



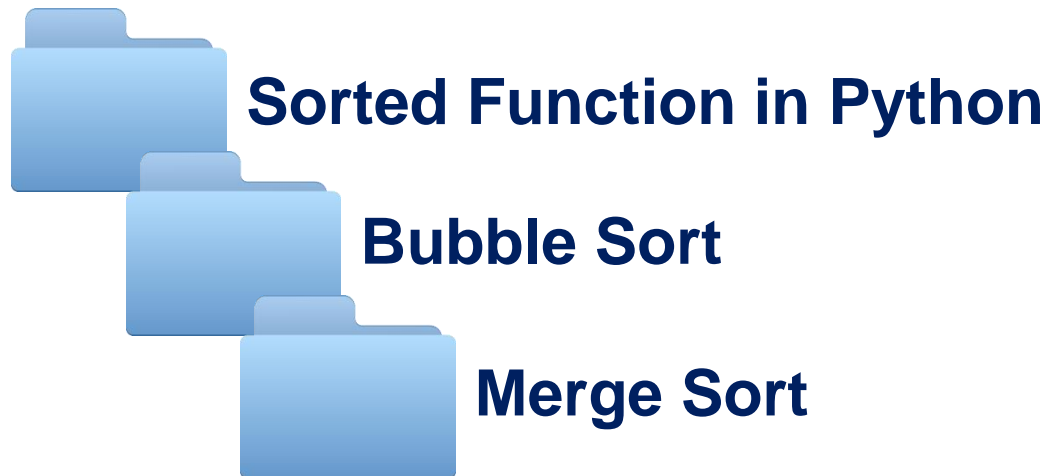
# Algorithm Implementation: Sorting in Python



## At the end of this lesson, you should be able to:

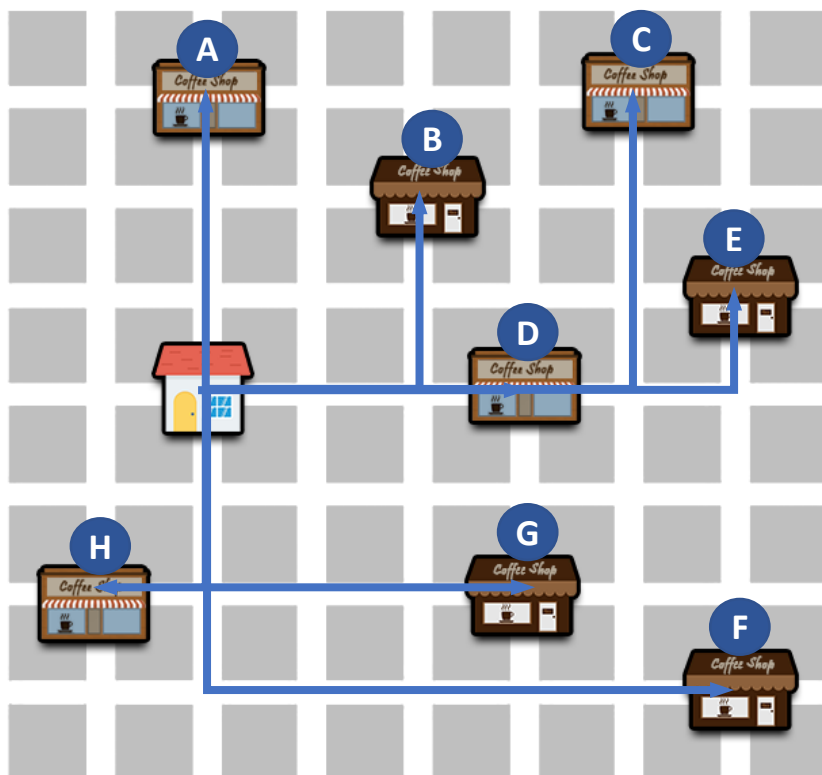
- Use sorted function in Python
- Explain the importance of coding sort algorithms in Python
- Code bubble sort in Python
- Code merge sort in Python
- Read and understand other sorting algorithms written in Python
- Apply your knowledge and understanding of sorting algorithms to your problem solving in Python

