Informatique en CPGE (2017-2018)
Corrigé exercices : polynômes

Opérations de base

1.

```
def print_poly(p):
    if len(p)==1:
        print(p[0])
        return None
    c=""
    indice=0
    while p[indice]==0:
        indice+=1
    if indice==0:
        c+=str(p[indice])
    else:
        if p[indice]==1:
            c+="X^"+str(indice)
        elif p[indice]==-1:
            c+="- X^"+str(indice)
        else:
            c+=str(p[indice])+" X^"+str(indice)
    for i in range(indice+1,len(p)):
        if p[i]>0:
            if p[i]!=1:
                c+=" + "+str(p[i])+" X^"+str(i)
            else:
                c+=" + X^"+str(i)
        elif p[i]<0:
            if p[i]!=-1:
                c+=" - "+str(-p[i])+" X^"+str(i)
            else:
                c+=" - X^"+str(i)
    print(c)
p=[0,0,-3,1,0,2]
print_poly(p)
```

2.

```
def oppose(p):
    return [-p[i] for i in range(len(p))] # ou [-c for c in p]
p=[3,2,-4]
q=oppose(p)
print_poly(p)
print_poly(q)
```

3.

```
def prod_poly_reel(p,r):
    return [r*p[i] for i in range(len(p))] # ou [r*c for c in p]
p=[3,2,-4]
q=prod_poly_reel(p,3)
print_poly(p)
print_poly(q)
```

4.

```
def somme_poly(p,q):
    s=[]
    m=min(len(p),len(q))
    for i in range(m):
        s.append(p[i]+q[i])
    if m<len(p):
        for i in range(m,len(p)):
            s.append(p[i])
    elif m<len(q):
        for i in range(m,len(q)):
            s.append(q[i])
    return s

p=[3,2,-4]
q=[0,5,3,-2]
print("p =",end=" ")
print_poly(p)
print("q =",end=" ")
print_poly(q)
print("p+q =",end=" ")
print_poly(somme_poly(p,q))
```

5.

```
def produit_poly(p,q):
    prod=[0]*(len(p)+len(q)-1)
    for i in range(len(p)):
        for j in range(len(q)):
            prod[i+j]+=p[i]*q[j]
    return prod

p=[1,1]
q=[1,-1]
print("p =",end=" ")
print_poly(p)
print("q =",end=" ")
print_poly(q)
print("p*q =",end=" ")
print_poly(produit_poly(p,q))
```

6.

```
def puissance_poly(p,n):
    if n==0:
        return [1]
    elif n==1:
        return p
    else:
        q=produit_poly([1],p)
        for i in range(2,n+1):
            q=produit_poly(q,p)
        return q
p=[1,1]
q=[1,-1]
print("p =",end=" ")
print_poly(p)
print("q =",end=" ")
print_poly(q)
```

```
print("p^6 =",end=" ")
print_poly(puissance_poly(p,6))
print("q^5 =",end=" ")
print_poly(puissance_poly(q,5))
```

7.

```
def derive_poly(p):
    if len(p)==1:
        return [0]
    else:
        return [i*p[i] for i in range(1,len(p))]
p=[3,-2,3,-2,1]
print("p =",end=" ")
print_poly(p)
print("p' =",end=" ")
print_poly(derive_poly(p))
```

8.

```
def primitive_poly(p):
    q=[0]
    for i in range(len(p)):
        q.append(p[i]/(i+1))
    return q
p=[3,-2,3,-2,1]
print("p =",end=" ")
print_poly(p)
print("prim(p) =",end=" ")
print_poly(primitive_poly(p))
```

9.

```
def valeur_poly(p,x): # calcul avec le schéma de Hörner
    v=p[-1]
    for i in range(len(p)-2,-1,-1):
        v=x*v+p[i]
    return v
p=[1,0,-1]
print("p =",end=" ")
print_poly(p)
print("p(3) =",end=" ")
print(valeur_poly(p,3))
```

Polynômes de Legendre

1.

```
print("Base canonique")
n=10
base=[[1],[0,1]]
for i in range(2,n+1):
    base.append(produit_poly(base[1],base[i-1]))
print("base canonique")
for i in range(n+1):
    print_poly(base[i])
```

2.

```
    def scalaire(p,q):
        prod=produit_poly(p,q)
        prim=primitive_poly(prod)
        s=valeur_poly(prim,1)-valeur_poly(prim,-1)
        return s
    p=[1]
    print("produit scalaire <p,p>",end=" ")
    print(scalaire(p,p))
```

3.

```
    def norme(p):
        return (scalaire(p,p))**0.5
    print("norme de p",end=" ")
    print(norme(p))
```

4.

```
    legendre=[]
    legendre.append(prod_poly_reel(base[0],1/norme(base[0])))
    for i in range(1,n+1):
        q=base[i]
        for j in range(i):
            proj=prod_poly_reel(legendre[j],-scalaire(base[i],legendre[j]))
            q=somme_poly(q,proj)
        legendre.append(prod_poly_reel(q,1/norme(q)))
    #pour les polynomes de legendre, on multiplie par la norme
    """for i in range(n+1):
        legendre[i]=prod_poly_reel(legendre[i],(2/(2*i+1))**0.5)"""

    print("base legendre")
    for i in range(n+1):
        print("p_"+str(i)+"(X) = ",end=" ")
        print_poly(legendre[i])
```

5.

```
    import matplotlib.pyplot as plt

    xliste=[i/100-1 for i in range(201)]
    for i in range(6):
        yliste=[valeur_poly(legendre[i],x) for x in xliste]
        plt.plot(xliste,yliste)
    plt.show()
```