

*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
```

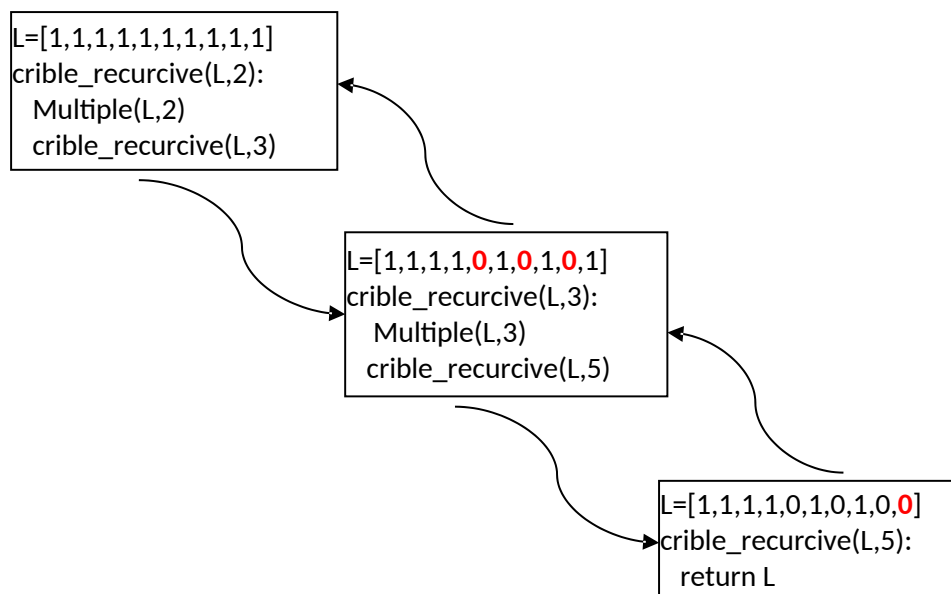
6- Fonction récursive **crible_recursive**

```

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 fonction **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 et passer à i suivant
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**:

```

1. # il ne faut pas modifier la lister P
2.
3. # Solution 1
4.

```

```

5. def opp_poly(P,deg):
6.     OP = []
7.     for i in range(deg+1):
8.         OP.append(-P[i])
9.     return(OP)
10.
11. # Solution 2
12.
13. def opp_poly(P,deg):
14.     return [-P[i] for i in range(deg+1)]
15.
16. # Solution 3
17.
18. def opp_poly(P):
19.     return [-i for i in P]

```

5- Fonction **add_poly** :

```

1. # Solution 1
2. def add_poly (P1, P2, deg1, deg2):
3.     S = []
4.     m = min(deg1,deg2)
5.
6.     for i in range(m+1):
7.         S.append(P1[i]+P2[i])
8.     if m == deg1: S = S+P2[m+1:]
9.     if m == deg2: S = S+P1[m+1:]
10.
11.     return(S)
12.
13. # Solution 2
14.
15. def add_poly (P1, P2, deg1, deg2):
16.
17.     while len(P1) != len(P2):
18.         if len(P1) < len(P2):
19.             P1.append(0)
20.         elif len(P2) < len(P1):
21.             P2.append(0)
22.
23.     return [P1[i] + P2[i] for i in range(len(P1))]
24.
25. # Solution 3
26. # Opérateur ternaire
27. # Résultat_if_cond_True if condition else Résultat_if_cond_False
28.
29. def add_poly (p1, p2, n1, n2):
30.     p11 = p1 + [0] * (n2-n1) if n2>n1 else p1
31.     p21 = p2 + [0] * (n1-n2) if n1>n2 else p2
32.     p3 = [0] * len(p11)
33.     for i in range(len(p3)):
34.         p3[i] = p11[i] + p21[i]
35.     return p3
36. # Solution 4
37.
38. def add_pol(p1,p2):
39.     return [
40.         p1[i]+p2[i] if i<len(p1) and i<len(p2) else
41.         p1[i] if i<len(p1) and i>=len(p2) else p2[i]
42.         for i in range(max(len(p1),len(p2)))
43.     ]

```

6- Fonction **mul_poly** :

```
1. # Solution 1
2.
3. def mul_poly (P1, P2, deg1, deg2):
4.     P = []
5.     for k in range(deg1+deg2+1):
6.         # calcul de ck
7.         c=0
8.         for i in range(deg1+1):
9.             for j in range(deg2+1):
10.                if k == i+j:
11.                    c += P1[i]*P2[j]
12.            P.append(c)
13.    return(P)
14.
15. # Solution 2
16.
17. def Mul_poly (P1, P2, deg1, deg2):
18.     P = [0] * (deg1+ deg2 + 1)
19.     for i in range(deg1+1):
20.         for j in range(deg2+1):
21.             P[i+j] += P1[i]*P2[j]
22.     return(P)
```

7- Programme principal

```
1. deg = saisie_deg()
2. P = saisie_poly(deg); print("P = ", P)
3. D = derive(P,deg); print("D = ", D)
4. O = opp_poly(P,deg); print("O = ", O)
5. A = add_poly(P,D,deg,deg-1); print("A = ", A)
6. M = mul_poly(P,D,len(P)-1,len(D)-1); print("M = ", M)
```