



**Concours Mathématiques et Physique, Physique et Chimie et Technologie
Alternative de correction de l'épreuve d'Informatique**

Barème sur 100

EXERCICE 1 (40 points)

Question 1 : ajout_monome = 5 pts

```
while (True) :  
    try :  
        degre = int(input("donner le degré du monôme: "))  
        if degre >= 0 :  
            break  
    except :  
        print("erreur de saisie") ou pass ou continue  
while (True) :  
    try :  
        coeff = float(input("donner le coefficient : "))  
        if coeff >= 0 :  
            break  
    except :  
        print('erreur de saisie') ou pass ou continue  
self.data.update({degre : coeff})
```

Question 2 : degree = 2,5 pts

```
return max(list(self.data.keys())) #possible avec la programmation
```

Question 3 : __call__ = 2,5 pts

```
Val = 0  
for degre in self.data.keys() :  
    Val += self.data[degre] * x0 ** degre  
return Val
```

Question 4 : __add__ = 5 pts

```
p = PolynomeCreux()  
for d in self.data.keys() :  
    if not ( d in other.data.keys()) :  
        p.ajout_monome({d:self.data[d]})  
    elif self.data[d] != -other.data[d] :  
        p.data.update({d : self.data[degre] +  
other.data[degre]})  
return p
```

```

p = PolynomeCreux()
for d1 in self.data.keys() :
    for d2 in other.data.keys() :
        d=d1+d2
        c=self.data[d1]*other.data[d2]
        if not(d in p.data.keys()) :
            p.ajout_monome({d:c})
        else :
            p.data[d]+=c
    if p.data[d]==0:
        del p.data[d]

return p

```

Question 6 : __str__ = 7,5 pts

```

P=sorted(self.data.items(),reverse=True)
Ch=""
for d,c in P :
    if c==1:
        Ch+=" + "
    elif c>0:
        Ch+=" + " + str(c)
    elif c== -1:
        Ch+=" - "
    elif c<0:
        Ch+=" - " + str(abs(c))
    if d!=0:
        if c!=1 and c!=-1:
            Ch+="*"
        Ch+="x"
        if d!=1:
            Ch+="**" + str(d)
if Ch[0:2]==" +": #pour enlever l'espace et le premier
plus
return Ch[2:]

```

Question 7 : primitive = 5 pts

```

p=PolynomeCreux()
for d in self.data.keys() :
    p.ajout_monome({d+1:self.data[d]/(d+1)})

return p

```

Question 8 : integrale = 5 pts

```

def integarle(P,a,b):
    return P.primitive()(b) - P.primitive()(a)

```

Question 1 : 5 pts

```
import sqlite3
base=sqlite3.connect(' EXERCICE2.db ')
curseur = base.cursor()
curseur.execute(' ' ' CREATE TABLE Utilisateur (IdU
INTEGER PRIMARY KEY , Nom TEXT , Prenom TEXT; ' ' ')
curseur.commit()
curseur.close()
base.close()
```

Question 2 : 2,5 pts

```
SELECT Nom , Prenom
FROM Utilisateur ;
```

Question 3 : 2,5 pts

```
SELECT Count(*)
FROM Utilisateur ;
```

Question 4 : 2,5 pts

```
SELECT IdCreateur , Count(*)
FROM Table
GROUP BY IdCreateur ;
```

Question 5 : 2,5 pts

```
SELECT IdU
FROM Privilege
WHERE Droit = 'CREATE ' OR Droit = ' ALL ' ;
```

Question 6 : 2,5 pts

```
SELECT Nom
FROM Utilisateur
ORDER BY Nom DESC ;
```

Question 7 : 5 pts

```
SELECT Nom , Prenom
FROM Privilege , Utilisateur
WHERE IdTable = ' Produit ' AND (Droit = ' INSERT ' OR Droit = '
ALL ' ) AND Privilege.IdU = Utilisateur.IdU ;
```

Question 8 : 2,5

$$\prod_{\substack{(\text{IdTable}=\text{'Produit'}) \\ \text{AND} \\ (\text{Droit}=\text{'INSERT'} \text{ OR } \text{Droit} = \text{'ALL'})}} (\text{Nom}, \text{Prenom}) \left(\begin{array}{c} \text{Privilege} \bowtie \text{Utilisateur} \\ \text{Privilege.IdU} = \text{Utilisateur.IdU} \end{array} \right)$$

EXERCICE 3 (35 points)

Question 1 : 5 pts

```
def puiss(x,p):  
    return [x**i for i in range(2*p+1)]
```

Question 2 : 5 pts

```
def list_puiss(L,p):  
    return [puiss(L[i],p) for i in range(len(L))]
```

Question 3 : 10 pts

```
def calcul_mat(Lp):  
    n=len(Lp)  
    m=n-1  
    M=np.ones((n,n))  
    Lx=[Lp[i][0] for i in range(n)]  
    Lpx=list_puiss(Lx,m)  
    for i in range(n):  
        for j in range(n):  
            U[i,j]=sum([v[i+j] for v in Lpx])  
    return U
```

Question 4 : 10 pts

```
def calcul_vect(Lp):  
    n=len(Lp)  
    m=n-1  
    v=np.ones((n))  
    Lx=[Lp[i][0] for i in range(n)]  
    Ly=[Lp[i][1] for i in range(n)]  
    Lpx=list_puiss(Lx,m)  
    Lsxy=[sum([Lpx[i][j]*Ly[i] for i in range(n)]) for j  
in range(m+1)]  
    for i in range(n):  
        v[i]=Lsxy[i]  
    return v
```

Question 5 : 5 pts

```
import numpy as np  
U=calcul_mat(Lp)  
v=calcul_vect(Lp)  
a=np.linalg.solve(U,v)
```