Algorithm

Quicksort

Code Walkthrough

partition()

Quicksort

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

Quicksort

```
[7, 2, 8, 1, 0, 3, 5]
 [0] [1] [2] [3] [4] [5] [6]
```

```python
def quicksort(lst, low, high):
    if low < high:

        pivot_index = partition(lst, low, high)

        quicksort(lst, low, pivot_index-1)
        quicksort(lst, pivot_index+1, high)
```

```
[7, 2, 8, 1, 0, 3, 5]
```

[0] [1] [2] [3] [4] [5] [6]

```python
def quicksort(lst, low, high):
    if low < high:

        pivot_index = partition(lst, low, high)

        quicksort(lst, low, pivot_index-1)
        quicksort(lst, pivot_index+1, high)
```

```
a = [7, 2, 8, 1, 0, 3, 5]
quicksort(a, 0, len(a)-1)
```

```
[7, 2, 8, 1, 0, 3, 5]
```

[0] [1] [2] [3] [4] [5] [6]

```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

    quicksort(lst, low, pivot_index-1)
    quicksort(lst, pivot_index+1, high)
```

```
quicksort(a, 0, len(a)-1)
```

```
[7, 2, 8, 1, 0, 3, 5]
```

[0] [1] [2] [3] [4] [5] [6]

```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

    quicksort(lst, low, pivot_index-1)
    quicksort(lst, pivot_index+1, high)
```

```
quicksort(a, 0, len(a)-1)
```

```
[7, 2, 8, 1, 0, 3, 5]
```

[0] [1] [2] [3] [4] [5] [6]

```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

    quicksort(lst, low, pivot_index-1)
    quicksort(lst, pivot_index+1, high)
```

```
quicksort(a, 0, len(a)-1)
```

