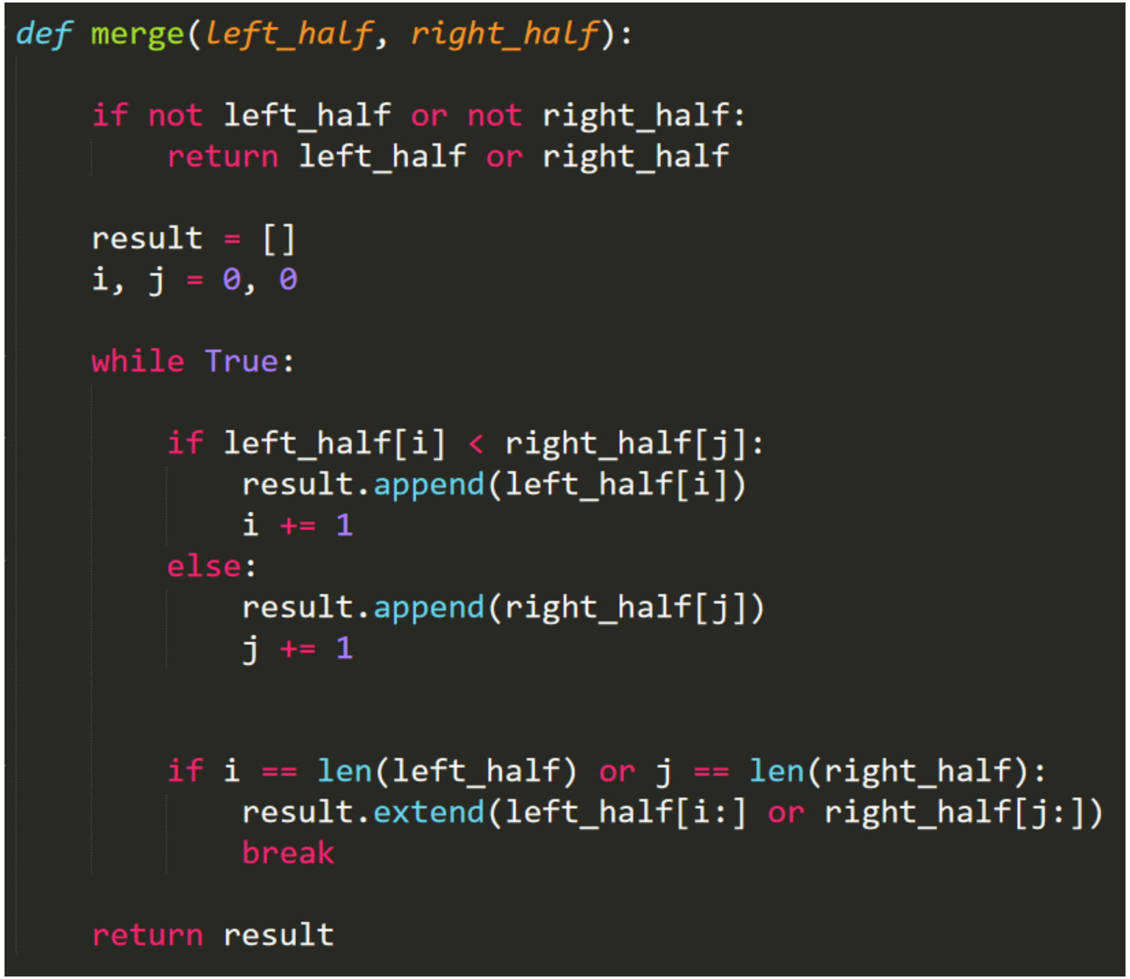
Algorithm

Merge Sort

Code Walkthrough

Part 2

Merge Sort



```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

Merge Sort

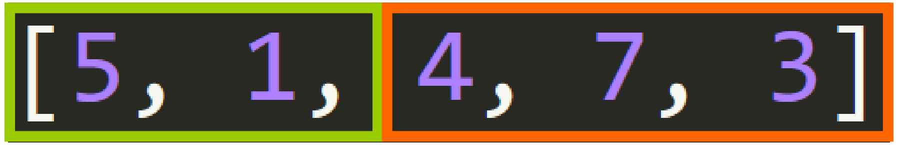
```python
def merge_sort(lst):

    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])
```

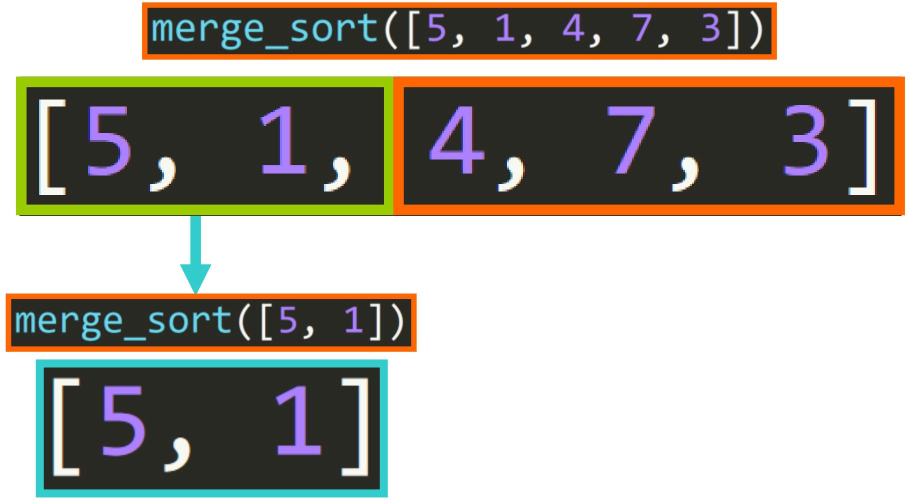
```
[5, 1, 4, 7, 3]
```

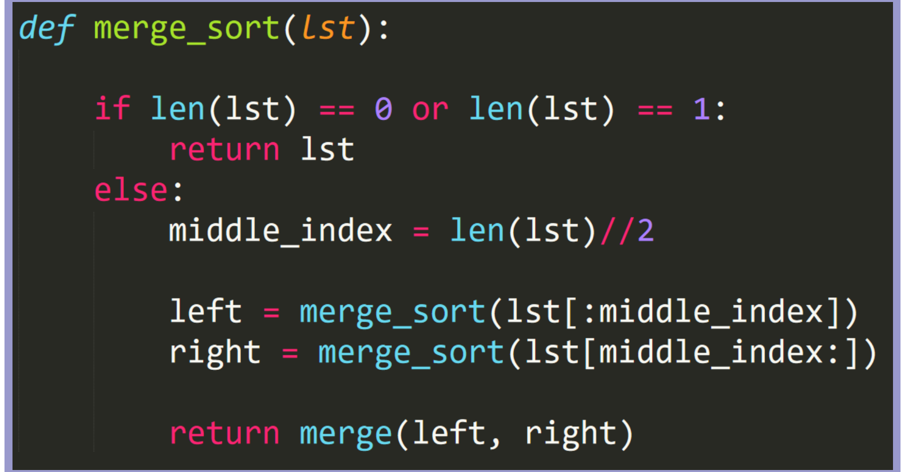
```python
def merge_sort(lst):

    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)
```

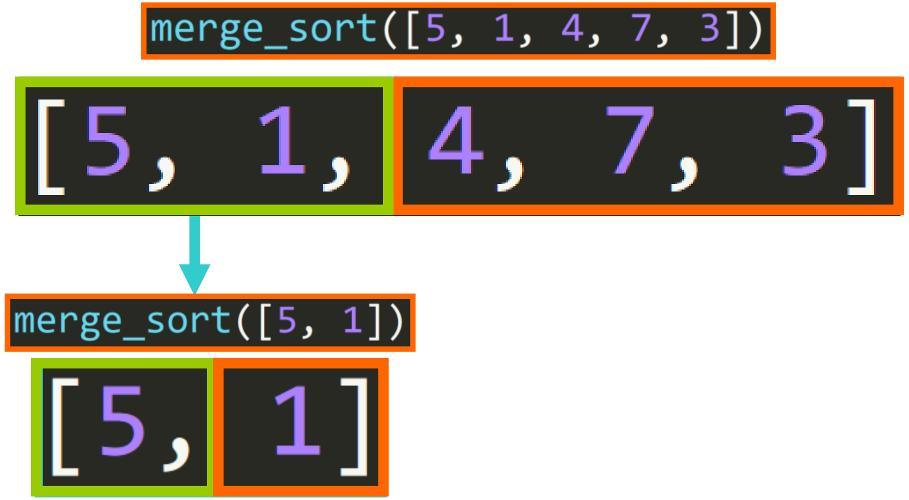
```python
merge_sort([5, 1, 4, 7, 3])
```



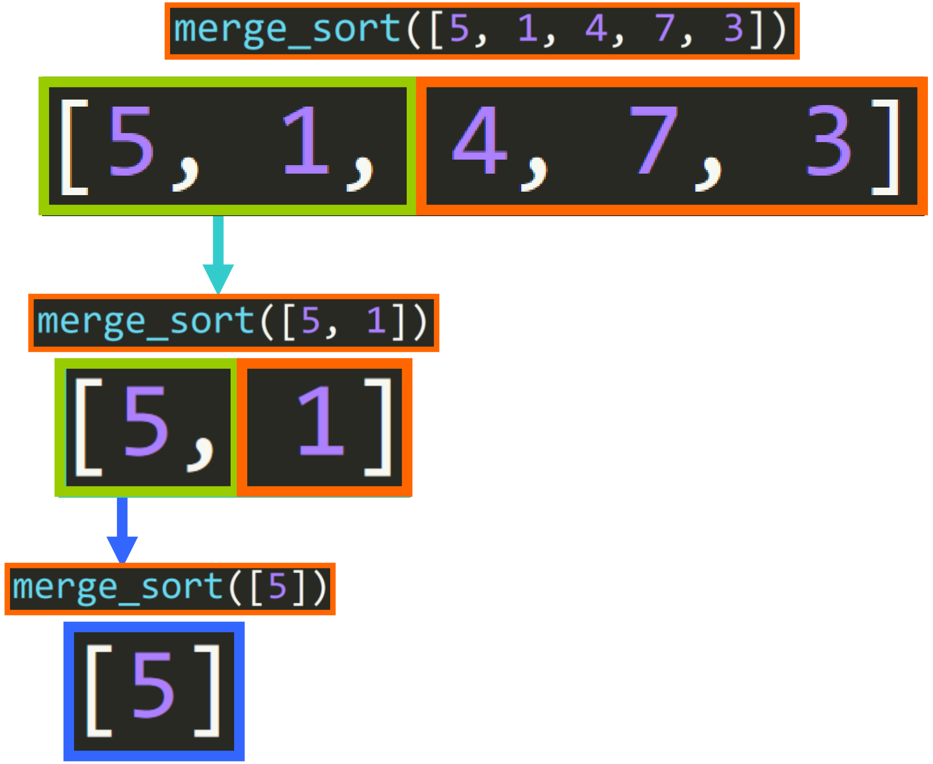
```python
def merge_sort(lst):

    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)
```



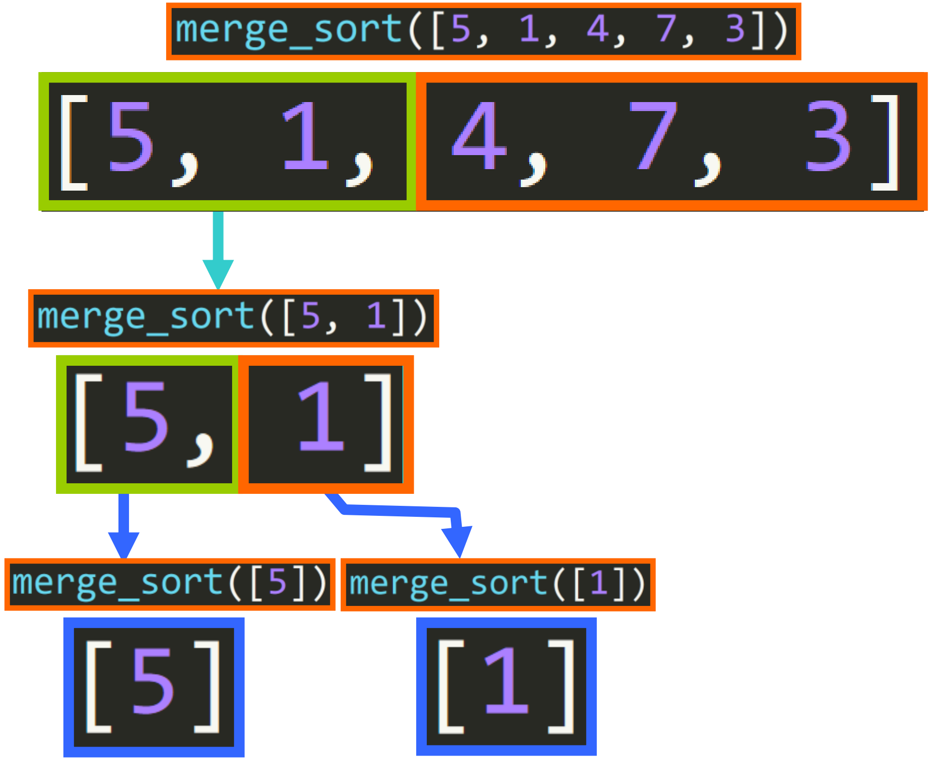

```python
def merge_sort(lst):

    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)
```



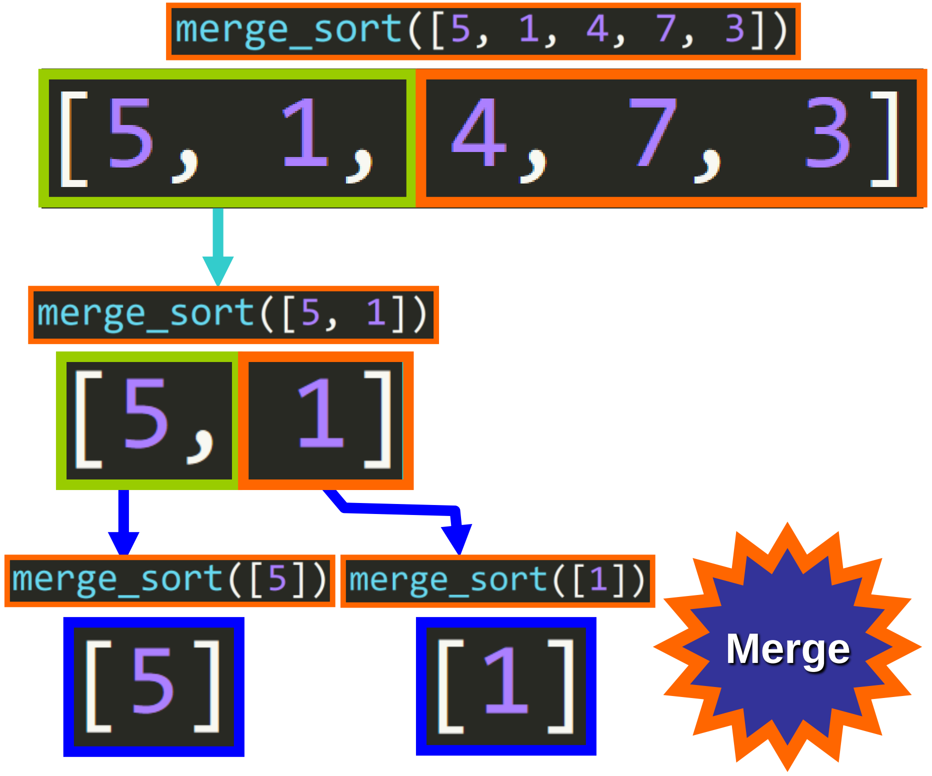
```python
def merge_sort(lst):

    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)
```

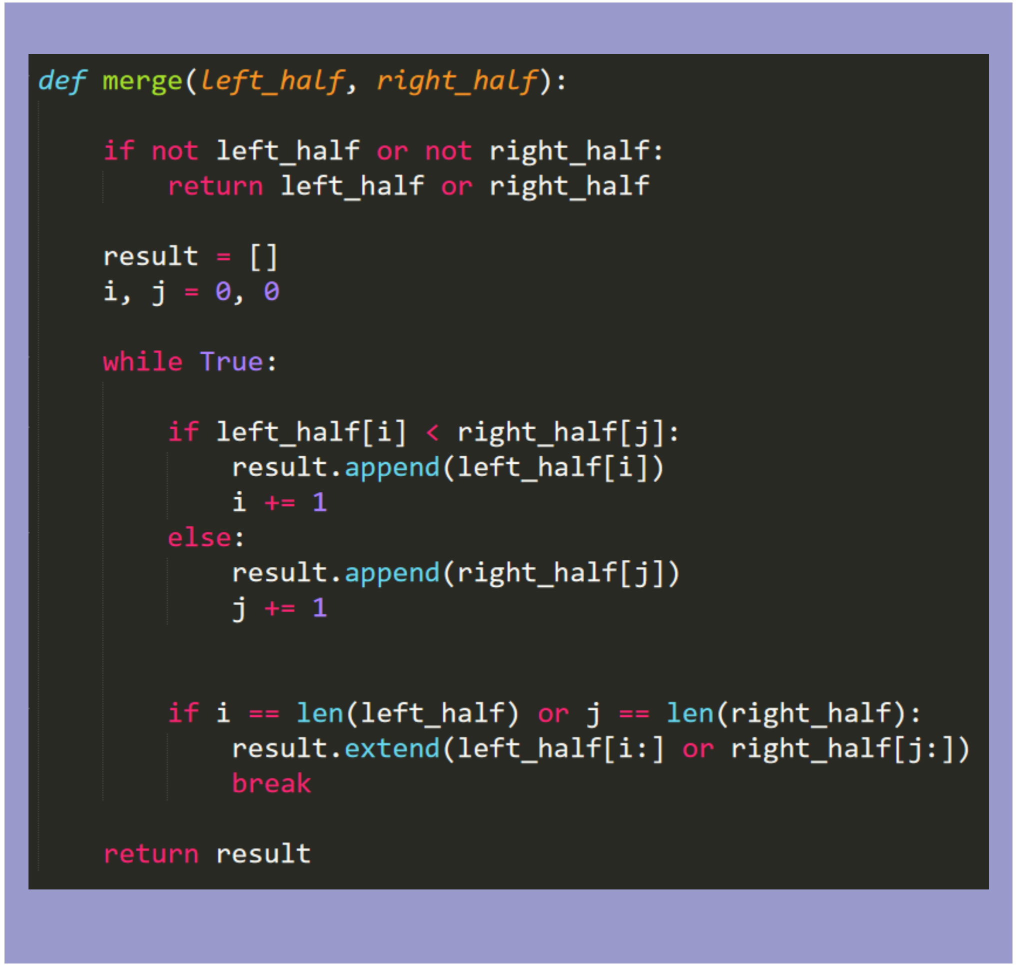


