Un polynôme sera représenté par une liste.
Exemple : $1 + 2X + 4X^3 + 2X^4$ sera représenté par $p = [1, 2, 0, 4, 2]$

Exercice 1

1) Ecrire une fonction **def coef(p,k)** donnant le coefficient du monôme de degré k si $k < len(p)$ et 0 sinon.

2) Ecrire une fonction **def degre(p)** donnant le degré du polynôme p. Attention car $p = [1, 2, 1, 3, 4, 0, 0, 0, 0]$ par exemple est de degré 4. Ce phénomène peut se produire par exemple à la suite d'une addition de polynômes. On choisira -10 pour le degré du polynôme nul.

Exercice 2

Ecrire une fonction **def somme(p,q)** calculant la somme de deux polynômes p et q. On pourra utiliser la fonction max qui est une fonction Python.

Exercice 3

1) Ecrire une fonction **def multconstante(a,p) :** qui multiplie une constante par le polynôme p.

2) Ecrire une fonction **def multmonôme(p,d) :** qui multiplie le polynôme p par le monôme X^d.

3) Ecrire une fonction **def multpoly(p,q) :** qui calcule le polynôme produit des polynomes p et q.

Exercice 4

Ecrire une fonction **def diveuc(A,B)** qui renvoie le quotient et le reste dans la division euclidienne des polynômes A par B. Utliser les exercices précédents.

Exercice 5

Ecrire une fonction **def horner(p,x) :** mettant en oeuvre l'algorithme de Horner, qui calcule la valeur de p pour le réel x.

Exercice 6

```
# Exponentielle rapide
def expo(a,m):
    c=bin(m)
    d=1
    p=len(c)
    for i in range(p):
        d=d**2
        if c[i]=='1':
            d=d*a
    return d
```

Montrer que la complexité de cet algorithme, qui calcule a^m, est en $O(\log_2 n)$.