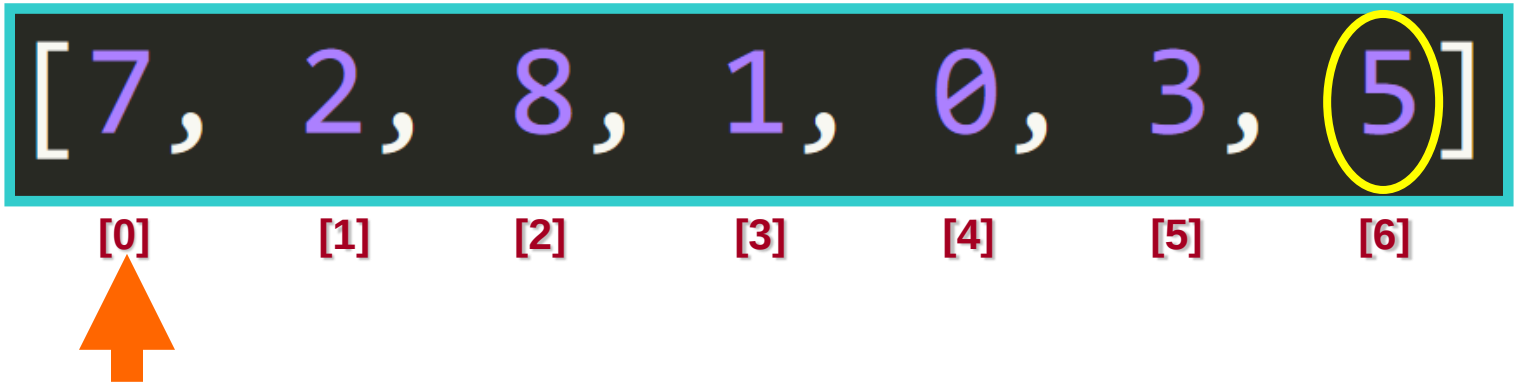

```
partition(lst, 0, 6)
```

pivot = 5

i = -1

j = 0

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 5

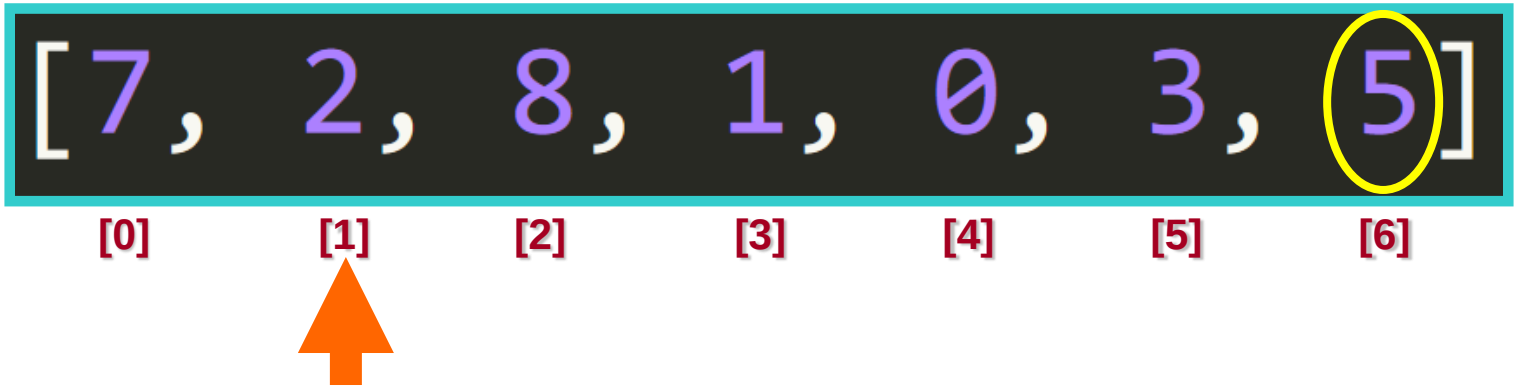
i = -1

j = 0

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

7 <= 5?

No!



pivot = 5

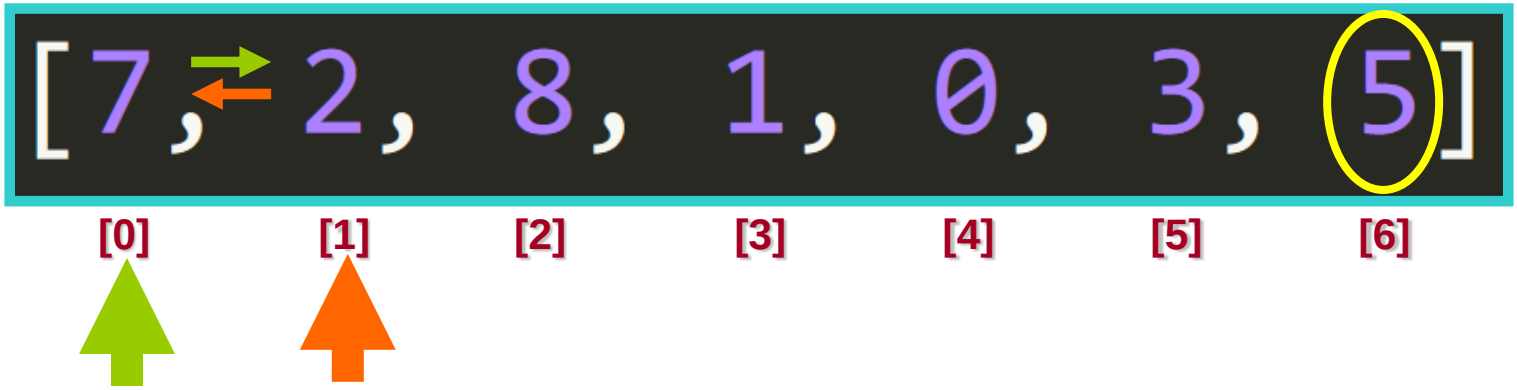
i = -1

j = 1

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

2 <= 5?