```python
def merge_sort(lst):

    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)
```

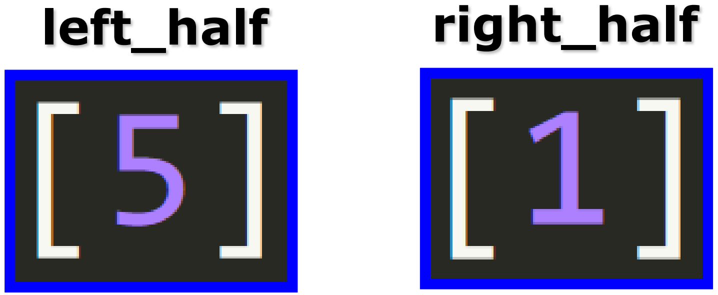


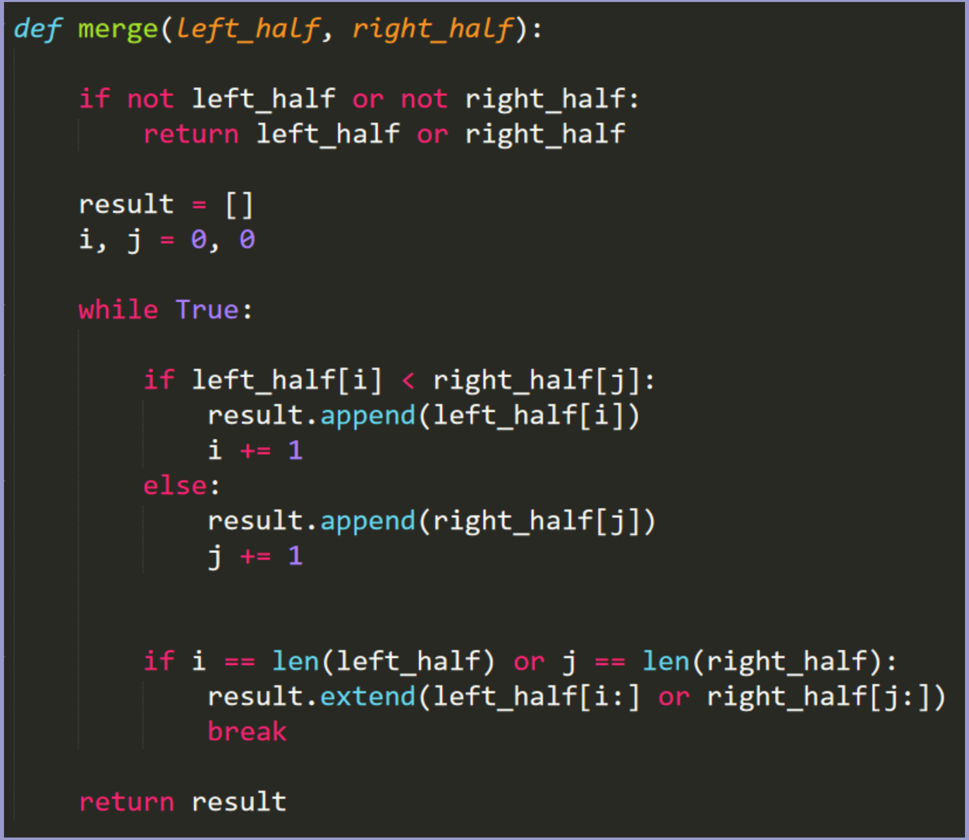
```python
def merge_sort(lst):

    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(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```





```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 0   j = 0

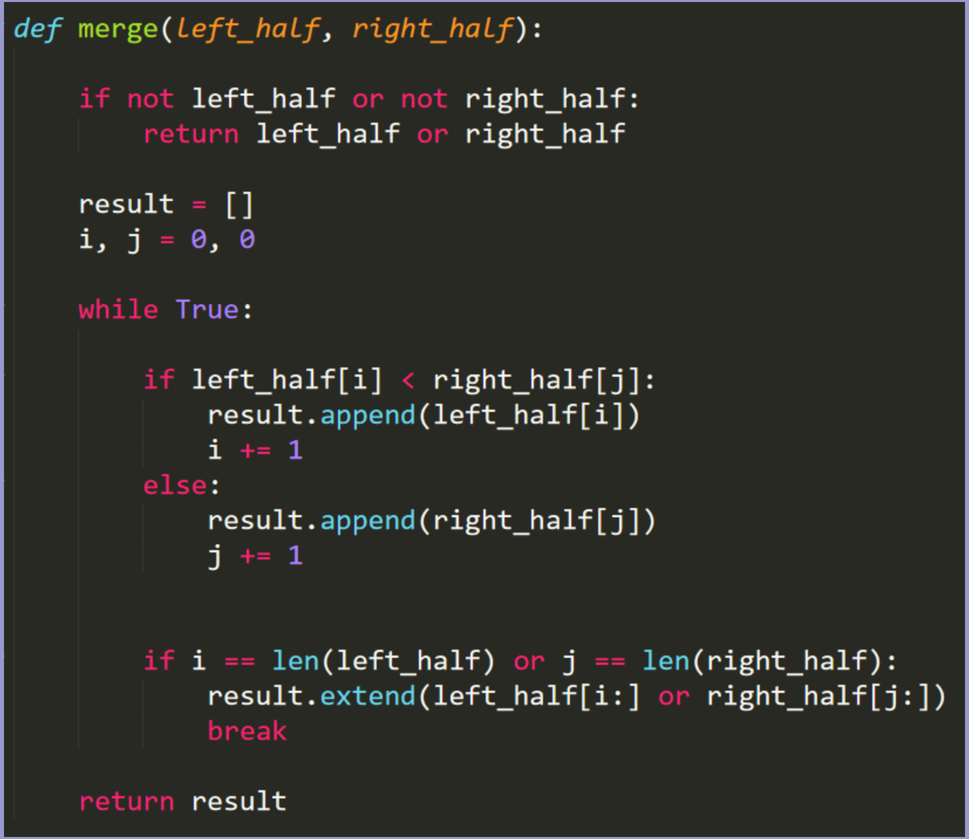
left_half

[5]

right_half

[1]

result = [ ]

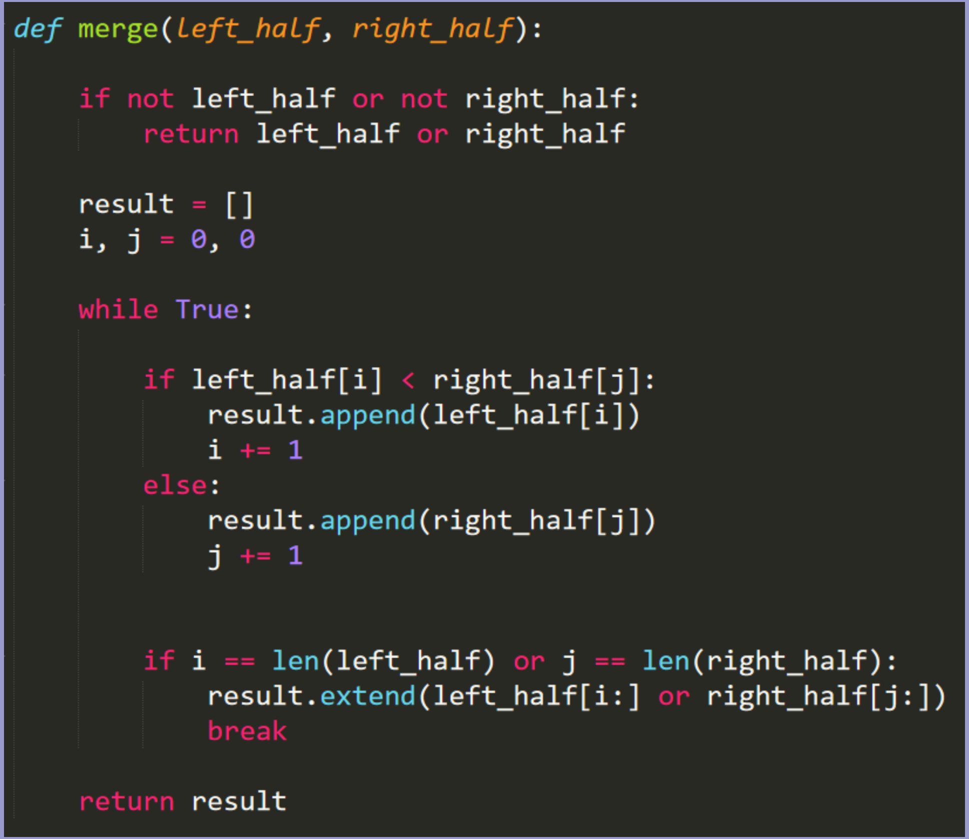


```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 0   j = 0

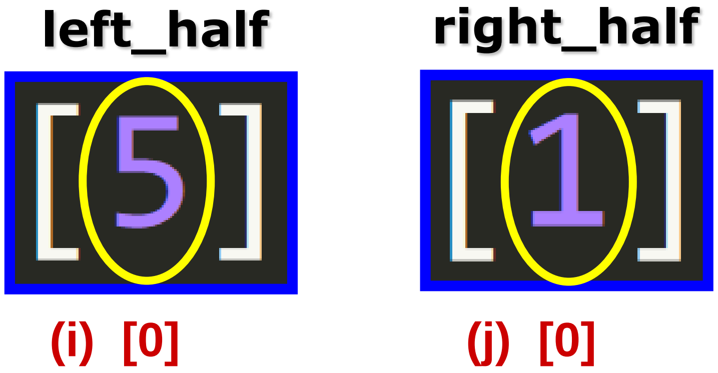
left_half	right_half
[5]	[1]
(i) [0]	(j) [0]

result = [ ]



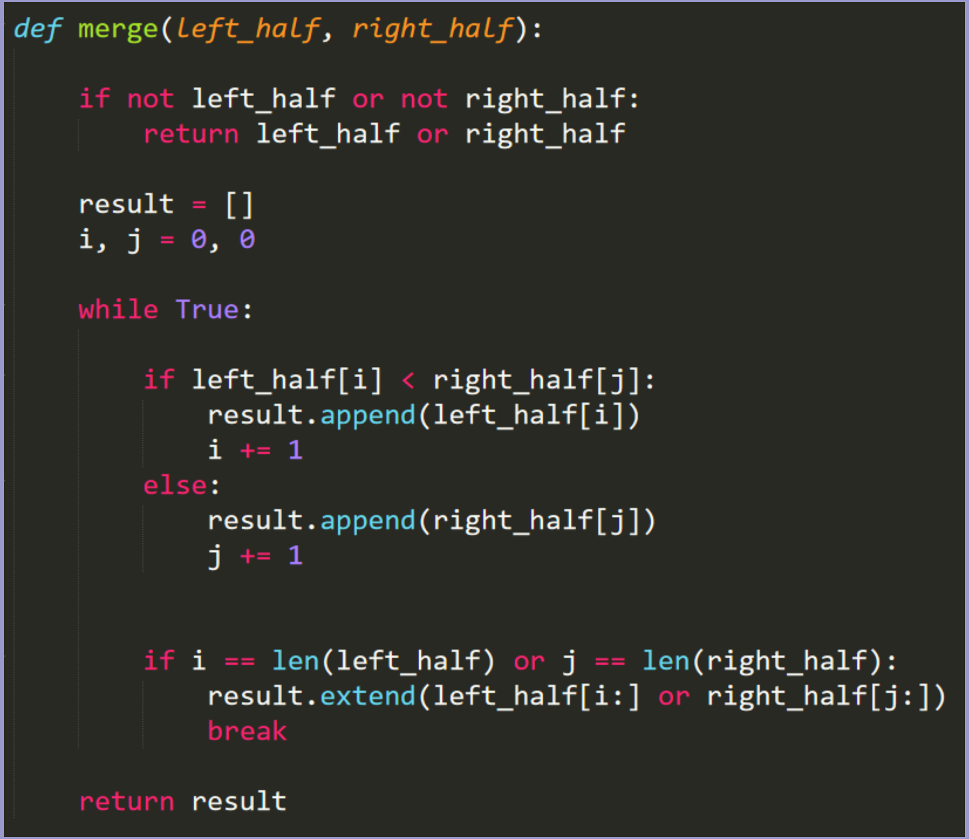
```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 0   j = 0



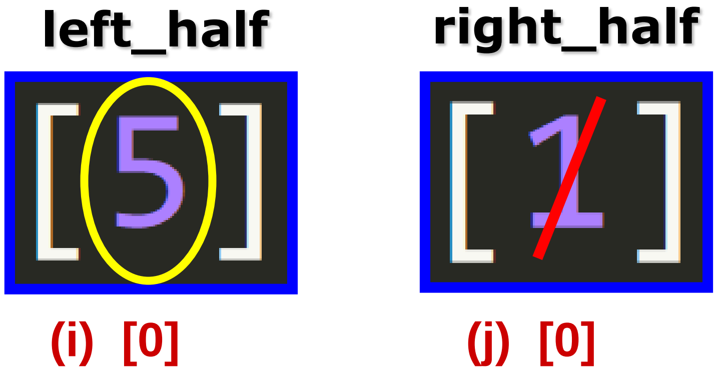
5 < 1?

result = [ ]

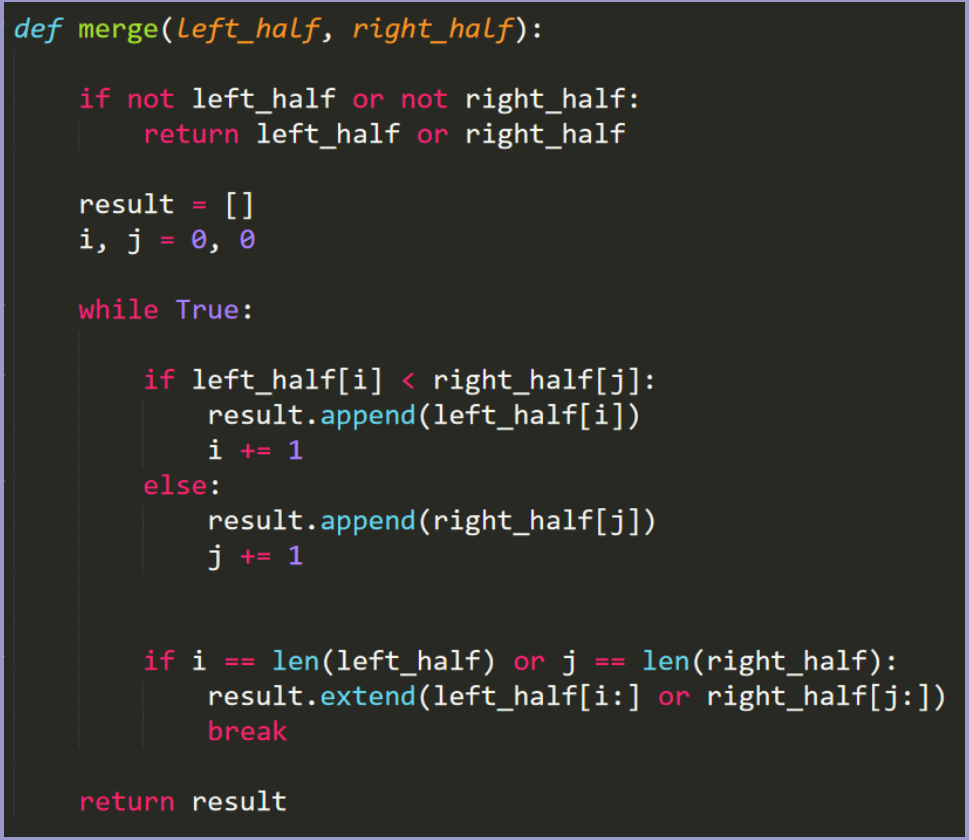


```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 0   j = 1

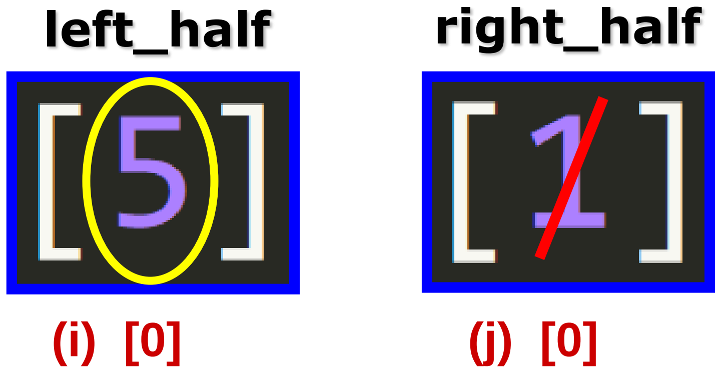


result = [1]