# Topic Outline



# Recall: Application of Looping

## Find the Distance to N Locations

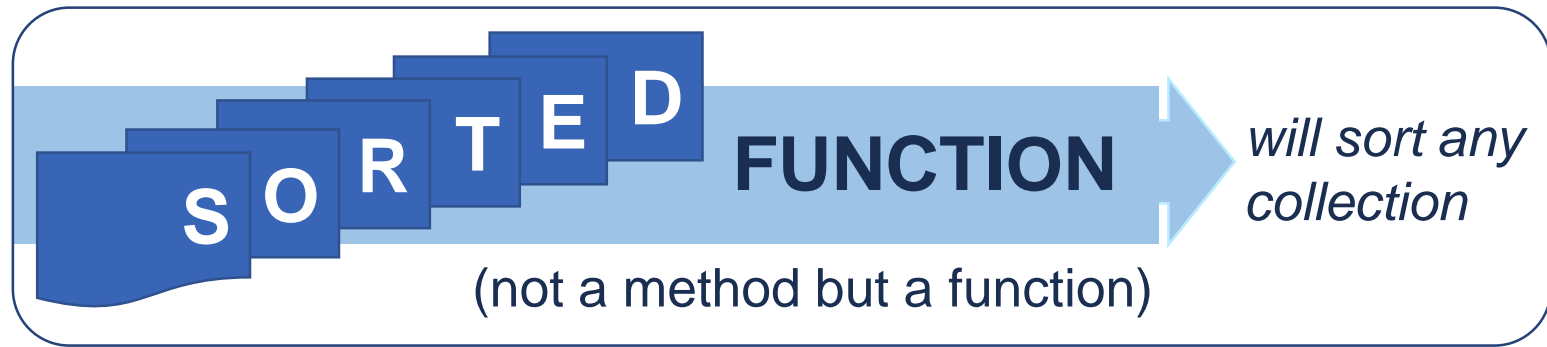


Coffeeshop	Distance
Coffeeshop A	3
Coffeeshop B	4
Coffeeshop C	7
Coffeeshop D	3
Coffeeshop E	6
Coffeeshop F	8
Coffeeshop G	5
Coffeeshop H	3

**Sort by  
distance**



Coffeeshop	Distance
Coffeeshop A	3
Coffeeshop D	3
Coffeeshop H	3
Coffeeshop B	4
Coffeeshop G	5
Coffeeshop E	6
Coffeeshop C	7
Coffeeshop F	8



Separates the collection into individual elements

- Sorts those elements


Returns the elements in a sorted list

- Returns a new sorted list without changing the original list



# Sort by Distance

Python provides a module named `operator` that contains the useful `itemgetter` for sorting values other than the first and allows sorting on multiple values.




```
from operator import itemgetter
restaurant_info = [['Kentucky', 15, 6, 'Fried chicken'],
                   ['Macdonald', 12, 5, 'Burger'],
                   ['Subway', 13, 7, 'Sandwiches']]

# field 1 (index 0): name of restaurant
# field 2 (index 1): distance of restaurant
# field 3 (index 2): average price per person of the restaurant
# field 4 (index 3): signature dish of the restaurant

sort_info = sorted(restaurant_info, key = itemgetter(1))
print('sort by distance', sort_info)
```

# Sort by Any Field



```
from operator import itemgetter
restaurant_info = [['Kentucky', 15, 6, 'Fried chicken'],
                   ['Macdonald', 12, 5, 'Burger'],
                   ['Subway', 13, 7, 'Sandwiches']]

# field 1 (index 0): name of restaurant
# field 2 (index 1): distance of restaurant
# field 3 (index 2): average price per person of the restaurant
# field 4 (index 3): signature dish of the restaurant
sort_field = ['name', 'distance', 'price', 'food']

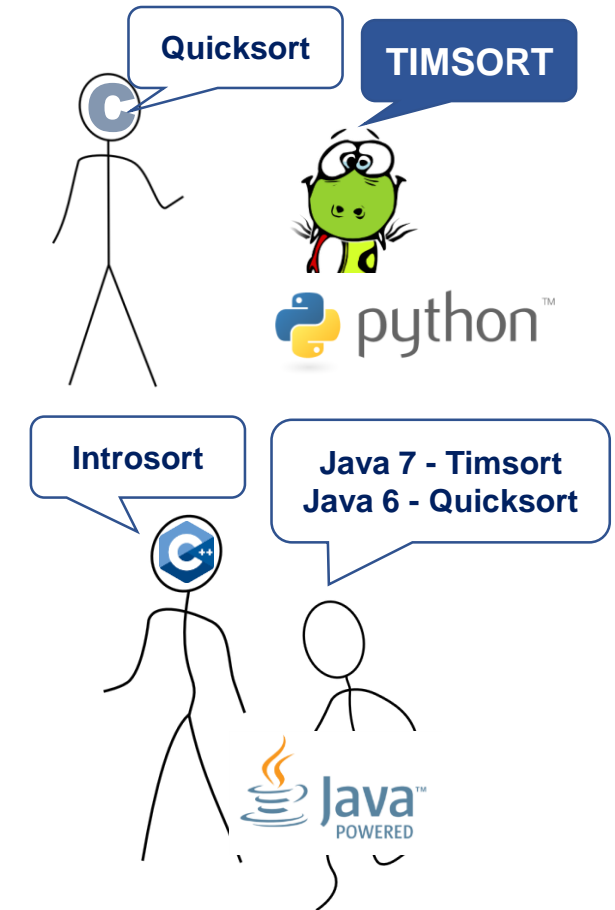
for i in range (len(sort_field)):
    sort_info = sorted (restaurant_info, key = itemgetter(i))
    print('sort by', sort_field[i], sort_info)
```



