Algorithm

Merge Sort

Code Walkthrough

Part 1

Merge Sort

```
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

Merge Sort

```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

```
merge_sort([5, 1, 4, 7, 3])
```

```
[5, 1, 4, 7, 3]
```

```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

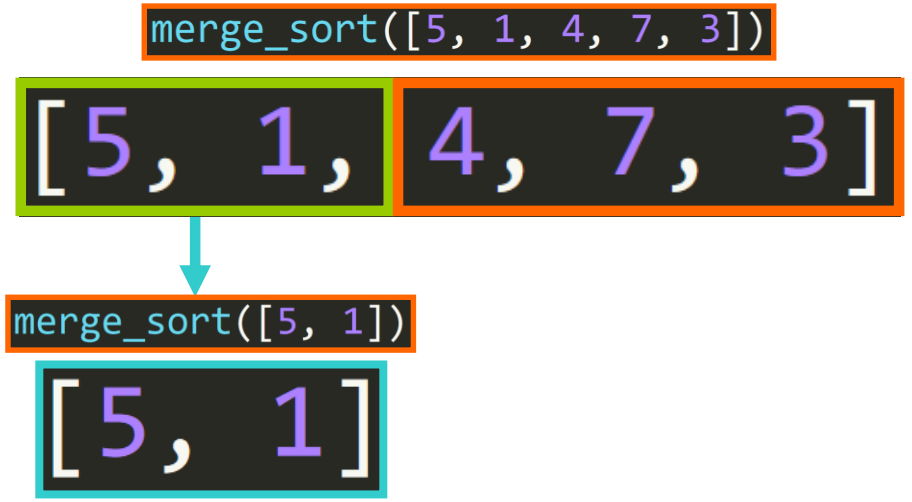
```
merge_sort([5, 1, 4, 7, 3])
```