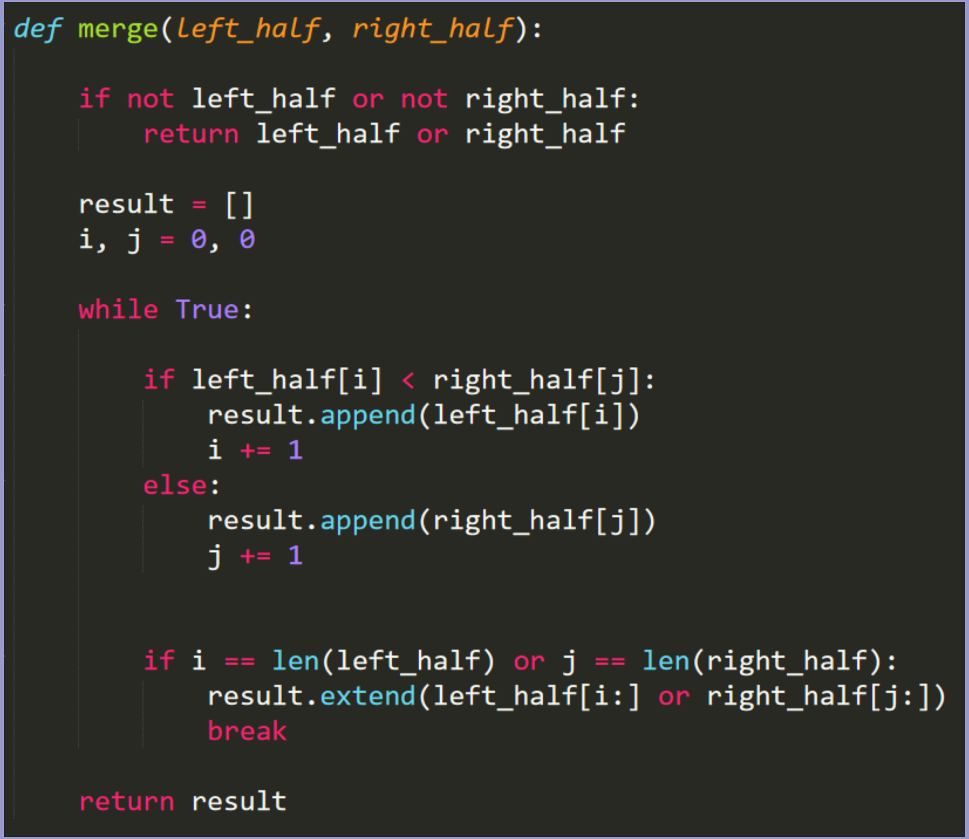


```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 0   j = 1



result = [1, 5]



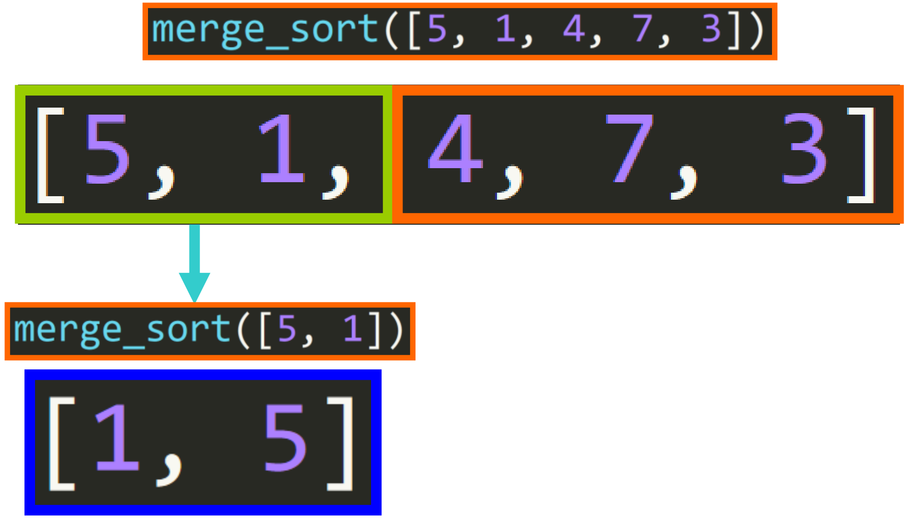
```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

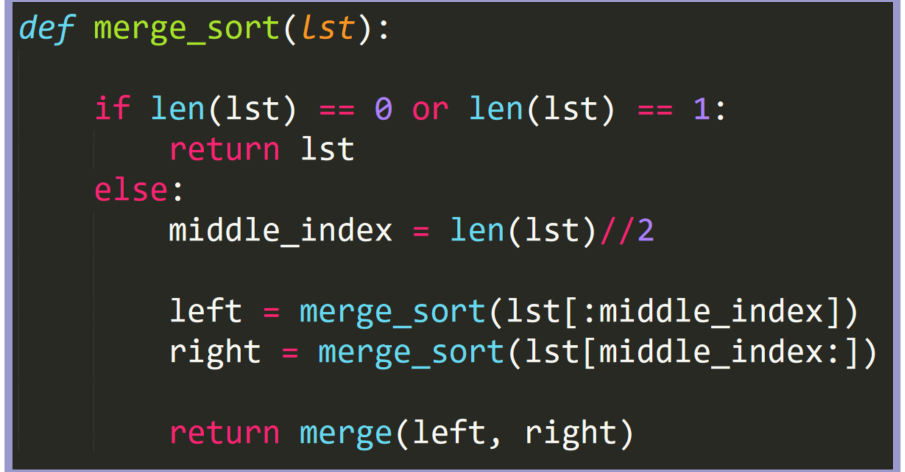
i = 0   j = 1

[1, 5]

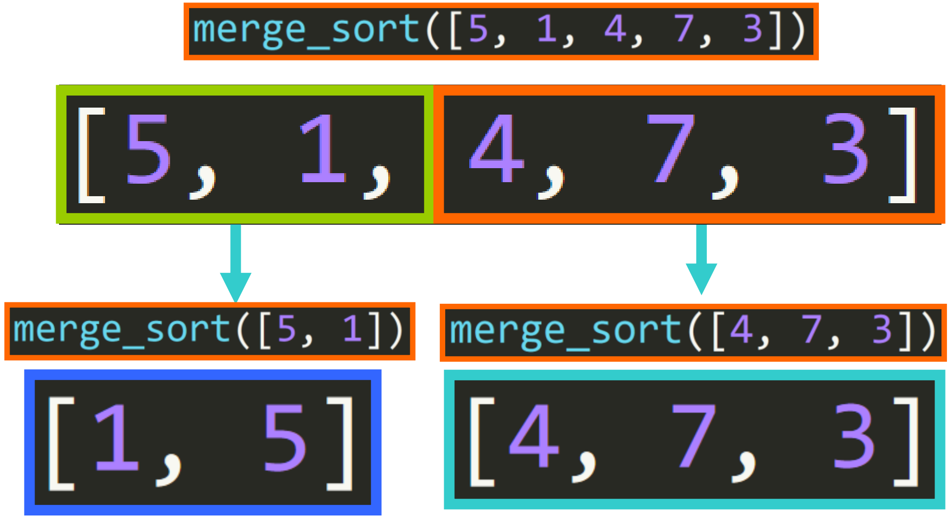
result = [1, 5]

```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)
```

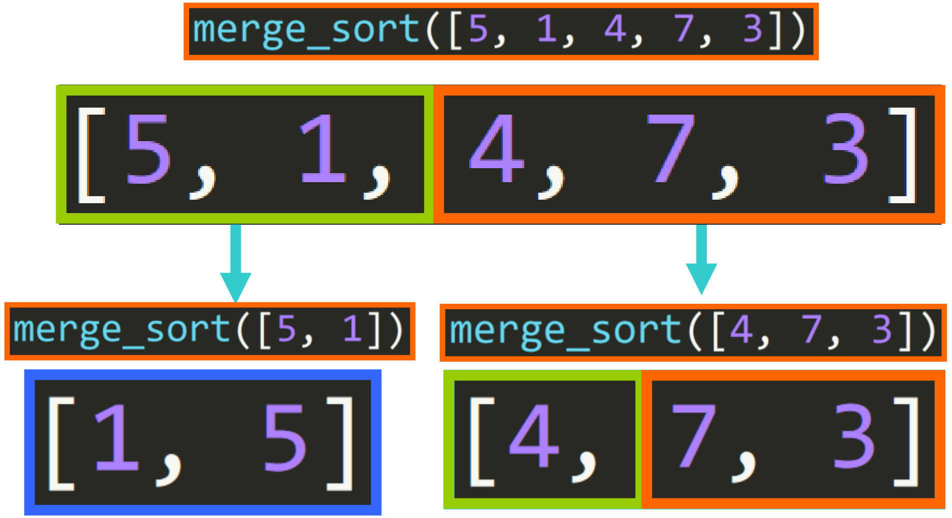




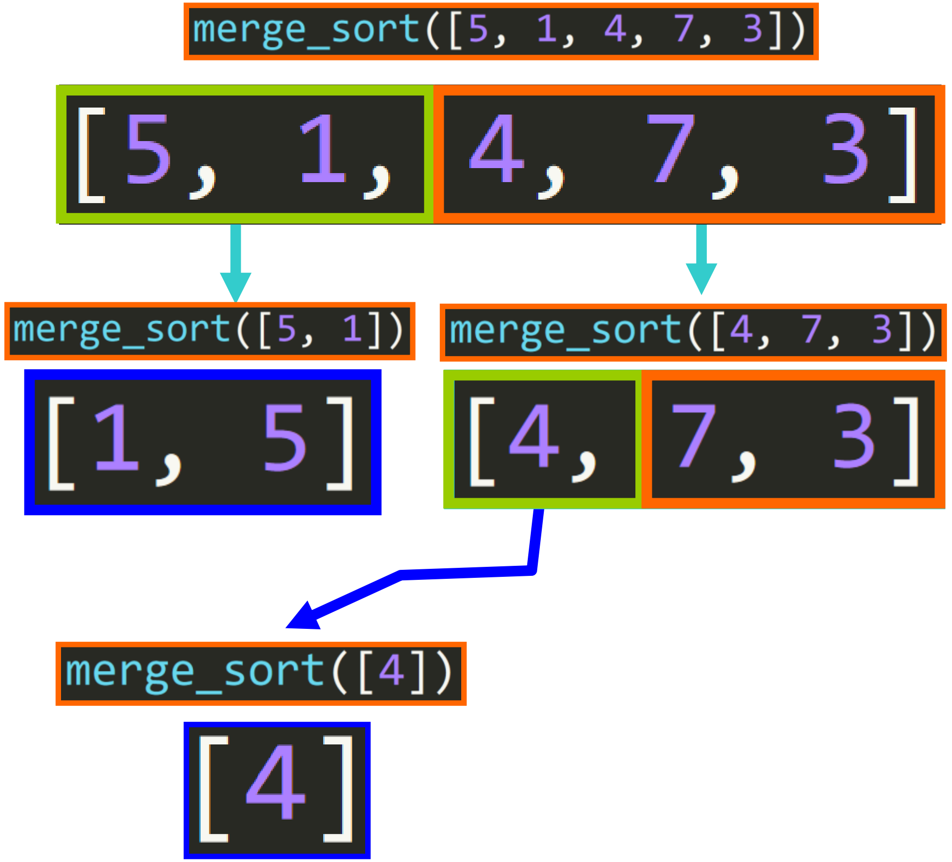
```python
def merge_sort(lst):

    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)
```



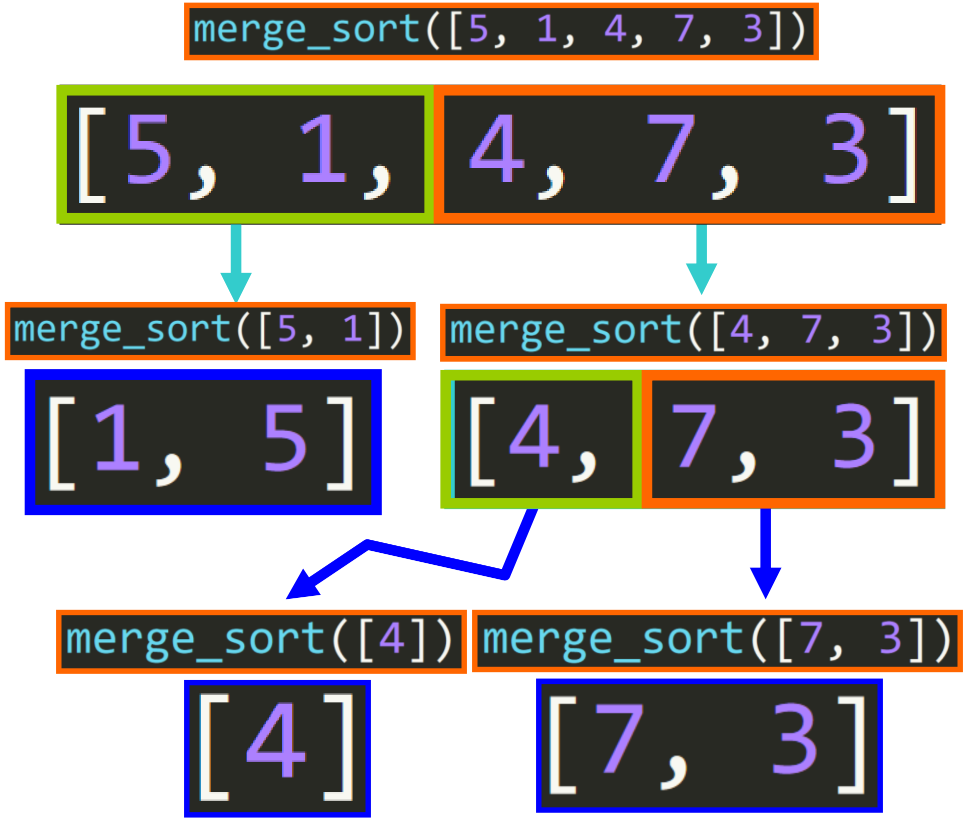
```python
def merge_sort(lst):

    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)
```



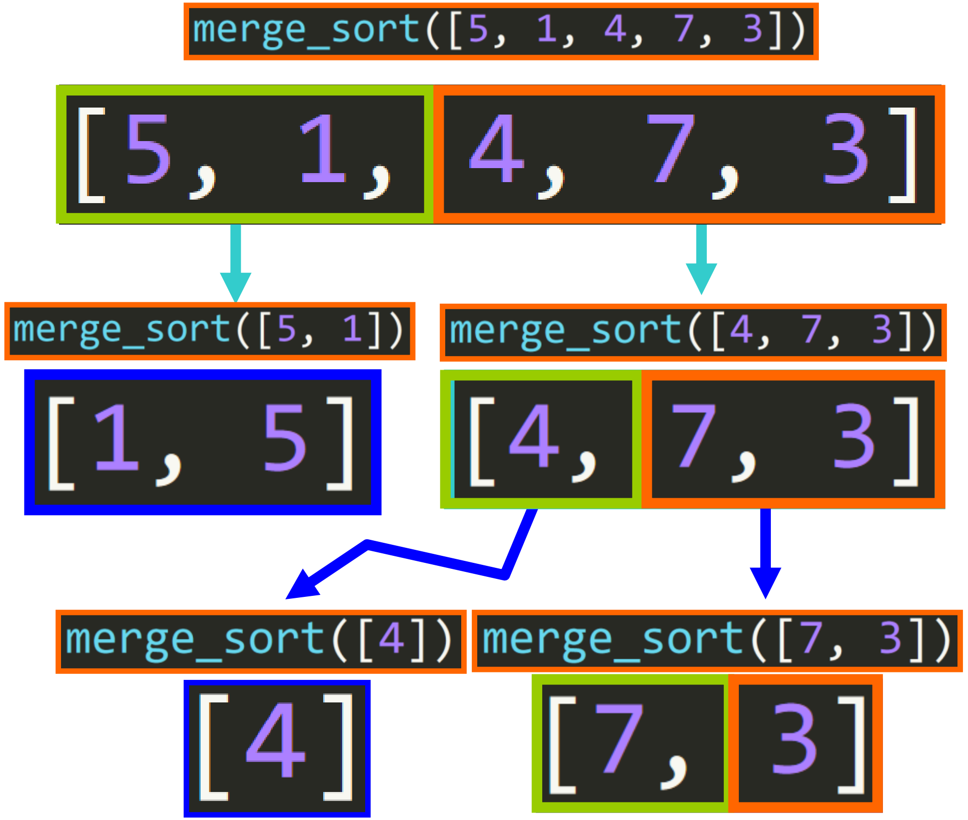
```python
def merge_sort(lst):

    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)
```

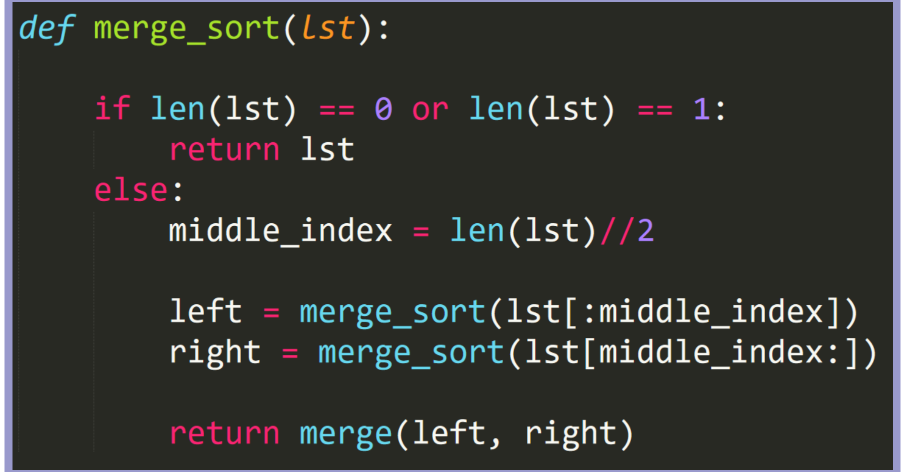


```python
def merge_sort(lst):

    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)
```

