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)
```

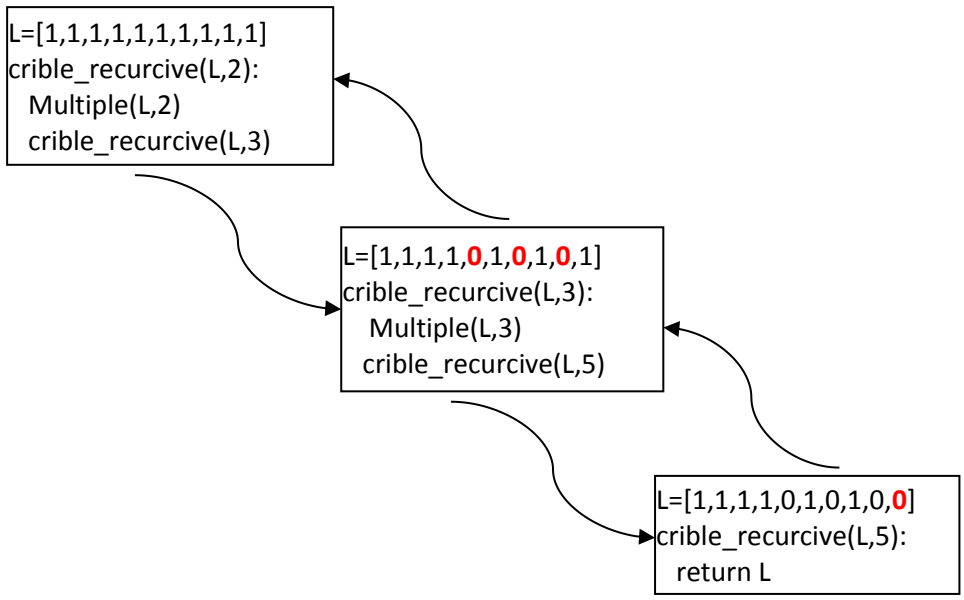
5- Fonction **crible :**

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

Problème 2 :

1- La function **saisie_deg** :

```
def saisie_deg():
    while True:
        try:
            n = int(input("donner le degré du polynôme :"))
            if n>0:
                break #fin de la boucle while
        except ValueError:
            continue #continuer dans la boucle while
    return(n)
```

2- Fonction **saisie_poly**:

```
def saisie_poly(deg):
    P = []
    for i in range(deg+1):
      while True:
        try:
           coef = float(input("Donner le coefficient P["+str(i)+"]="))
        except ValueError:
           print ("coefficient invalide")
        else:
           #il faut éviter d'avoir un coef nul à la fin du polynôme
           if i < deg or (i == deg and coef != 0):
               P.append(c)
               break #arrêt de la boucle while et passer à i suivant
    return(P)
```

3- Fonction **derive**:

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

```
#solution 1

def derive(p):
    d = []
    for i in range(1, len(p)):
        d.append(i * p[i])
    return(d)

#solution 2

def derive(p):
   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

```

```python
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** :

```python
# 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)
```