Algorithm: Bubble Sort

Bubble Sort

The list is sorted in <u>ascending</u> order

<u>Key Aspects:</u>

- Simplest sorting algorithm.
- Elements move "up" like bubbles.
- Uses a "swapping" mechanism.
- Can be used for small lists.
- Inefficient for large lists.

<u>Algorithm:</u>

- Start with the first element (index 0).
- For each item in the list (except the last one):
    - Compare each element (except the last element) with the element to its right.
    - If the elements are not ordered (if left-item > right-item), swap them.
- Repeat these steps until the list is ordered (no swaps are made).

<u>Optimizations:</u>

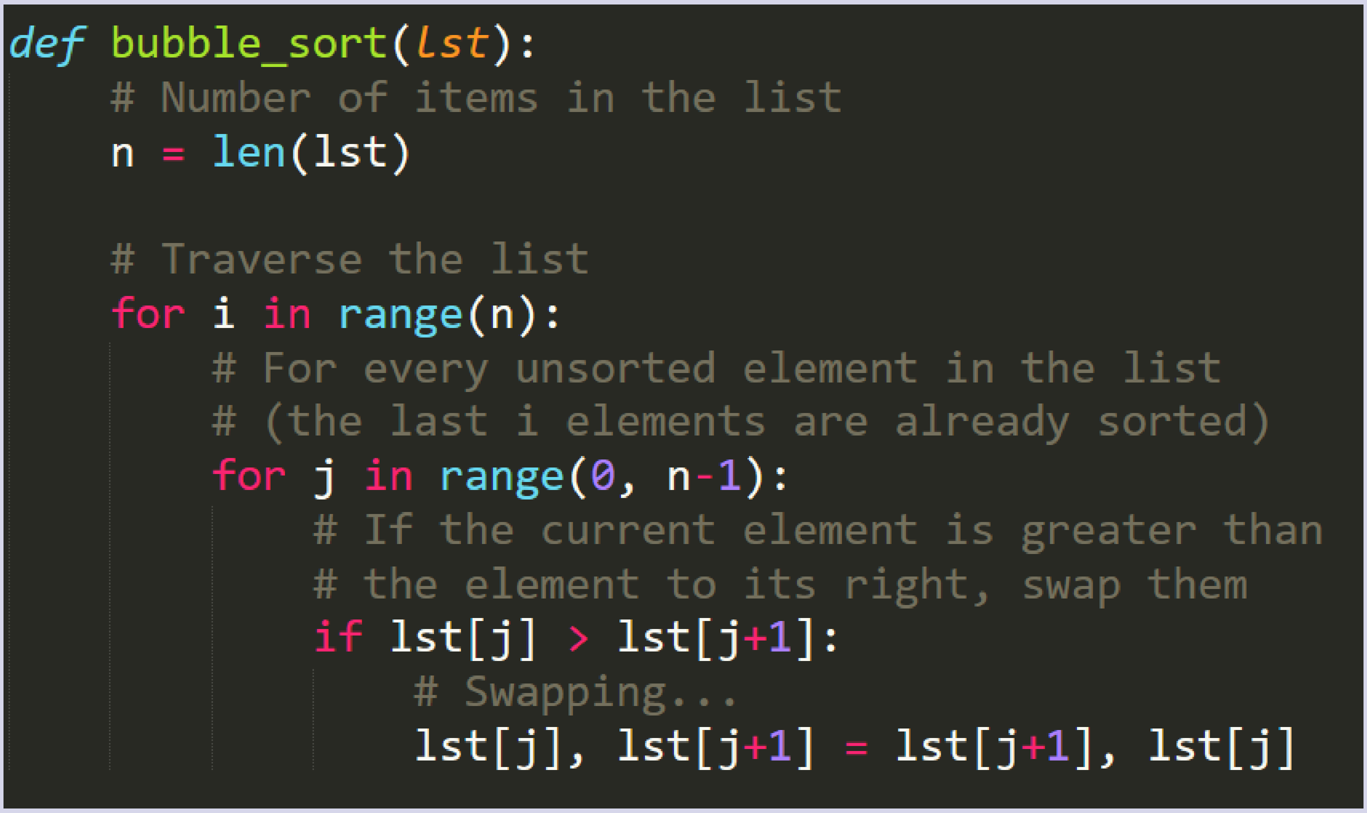
- Add a flag variable "swapped" to stop the process once the list is sorted (when no swaps were made).
- Use n – i – 1 as the end index of the range() function in the inner loop. This will avoid unnecessary repetitions.

