Série de révision Python – Exercices extraits de concours

Corrigé

Problème 1 :

1- Fonction **init** permettant d'initialiser la liste :

```
1- def init():
2-     return [1]*100
```

3- Fonction **multiple :**

```
1. def multiple(L,ind):
2.     for i in range(ind+1,len(L)):
3.         if i % ind == 0:
4.             L[i] = 0
```

4- Fonction **suivant :**

```
1. def suivant(L,ind):
2.     i = ind + 1
3.     while L[i] != 1:
4.         i += 1
5.     return(i)
```

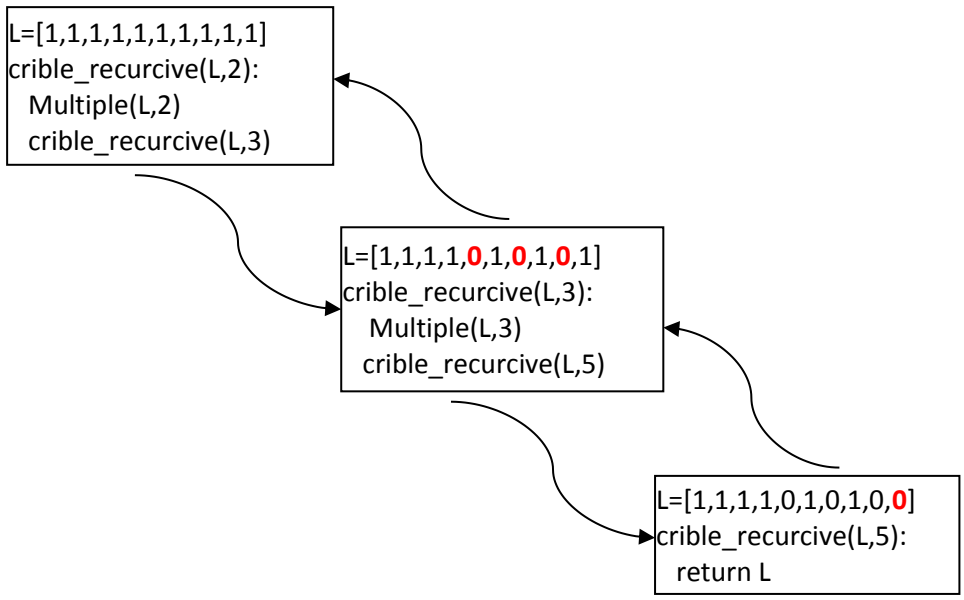
5- Fonction **crible :**

```
1. from math import sqrt
2.
3. def crible(L):
4.
5.     index = 2
6.     while (index <= int(sqrt(len(L)))):
7.         multiple(L, index)
8.         index = suivant(L, index)
9.
10.    P = []
11.    for i in range(2,len(L)):
12.        if L[i] == 1:
13.            P.append(i)
14.
15.    return P
```

6- Fonction récursive **crible_recursive**

```
1. from math import sqrt
2.
3. i = 2
4. L = init()
5. limite= int(sqrt(len(L)))
6.
7. def crible_recursive(L,i):
8.     if i < limite:
9.         multiple(L,i)
10.        i = suivant(L,i)
11.        return crible_recursive(L,i)
12.    else:
13.        return L
```

7- Schéma d'exécution de la fonction récursive pour N = 10



8- Programme principal

```
1. #avec la fonction itérative
2. def main():
3.     L = init()
4.     p =  crible(L)
5.     print(p)
6.
7. #Avec la fonction récursive
8. if __main__ == '__name__':
9.         from math import sqrt
10.        L = init()
11.        index = 2
12.        crible_recursive(L,index)
13.        premiers = [i for i in range(2,len(L)) if L[i]]
14.        print(premiers)
```

Problème 2 :

1- La function **saisie_deg** :

```
1. def saisie_deg():
2.     while True:
3.         try:
4.             n = int(input("donner le degré du polynôme :"))
5.             if n>0:
6.                 break #fin de la boucle while
7.         except ValueError:
8.             continue #continuer dans la boucle while
9.     return(n)
```

2- Fonction **saisie_poly**:

```
1. def saisie_poly(deg):
2.     P = []
3.     for i in range(deg+1):
4.       while True:
5.         try:
6.            coef = float(input("Donner le coefficient P["+str(i)+"]="))
7.         except ValueError:
8.            print ("coefficient invalide")
9.         else:
10.           #il faut éviter d'avoir un coef nul à la fin du polynôme
11.           if i < deg or (i == deg and coef != 0):
12.               P.append(c)
13.               break #arrêt de la boucle while
14.    return(P)
```

3- Fonction **derive**:

Solution de $P'(x)=\sum_{i=1}^{n} i\, a_i\, x^{i-1}$

```
1. #solution 1
2.
3. def derive(p):
4.     d = []
5.     for i in range(1, len(p)):
6.         d.append(i * p[i])
7.     return(d)
8.
9. #solution 2
10.
11.def derive(p):
12.   return [i * p[i] for i in range(1,len(p))]
```

4- Fonction **opp_poly**:

```
# il ne faut pas modifier la lister P

# Solution 1

def opp_poly(P,deg):
    OP = []
    for i in range(deg+1):
        OP.append(-P[i])
    return(OP)

# Solution 2

def opp_poly(P,deg):
    return [-P[i] for i in range(deg+1)]

# Solution 3

def opp_poly(P):
    return [-i for i in P]
```