Yes!



pivot = 5

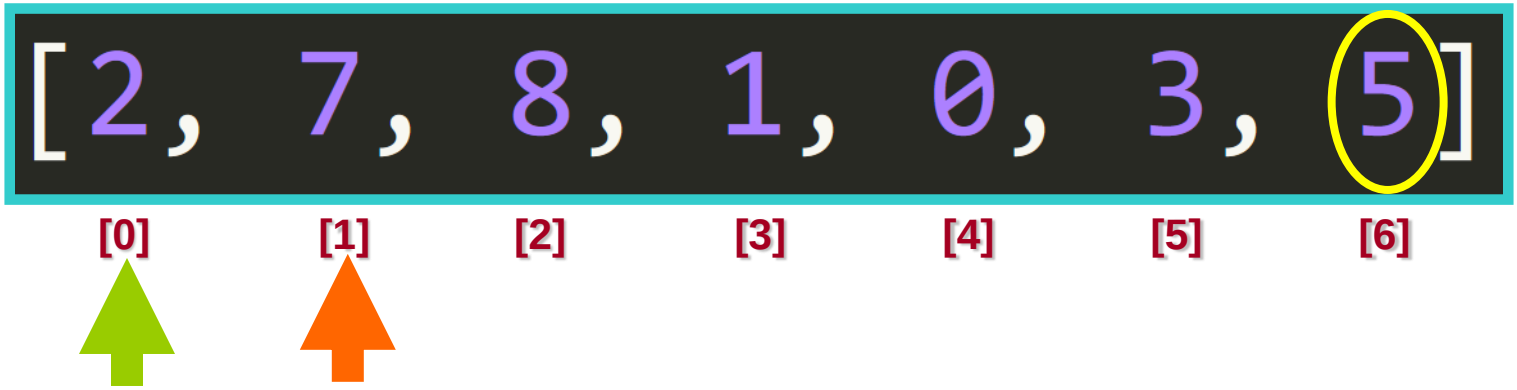
i = 0

j = 1

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

2 <= 5?

Yes!