```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

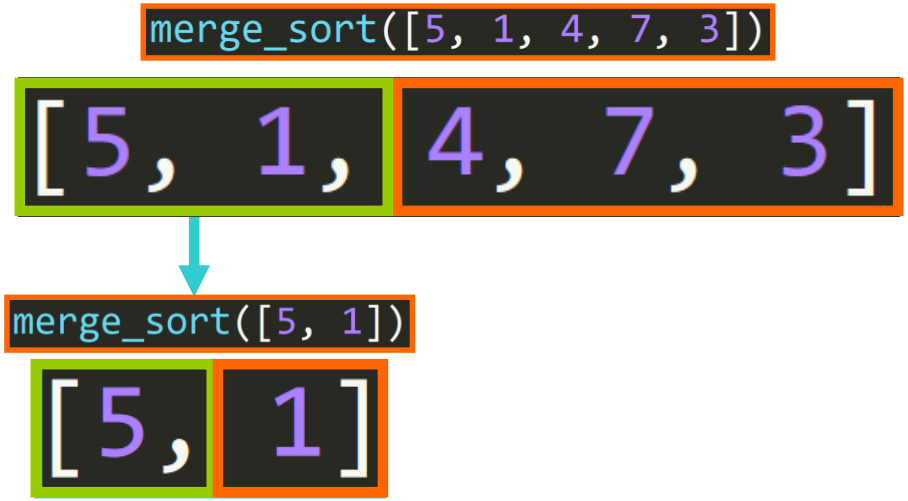


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

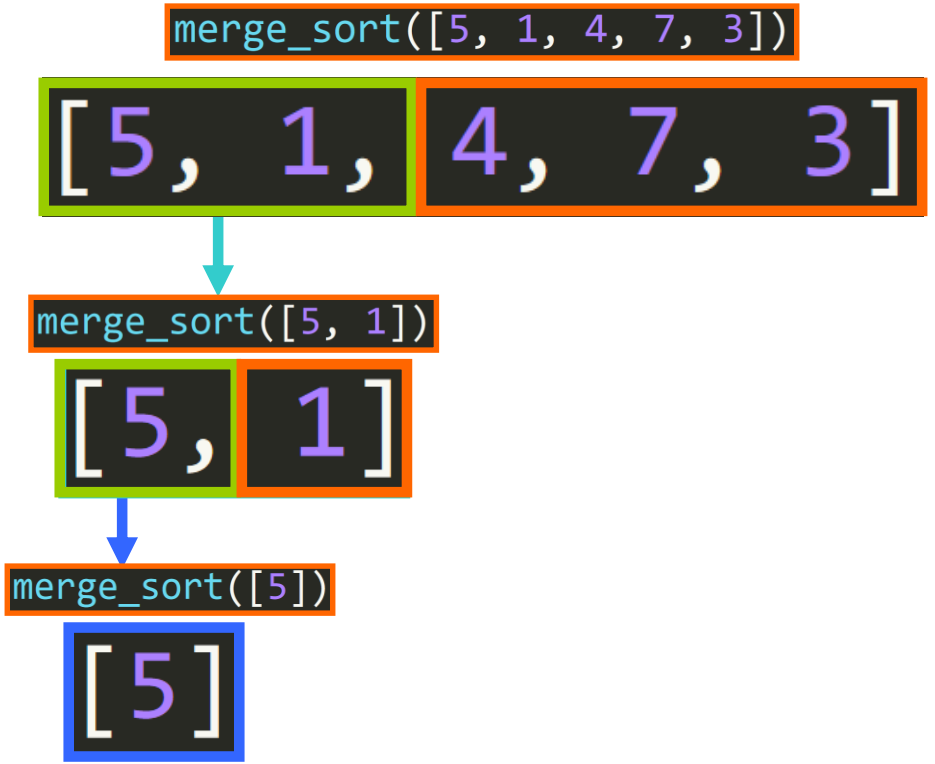


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

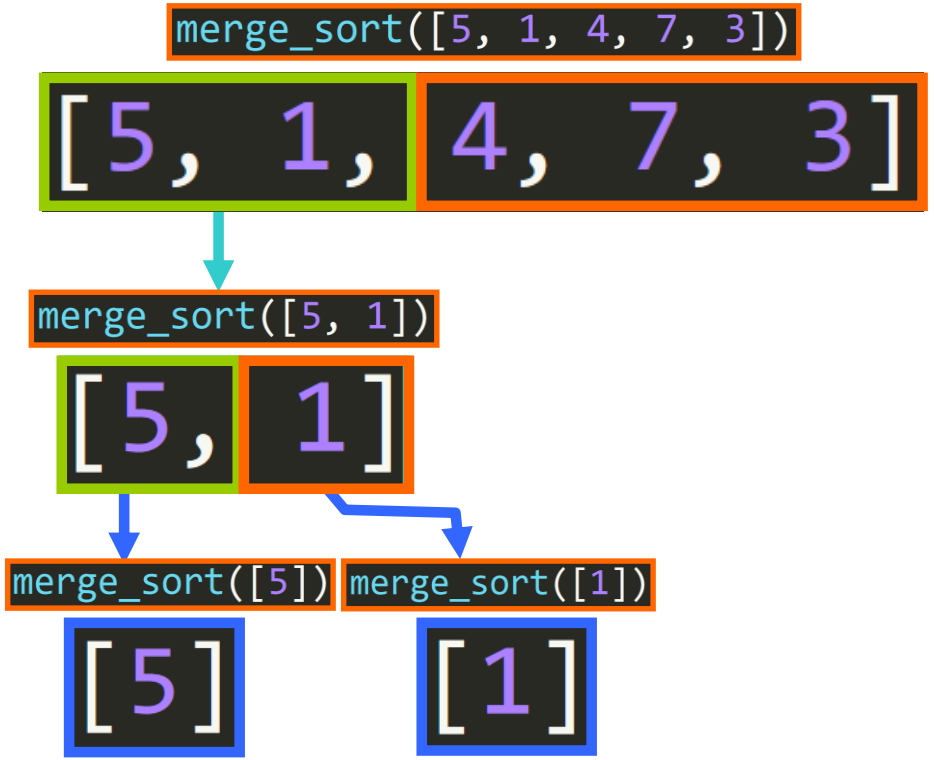


