction `parmi(k,n)` qui retourne $\binom{n}{k}$.

Exercice 7 Définir les deux fonctions suivantes :
1. `derive(P)` retourne P'
2. `integre(P)` retourne la primitive de P s'annulant en 0

Exercice 8 *Évaluation basique*
1. Compléter le script suivant :

```
def eval(P,x)
    e=0
    for i in range(...):
        e=e+...
    return e
```

2. Définir le polynôme $P = 1 + X - 2X^2 + 3X^4$ et vérifier que $P(2) = 43$.
3. Donner le nombre d'opérations réalisées en fonction de n, le degré de P :
- Nombre de multiplications :
- Nombre d'additions :
- Nombre d'affectations :

Exercice 9 *Évaluation optimisant le calcul de la puissance*
On remarque que : $x^{i+1} = x^i \times x$
Ainsi, une multiplication suffit pour le calcul x^{i+1}, si toutefois, on a sauvegardé dans une variable x^i.
1. Proposer une fonction `eval2(P,x)` tenant compte de cette amélioration.
2. Donner le nombre d'opérations réalisées en fonction de n, le degré de P :
- Nombre de multiplications :
- Nombre d'additions :
- Nombre d'affectations :

Exercice 10 *Algorithme de Hörner* **Prog**
1. Donner la fonction répondant à l'algorithme suivant :

Entrée :	P un polynôme de degré n (liste des coeff) x un réel
	$e \leftarrow a_n$ Pour i variant de $n-1$ à 0 faire $\quad e \leftarrow e * x + a_i$ FinPour
Sortie :	e égal à $P(x)$.

2. Donner le nombre d'opérations réalisées en fonction de n, le degré de P :
- Nombre de multiplications :
- Nombre d'additions :
- Nombre d'affectations :

Lycée Saint-Exupery – maffre.saintexupery@gmail.com – MPSI