# What Algorithm does Python's Sorted() Use?

## **TIMSORT** Hybrid sorting algorithm used by Python

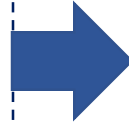
- Derived from merge sort and insertion sort and designed to perform well on many kinds of real-world data
- Invented by Tim Peters in 2002 for use in the Python programming language
- Finds subsets of the data that are already ordered, and uses the subsets to sort the data more efficiently. This is done by merging an identified subset, called a run, with existing runs until certain criteria are fulfilled
- Timsort has been Python's standard sorting algorithm since version 2.3
- Now, also used to sort arrays in Java SE 7 and on the Android platform



# Importance of Basic Algorithm Coding



**Why is there a need  
to learn how to code  
basic sort algorithm?**



**Training for the Future**

# Bubble Sort Algorithm

**Step 1:** Get the length of the sequence

**Step 2:** Take the first element and compare it with the immediate neighbor to the right:  $a_i > a_{i+1}$

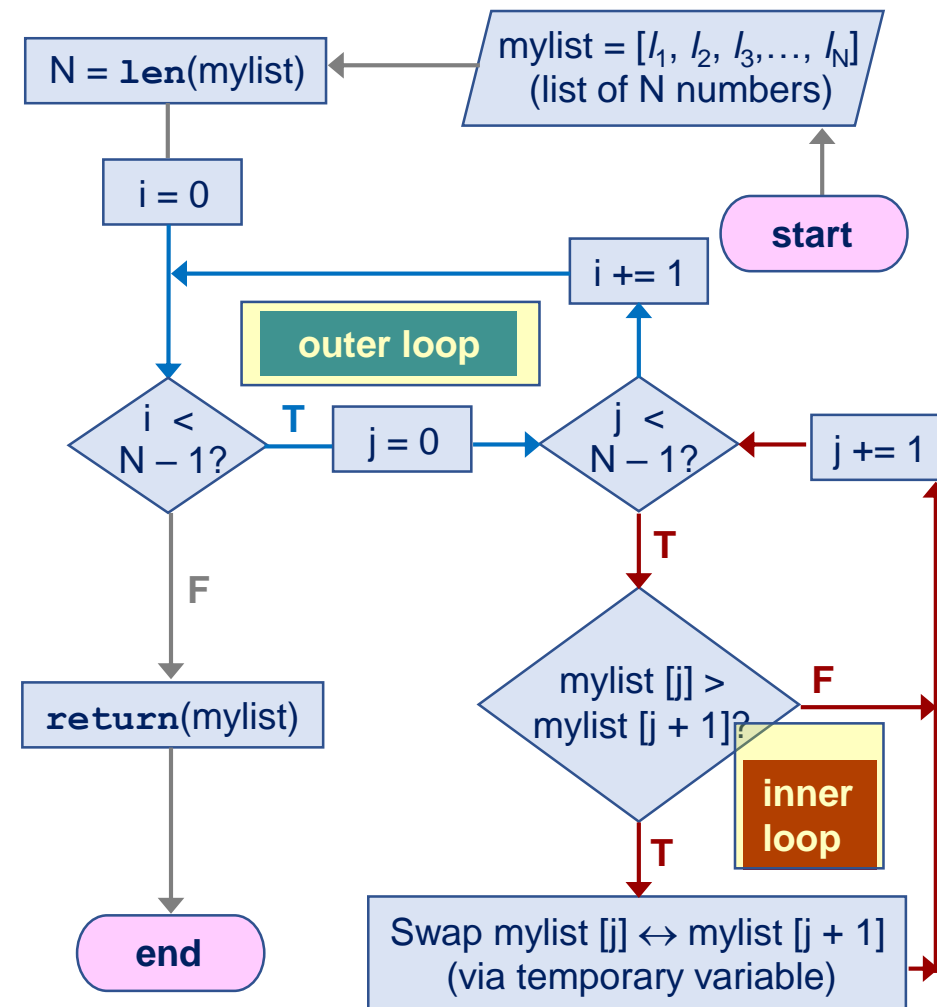
- If true: swap and increment  $i$  by one
- If false: increment  $i$  by one