<u>Time Complexity:</u>

- Worst-Case Time Complexity: O(n*n) (Quadratic).
- Average-Case Time Complexity: O(n*n) (Quadratic).
- Best-Case Time Complexity: O(n) (Constant).

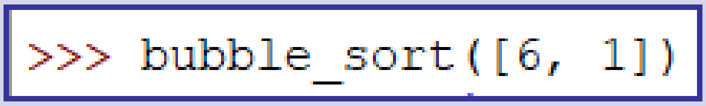
Bubble Sort

Code (without optimizations):

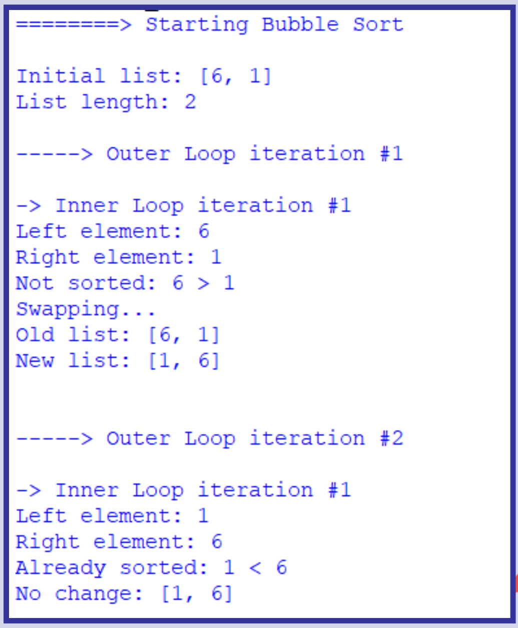


```python
def bubble_sort(lst):
    # Number of items in the list
    n = len(lst)

    # Traverse the list
    for i in range(n):
        # For every unsorted element in the list
        # (the last i elements are already sorted)
        for j in range(0, n-1):
            # If the current element is greater than
            # the element to its right, swap them
            if lst[j] > lst[j+1]:
                # Swapping...
                lst[j], lst[j+1] = lst[j+1], lst[j]
```

Example:



