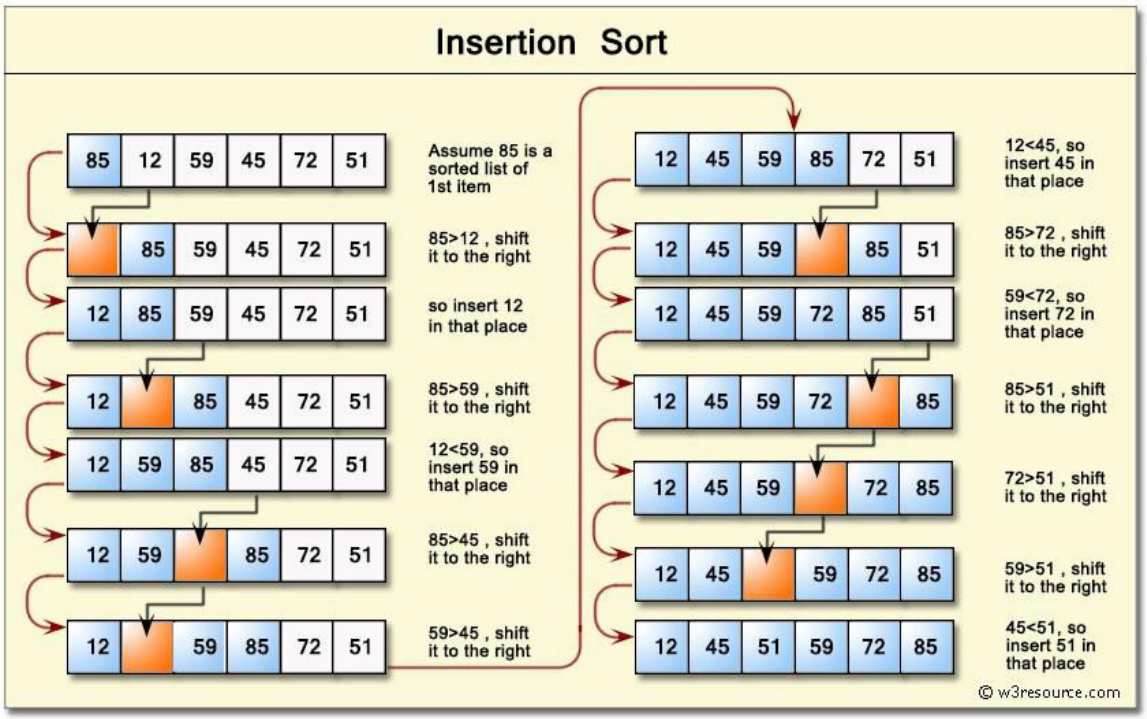
Entwicklung eines Algorithmus InsertionSort

Zu einer natürlichen Zahl n ist ein Array **a** mit den **n** Komponenten **a[0], . . . , a[n-1]** gegeben (in Python läßt sich ein Array als Liste definieren), für die die Operationen **=** , **<** und **>** definiert sind.

Ziel: Die Inhalte der Komponenten sind gemäß dem Algorithmus „Sortieren durch direktes Einfügen" (InsertionSort) so anzuordnen, daß gilt:

a[0] ≤ a[1] ≤ . . . ≤ a[n-1]

Beispiel (n = 6):



a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
85	12	59	45	72	51

Die aus der Komponente **a[0]** bestehende 1-elementige Teilliste gilt als sortiert, die aus den Komponenten **a[1] , . . . , a[n-1]** bestehende Teilliste ist zu Anfang unsortiert.

Wir verwenden die Variable `current` als temporäre Variable.

1. Schritt:

```
current = a[1]
i = 1 - 1
if current < a[i]:
        a[i+1] = a[i]
        a[i]   = current
```

Ergebnis des 1. Schritts:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
12	85	59	45	72	51

Die aus den Komponenten **a[0], a[1]** bestehende Teilliste ist sortiert, der Bereich **a[2] , . . . , a[5]** unsortiert.

2. Schritt:

```
current = a[2]
i = 2 - 1
while i >= 0:
   if current < a[i]:
         a[i+1] = a[i]
         a[i]   = current
   i = i - 1
```

Ergebnis des 2. Schritts:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
12	59	85	45	72	51

Die aus den Komponenten **a[0], a[1], a[2]** bestehende Teilliste ist sortiert, der Bereich **a[3] , . . . , a[5]** unsortiert.

3. Schritt:

```
current = a[3]
i = 3 - 1
while i >= 0:
   if current < a[i]:
         a[i+1] = a[i]
         a[i]   = current
   i = i - 1
```

Ergebnis des 3. Schritts:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
12	45	59	85	72	51

Die aus den Komponenten **a[0], . . . , a[3]** bestehende Teilliste ist sortiert, der Bereich **a[4] , a[5]** unsortiert.

4. Schritt:

```
current = a[4]
i = 4 - 1
while i >= 0:
   if current < a[i]:
         a[i+1] = a[i]
         a[i]   = current
   i = i - 1
```

Ergebnis des 4. Schritts:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
12	45	59	72	85	51

Die aus den Komponenten **a[0], . . . , a[4]** bestehende sortierte Teilliste ist mit der Komponente **a[5]** zu einer sortierten Gesamtliste zu verschmelzen.

5. Schritt:

```
current = a[5]
i = 5 - 1
while i >= 0:
    if current < a[i]:
          a[i+1] = a[i]
          a[i]   = current
    i = i - 1
```

Ergebnis des 5. Schritts:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
12	45	51	59	72	85

Bei dem gewählten Beispiel (n = 6) ist der Anweisungsblock

```
current = a[j]
i = j - 1
while i >= 0:
    if current < a[i]:
          a[i+1] = a[i]
          a[i]   = current
    i = i - 1
```

für j = 1, 2, . . . , 5 zu wiederholen.

Allgemein halten wir fest:

Die zu sortierende Gesamtliste besteht vor jedem Schritt aus einer bereits sortierten Teilliste und einer unsortierten Teilleiste; vor dem ersten Schritt ist die aus dem einen Element a[0] bestehende Liste sortiert und die Liste a[1] , . . , a[n-1] unsortiert. Nachfolgend wird das jeweils erste Element der unsortierten Teilliste an der richtigen Stelle in die sortierte Teilliste eingefügt, so daß der sortierte Bereich mit jedem Schritt wächst, bis die gesamte Liste sortiert ist.

Falls das Array **a** aus den **n** Komponenten **a[0] , . . . , a[n-1]** besteht, ist der Anweisungsblock

```
current = a[j]
i = j - 1
while i >= 0:
    if current < a[i]:
          a[i+1] = a[i]
          a[i]   = current
    i = i - 1
```

nacheinander für j = 1, . . . , n–1 zu wiederholen. Folglich implementieren wir diesen Anweisungsblock als Schleifenrumpf einer geeignet initialisierten `for`- oder `while`-Schleife.