**Step 3:** Repeat step 2,  $N - 1$  times

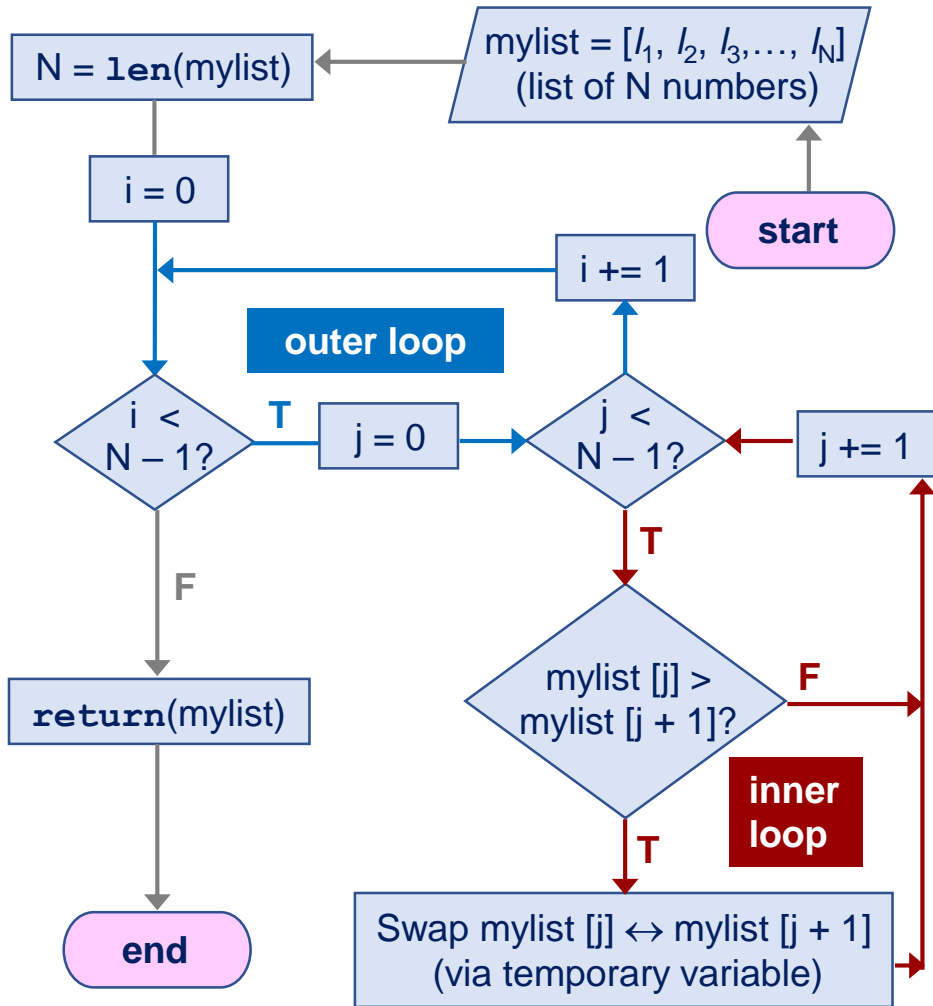
## Pseudocode

```

while k < N
    for i = 0 to N - 1
        if a[i] > a[i + 1]
            then
                swap()
        end
    k = k + 1
end
    
```



# Suggested Code



```
def bubbleSort(alist):
    for passnum in range(len(alist)-1):
        for i in range(len(alist)-1):
            if alist[i]>alist[i+1]:
                temp = alist[i]
                alist[i] = alist[i+1]
                alist[i+1] = temp
```

```
alist = [6,5,4,3]
bubbleSort(alist)
print(alist)
```



- Is it efficient?
- Any possible improvement?