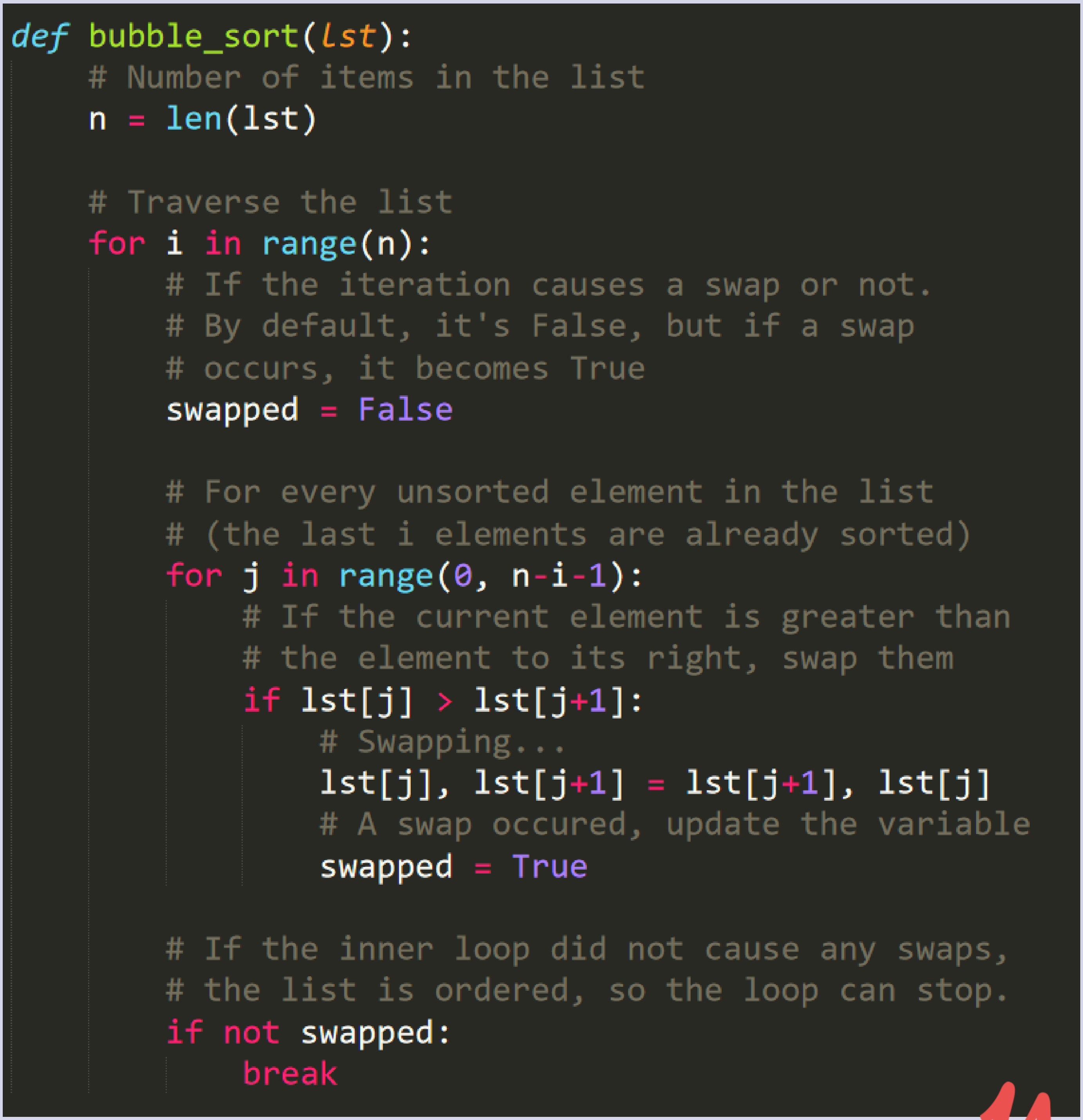
```
>>> bubble_sort([6, 1])
```



```
========> Starting Bubble Sort

Initial list: [6, 1]
List length: 2

-----> Outer Loop iteration #1

-> Inner Loop iteration #1
Left element: 6
Right element: 1
Not sorted: 6 > 1
Swapping...
Old list: [6, 1]
New list: [1, 6]


-----> Outer Loop iteration #2

-> Inner Loop iteration #1
Left element: 1
Right element: 6
Already sorted: 1 < 6
No change: [1, 6]
```

Bubble Sort

Code (with optimizations):