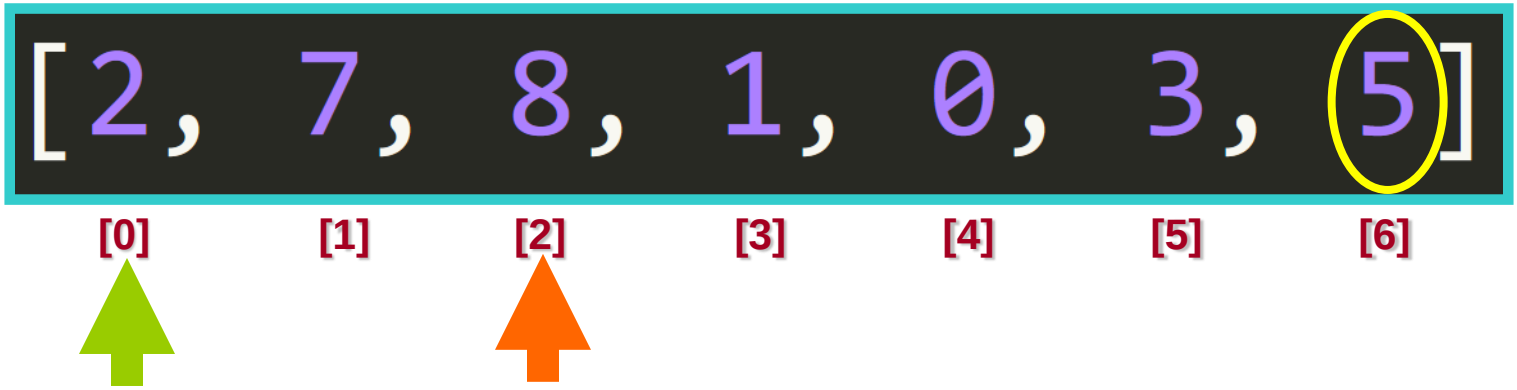


pivot = 5

i = 0

j = 1

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

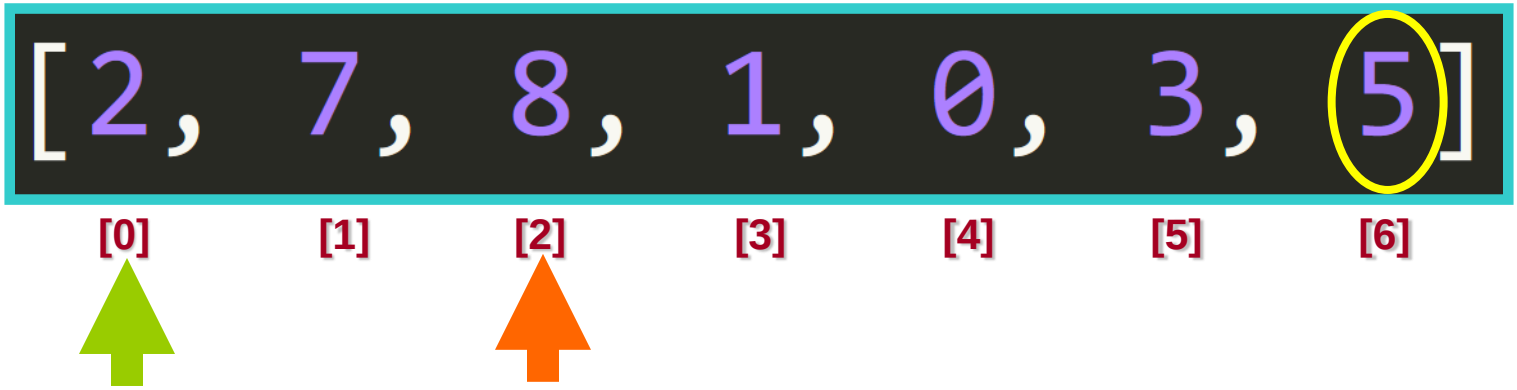


pivot = 5

i = 0

j = 2

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 5

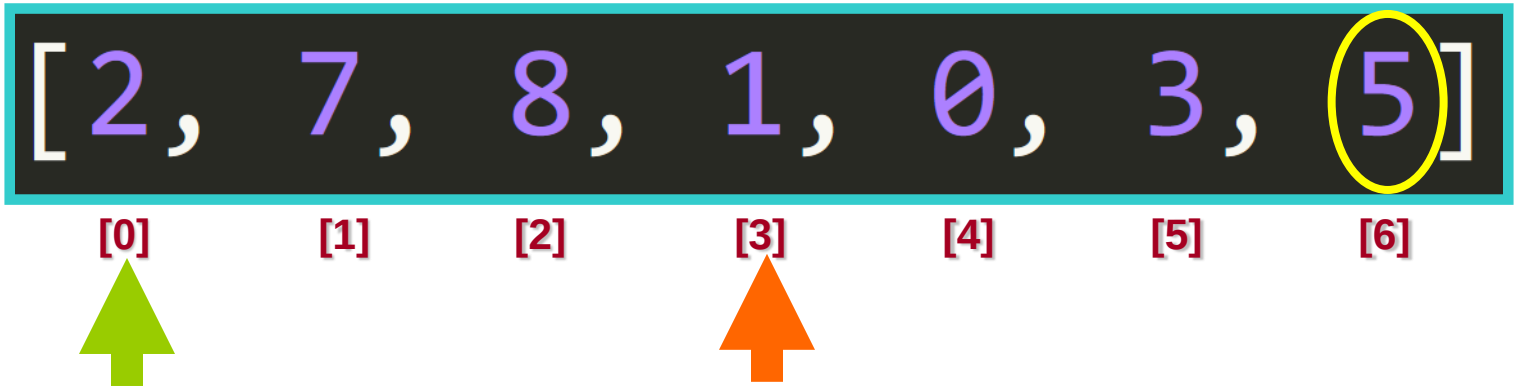
i = 0

j = 2

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

8 <= 5?