# Bubble Sort: Second Pass - End

- At the end of the first pass, there are  $n - 1$  items left to sort.
- At the end of the second pass, there are  $n - 2$  items left to sort.
- Number of items to be sorted increases while the number of pass increases.



93 in place after  
the first pass



**Second Pass**



# Improved Code of Bubble Sort

```
def bubbleSort(alist):
    for passnum in range(len(alist)-1):
        for i in range(len(alist)-passnum-1):
            if alist[i]>alist[i+1]:
                temp = alist[i]
                alist[i] = alist[i+1]
                alist[i+1] = temp

alist = [54,26,93,17,77,31,44,55,20]
bubbleSort(alist)
print(alist)
```



- The inner loop executes  **$n - 1$**  times at first, linearly dropping to just once
- On average, inner loop executes about  **$n/2$**  times for each execution of the outer loop

 **Any possible improvement?**



**Recall**

Since each pass places the next largest value in place, the total number of passes necessary will be  $n - 1$ .





What is the output of the following Python code?

```
def bubbleSort(alist):  
    for passnum in range(len(alist)-1):  
        for i in range(len(alist)-passnum-1):  
            if alist[i]>alist[i+1]:  
                temp = alist[i]  
                alist[i] = alist[i+1]  
                alist[i+1] = temp  
        print("pass",passnum+1, ":",alist)  
  
alist = [1,2,3,4,5,6]  
bubbleSort(alist)  
print(alist)
```



What is the output of the following Python code?

```
def bubbleSort(alist):  
    for passnum in range(len(alist)-1):  
        for i in range(len(alist)-passnum-1):  
            if alist[i]>alist[i+1]:  
                temp = alist[i]  
                alist[i] = alist[i+1]  
                alist[i+1] = temp  
        print("pass",passnum+1, ":",alist)  
  
alist = [1,2,3,4,5,6]  
bubbleSort(alist)  
print(alist)
```

Answer

```
pass 1 : [1, 2, 3, 4, 5, 6]  
pass 2 : [1, 2, 3, 4, 5, 6]  
pass 3 : [1, 2, 3, 4, 5, 6]  
pass 4 : [1, 2, 3, 4, 5, 6]  
pass 5 : [1, 2, 3, 4, 5, 6]  
[1, 2, 3, 4, 5, 6]
```

# Break: Early Exit to be More Efficient

```
procedure bubbleSort( list : array of items )
  loop = list.count;
  for i = 0 to loop-1 do:
    swapped = false
    for j = 0 to loop-i-1 do:
      /* compare the adjacent elements */
      if list[j] > list[j+1] then
        /* swap them */
        swap( list[j], list[j+1] )
        swapped = true
      end if
    end for
    /*if no number was swapped that means array is sorted now, break the loop.*/
    if(not swapped) then
      break
    end if
  end for
end procedure return list
```

- How can we know if the list is sorted?
- The pass go through the list but no swaps are needed, which indicates that the list is sorted.



