

...cont'd

- 1 Start a new program by declaring a function to receive a list reference as input and begin an outer loop to repeatedly iterate through the array list
`def bubble_sort(array) :`

```
    for index in range( len( array ) ) :
```

```
        # Algorithm sequence to be added here.
```

- 2 Next, add the algorithm sequence to iterate through the array list elements, up to the penultimate element, and swap values if the next is greater than the current value

```
    for element in range( ( len( array ) - 1 ) - index ) :  
        if array[ element ] > array[ element + 1 ] :  
            array[ element ] , array[ element + 1 ] = \  
                array[ element + 1 ] , array[ element ]
```

```
        print( '\tResolving element[', element , \  
                ']' to ' , array )
```

- 3 Now, add statements to create and display an unsorted list
`array = [5 , 3 , 1 , 2 , 6 , 4]`
`print('Bubble 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
`bubble_sort(array)`
`print('Array :', array)`



bubble.py

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



In average simple cases, insertion sort outperforms selection sort, and selection sort outperforms bubble sort.