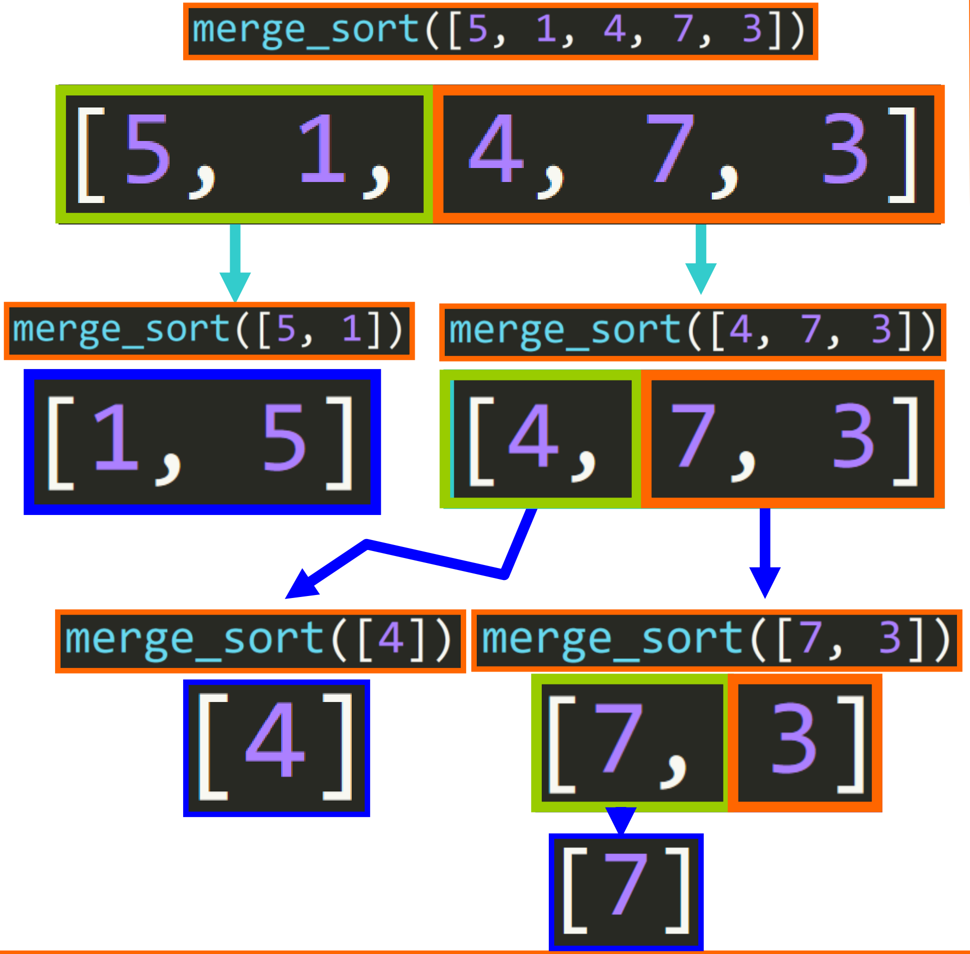


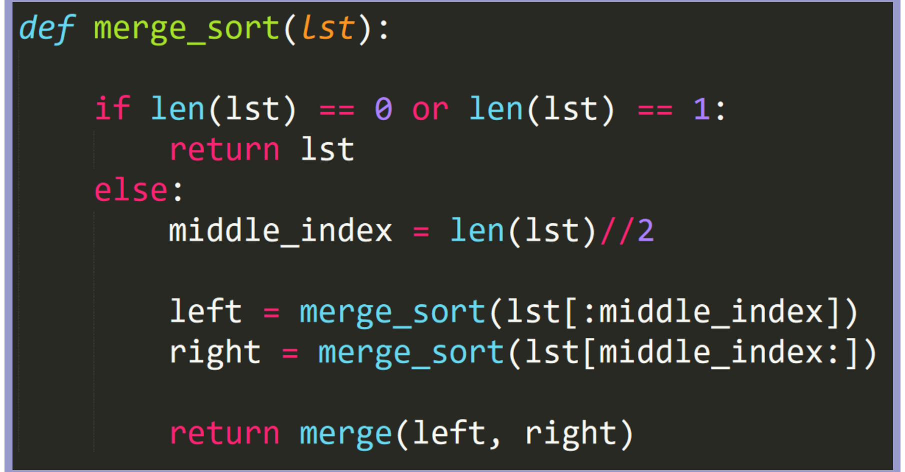
```python
def merge_sort(lst):

    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)
```



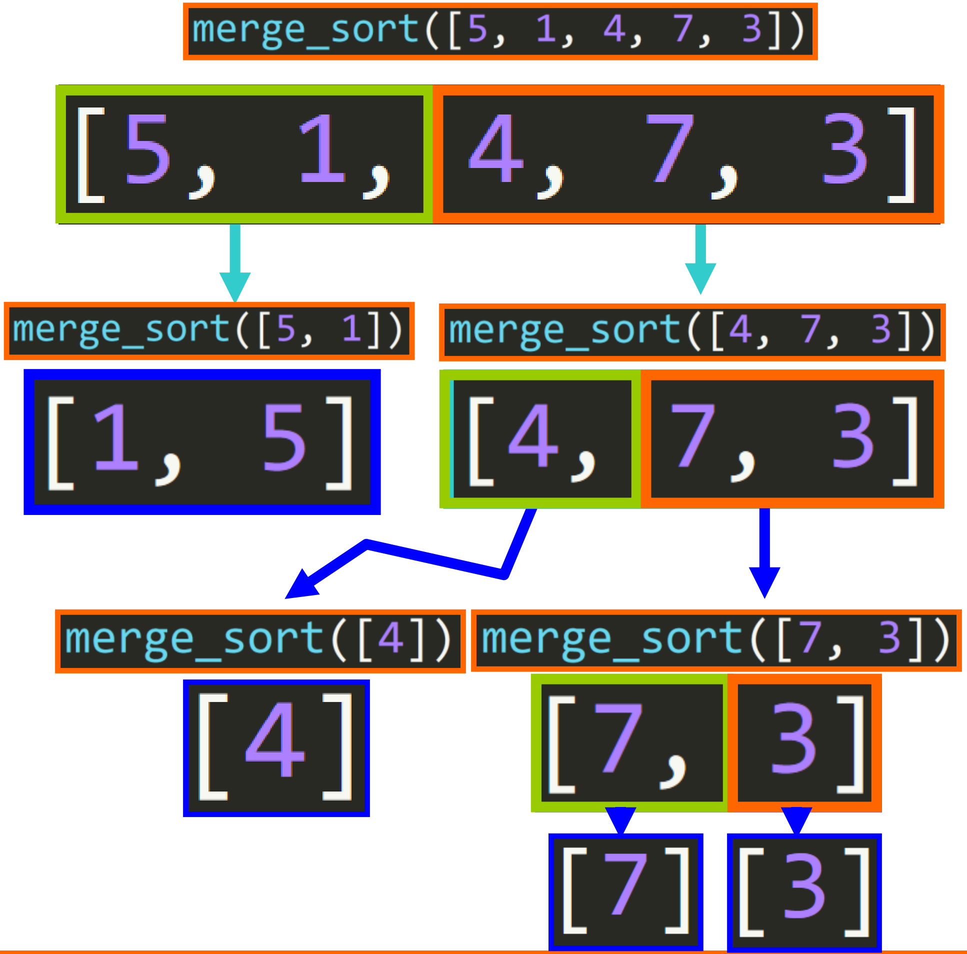


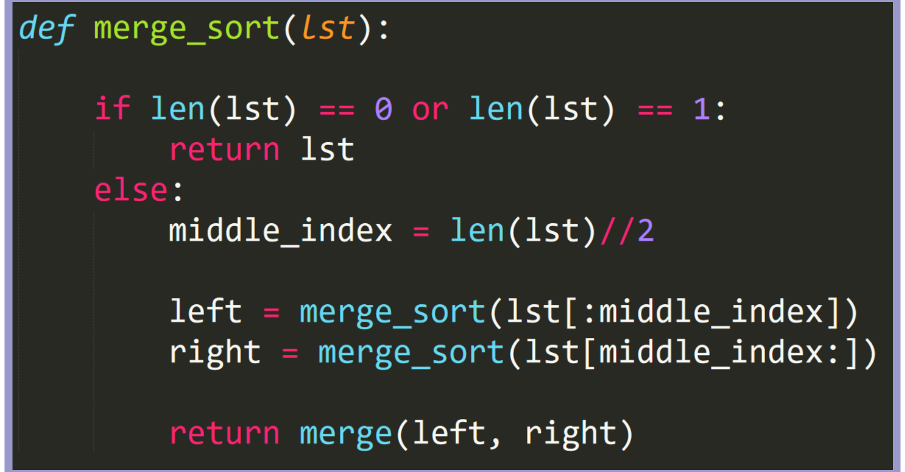
```python
def merge_sort(lst):

    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)
```

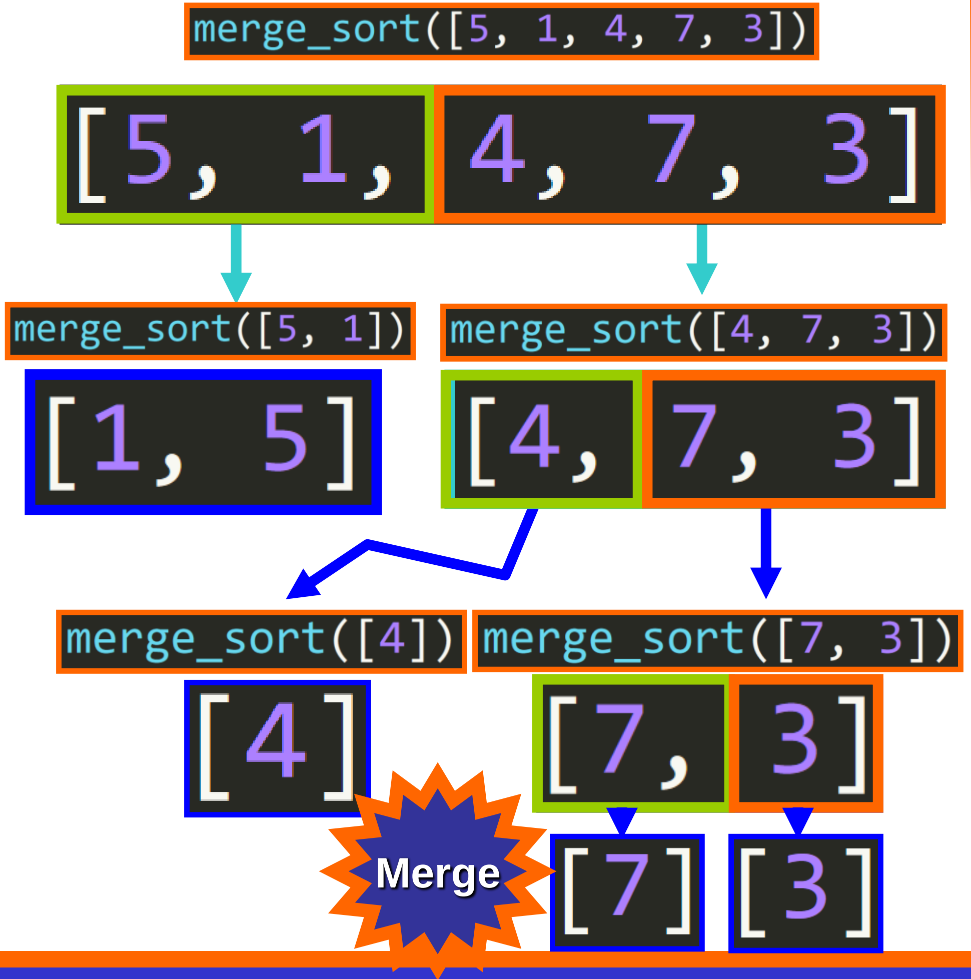




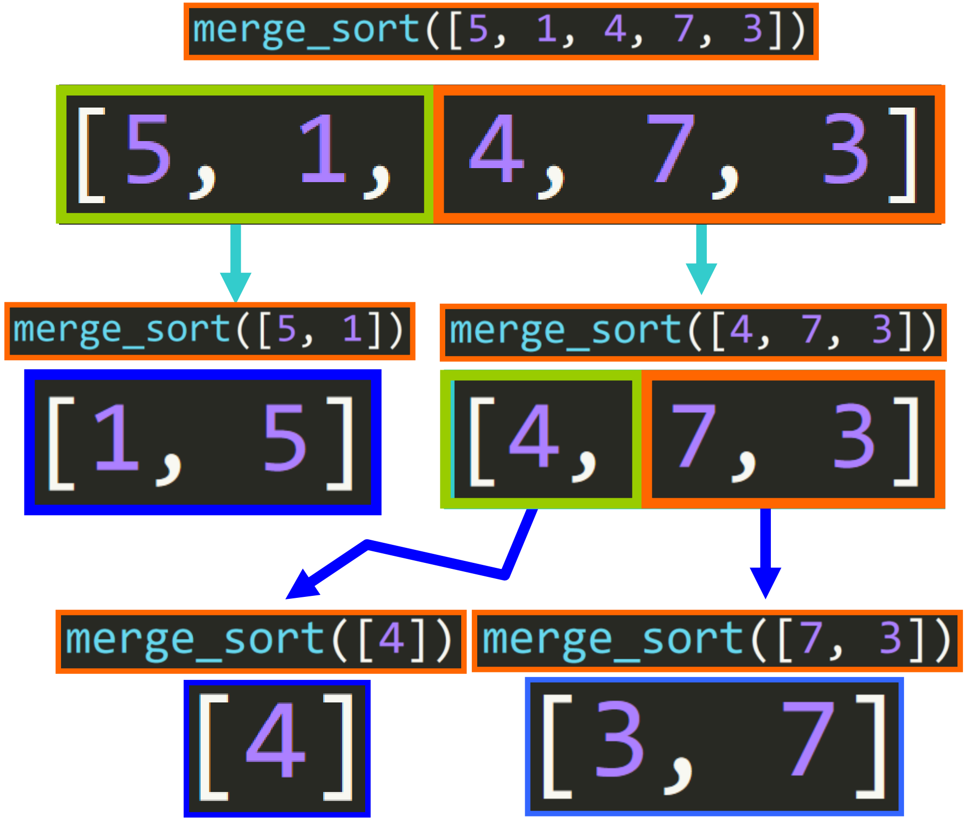
```
def merge_sort(lst):

    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)
```



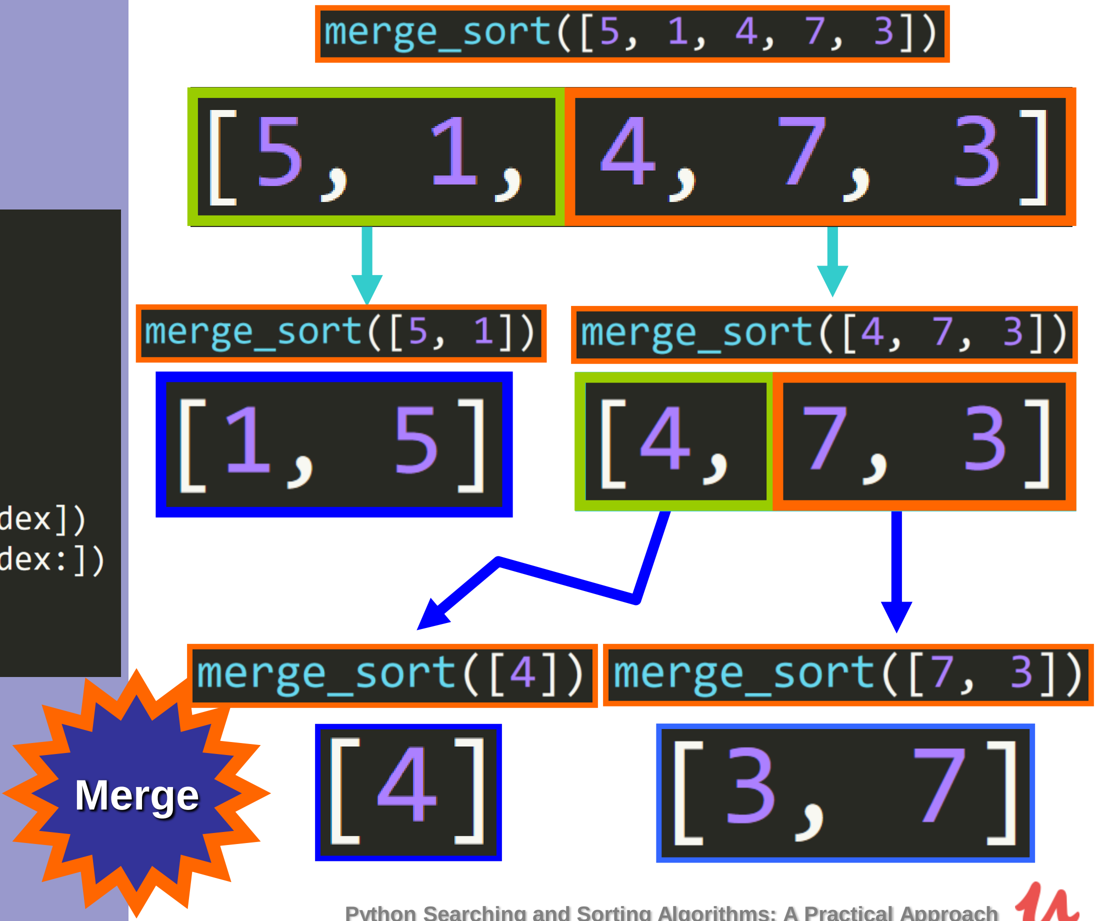

```python
def merge_sort(lst):

    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)
```