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

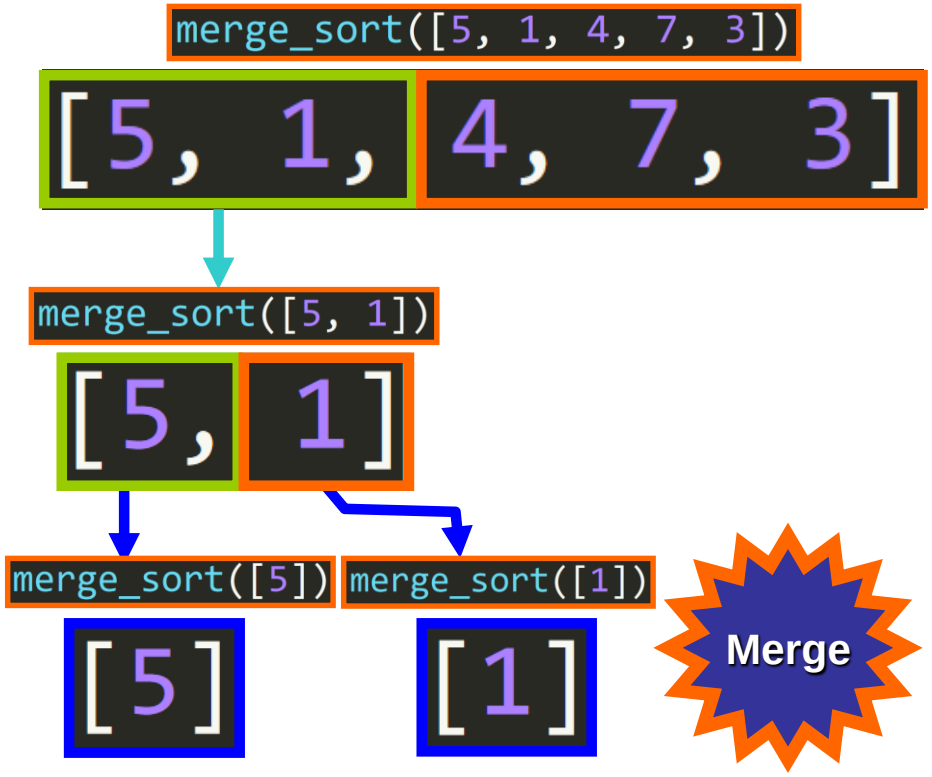


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

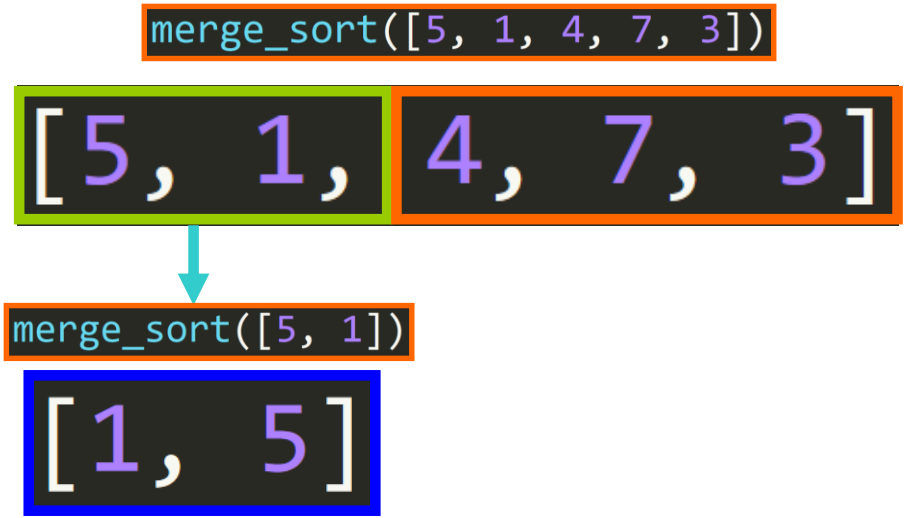


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

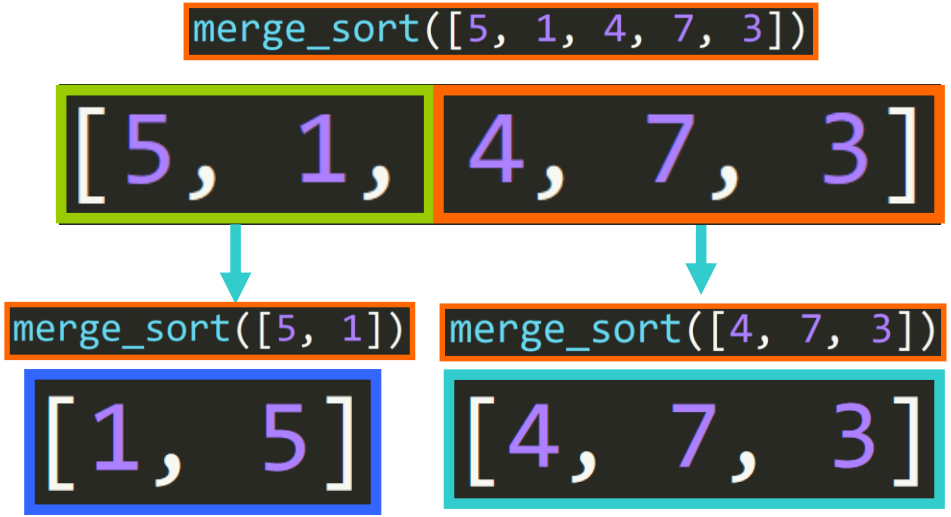