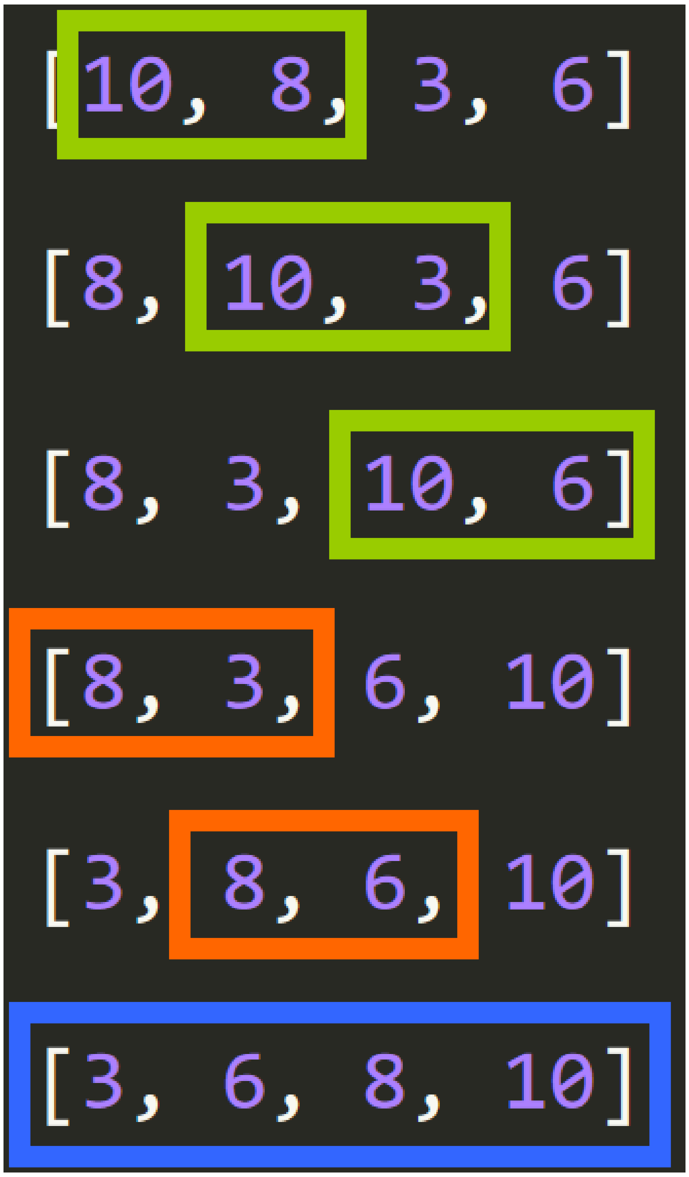


```python
def bubble_sort(lst):
    # Number of items in the list
    n = len(lst)

    # Traverse the list
    for i in range(n):
        # If the iteration causes a swap or not.
        # By default, it's False, but if a swap
        # occurs, it becomes True
        swapped = False

        # For every unsorted element in the list
        # (the last i elements are already sorted)
        for j in range(0, n-i-1):
            # If the current element is greater than
            # the element to its right, swap them
            if lst[j] > lst[j+1]:
                # Swapping...
                lst[j], lst[j+1] = lst[j+1], lst[j]
                # A swap occured, update the variable
                swapped = True

        # If the inner loop did not cause any swaps,
        # the list is ordered, so the loop can stop.
        if not swapped:
            break
```

Bubble Sort

Example:

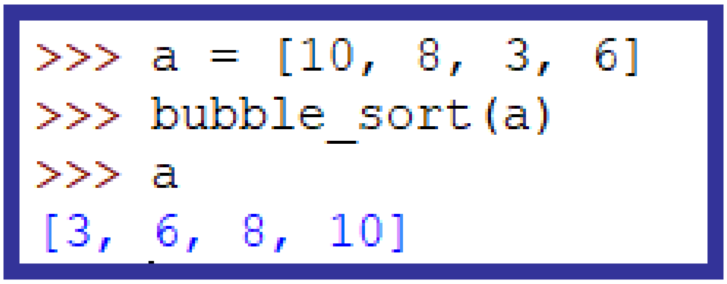
```
>>> a = [10, 8, 3, 6]
>>> bubble_sort(a)
>>> a
[3, 6, 8, 10]
```



