

```
# Ch3-Prog.py
```

```
01| def plus_proches1(L):
02|     n=len(L)
03|     paire=[0,1]
04|     dmin=abs(L[0]-L[1])
05|     for i in range(0,n):
06|         for j in range(0,n):
07|             if i!=j and dmin>abs(L[i]-L[j]):
08|                 dmin=abs(L[i]-L[j])
09|                 paire=[i,j]
10|     return(paire)
11|
12| def plus_proches2(L):
13|     n=len(L)
14|     paire=[0,1]
15|     dmin=abs(L[0]-L[1])
16|     for i in range(0,n-1):#toutes les paires [...,n] seront testées
grâce à la seconde boucle
17|         for j in range(i+1,n):#toutes les paires [i,j] avec j<i ont déjà
été testées sous la forme [j,i] dans les itérations précédentes
18|             if dmin>abs(L[i]-L[j]):
19|                 dmin=abs(L[i]-L[j])
20|                 paire=[i,j]
21|     return(paire)
22|
23|
#-----#
24|
25| def tri_bulles(L):
26|     j=len(L)-1
27|     while j>0:
28|         tri=True
29|         for i in range(0,j):
30|             if L[i]>L[i+1]:
31|                 L[i],L[i+1]=L[i+1],L[i]
32|                 tri=False
33|         j=j-1
34|         if tri:
35|             j=0
36| #il n'est pas nécessaire que la fonction effectue un renvoi puisque les
modifications sont appliquées directement à la liste L
37|
38|
#-----#
39|
40| def recherche(motif, texte):
41|     n=len(texte)
42|     m=len(motif)
43|     for j in range(0,n-m+1):
44|         i=0
45|         while i<m and texte[j+i]==motif[i]:
46|             i=i+1
47|         if i==m:
48|             return(j)
49|
50| def distances(motif):
51|     m=len(motif)
```

```

52|     dico={chr(i):m for i in range(0,256)}#on initialise le tableau en
    considérant qu'aucun caractère n'apparaît dans le motif
53|                                     #char(i) permet de convertir le
    nombre i en caractère ASCII
54|     for i in range(0,m-1):#le caractère de rang m-1 est exclu car, s'il
    n'apparaît pas avant dans le motif, sa distance reste m
55|         dico[motif[i]]=m-1-i#pour les caractères apparaissant dans le
    motif (sauf le dernier), on modifie la distance
56|     return(dico)
57|
58| def boyer_moore(texte,motif):
59|     m=len(motif)
60|     n=len(texte)
61|     d=distances(motif) # on effectue le pré-traitement du motif
62|     s=m-1 # on commence par le dernier caractère du motif
63|     sol=[] # on crée une liste afin de stocker les rangs des occurrences
    du motif dans le texte
64|     while s<n:
65|         i=m-1
66|         while i>=0 and motif[i]==texte[s]:# tant qu'il y a
    correspondance entre les caractères du motif et du texte
67|             s=s-1 # on recule d'un cran afin de
    comparer le caractère précédent du motif
68|             i=i-1 # au caractère précédent du
    texte
69|         if i<0: # si le motif est trouvé (i=-1
    car m passages dans la boucle précédente)
70|             sol.append(s+1) # on stocke dans la liste sol
    le rang dans le texte du premier terme du motif
71|             s=s+max(d[texte[s]],m-i)
72|     return(sol)
73|
74|

```