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

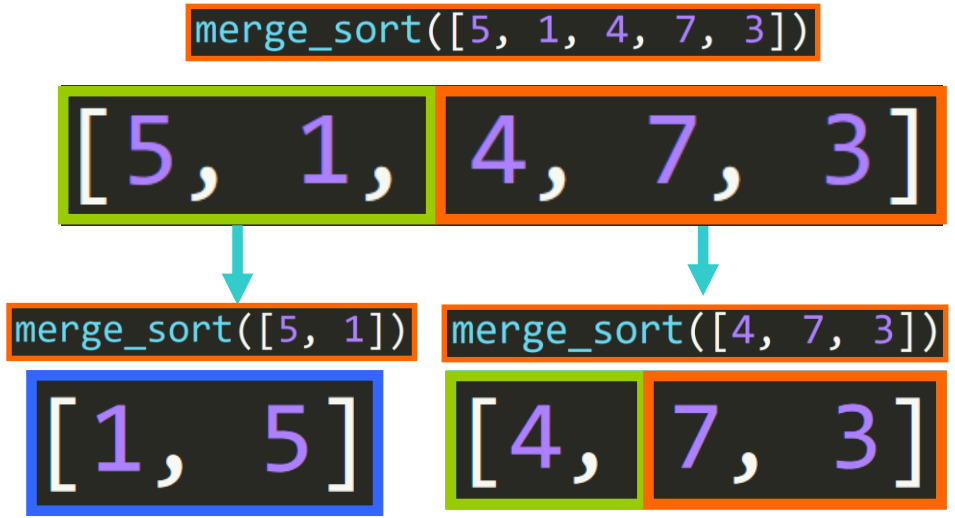


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

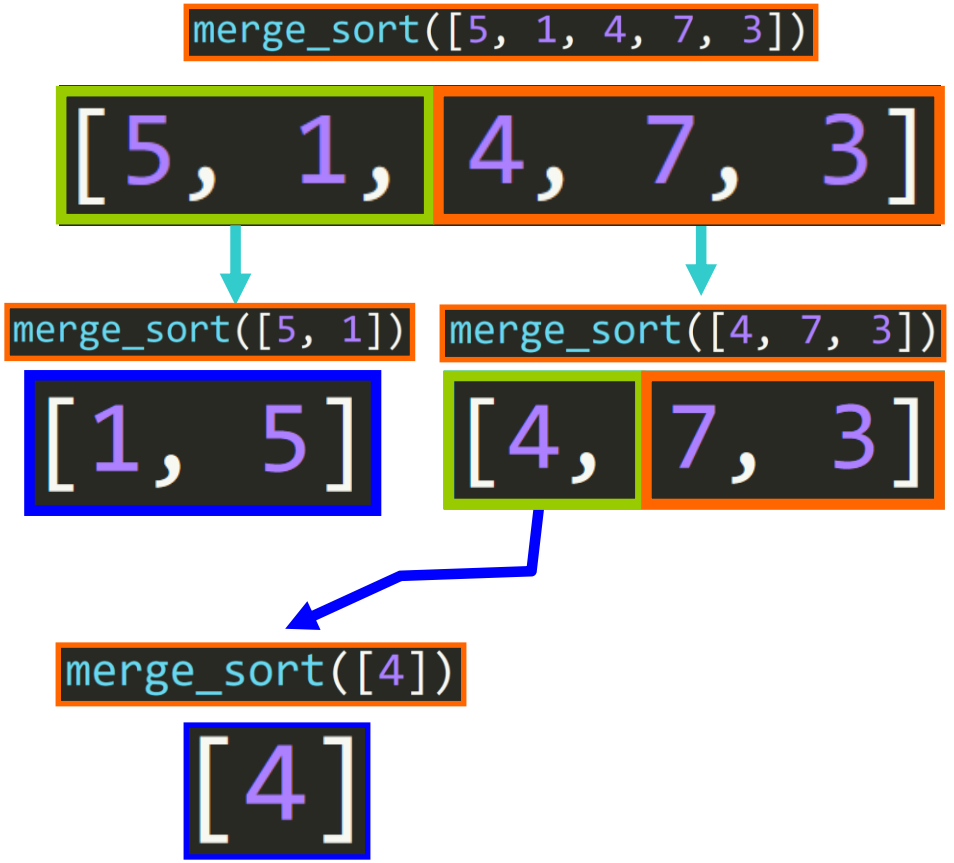


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

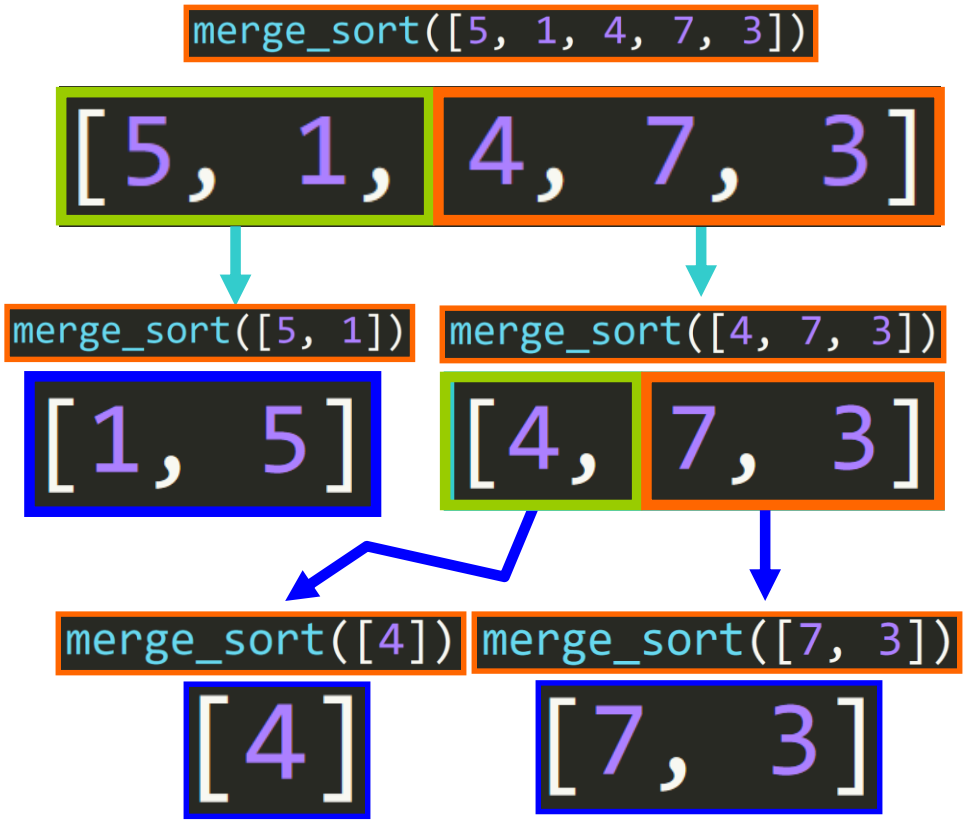


