Bubble Sort for All

Contents

Time required: 90 minutes

DRY

Don’t Repeat Yourself

Bubble Sort

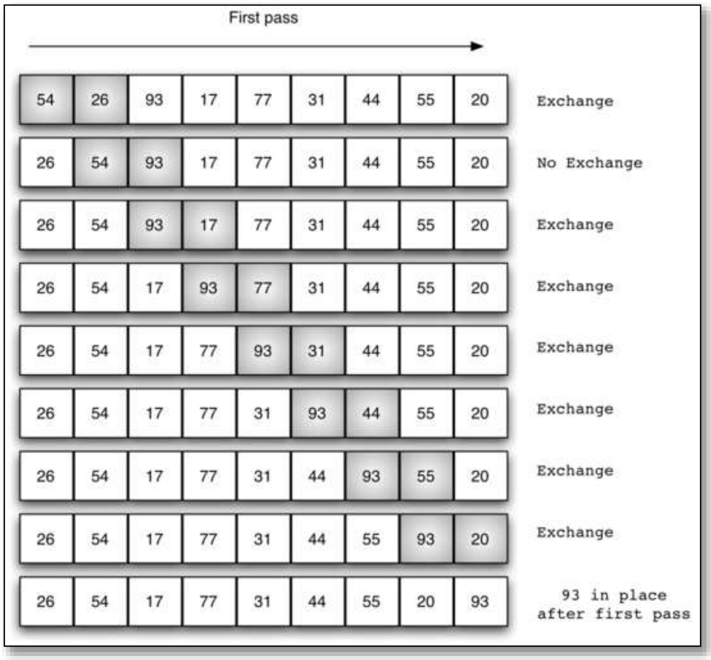
The bubble sort makes multiple passes through an array. It compares adjacent items and exchanges those that are out of order. Each pass through the array places the next largest value in its proper place. In essence, each item “bubbles” up to the location where it belongs.

- Bubble Sort Video
- Yet Another Bubble Sort Video

Bubble Sort Algorithm

```
bubbleSort(array)
  for i <- 1 to indexOfLastUnsortedElement-1
    if leftElement > rightElement
      swap leftElement and rightElement
end bubbleSort
```

The figure below shows the first pass of a bubble sort. The shaded items are being compared to see if they are out of order. If there are n items in the array, then there are n−1 pairs of items that need to be compared on the first pass. It is important to note that once the largest value in the array is part of a pair, it will continually be moved along until the pass is complete.



54	26	93	17	77	31	44	55	20	Exchange
26	54	93	17	77	31	44	55	20	No Exchange
26	54	93	17	77	31	44	55	20	Exchange
26	54	17	93	77	31	44	55	20	Exchange
26	54	17	77	93	31	44	55	20	Exchange
26	54	17	77	31	93	44	55	20	Exchange
26	54	17	77	31	44	93	55	20	Exchange
26	54	17	77	31	44	55	93	20	Exchange
26	54	17	77	31	44	55	20	93	93 in place after first pass

At the start of the second pass, the largest value is now in place. There are n−1 items left to sort, meaning that there will be n−2 pairs. Since each pass places the next largest value in place, the total number of passes necessary will be n−1. After completing the n−1 passes, the smallest item must be in the correct position with no further processing required.

Typically, swapping two elements in an array requires a temporary storage location (an additional memory location). A code fragment such as the following will exchange the i and j items in the array. Without the temporary storage, one of the values would be overwritten.

```
temp = alist[i];
alist[i] = alist[j];
alist[j] = temp;
```