5- Fonction **add_poly** :

```
# Solution 1

def add_poly (P1, P2, deg1, deg2):
    S = []
    m = min(deg1,deg2)

    for i in range(m+1):
        S.append(P1[i]+P2[i])
    if m == deg1: S = S+P2[m+1:]
    if m == deg2: S = S+P1[m+1:]

    return(S)

# Solution 2

def add_poly (P1, P2, deg1, deg2):

    while len(P1) != len(P2):
        if len(P1) < len(P2):
            P1.append(0)
        elif len(P2) < len(P1):
            P2.append(0)

    return [P1[i] + P2[i] for i in range(len(P1))]

# Solution 3
# Opérateur ternaire
# Résultat_if_cond_True if condition else Résultat_if_cond_False

def add_poly (p1, p2, n1, n2):
    p11 = p1 + [0] * (n2-n1) if n2>n1 else p1
    p21 = p2 + [0] * (n1-n2) if n1>n2 else p2
    p3 = [0] * len(p11)
    for i in range(len(p3)):
        p3[i] = p11[i] + p21[i]
    return p3
```

```
# Solution 4

def add_pol(p1,p2):
    return [
                p1[i]+p2[i] if i<len(p1) and i<len(p2) else
                p1[i] if i<len(p1) and i>=len(p2) else p2[i]
                for i in range(max(len(p1),len(p2)))
            ]
```

6- Fonction **mul_poly** :

```
# Solution 1

def mul_poly (P1, P2, deg1, deg2):
     P = []
     for k in range(deg1+deg2+1):
         # calcul de ck
         c=0
         for i in range(deg1+1):
             for j in range(deg2+1):
                 if k == i+j:
                     c += P1[i]*P2[j]
         P.append(c)
    return(P)

# Solution 2

def Mul_poly (P1, P2, deg1, deg2):
    P = [0] * (deg1+ deg2 + 1)
    for i in range(deg1+1):
        for j in range(deg2+1):
            P[i+j] += P1[i]*P2[j]
    return(P)
```

7- Programme principal

```
deg = saisie_deg()
P = saisie_poly(deg); print("P = ", P)
D = derive(P,deg); print("D = ", D)
O = opp_poly(P,deg); print("O = ", O)
A = add_poly(P,D,deg,deg-1); print("A = ", A)
M = mul_poly(P,D,len(P)-1,len(D)-1); print("M = ", M)
```

Problème 3 :

1. Fonction **saisie** permettant de saisir un entier strictement positif

```
def saisie():
    while True:
        try:
            n=int(input("donner un entier strictement positif"))
            if n>0:
                return(n)
        except ValueError:
            continue
```

10. Fonction **ajout_titre :**

```
def ajout_titre(Ltitre,Lnum,Lexp):
    t = input('Donner un titre')
    while True:
        try:
            n = int(input("donner le numero du livre"))
            if n>0 and not(n in Lnum):
                break
        except ValueError:
            continue
nexp = Saisie()
Ltitre.append(t)
Lnum.append(n)
Lexp.append(nexp)
```

11. Fonction **etat_biblio**

```
def etat_biblio(Ltitre,Lnum,Lexp):
    for i in range(len(Ltitre)):
        print(Lnum[i],Ltitre[i],Lexp[i],sep=' ')
```

12. Fonction **auto_emprunt**

```
def aut_emprunt(CIN, LE,LP):
    # vérifier qu'il n'a pas déjà un livre à rendre
    i = 0
    B1 = False
    while B1==False and i<=len(LE)-1:
        if CIN==LE[i][1]:
            B1=True
        else:
            i+=1
    # Vérifier qu'il n'est pas pénalisé
    B2=CIN not in LP
    return(not B1 and B2)
```

14. Fonction **calcul_date**

```
# Solution 1

def calcul_date(j,m):
    jr=j+10
    mr=m
    if jr>31 and (m==1 or m==3 or m==5 or m==7 or m==8 or m==10 or m==12):
        jr=jr-31
        mr+=1
        mr = mr if mr<=12 else 1
    if jr>30 and (m==4 or m==6 or m==9 or m==11):
        jr=jr-30
        mr+=1
    if jr>28 and m==2:
        jr=jr-28
        mr=3
    return(jr,mr)

# Solution 2 : bibliothèque datetime

def calcul_date(j,m):

    import datetime

    annee_emprunt = 2020
    mois_emprunt = m
    jour_emprunt = j

    date_emprunt = datetime.date(annee_emprunt,mois_emprunt,jour_emprunt)
    date_retour = date_emprunt.fromordinal(date_emprunt.toordinal()+10)

    return(date_retour.year , date_retour.month,date_retour.day)
```

1. Fonction **emprunt**

```
def emprunt(LP,LE,Lexp,Lnum):
    while True:
        try:
            CIN=int(input("donner le cin de l'emprunteur"))
            if type(CIN)==int and CIN>10000000 and CIN< 99999999:
                break
        except ValueError:
            continue
    if aut_emprunt(CIN,LE,LP):
        while True:
            num=int(input('saisir le numero du livre'))
            if num in Lnum:
                break
        k=Lnum.index(num)
        if Lexp[k]>0:
            je=int(input("saisir la date de l'emprunt"))
            me=int(input("saisir le mois de l'emprunt"))
            Lexp[k]-=1
            jr,mr=Calcul_date(je,me)
            LE.append([CIN,num,je,me,jr,mr,0])
```

21. Fonction **maj_penalite**

```
def maj_penalite(LE,LP,j,m):
    for l in LE:
        if l[5]>m or l[5]==m and L[4]>j:
            l[6]=1
            if not L[0] in LP:
                LP.append(CIN)
```

7. Fonction **retour**

```
def retour(num,j,m,LE,Lexp,LP):
    MAJ_penalite(LE,LP,j,m)
    i=0
    while LE[i][1]!=num and i< len(LE):
        i+=1
    del LE[i]
    k=Lexp.index(num)
    Lexp[k]+=1
```