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

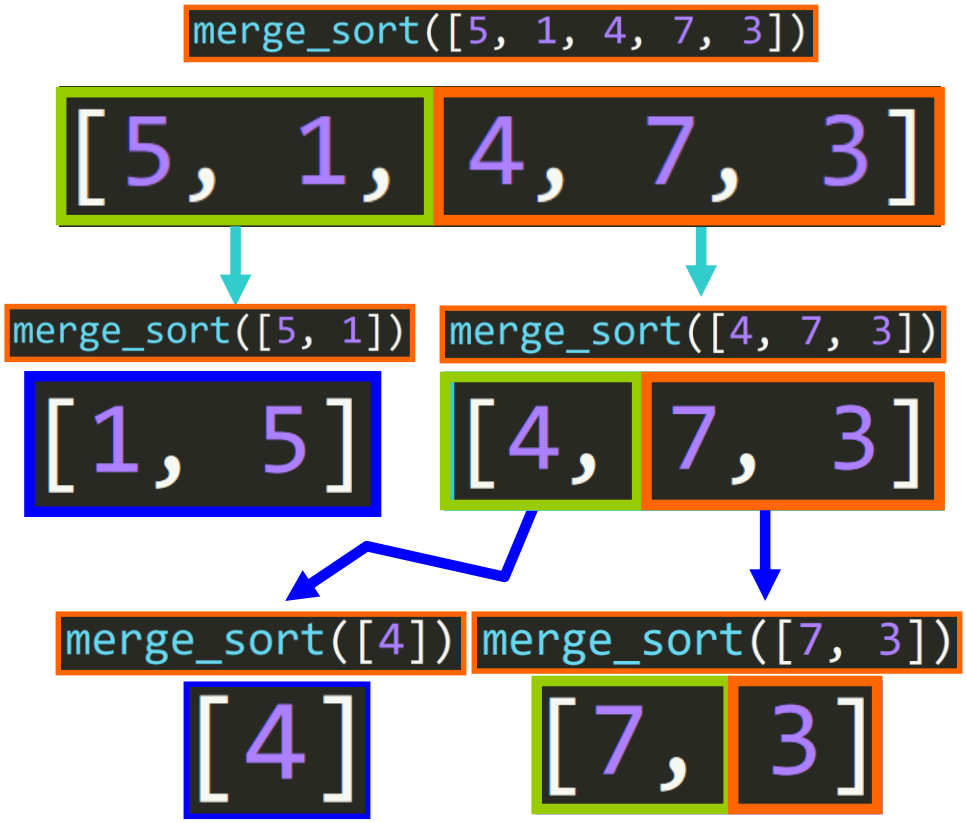


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

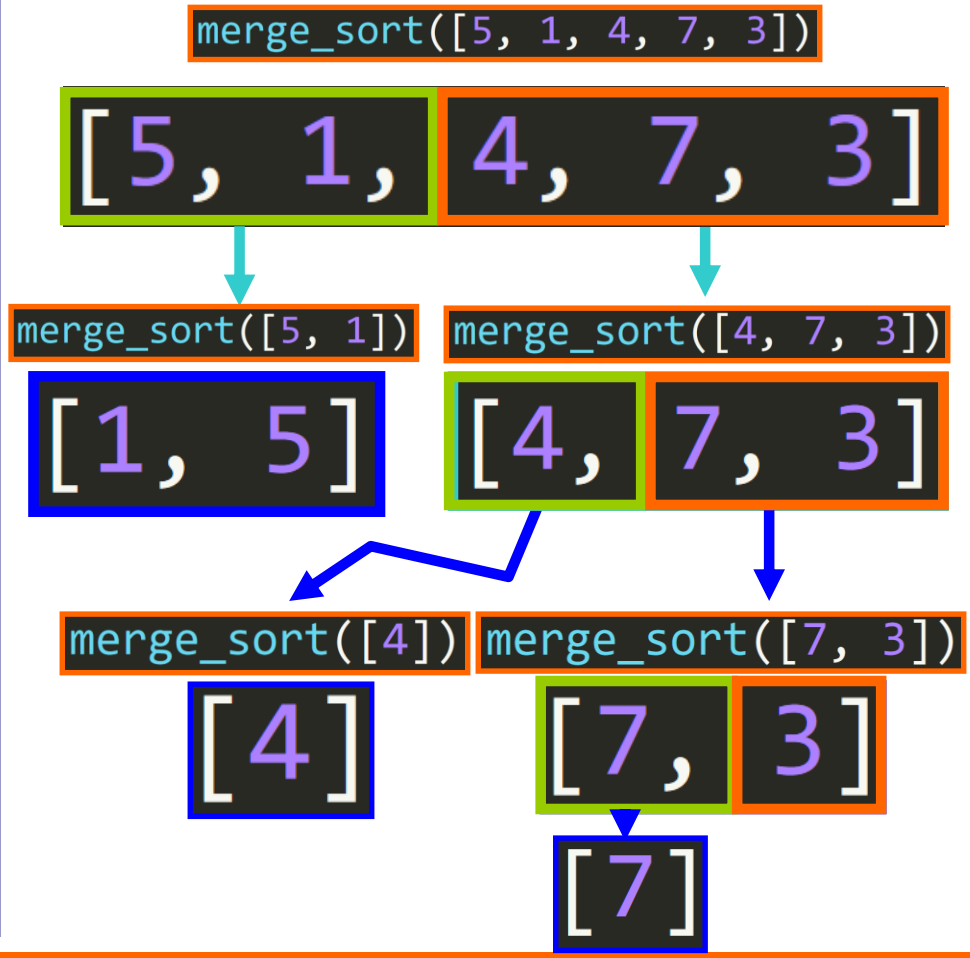


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

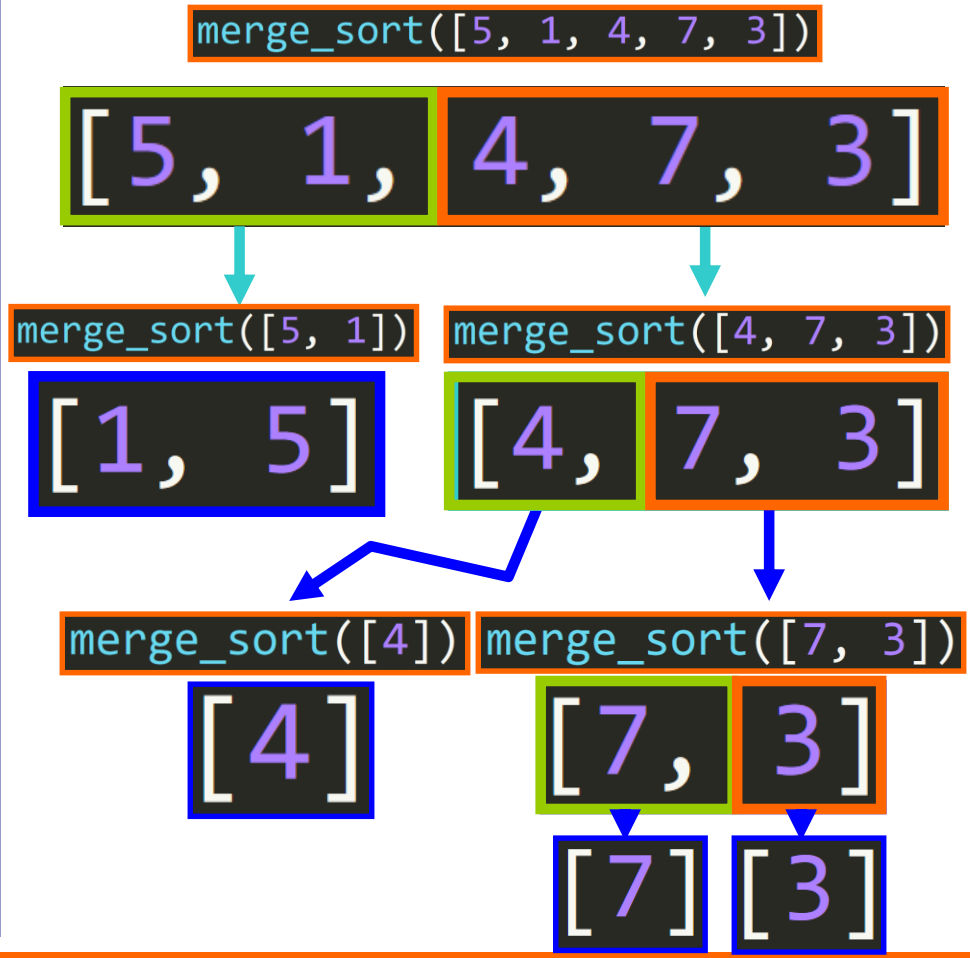