No!



pivot = 5

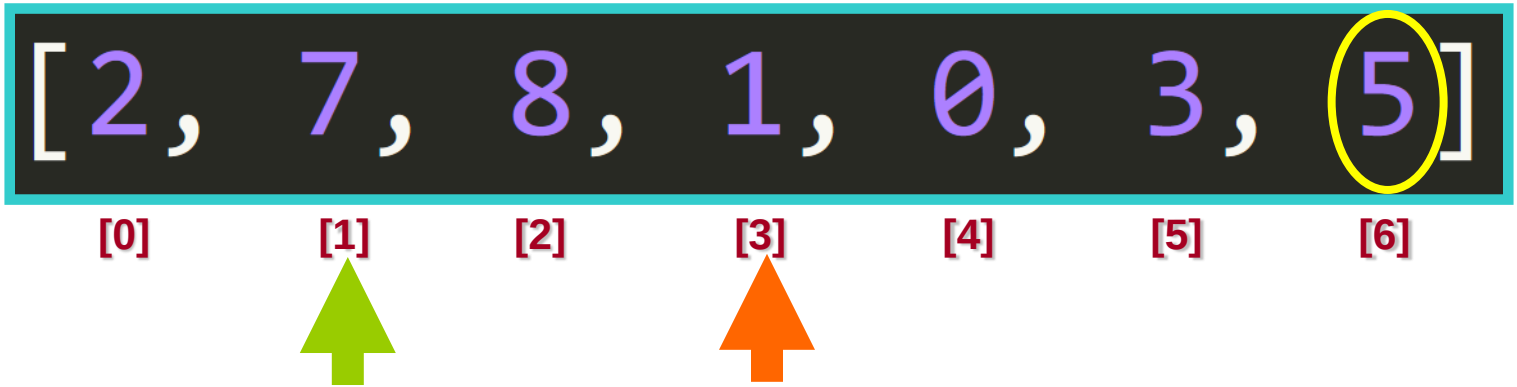
i = 0

j = 3

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

1 <= 5?

Yes!