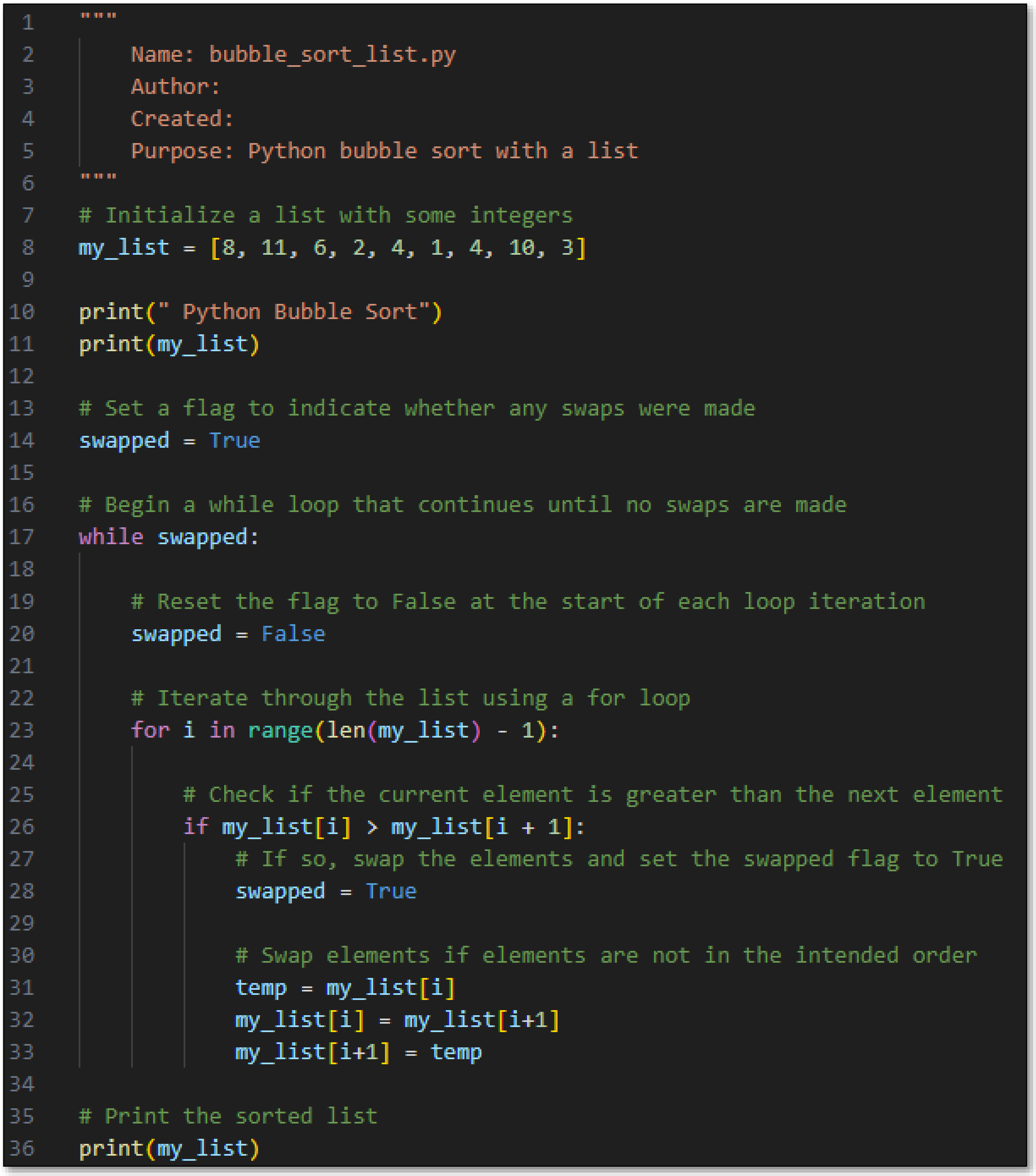
Pseudocode for bubblesort.

```
Step 1: For i = 0 to N-1 repeat Step 2
    Step 2: For J = i + 1 to N – I repeat
         Step 3: if A[J] > A[i]
            Swap A[J] and A[i]
            [End of Inner for loop]
        [End if Outer for loop]
Step 4: Exit
```

Tutorial: Python Bubble Sort

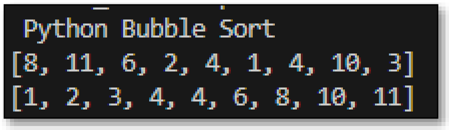
We are going to bubble sort a list in Python. The Python list has a **list.sort()** function. This tutorial is about understanding what might be going on behind the scenes. We aren’t going to use the list.sort() or any other builtin methods, we are going to manually sort using bubble sort.

Create a Python file named **bubble_sort_function.py** Add the following code.



```python
"""
    Name: bubble_sort_list.py
    Author:
    Created:
    Purpose: Python bubble sort with a list
"""
# Initialize a list with some integers
my_list = [8, 11, 6, 2, 4, 1, 4, 10, 3]

print(" Python Bubble Sort")
print(my_list)

# Set a flag to indicate whether any swaps were made
swapped = True

# Begin a while loop that continues until no swaps are made
while swapped:

    # Reset the flag to False at the start of each loop iteration
    swapped = False

    # Iterate through the list using a for loop
    for i in range(len(my_list) - 1):

        # Check if the current element is greater than the next element
        if my_list[i] > my_list[i + 1]:
            # If so, swap the elements and set the swapped flag to True
            swapped = True

            # Swap elements if elements are not in the intended order
            temp = my_list[i]
            my_list[i] = my_list[i+1]
            my_list[i+1] = temp

# Print the sorted list
print(my_list)
```

Example run:

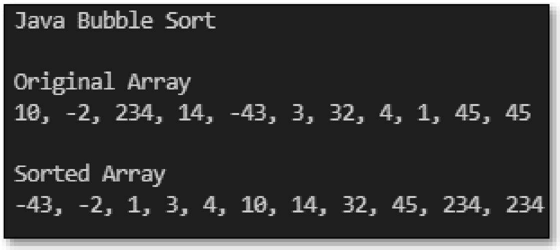


```
 Python Bubble Sort
[8, 11, 6, 2, 4, 1, 4, 10, 3]
[1, 2, 3, 4, 4, 6, 8, 10, 11]
```

Assignment 1: Java Bubble Sort

Same idea, different language. Again, we aren’t going to use any built-in methods, we are going to create a manual bubble sort based on the previous examples.

Example run:

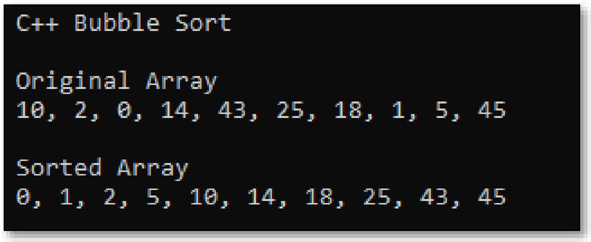


```
Java Bubble Sort

Original Array
10, -2, 234, 14, -43, 3, 32, 4, 1, 45, 45

Sorted Array
-43, -2, 1, 3, 4, 10, 14, 32, 45, 234, 234
```

Assignment 2: CPP Bubble Sort

One more time. Bubble sort time.

Example run:



```
C++ Bubble Sort

Original Array
10, 2, 0, 14, 43, 25, 18, 1, 5, 45

Sorted Array
0, 1, 2, 5, 10, 14, 18, 25, 43, 45
```

Assignment

1. Create and test each program.

Assignment Submission

Submit all code files and a screenshot of each program showing that they work.