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

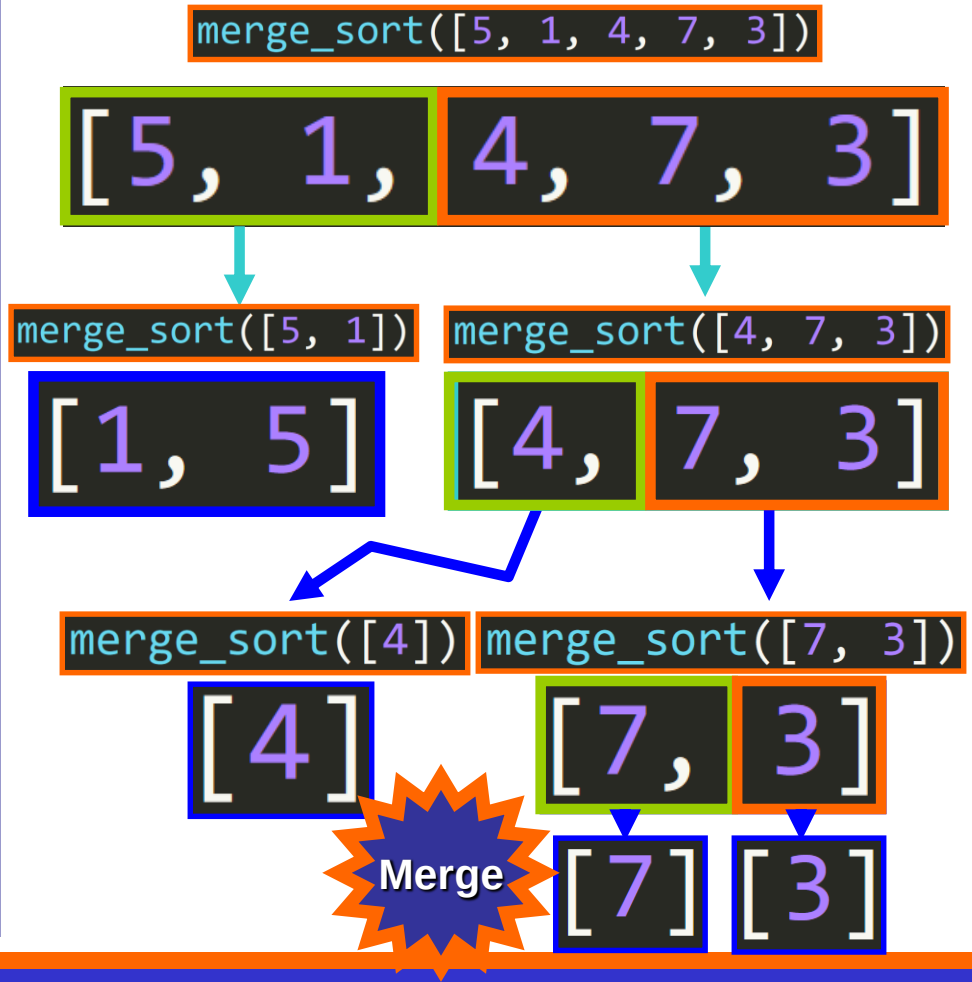


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

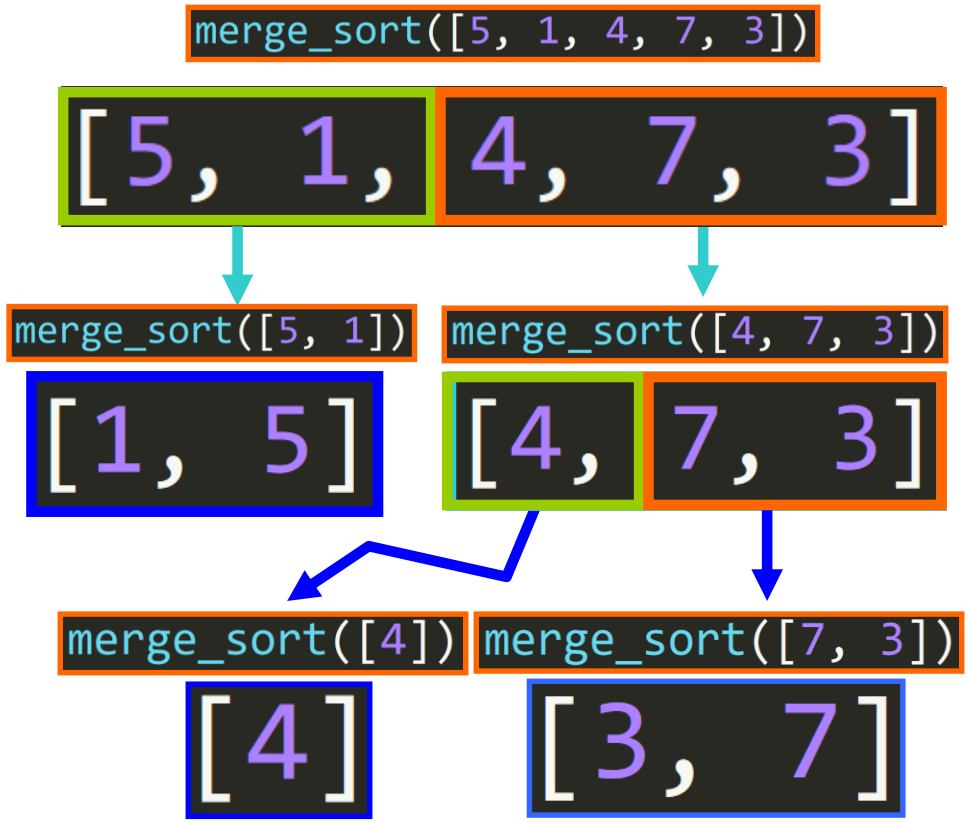


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

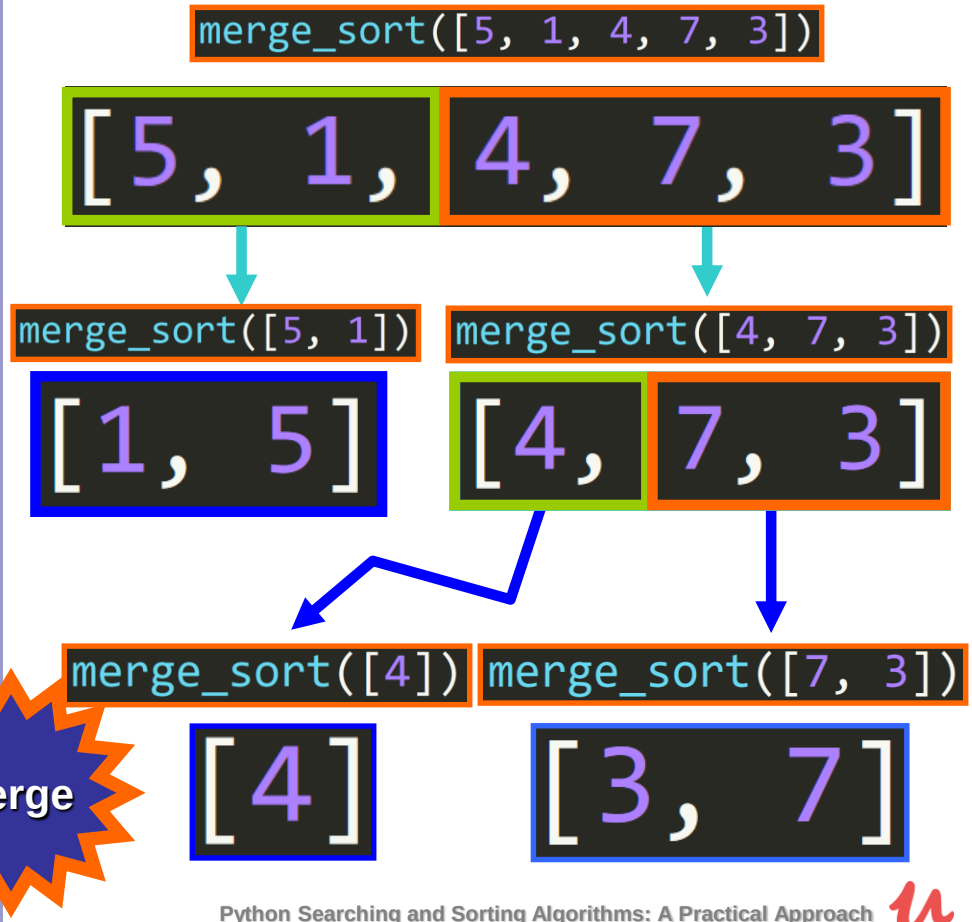