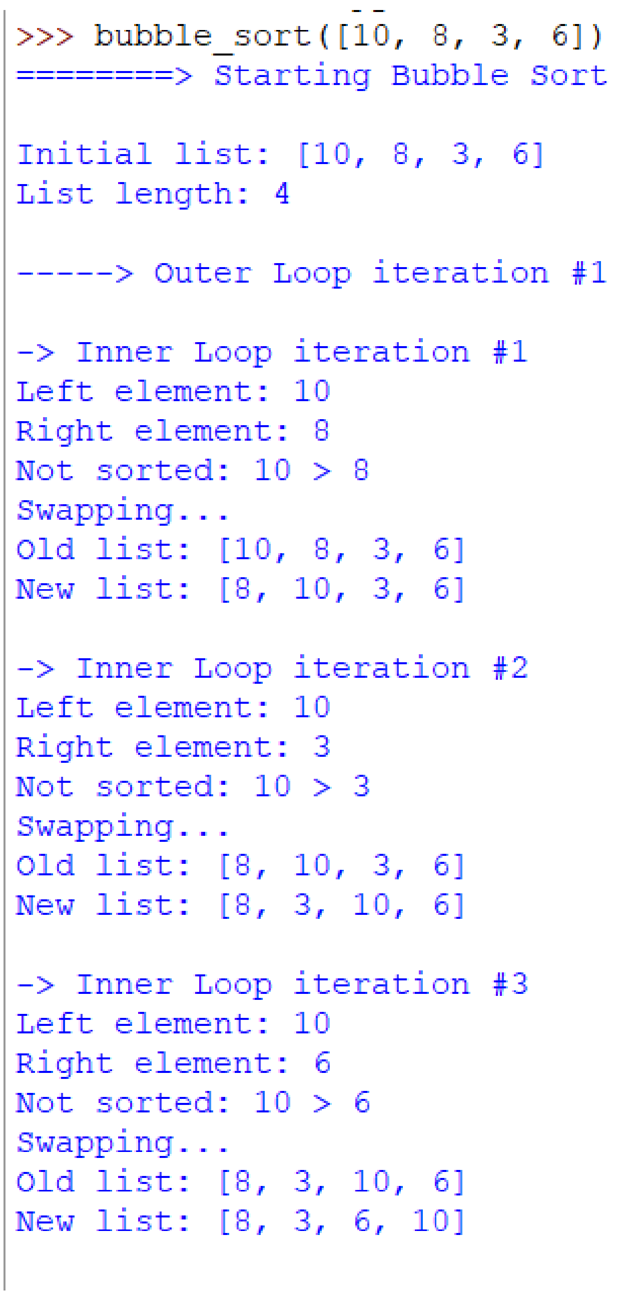
```
[10, 8, 3, 6]

[8, 10, 3, 6]

[8, 3, 10, 6]

[8, 3, 6, 10]

[3, 8, 6, 10]

[3, 6, 8, 10]
```

Bubble Sort

Example:



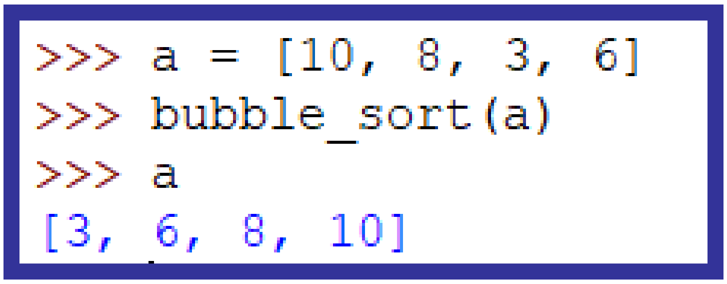
```
>>> a = [10, 8, 3, 6]
>>> bubble_sort(a)
>>> a
[3, 6, 8, 10]
```



