TD 19 : Polynômes

On ne considère que des polynômes à une indéterminée à coefficients dans $\mathbb{R}$:

$$P = \sum_{k=0}^{n} a_k X^k$$

où n est le degré du polynôme P et $(a_k)_{k\in\mathbb{N}}$ la suite réelle à support fini des coefficients de P. Le polynôme P est représenté par une liste en Python (type `list`) :

$$\texttt{P} = [a_0, a_1, \cdots, a_{n-1}, a_n]$$

La liste représentant un polynôme est dite réduite si son dernier coefficient est non nul : un polynôme de degré n est toujours représenté sous forme réduite par une liste de $n+1$ éléments. Les polynômes constants sont représentés, lorsqu'ils sont réduits, par une liste ne contenant qu'un seul élément : $\texttt{P} = [a_0]$. Le polynôme nul réduit est représenté par la liste vide [ ], son degré est pris égal à -1 (la convention d'un degré égal à $-\infty$ est plus difficile à manipuler).
Le coefficient dominant a_n du polynôme P s'obtient à partir de la représentation réduite par `P[-1]`, après avoir vérifié que `P` n'est pas le polynôme nul. Un polynôme unitaire a son coefficient dominant égal à 1. L'élément `P[k]` de la liste `P` représente le coefficient a_k du polynôme P.

Exercice 1 (Fonctions élémentaires sur les polynômes (20 minutes))

1. Écrire une fonction `reduit` qui réduit un polynôme **en place** (la fonction retourne None), en supprimant les zéros inutiles. Le cas du polynôme nul est pris en compte.

```
>>> P = [1, 2, 3, 0, 0]
>>> reduit(P)
>>> P
[1, 2, 3]
```

2. Écrire une fonction `degre` qui retourne le degré du polynôme transmis. Le cas du polynôme nul doit être pris en compte : `degre([])` doit retourner -1.

```
>>> degre(P)
2
```

3. Écrire une fonction `monome` qui retourne le monôme $P = a\,X^n$. Le cas $a = 0$ et le cas $n = 0$ sont pris en compte.

```
>>> monome(5, 3)
[0, 0, 0, 5]
```

4. Écrire une fonction `mul_Xn` qui retourne le polynôme $Q = X^n P$, avec $n \in \mathbb{N}$. Le cas d'un polynôme nul doit être pris en compte, comme le cas $n = 0$.

```
>>> mul_Xn(P, 2)
[0, 0, 1, 2, 3]
```

Exercice 2 (Opérations sur les polynômes (40 minutes))

1. Écrire une fonction `evalue` qui retourne $P(x)$. Le cas du polynôme nul doit être pris en compte.

```
>>> evalue(P, 5)
86
>>> evalue([],2)
0
```

2. Écrire une fonction `mult_scalaire` qui multiplie le polynôme P par a. Le polynôme nul et le cas $a = 0$ sont à traiter.

```
>>> mult_scalaire(P, 3)
[3, 6, 9, 0, 0]
>>> mult_scalaire(P, 0)
[]
>>> mult_scalaire([], 3)
[]
```

3. Écrire une fonction `additionne` qui retourne $P + Q$. On veille à retourner un polynôme réduit. P et Q peuvent être nuls.

```
>>> additionne(P,[1,-2,-3])
[2]
>>> additionne(P,[4,3,2,1])
[5, 5, 5, 1]
>>> additionne(P,[])
[1, 2, 3]
>>> additionne([2],P)
[3, 2, 3]
>>> additionne([],[])
[]
```

4. Écrire une fonction `soustrait` qui retourne $P-Q$. On veille à retourner un polynôme réduit. P et Q peuvent être nuls.

5. On pose $A = \sum_{i=0}^{n} a_i X^i$ et $B = \sum_{j=0}^{m} b_j X^j$, d'où $C = AB = \sum_{k=0}^{n+m} c_k\, X^k$, avec

$$c_k = \sum_{\substack{i+j=k \\ 0 \leq i \leq n \\ 0 \leq j \leq m}} a_i\, b_j$$

   Écrire une fonction `multiplie` qui retourne le produit AB de deux polynômes A et B. Les polynômes A et B peuvent être nuls.

6. Évaluer la complexité temporelle de la multiplication polynomiale ainsi programmée pour deux polynômes de degrés n et m. Conclure pour des polynômes de taille n en général.

7. À l'aide de la fonction `multiplie`, écrire une fonction `puissance`, qui retourne P^n.