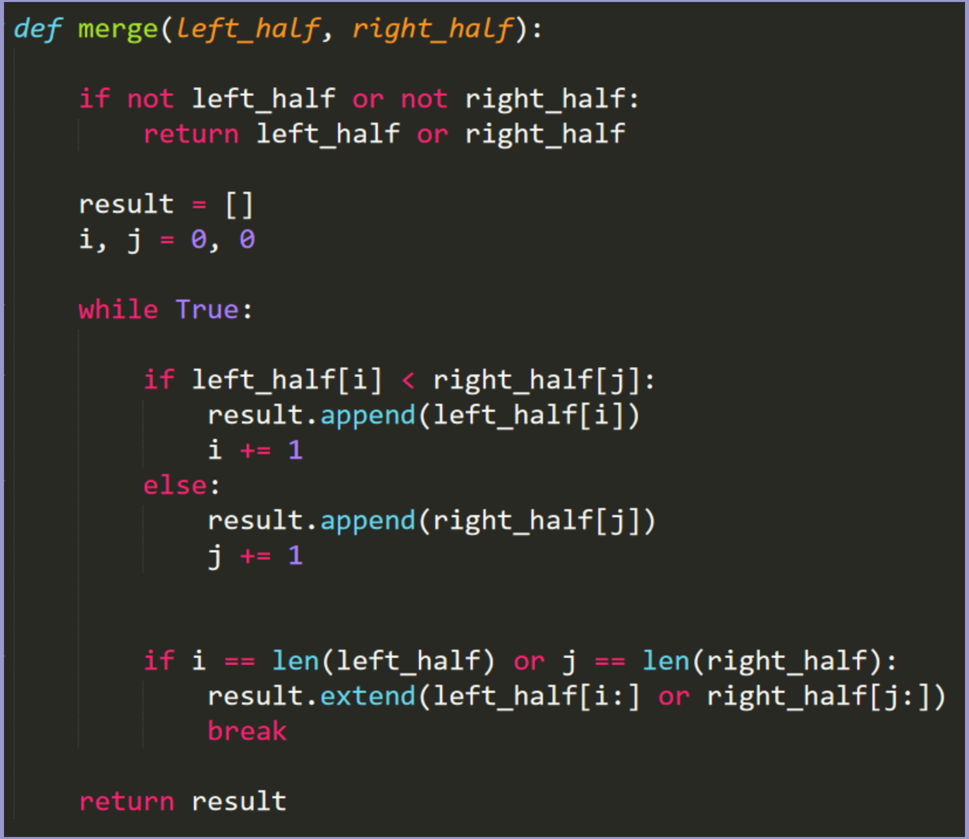


```python
def merge_sort(lst):

    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)
```

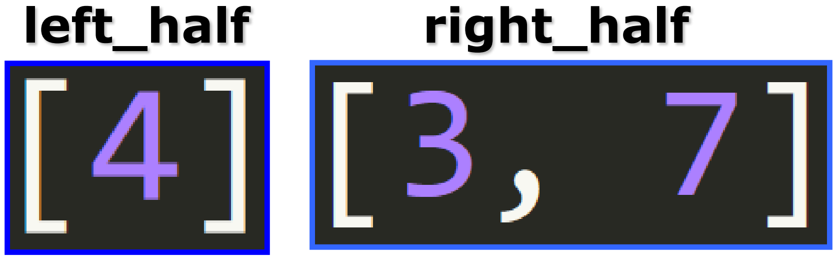


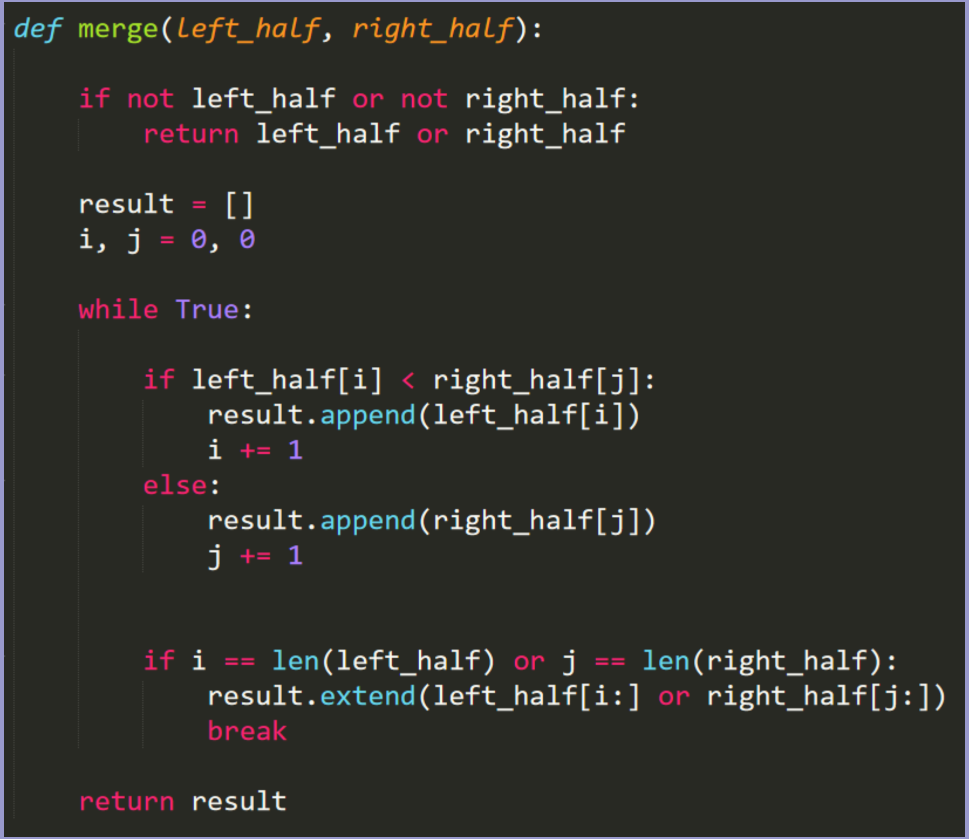
```python
def merge_sort(lst):

    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(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```



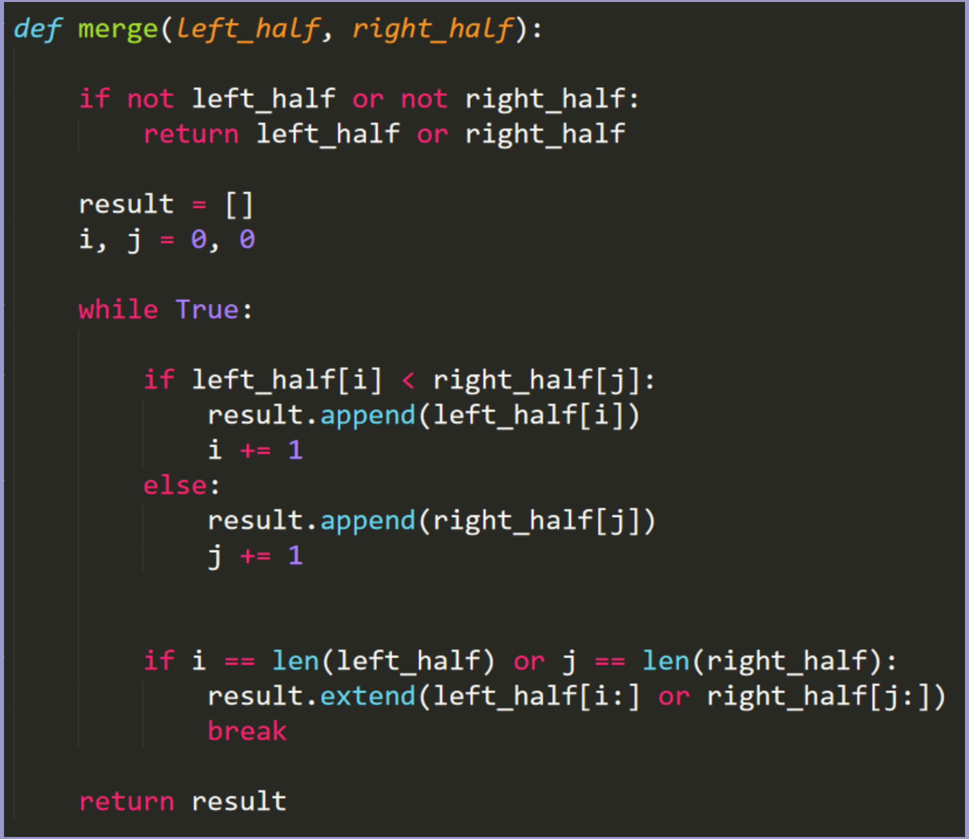


```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 0   j = 0

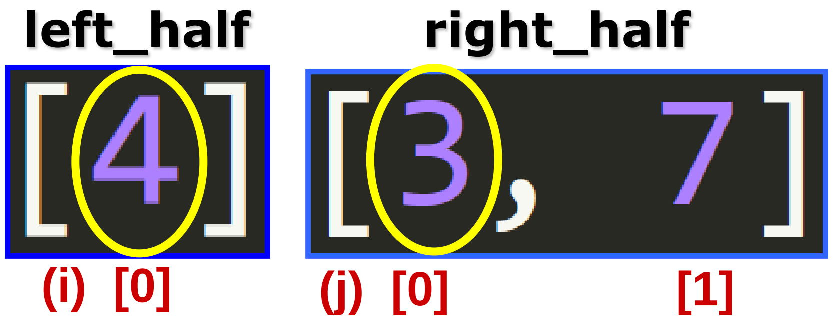
left_half	right_half
[4]	[3, 7]
(i) [0]	(j) [0] [1]

result = [ ]



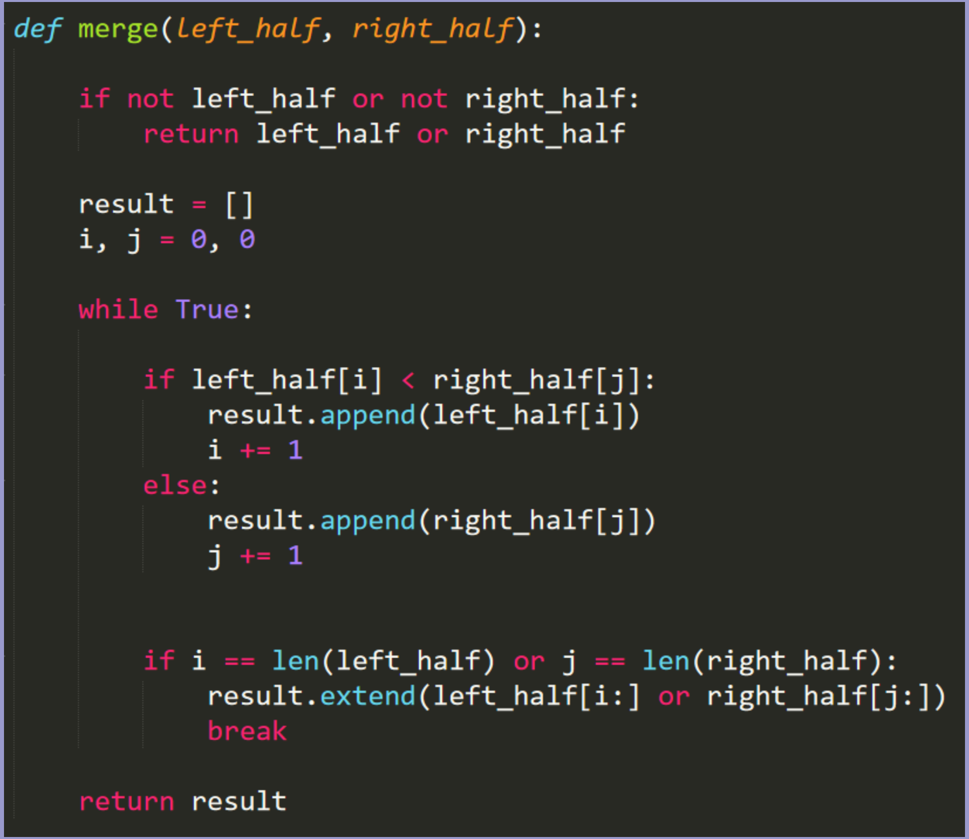
```
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 0 j = 0



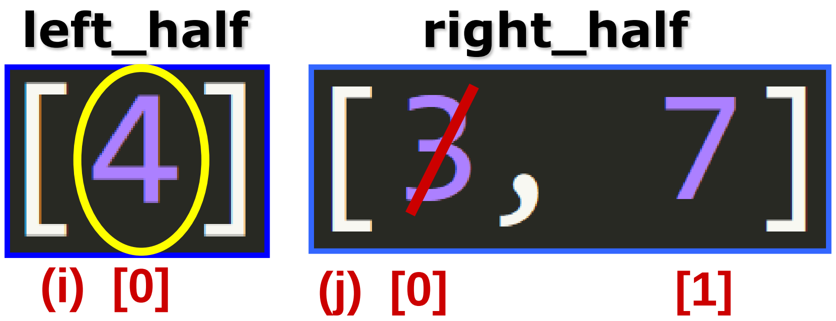
4 < 3?

result = [ ]

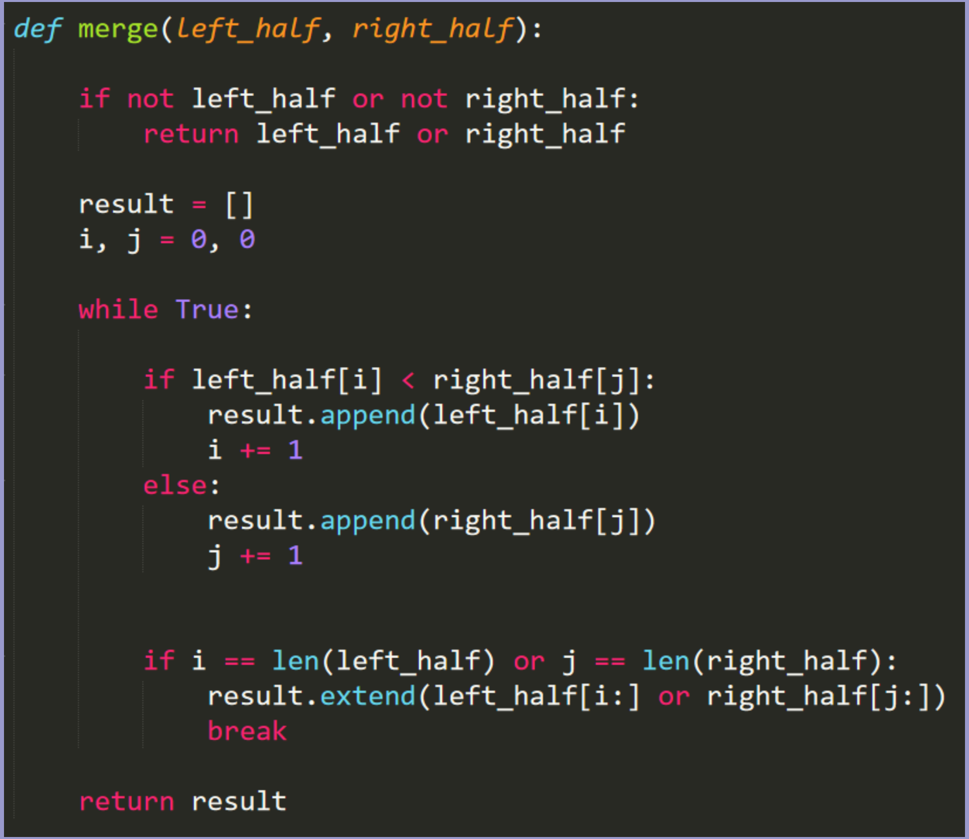


```
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 0   j = 1

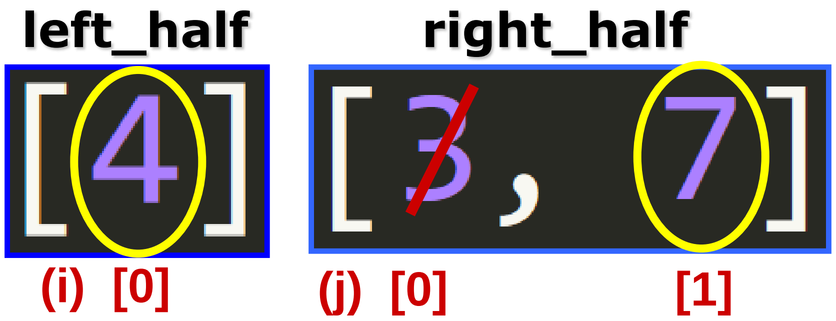


result = [3]