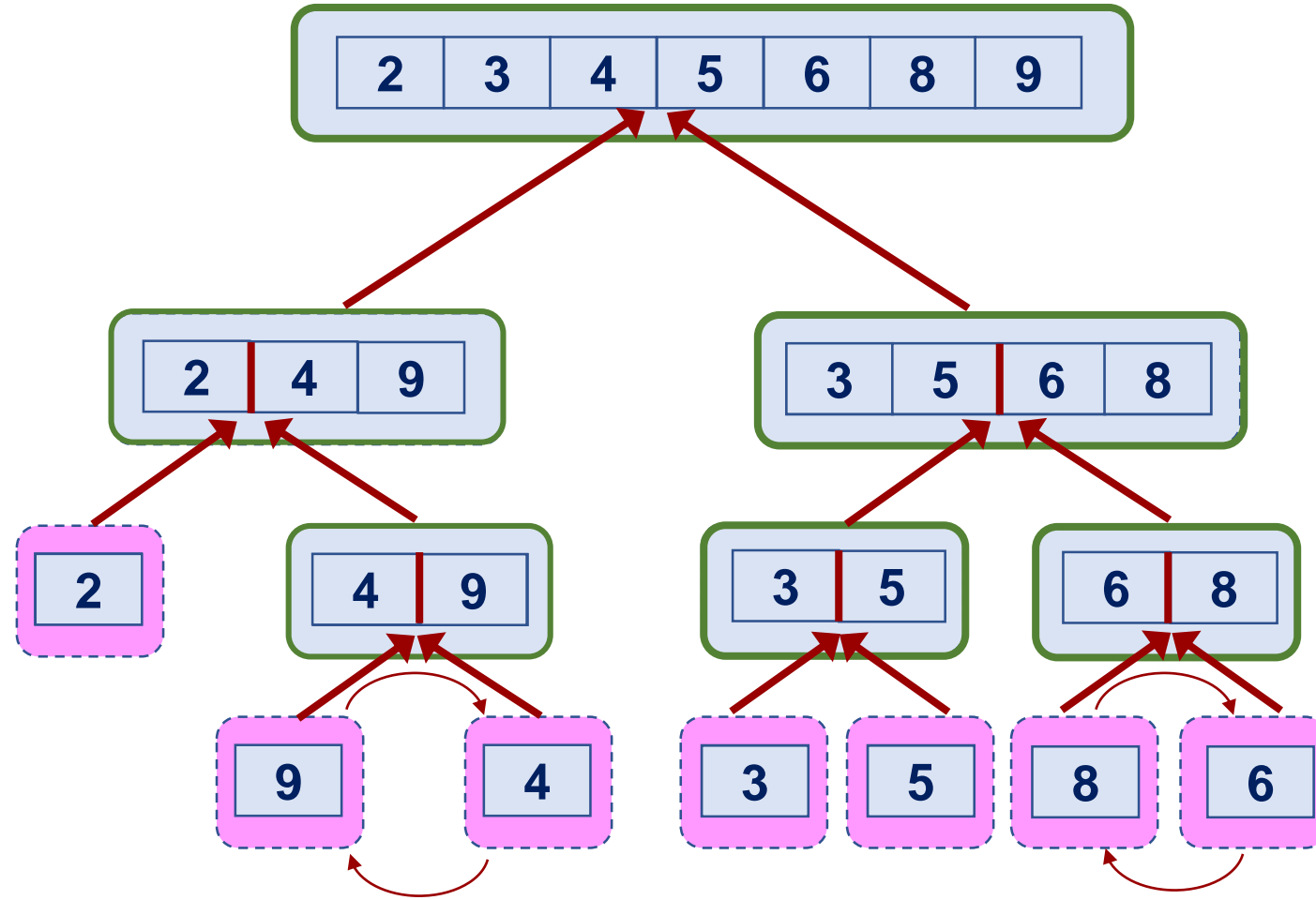
# Improved Code of Bubble Sort: Run

```
def bubbleSort(alist):  
    for passnum in range(len(alist)-1):  
        swapped = False  
        for i in range(len(alist)-1):  
            if alist[i]>alist[i+1]:  
                temp = alist[i]  
                alist[i] = alist[i+1]  
                alist[i+1] = temp  
                swapped = True  
        print("pass",passnum+1, ":",alist)  
        if not swapped:  
            break;  
  
alist = [1,2,3,4,5,6]  
bubbleSort(alist)  
print(alist)
```



pass 1 : [1, 2, 3, 4, 5, 6]  
[1, 2, 3, 4, 5, 6]


# Recall: Merge Sort - Execution Example



**Step 4:** Merge the two sorted subsequences to form the output sequence.



# Merge Sort Function



```
def mergesort(list_of_items):  
    list_len = len(list_of_items)  
  
    # base case  
    if list_len < 2:  
        return list_of_items  
  
    left_list = list_of_items[:list_len // 2]    # //  
    right_list = list_of_items[list_len // 2:]    # "//" to force division  
  
    # merge sort left and right list recursively  
    left_list = mergesort(left_list)  
    right_list = mergesort(right_list)  
    return merge(left_list, right_list)
```

# Merge Adjacent List

This means we have a list on the left and a list on the right.

## Step 1

Compare the first elements of both lists one by one.

