```python
''' Lycée Vaugelas MP 2023-2024
    TP n° 1 : Exercice 2, tri par insertion
'''

'''Le tri d'un tableau s'effectue en place en insérant successivement chaque
    L[i], pour i = 1 à n-1 dans L[0:i] déjà trié. Une boucle for permet de
    parcourir le tableau et une boucle while décale vers la droite les éléments
    à gauche de L[i], tant qu'ils sont supérieurs à L[i]. Il reste à positionner
    L{i] dans le trou laissé vide.
'''
# première version
def tri_insertion(L):
    for i in range(1, len(L)):
        x = L[i]
        j = i - 1
        while j >= 0 and L[j] > x:
            L[j + 1] = L[j]
            j -= 1
        L[j + 1] = x


arr = [12, 11, 13, 5, 6]
tri_insertion(arr)
print(arr)
arr = [1, 2, 3, 4, 5]
tri_insertion(arr)
print(arr)
arr = [5, 4, 3, 2, 1]
tri_insertion(arr)
print(arr)

# seconde verion
def tri_insertion(L):
    for i in range(1, len(L)):
        x = L[i]
        j = i
        while j and L[j - 1] > x:
            L[j] = L[j - 1]
            j -= 1
        L[j] = x

arr = [12, 11, 13, 5, 6]
tri_insertion(arr)
print(arr)
arr = [1, 2, 3, 4, 5]
tri_insertion(arr)
print(arr)
arr = [5, 4, 3, 2, 1]
tri_insertion(arr)
print(arr)

# troisième version
def tri_insertion(L):
    for i in range(1, len(L)):
        x, y = L[i], L[i - 1]
        j = i
        while y > x:
            L[j] = y
            j -= 1
            if j == 0 : break
            y = L[j - 1]
        L[j] = x

arr = [12, 11, 13, 5, 6]
tri_insertion(arr)
print(arr)
arr = [1, 2, 3, 4, 5]
tri_insertion(arr)
print(arr)
arr = [5, 4, 3, 2, 1]
tri_insertion(arr)
print(arr)
```

```
73
74  ''' L'usage d'une sentinelle (élément plus petit que tous les autres)en début de
75      tableau permet d'éviter de tester j
76      Le nombre de comparaisons est toujours égal au nombre d'affectations et de
77      lecture :
78      ~ n dans le meilleur des cas
79      ~  n²/2 dans le pire des cas, n(n-1)/2 plus précisément
80      ~ n² / 4 en moyenne
81      Le tri par insertion d'un tableau de réalise en place, il est stable et
82      online.
83      Correction : l'invariant de boucle "L[0:i] trié" est facile à montrer,
84      il démontre la correction en sortie.
85  '''
86
87  # quatrième version avec sentinelle supposée présente
88  def tri_insertion(L):
89      for i in range(1, len(L)):
90          x, y = L[i], L[i - 1]
91          j = i
92          while y > x:
93              L[j] = y
94              j -= 1
95              y = L[j - 1]
96          L[j] = x
97
98  arr = [-100, 12, 11, 13, 5, 6]
99  tri_insertion(arr)
100 print(arr)
101 arr = [-100, 1, 2, 3, 4, 5]
102 tri_insertion(arr)
103 print(arr)
104 arr = [-100, 5, 4, 3, 2, 1]
105 tri_insertion(arr)
106 print(arr)
107
108 # cinquième version droite-gauche, avec sentinelle dynamique
109 def tri_insertion(L):
110     L.append(max(L))
111     for i in range(len(L) - 3, -1, -1):
112         x, y = L[i], L[i + 1]
113         j = i
114         while y < x:
115             L[j] = y
116             j += 1
117             y = L[j + 1]
118         L[j] = x
119     L.pop()
120
121 arr = [12, 11, 13, 5, 6]
122 tri_insertion(arr)
123 print(arr)
124 arr = [1, 2, 3, 4, 5]
125 tri_insertion(arr)
126 print(arr)
127 arr = [5, 4, 3, 2, 1]
128 tri_insertion(arr)
129 print(arr)
130 arr = [5, 2, 3, 4, 1]
131 tri_insertion(arr)
132 print(arr)
133 ''' Dans ce cas l'invariant est L[i+1:] trié.
134     On choisit max(L) comme sentinelle car la complexité n'en est pas pénalisée
135     et l'on garantit ainsi le polymorphisme de la fonction. '''
136
```