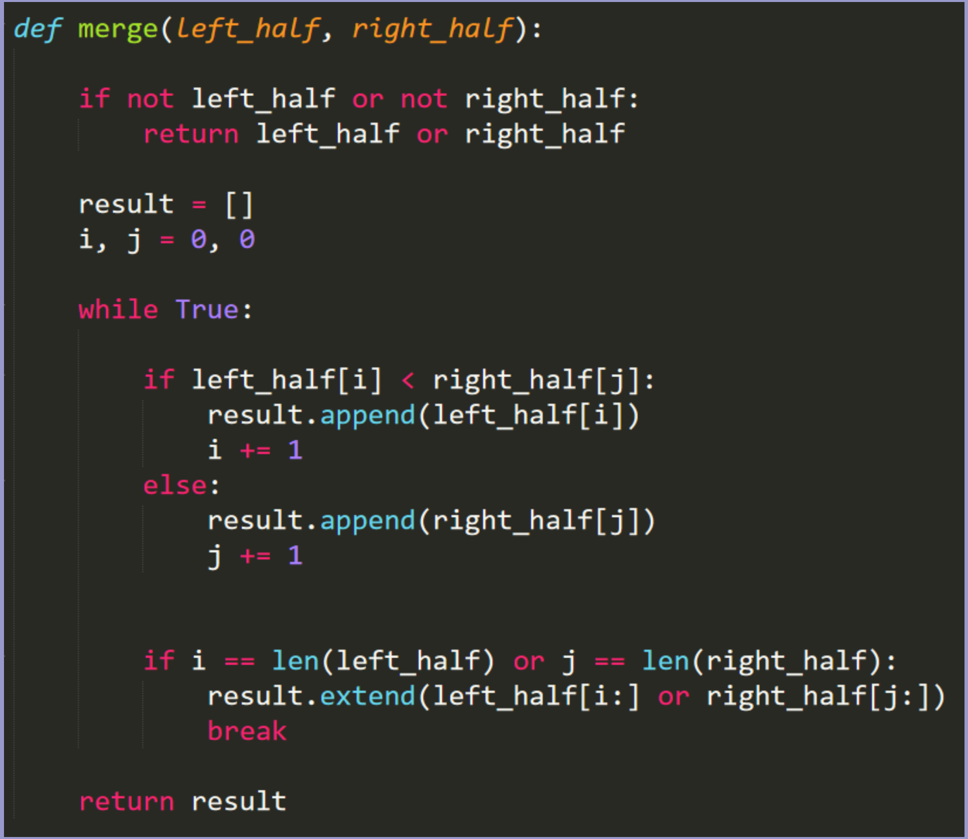
```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 0   j = 1



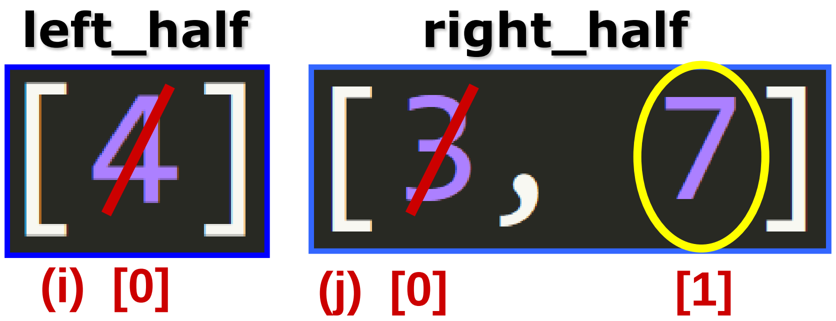
4 < 7?

result = [3]



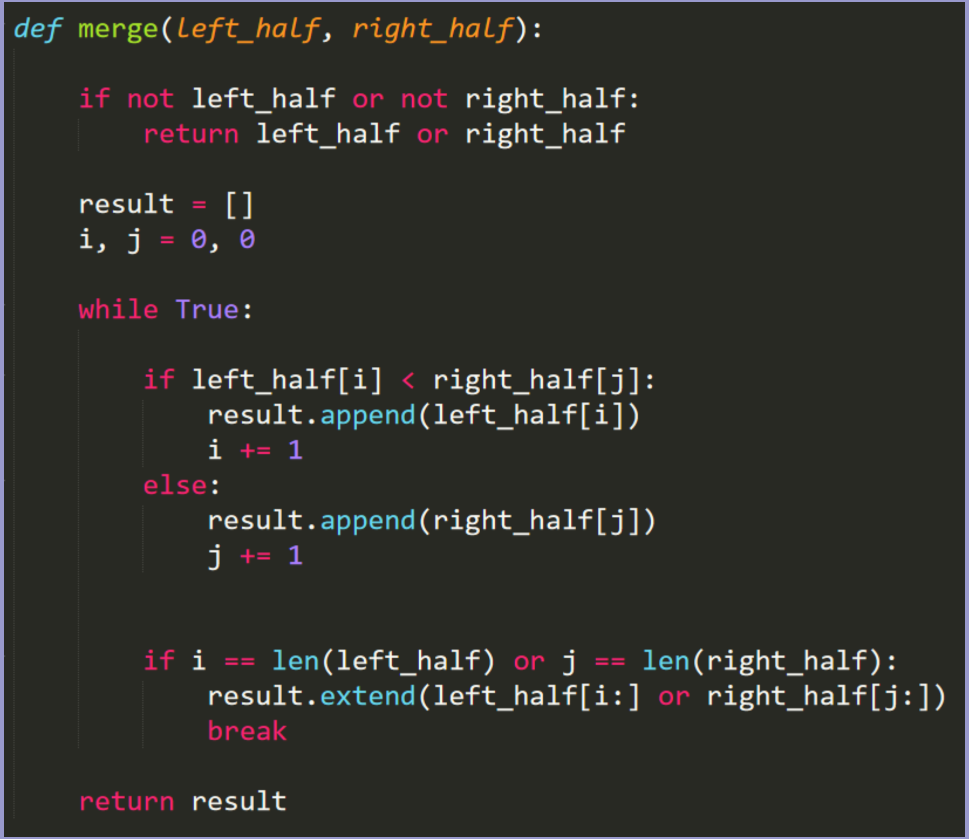
```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 1   j = 1



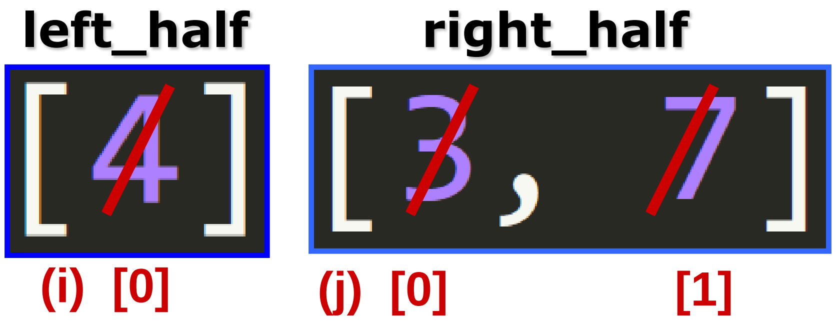
4 < 7?

result = [3, 4]

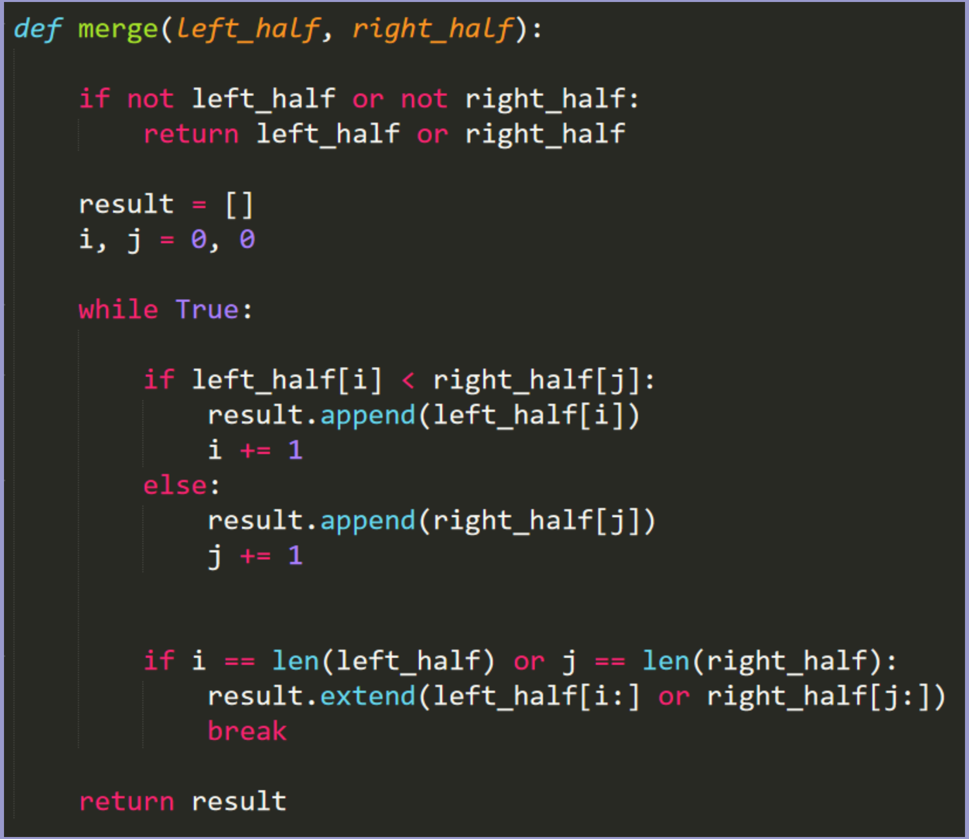


```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 1   j = 1



result = [3, 4, 7]



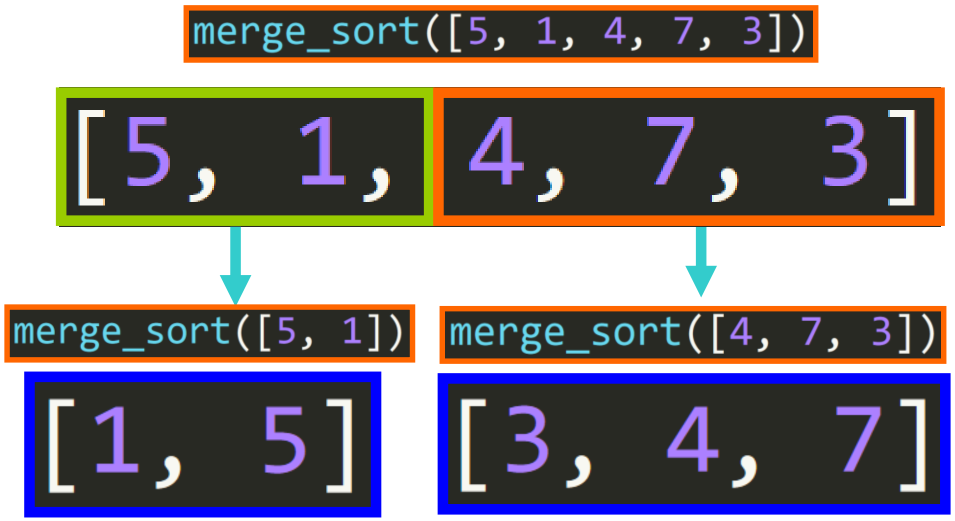
```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

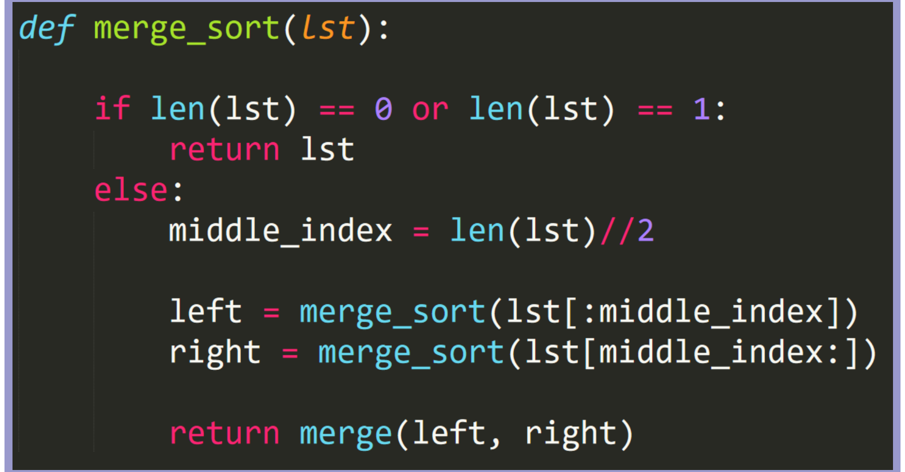
i = 1   j = 1

[3, 4, 7]

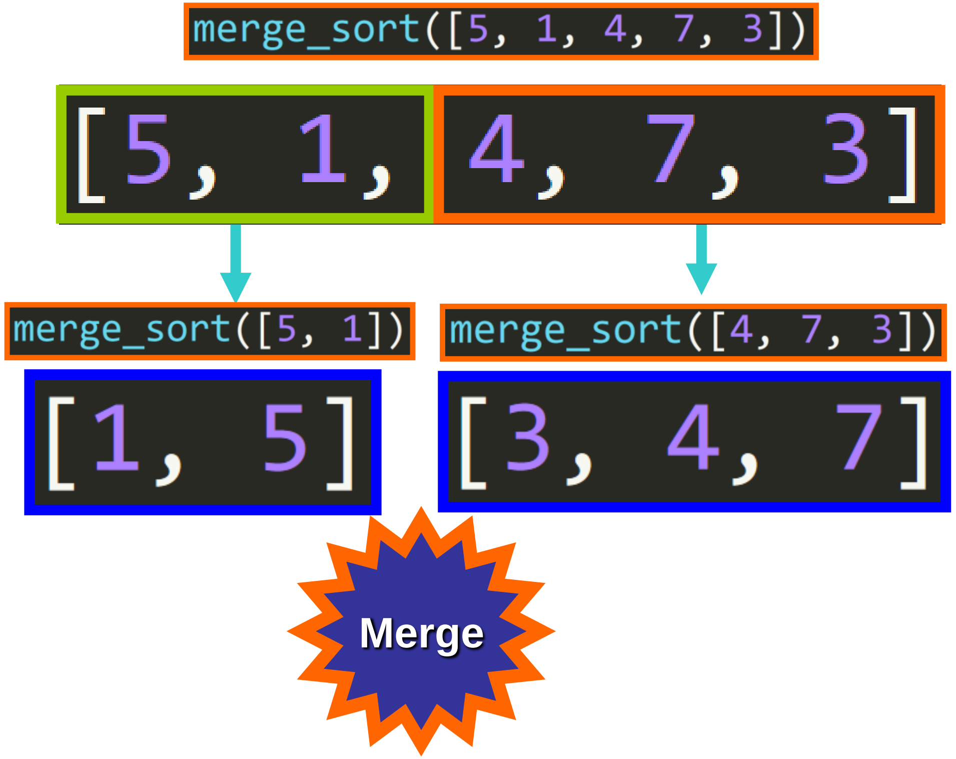
result = [3, 4, 7]

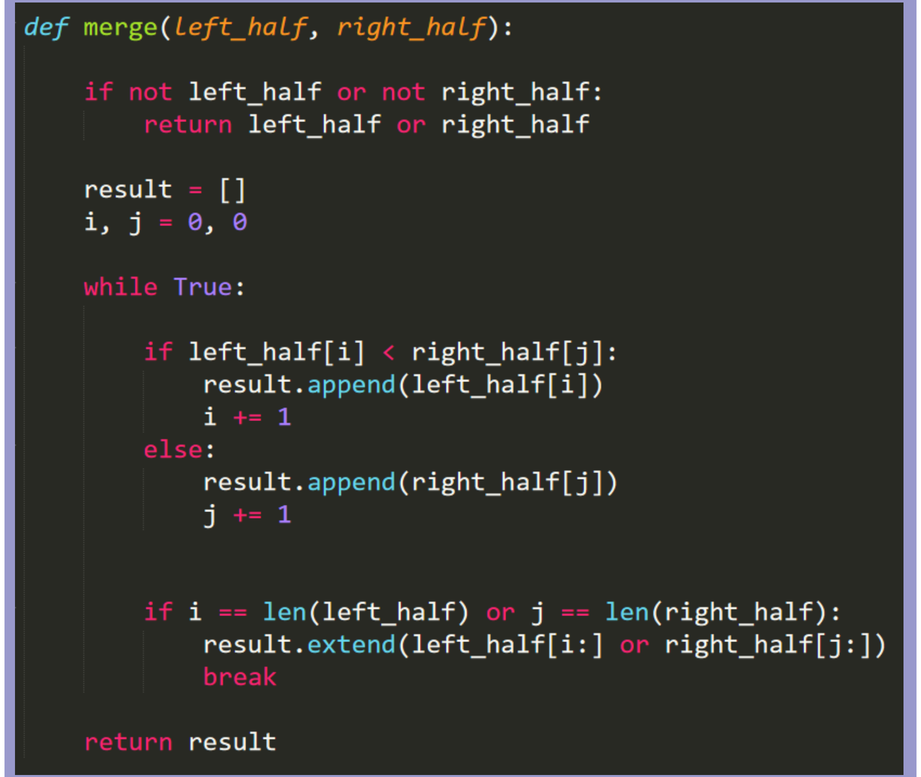
```python
def merge_sort(lst):

    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)
```



