

...cont'd

- 1 Start a new program by declaring a function to receive a list reference as input and begin a loop to store the current element's value

```
def insertion_sort( array ) :  
  
    for index in range( 1 , len( array ) ) :  
        value = array[ index ]  
  
        # Algorithm sequence to be added here.  
  
        print( '\tResolving element[', index , ']' to ' , array )
```



insertion.py

- 2 Next, add the algorithm sequence to repeatedly insert the current value if smaller than that in the current element

```
while array[ index-1 ] > value and index >= 1 :  
    array[ index ] = array[ index-1 ]  
    index -=1  
array[ index ] = value
```

- 3 Now, add statements to create and display an unsorted list

```
array = [ 5 , 3 , 1 , 2 , 6 , 4 ]  
print( 'Insertion Sort...\nArray :', array )
```

- 4 Finally, add statements to call the algorithm function and display the list once more – to see the list sorted in place

```
insertion_sort( array )  
print( 'Array :', array )
```

```
Python Shell  
File Edit Shell Debug Options Windows Help  
>>> ----- RESTART -----  
>>>  
Insertion Sort...  
Array : [5, 3, 1, 2, 6, 4]  
    Resolving element[ 0 ] to [3, 5, 1, 2, 6, 4]  
    Resolving element[ 0 ] to [1, 3, 5, 2, 6, 4]  
    Resolving element[ 1 ] to [1, 2, 3, 5, 6, 4]  
    Resolving element[ 4 ] to [1, 2, 3, 5, 6, 4]  
    Resolving element[ 3 ] to [1, 2, 3, 4, 5, 6]  
Array : [1, 2, 3, 4, 5, 6]  
>>> |
```



On some iterations this algorithm recognizes that elements in the “unsorted” part are already sorted following earlier insertions.