Polynômes

Exercice 1

Écrire des fonctions en python qui permettent de calculer la somme et la différence de deux polynômes.

Exercice 2

On considère les polynômes

$$p(x) = x^5 - x^4 + x^3 - 3\,x^2 + 2\,x - 1 \quad \text{et} \quad q(x) = -3\,x^3 + 5x^2 - 2x + 3$$

a) Ecrire les deux listes python `p` et `q` correspondant à ces polynômes.

b) A l'aides des fonctions de l'exercice précédent, faire calculer les listes correspondant à $p+q$, $p-q$ et $q-p$.

Exercice 3

Même exercice pour

$$p(x) = 2\,x^3 - 3\,x^2 + 5\,x - 1 \quad \text{et} \quad q(x) = 3\,x^3 + 2\,x^2 - 4\,x + 2$$

Exercice 4

Même exercice pour

$$p(x) = x^7 - x^6 + 3\,x^5 - 4\,x^4 - x^3 - 2\,x^2 - 3\,x - 1 \quad \text{et} \quad q(x) = 3\,x^3 + 2\,x^2 - 4\,x + 2$$

Exercice 5

Même exercice pour $p(x) = x^2 + x + 2$ et $q(x) = x^3 - 2\,x$

Exercice 6

Soit

$$a(x) = -2\,x^2 + 2\,x - 1 \quad \text{et} \quad b(x) = 2\,x^3 - \,x^2 - 2\,x + 2$$

a) Dessiner le tableau qui illustre la multiplication de a par b.

b) Donner les deux listes python `a` et `b`, correspondant à ces deux polynômes.

c) Créer une liste `p` dans le shell python permettant de stocker les coefficients du polynôme $a \cdot b$. Quelle doit-être sa longueur ?

d) Pour calculer les quatre premiers éléments de `p`, on écrit

```
p[0] = a[0]*b[0]
p[1] = a[0]*b[1] + a[1]*b[0]
p[2] = a[0]*b[2] + a[1]*b[1] + a[2]*b[0]
p[3] = a[0]*b[3] + a[1]*b[2] + a[2]*b[1] + a[3]*b[0]
```

À l'aide d'instructions analogues, faire calculer les deux derniers coefficients du produit.

e) Vérifier que la liste obtenue correspond bien au polynôme $a \cdot b$.

Exercice 7

Soit p et q deux polynômes en x. On note `p` et `q` les listes python qui correspondent à ces deux polynômes.
Écrire une fonction `produit` qui prend les deux listes `p` et `q` en paramètre et qui renvoie une liste qui correspond au produit

$$p \cdot q$$

On pourra procéder comme suit :

a) Créer une liste vide `t`, qui contiendra les lignes du tableau de tous les produits des coefficients de p et q.

b) Créer une liste de la bonne longueur qui pourra contenir les coefficients du produit

$$p \cdot q$$

Cette liste contient le bon nombre de fois 0.

c) En parcourant le tableau `t`, ajouter chaque `t[i][j]` au coefficient correspondant du résultat `r`.

d) Retourner le résultat `r` à l'utilisateur

Exercice 8

Écrire une fonction qui permet de faire calculer p^n pour p un polynôme et n un entier positif.

Exercice 9

Soit les polynômes

$$a(x) = 3\,x^2 - 4\,x + 3 \text{ , } p(x) = x^4 + 2\,x^3 - 2\,x^2 - 4\,x + 17 \text{ et } q(x) = 2\,x^3 - 3\,x^2 - 5\,x + 18$$

faire calculer automatiquement :

a) les polynômes a^8 et $a^6 \cdot p^6 \cdot q^6$;

b) le coefficient du polynôme $p^{10} \cdot q^{20}$ de degré 77.

Faire vérifier le résultat par le serveur WolframAlpha.

Exercice 10

Soit le polynôme $p(x) = x + 1$. Faire calculer automatiquement

$$q_n = p^n$$

pour n variant entre 0 et 15. Que peut-on constater ?

Exercice 11

Soit le polynôme $p(x) = 2x^2 - x + 2$. Faire calculer automatiquement

$$q_n = p^n$$

pour n variant entre 1 et 15. Que peut-on constater ?

Exercice 12

Soit le polynôme $p(x) = 1 - x$. Faire calculer automatiquement

$$q_n(x) = p(x) \cdot (1 + x + x^2 + \ldots + x^n)$$

pour n variant entre 1 et 20. Que peut-on constater ?

Solutions

1

```
def somme(p, q):
    if len(q) > len(p):
        p, q = q, p
    r = p[:]
    for i in range(len(q)):
        r[i] += q[i]
    return r
def diff(p, q):
    q = [-q[i] for i in range(len(q))]
    if len(q) > len(p):
        p, q = q, p
    r = p[:]
    for i in range(len(q)):
        r[i] += q[i]
    return r
```

2

```
def somme(p, q):
    if len(q) > len(p):
        p, q = q, p
    r = p[:]
    for i in range(len(q)):
        r[i] += q[i]
    return r
def diff(p, q):
    q = [-q[i] for i in range(len(q))]
    if len(q) > len(p):
        p, q = q, p
    r = p[:]
    for i in range(len(q)):
        r[i] += q[i]
    return r

p = [-1, 2, -3, 1, -1, 1]
q = [3, -2, 5, -3]
print(somme(p, q))
print(diff(p, q))
print(diff(q, p))
```

3 –

4 –

5 –

6

```
a = [-1, 2, -2]
b = [2, -2, -1, 2]

deg = (len(a) - 1) + (len(b) - 1)

p = [0 for i in range(deg + 1)]
p[0] = a[0]*b[0]
p[1] = a[0]*b[1] + a[1]*b[0]
p[2] = a[0]*b[2] + a[1]*b[1] + a[2]*b[0]
p[3] = a[0]*b[3] + a[1]*b[2] + a[2]*b[1]
p[4] = a[1]*b[3] + a[2]*b[2]
p[5] = a[2]*b[3]

p_ = [0 for i in range(deg + 1)]
for k in range(len(p_)):
    for i in range(len(a)):
        for j in range(len(b)):
            if (i + j) == k:
                p_[k] += a[i]*b[j]

##Vérification p = p_= [-2, 6, -7, 0, 6, -4]
import sympy as sp
x = sp.var("x")
sp.expand((-2*x**2+2*x-1)*(2*x**3-x**2-2*x+2))
-4*x**5 + 6*x**4 - 7*x**2 + 6*x - 2
```

7

```
def produit(p, q):
    t =[]
    for coeff_p in p:
        ligne = []
        for coeff_q in q:
            ligne.append(coeff_p * coeff_q)
        t.append(ligne)
    r = [0 for k in range(len(p) + len(q) - 1)]
    for i in range(len(p)):
        for j in range(len(q)):
            r[i + j] += t[i][j]
    return r
```

8

```
def pui(p, n):
    if n == 0:
        return [1]
```

```
    else:
        q = p
        for i in range(n - 1):
            q = produit(q, p)
        return q
```

9

```
a = [3, -4, 3]
p = [17, -4, -2, 2, 1]
q = [18, -5, -3, 2]
print(pui(a, 8))
print(produit(produit(pui(a, 6), pui(p, 6)), pui(q, 6)))
print(produit(pui(p,10),pui(q,20))[77])
```

10 On obtient le début du triangle de Pascal.

11 Les listes de coefficients sont toutes « symétriques par rapport au coefficient central ».

```
p = [2, -1, 2]
for i in range(1, 16):
    print(pui(p, i))
```

12 $q_n(x) = 1 - x^{n+1}$