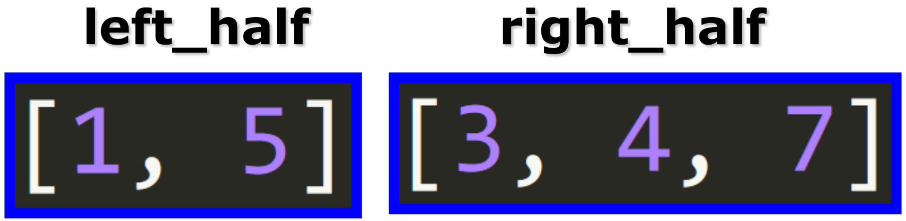


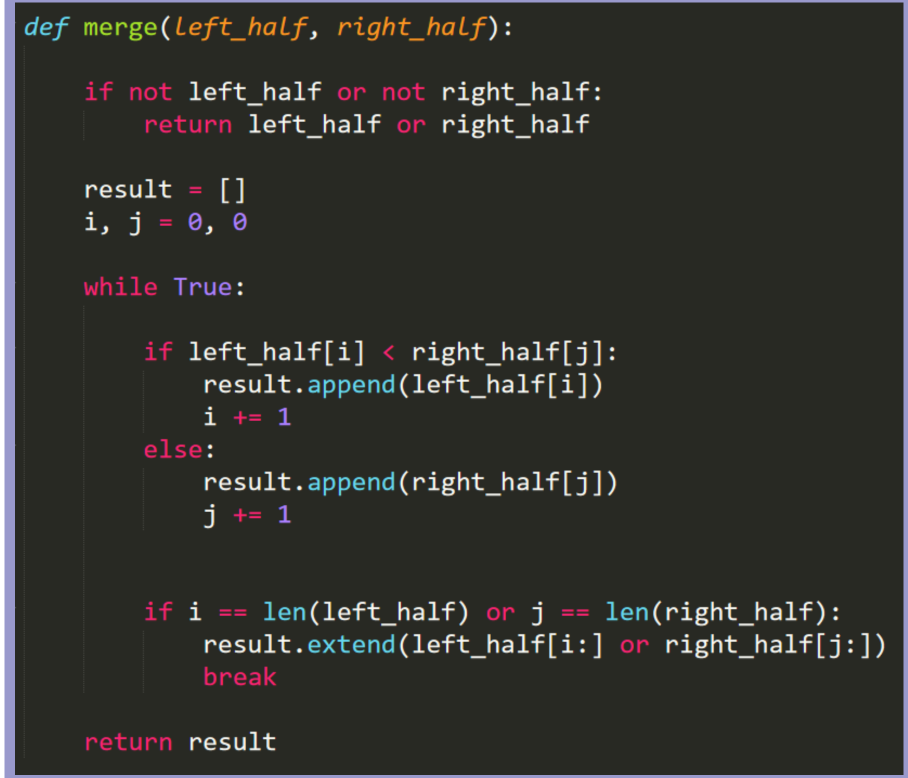
```python
def merge_sort(lst):

    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)
```





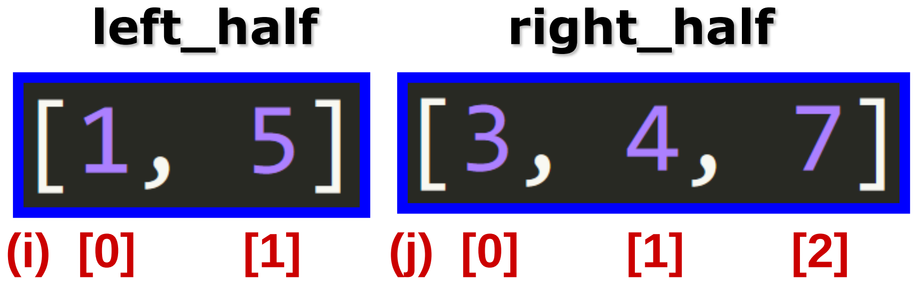
```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```



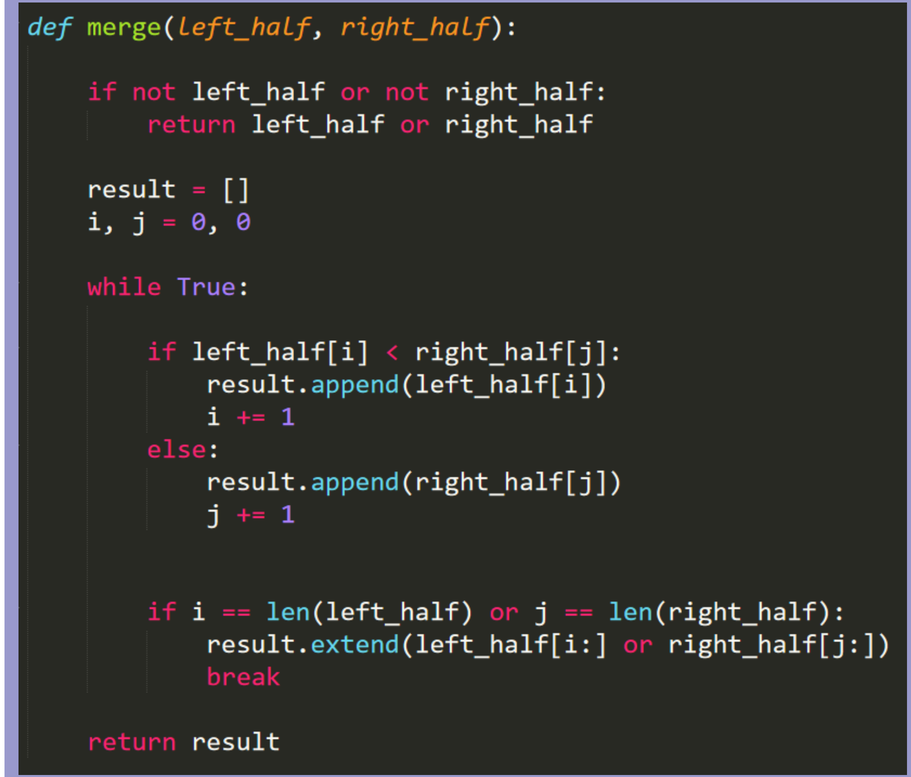


```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 0   j = 0

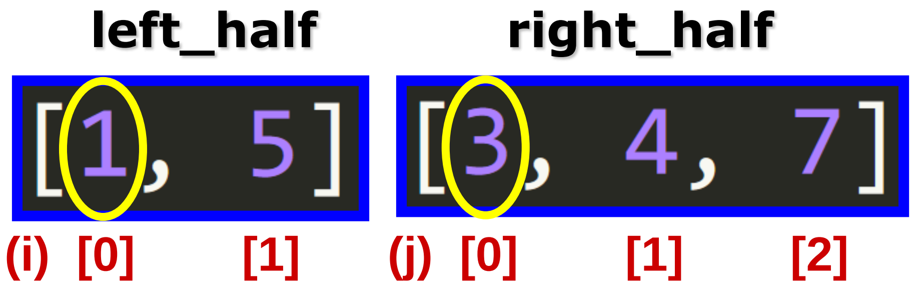


result = [ ]



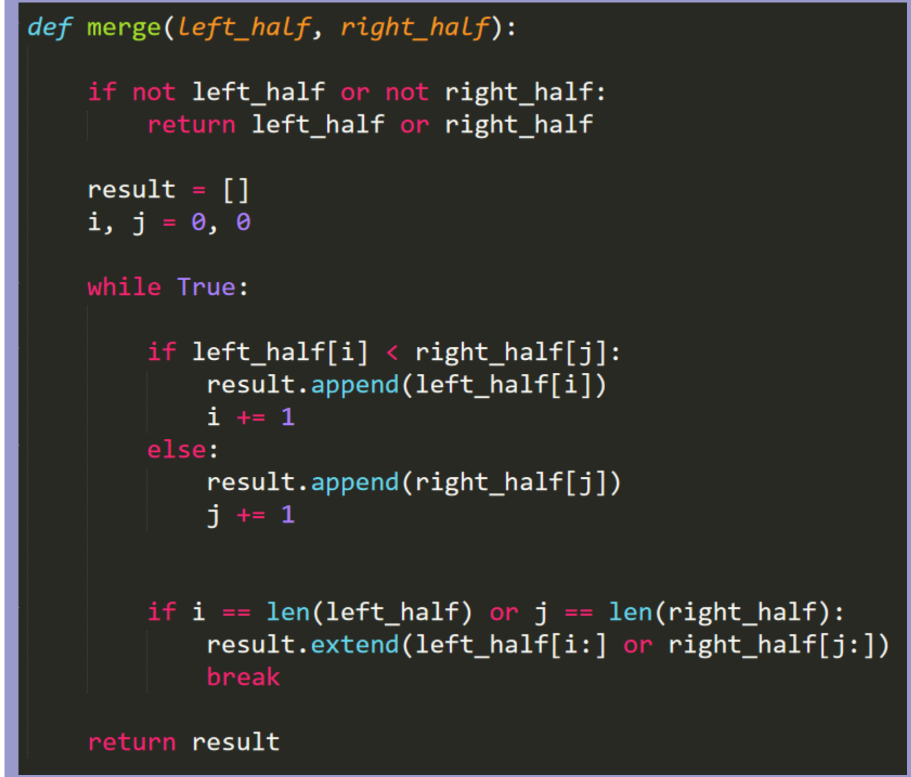
```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 0   j = 0



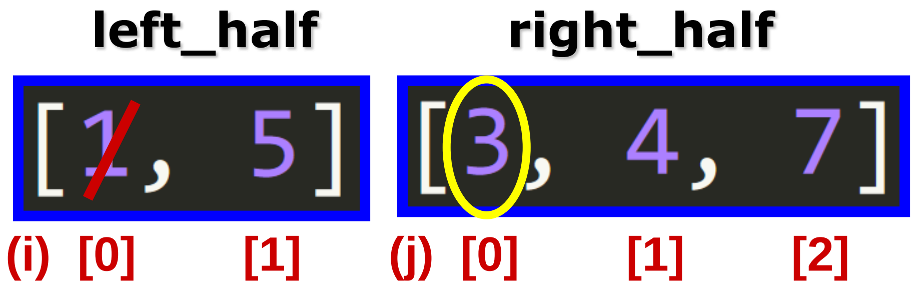
1 < 3?

result = [ ]

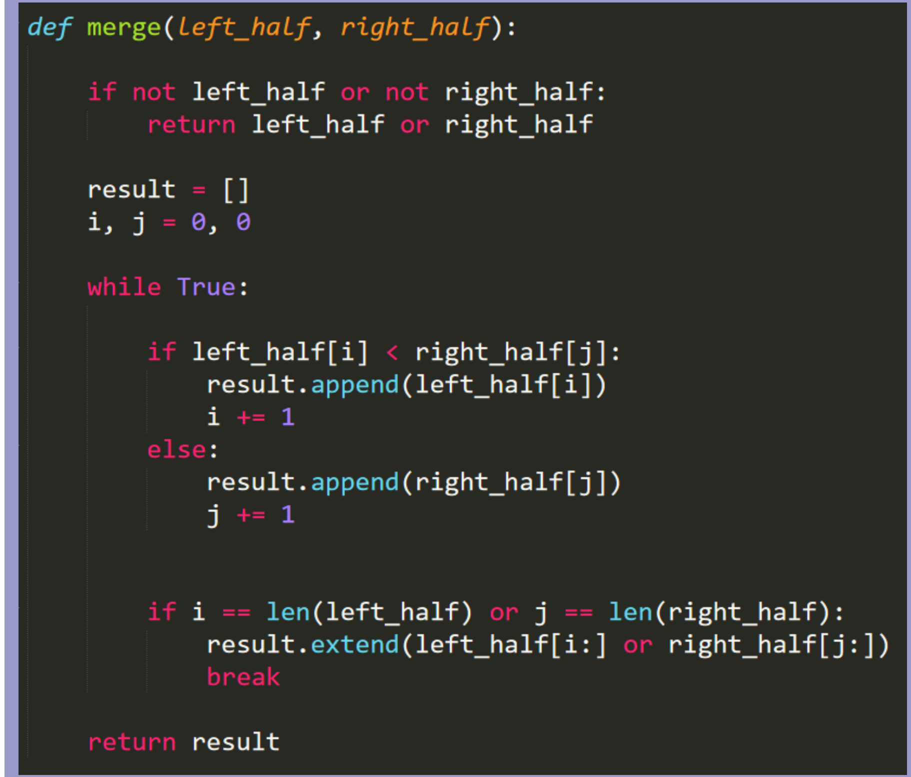


```
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 1   j = 0



result = [1]



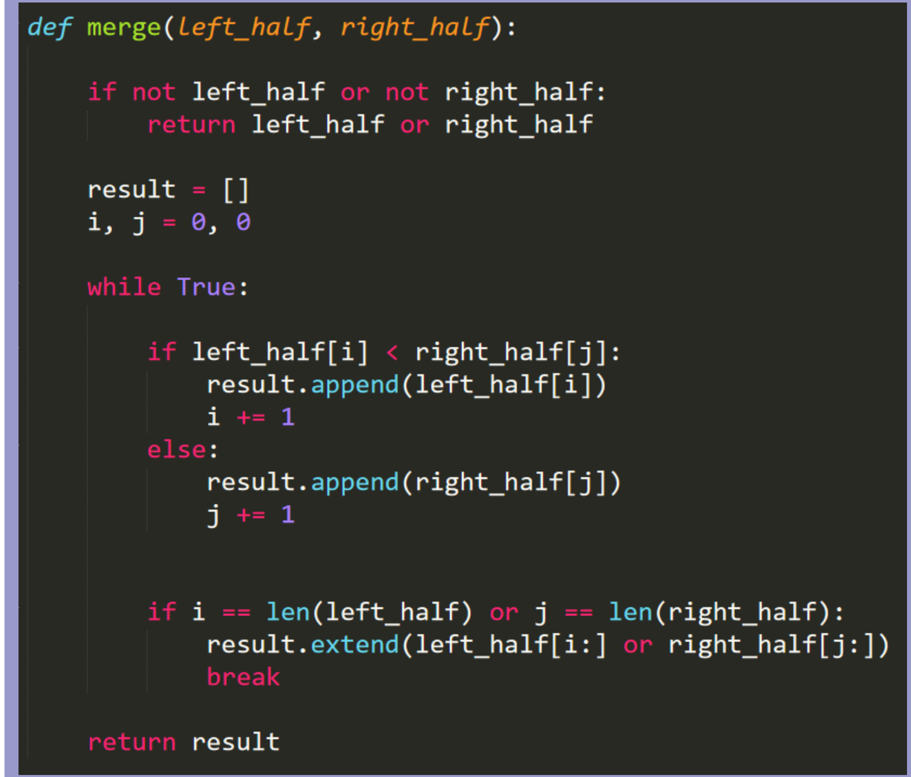
```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 1   j = 0

left_half		right_half		
~~1~~	5	3	4	7
(i) [0]	[1]	(j) [0]	[1]	[2]

5 < 3?

result = [1]



```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 1   j = 1

left_half

[$\cancel{1}$, 5]

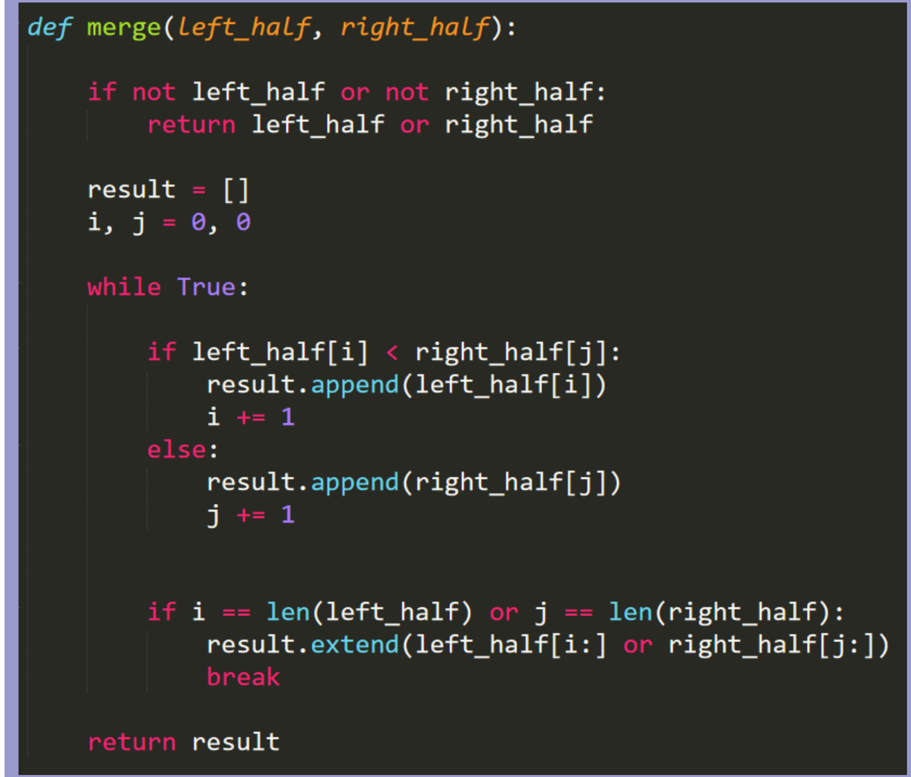
(i) [0] [1]

right_half

[$\cancel{3}$, 4, 7]

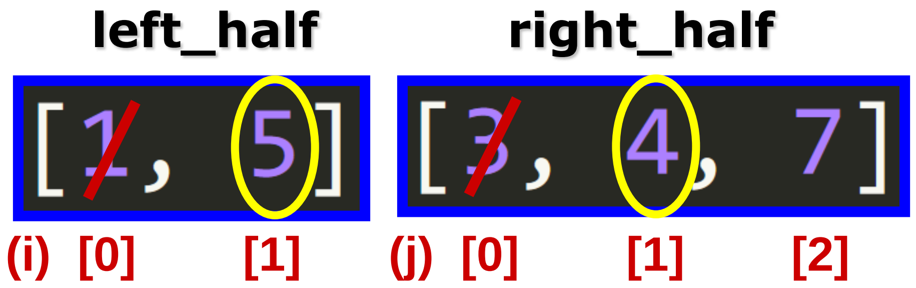
(j) [0] [1] [2]

result = [1, 3]