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

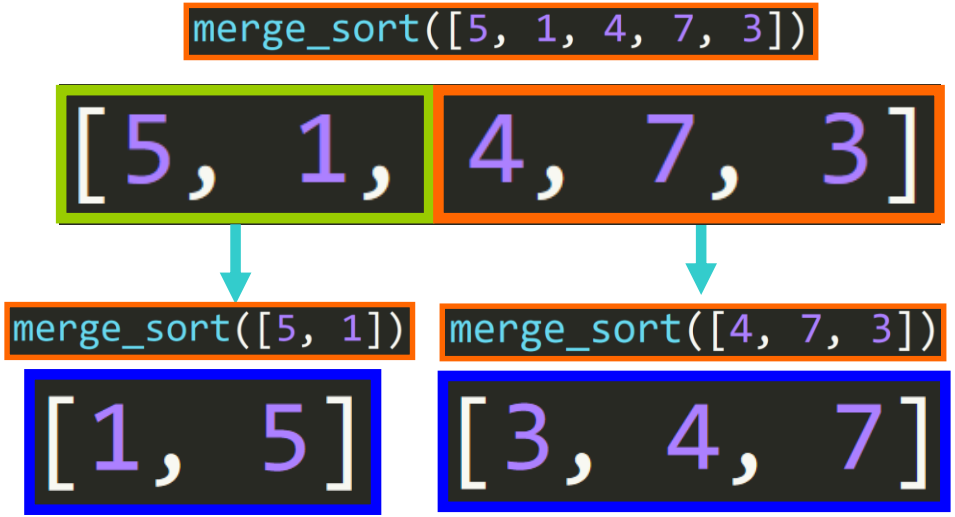


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```

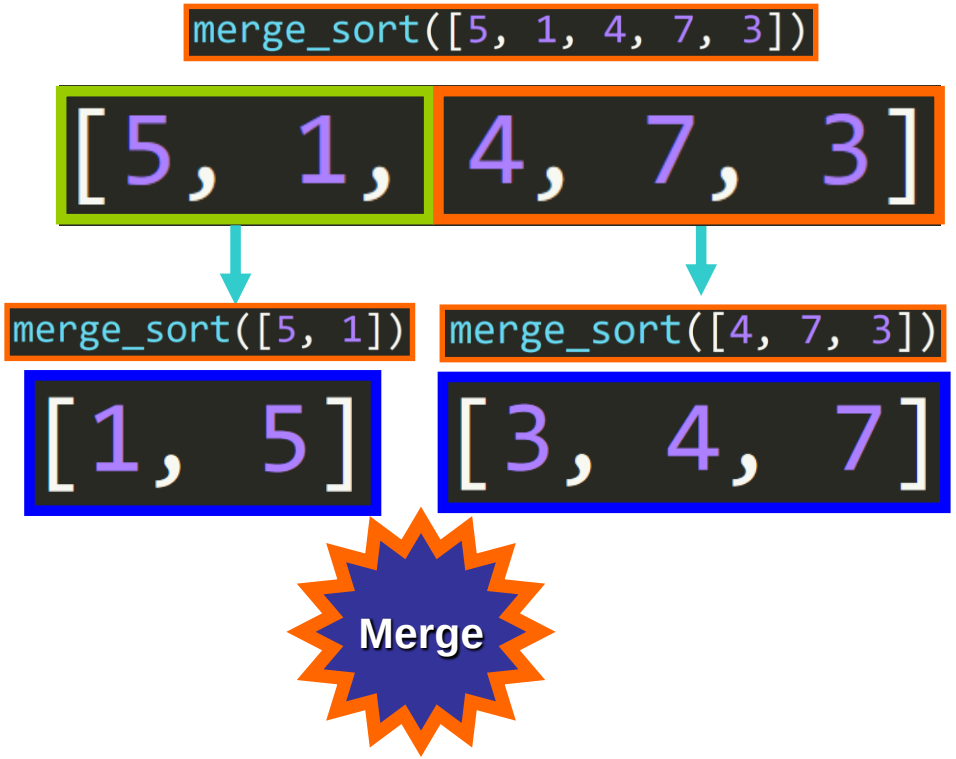


```python
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```



```
def merge_sort(lst):

    if len(lst) == 0 or len(lst) == 1:
        return lst
    else:
        middle_index = len(lst)//2

        left = merge_sort(lst[:middle_index])
        right = merge_sort(lst[middle_index:])

        return merge(left, right)
```


```
[1, 3, 4, 5, 7]
```

To merge()