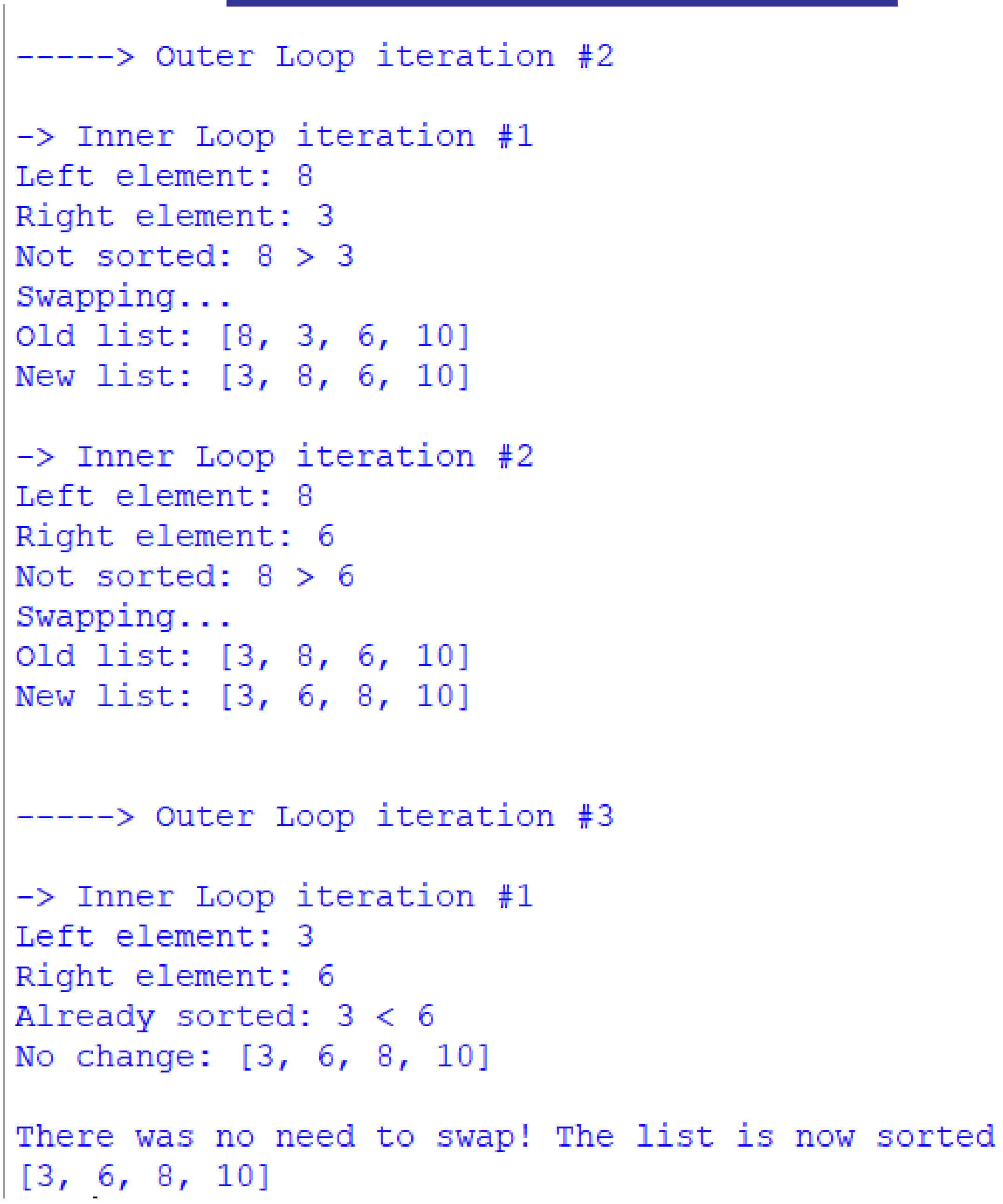
```
>>> bubble_sort([10, 8, 3, 6])
========> Starting Bubble Sort

Initial list: [10, 8, 3, 6]
List length: 4

-----> Outer Loop iteration #1

-> Inner Loop iteration #1
Left element: 10
Right element: 8
Not sorted: 10 > 8
Swapping...
Old list: [10, 8, 3, 6]
New list: [8, 10, 3, 6]

-> Inner Loop iteration #2
Left element: 10
Right element: 3
Not sorted: 10 > 3
Swapping...
Old list: [8, 10, 3, 6]
New list: [8, 3, 10, 6]

-> Inner Loop iteration #3
Left element: 10
Right element: 6
Not sorted: 10 > 6
Swapping...
Old list: [8, 3, 10, 6]
New list: [8, 3, 6, 10]
```

Bubble Sort

Example:



```
>>> a = [10, 8, 3, 6]
>>> bubble_sort(a)
>>> a
[3, 6, 8, 10]
```



```
-----> Outer Loop iteration #2

-> Inner Loop iteration #1
Left element: 8
Right element: 3
Not sorted: 8 > 3
Swapping...
Old list: [8, 3, 6, 10]
New list: [3, 8, 6, 10]

-> Inner Loop iteration #2
Left element: 8
Right element: 6
Not sorted: 8 > 6
Swapping...
Old list: [3, 8, 6, 10]
New list: [3, 6, 8, 10]


-----> Outer Loop iteration #3

-> Inner Loop iteration #1
Left element: 3
Right element: 6
Already sorted: 3 < 6
No change: [3, 6, 8, 10]

There was no need to swap! The list is now sorted
[3, 6, 8, 10]
```