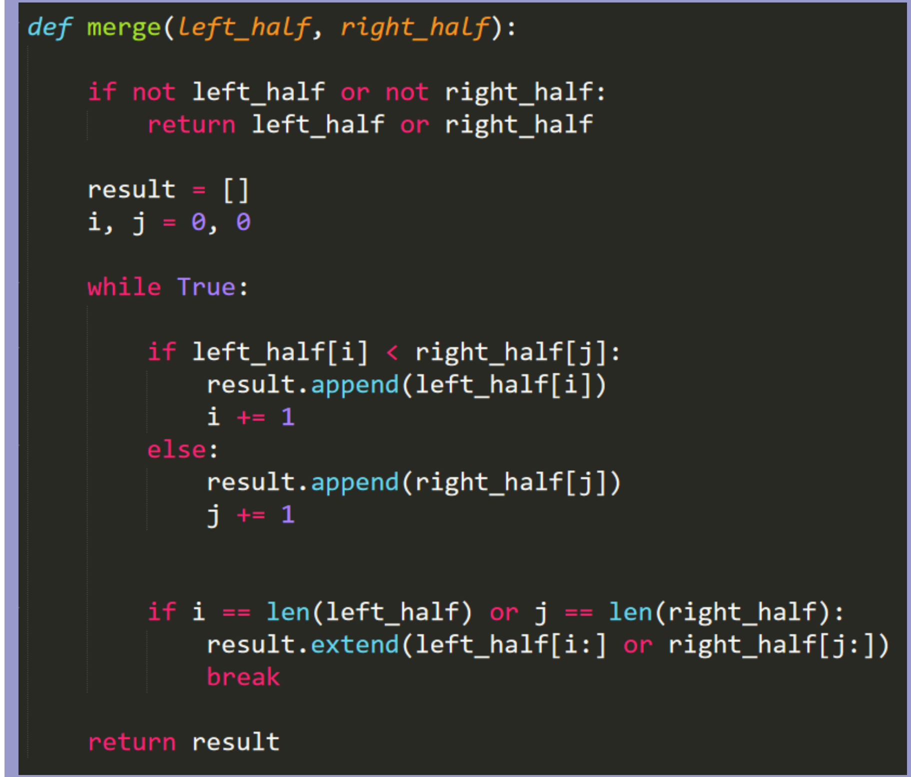
```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 1 j = 1



5 < 4?

result = [1, 3]

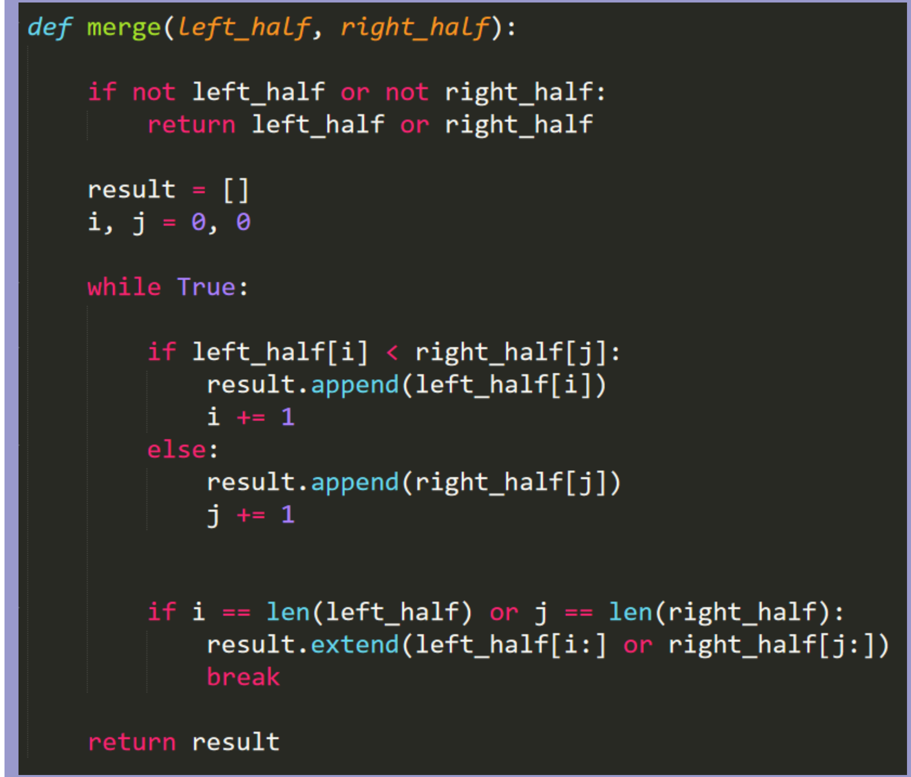


```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 1 j = 2

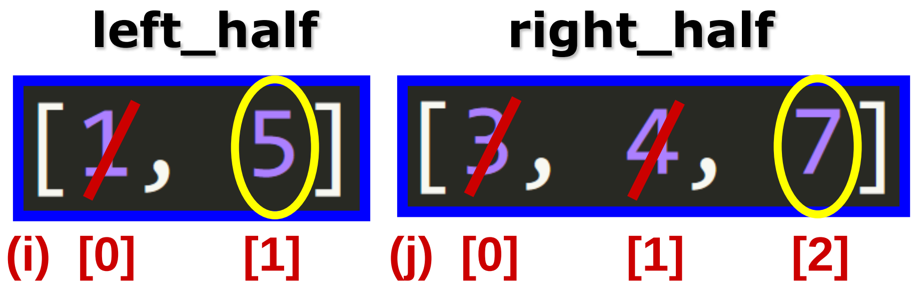
left_half		right_half		
[~~1~~,	5]	[~~3~~,	~~4~~,	7]
(i) [0]	[1]	(j) [0]	[1]	[2]

result = [1, 3, 4]



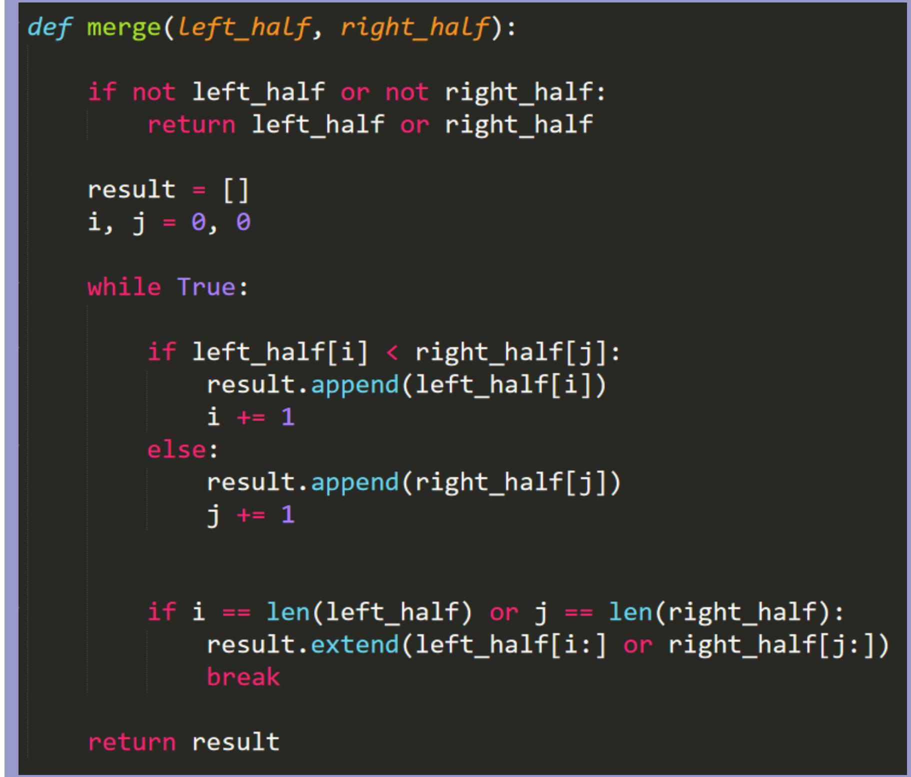
```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 1   j = 2



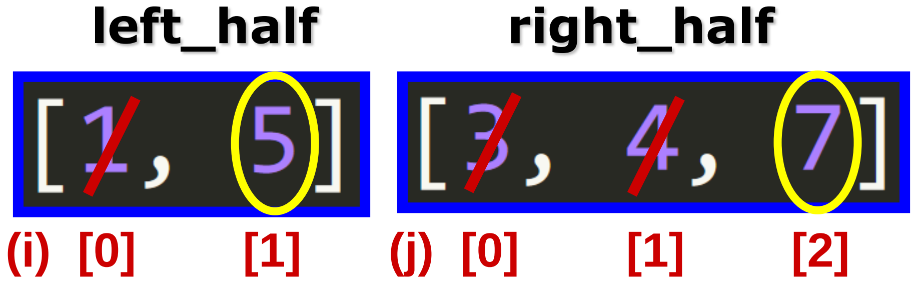
5 < 7?

result = [1, 3, 4]



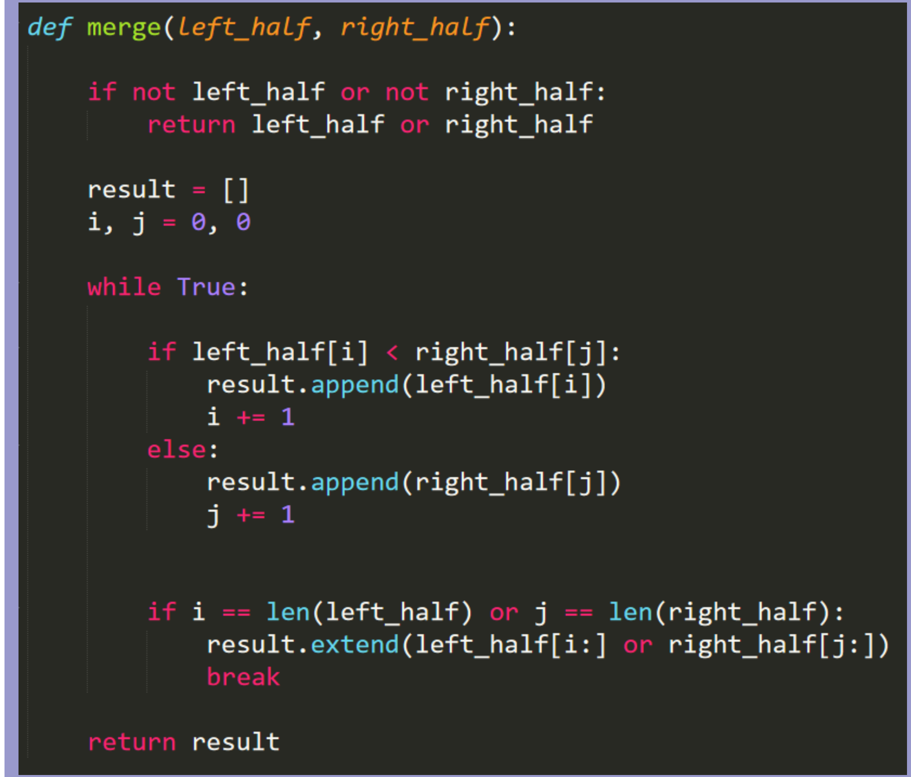
```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 1   j = 2



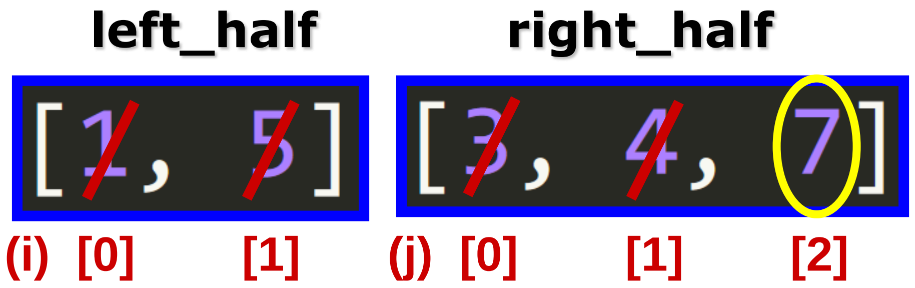
5 < 7?

result = [1, 3, 4, 5]

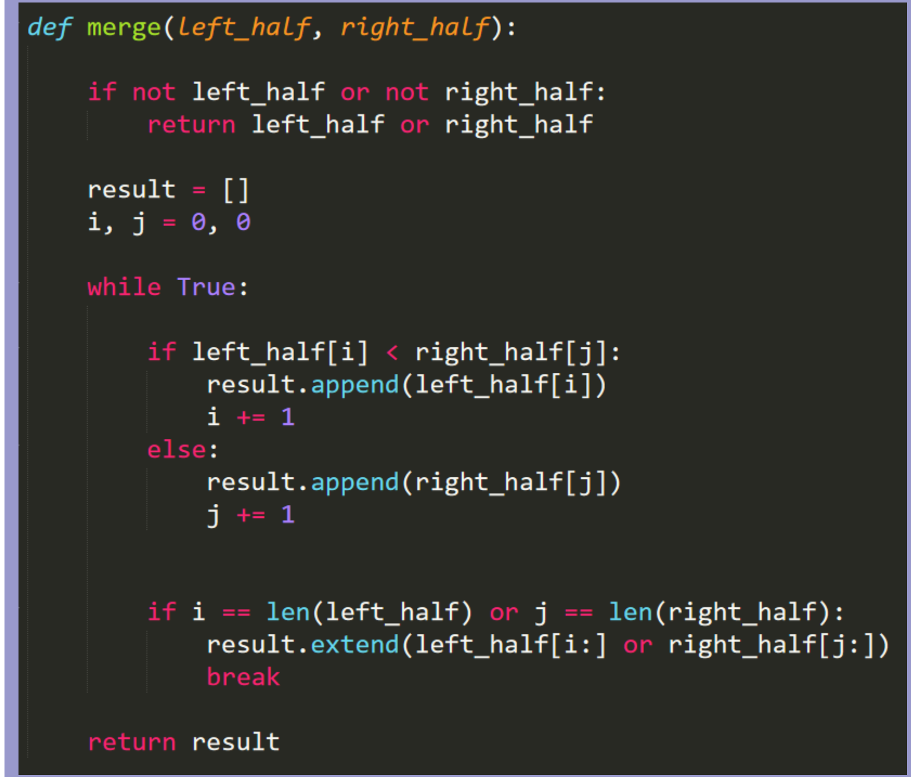


```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 2   j = 2

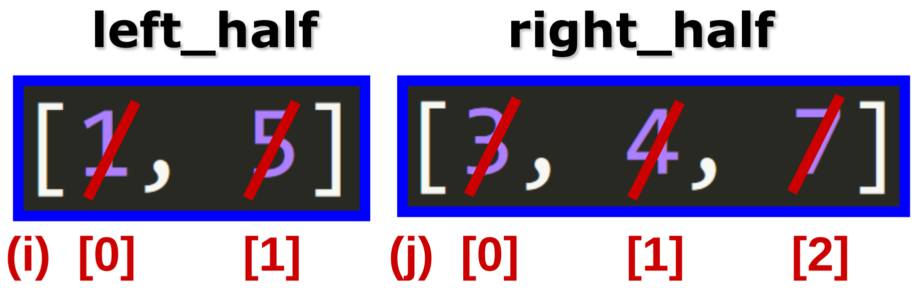


result = [1, 3, 4, 5]

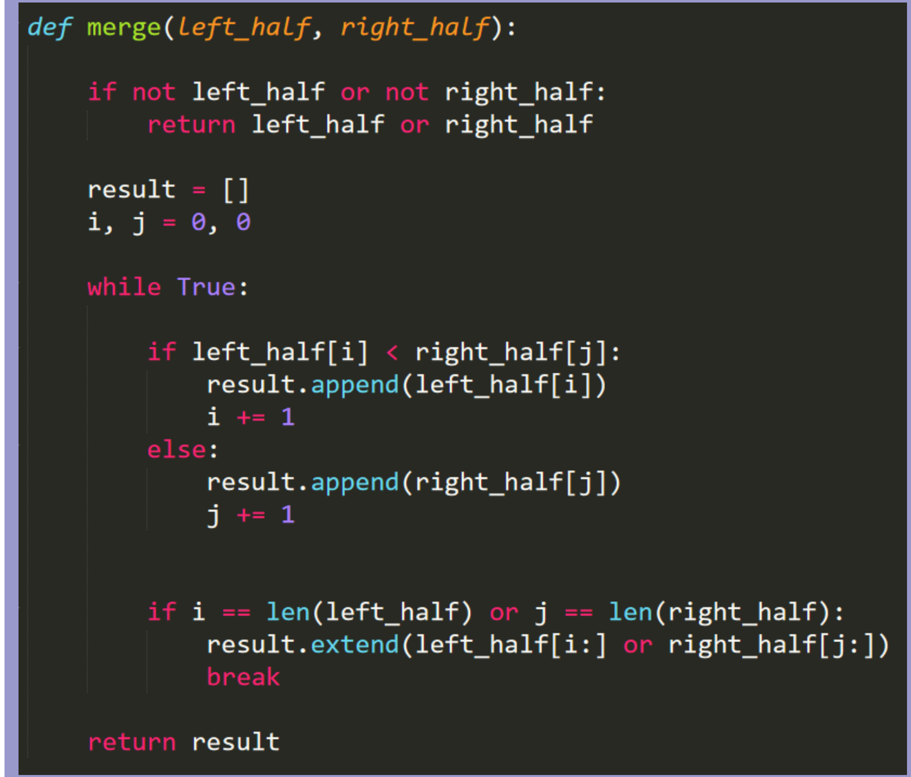


```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 2   j = 2



result = [1, 3, 4, 5, 7]



```python
def merge(left_half, right_half):

    if not left_half or not right_half:
        return left_half or right_half

    result = []
    i, j = 0, 0

    while True:

        if left_half[i] < right_half[j]:
            result.append(left_half[i])
            i += 1
        else:
            result.append(right_half[j])
            j += 1


        if i == len(left_half) or j == len(right_half):
            result.extend(left_half[i:] or right_half[j:])
            break

    return result
```

i = 2   j = 2

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

result = [1, 3, 4, 5, 7]

```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)
```

Sorted

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

Time to Practice