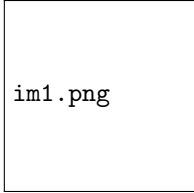


im2.jpeg



im1.png

## ETUDE DE POLYNOMES

NOM1 :KENTSA MELI HABIB EDGAR  
NOM2 :HAMDANE AMINE  
NOM DU PROF : CARDINAL JEAN-PAUL  
ANNÉE :2017/2018

Dans tous nos exemples, nous utiliserons les trois polynomes suivants :  
 $P(x) = -1 + 3x + 2x^2$ ,  
 $Q(x) = 3x$ ,

$$R(x) = 1 - x^2$$

Dans le code, nos polynomes seront représentés de la façon suivante :

```
P = [-1,3,2]
Q = [0,3]
R = [1,0,-1]
```

## 1 Valeur d'un polynome

Dans cette partie, nous calculons la valeur d'un polynome en un point. En pratique, il suffit seulement de remplacer la variable par l'abscisse du point dont on veut connaître l'image. L'implémentation de cette fonction est la suivante :

```
def poly_definition(P,x) :

    r = 0.0

    for i in range(0,len(P)):

        r = r + P[i]*(x**i)

    return r
```

Exemple :

Calculons la valeur du polynome en 2. On obtient donc en ligne de commande :

```
In [2]: poly_definition([-1,3,2],2)
Out[2]: 13.0
```

## 2 Somme de deux polynomes

Ici, on s'intéresse à additionner deux polynomes. le résultat final sera un polynome de degré équivalent au degré du polynome de plus haut degré. L'implémentation de la fonction est la suivante :

```
def poly_add(P1,P2) :

    #poly
    PAD=[]

    i=0

    while i<len(P1) and i<len(P2):

        PAD.append(P1[i]+P2[i])
        i = i+1

    if i<len(P1) :

        PAD = PAD + P1[i:len(P1)]

    if i<len(P2) :

        PAD = PAD + P2[i:len(P2)]

    return PAD
```

Exemple :

Ici nous calculons la somme de deux polynome P et Q donnés ainsi nous obtenons en ligne de commande :

```
In [3]: poly_add([-1,3,2],[0,-3])
Out[3]: [-1, 0, 2]
```

### 3 Degré d'un polynome

Nous avons écrit dans cette partie un code permettant de renvoyer le degré du monome de plus haut degré. son implémentation nous donne ceci :

```
def deg_poly(P):

    i = len(P)-1

    while i>=0 and P[i]==0:
        i = i-1

    return i
```

Exemple :

Le monome de plus haut est comme nous le constatons de degré 2.La fonction nous revoie le degrés de ce monome de tel manière :

```
In [5]: deg_poly([-1,3,2])
Out[5]: 2
```

## 4 coefficient d'un monome

Supposons dans un polynome du type  $a_0 + a_1x^1 + \dots + a_nx^n$  :  
le coefficient du monome est  $x^k$  est  $a_k$ .

Le code que nous avons écrit permet de determiner ce coefficient. L'implémentation est la suivante :

```
def coef(p,k):
    if k<len(p):
        return p[k]
    else:
        return 0
```

Exemple :

Ici nous allons demander quel est le coefficient du monome de degré 2 et de degré 3. Cela nous donne en ligne de commande :

```
In [6]: coef([-1,3,2],2)
Out[6]: 2

In [7]: coef([-1,3,2],3)
Out[7]: 0
```

## 5 Calcul de la dérivée

Pour un polynome donné, la fonction *derivee* calcule et renvoi le polynome dérivé. L'implémentation est la suivante :

```
def derivee(P) :
    PR = []

    if len(P) == 1:
        return [0]

    else:
        for i in range(1,len(P)):
            PR.append(i*P[i])

        return PR
```

Exemple :

Ici nous allons calculons tout bonnement la dérivé de notre polynome habituel, cela nous donne donc :

```
In [8]: derivee([-1,3,2])
Out[8]: [3, 4]
```

## 6 Produit de deux polynomes

Pour deux polynomes dont les monomes de plus hauts degré sont respectivement  $n$  et  $m$ , le monome de plus haut degré du polynome produit sera de degré  $n+m$  ou sera 0 si l'un des polynomes est le polynome nul. L'implémentation est la suivante :

```
def polyprod(P, Q) :
    R=[0]*(len(Q)+len(P)-1)

    for i in range(0,len(P)):
        for j in range(0,len(Q)):
            R[i+j]=R[i+j]+(P[i]*Q[j])

    return R
```

Exemple :

Ici nous allons calculer le produit du polynome P et R, le resultat de l'experience nous donne :

```
In [9]: polyprod([-1,3,2],[1,0,-1])
Out[9]: [-1, 3, 3, -3, -2]
```

## 7 Puissance d'un polynome

Dans cette partie il s'agit de calculer la puissance  $n$  d'un polynome dont le monome de plus haut degré est de degré  $m$ . Le degré du monome de plus haut degré du polynome résultant sera  $n+m$ . Pour ceci, nous avons utilisé une fonction récursive L'implémentation de cette fonction nous donne ceci :

```
def puis_poly(p,n):

    if n==0:
        return [1]
    else:
        return polyprod(p,puis_poly(p,n-1))
```

Exemple :

Calculons maintenant notre polynome  $P$  à la puissance 4 cela nous donne un nouveau polynome de degrés 8, nous avons donc en ligne de commande :

```
In [10]: puis_poly([1,3,2],4)
Out[10]: [1, 12, 62, 180, 321, 360, 248, 96, 16]
```

## 8 Dichotomie

Cette technique nous permet grace a une boucle d'approximer la ou les racines réelles d'un polynome de degré  $n$  sur un intervalle donné. La fonction nous renvoie un intervalle dans lequel est comprise la racine trouvé par le fonction.L'implémentation est la suivante :

```
def dichotomie(P,a,b,epsi):

    while b-a>epsi:
        m=(b+a)/2
        if poly_definition(P,m)*poly_definition(P,a)>0:
            a=m
        else:
            b=m
    return a,b
```

Exemple :

La recherche des racines du plynome  $Q$  dans l'intervalle  $[-10,10]$  avec un pas de 1 nous renvoie l'intervalle dans lequel la racine se trouve, en ligne de commande nous avons :

```
In [11]: dichotomie([0,3],-10,10,1)
Out[11]: (-1, 0)
```