## Step 2

Add this value to the list of “sorted items”. Move the smaller element out of the list that it was found in.

## Step 3

Repeat the process until only a single list remains.

## Step 4

One list should still contain elements. This list is sorted. Move its contents into the result list.

## Suggested code

```
while left_list and right_list:
    if left_list[0] < right_list[0]:
        result_list.append(left_list[0])
        left_list.pop(0)
    else:
        result_list.append(right_list[0])
        right_list.pop(0)

if left_list:
    result_list.extend(left_list)
else:
    result_list.extend(right_list)
```

# Merge Function

```
def merge(left_list, right_list):  
  
    result_list = []  
  
    # while left and right list has elements  
    while left_list and right_list:  
        if left_list[0] < right_list[0]:  
            result_list.append(left_list[0])  
            left_list.pop(0)  
        else:  
            result_list.append(right_list[0])  
            right_list.pop(0)  
  
    #left list still contain elements. Append its contents to end of the result  
    if left_list:  
        result_list.extend(left_list)  
    else:  
        #right list still contain elements. Append its contents to end of the result  
        result_list.extend(right_list)
```



# Application

```
list_test = [21, 1, 26, 45, 3, 3, 4, 27, 43, 34, 46, 40]  
new_list = mergesort(list_test)  
for item in new_list:  
    print(item, end = ', ')
```



**1, 3, 3, 4, 21, 26, 27, 34, 40, 43, 45, 46,**



## Bubble Sort Algorithm

### Pseudocode

```
while k < N
    for i = 0 to N = 1
        if a[i] > a[i + 1]
            then
                swap()
        end
    k = k + 1
end
```






## SORTING in PYTHON

## Merge Sort Algorithm

### Pseudocode




```
MergeSort(list, first, last)
if (first < last)
    middle = (first + last) div 2
    MergeSort(list, first, middle)
    MergeSort(list, middle+1, last)
    Merge(list, first, middle, last)
end if
```

# References for Images

No.	Slide No.	Image	Reference
1	All pages with Python codes		Python Logo [Online Image]. Retrieved April 18, 2018 from <a href="https://pixabay.com/en/language-logo-python-2024210/">https://pixabay.com/en/language-logo-python-2024210/</a> .
2	9		Search [Online Image]. July 9, 2018 from <a href="https://pixabay.com/en/snake-python-serpent-green-reptile-312561/">https://pixabay.com/en/snake-python-serpent-green-reptile-312561/</a> .
3	9		Python logo [Online Image]. Retrieved June 20, 2018 from <a href="https://www.python.org/static/community_logos/python-logo-master-v3-TM.png">https://www.python.org/static/community_logos/python-logo-master-v3-TM.png</a> .
4	9		Java log [Online Image]. Retrieved June 20, 2018 from <a href="http://www.oracle.com/us/technologies/java/java-licensing-logo-guidelines-1908204.pdf">http://www.oracle.com/us/technologies/java/java-licensing-logo-guidelines-1908204.pdf</a> .
5	9		By Jeremy Kratz - <a href="https://github.com/isocpp/logos">https://github.com/isocpp/logos</a> , Copyrighted free use, retrieved June 20, 2018 from <a href="https://commons.wikimedia.org/w/index.php?curid=62851110">https://commons.wikimedia.org/w/index.php?curid=62851110</a> .



# References for Images

No.	Slide No.	Image	Reference
6	10		By US Air Force - <a href="https://media.defense.gov/2010/May/20/2000360693/-1/-1/0/100520-F-9999B-111.JPG">https://media.defense.gov/2010/May/20/2000360693/-1/-1/0/100520-F-9999B-111.JPG</a> , Public Domain, retrieved July 9, 2018 from <a href="https://commons.wikimedia.org/w/index.php?curid=10404400">https://commons.wikimedia.org/w/index.php?curid=10404400</a> .
7	10, 12, 14		Question problem [Online Image]. Retrieved April 18, 2018 from <a href="https://pixabay.com/en/question-problem-think-thinking-622164/">https://pixabay.com/en/question-problem-think-thinking-622164/</a> .
8	14		Search [Online Image]. Retrieved April 18, 2018 from <a href="https://pixabay.com/en/database-search-database-search-icon-2797375/">https://pixabay.com/en/database-search-database-search-icon-2797375/</a> .