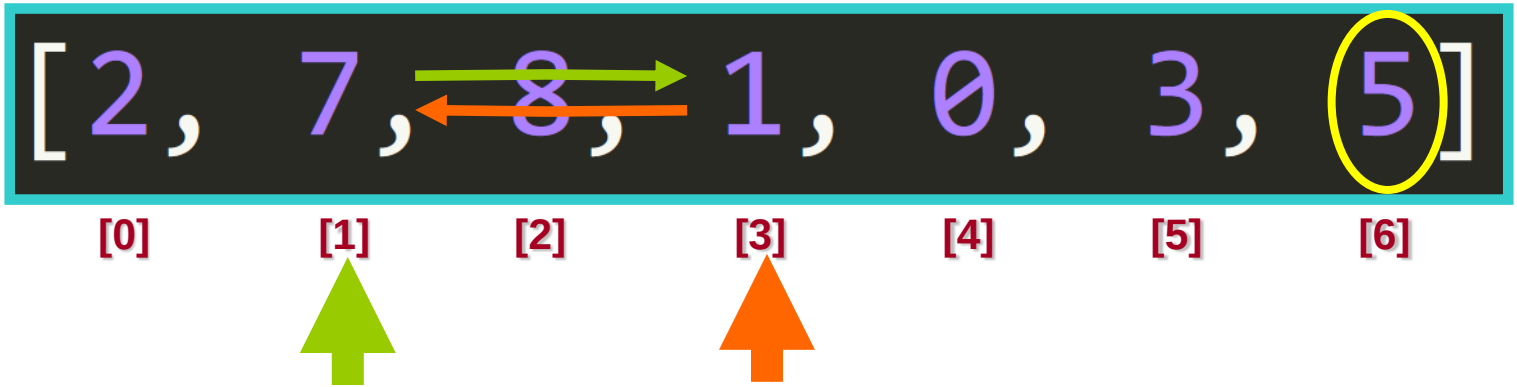


pivot = 5

i = 1

j = 3

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

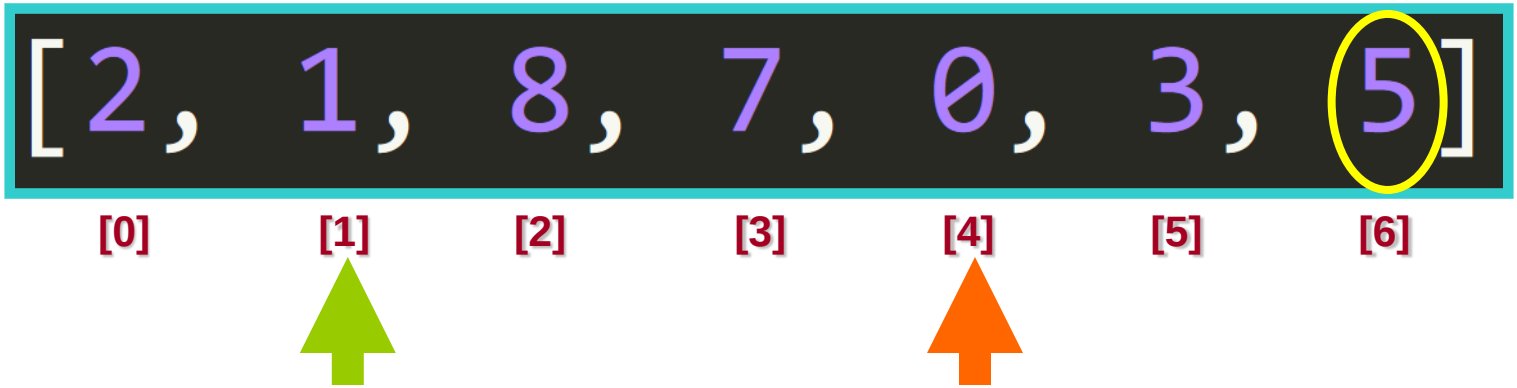


pivot = 5

i = 1

j = 3

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 5

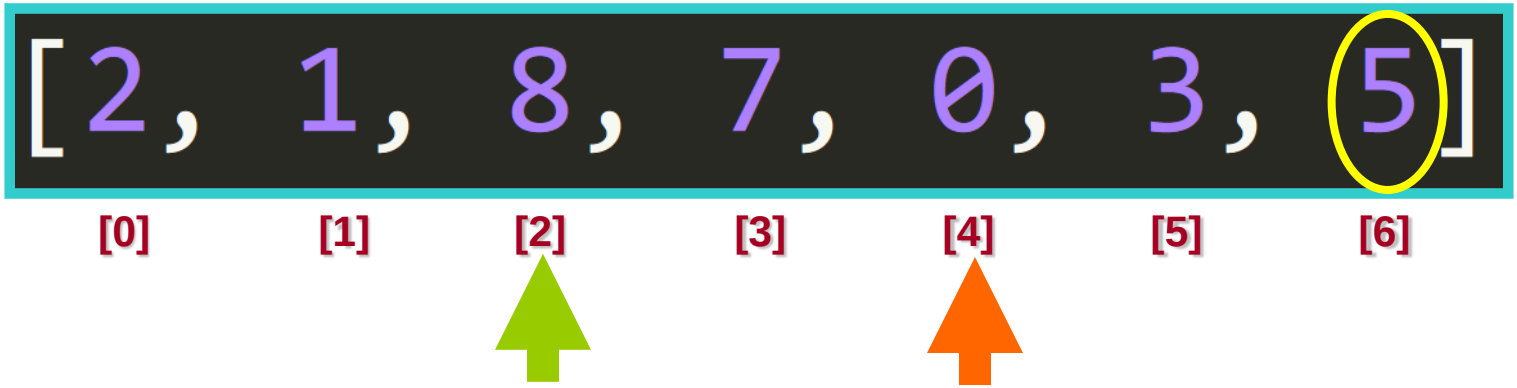
i = 1

j = 4

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

0 <= 5?

Yes!

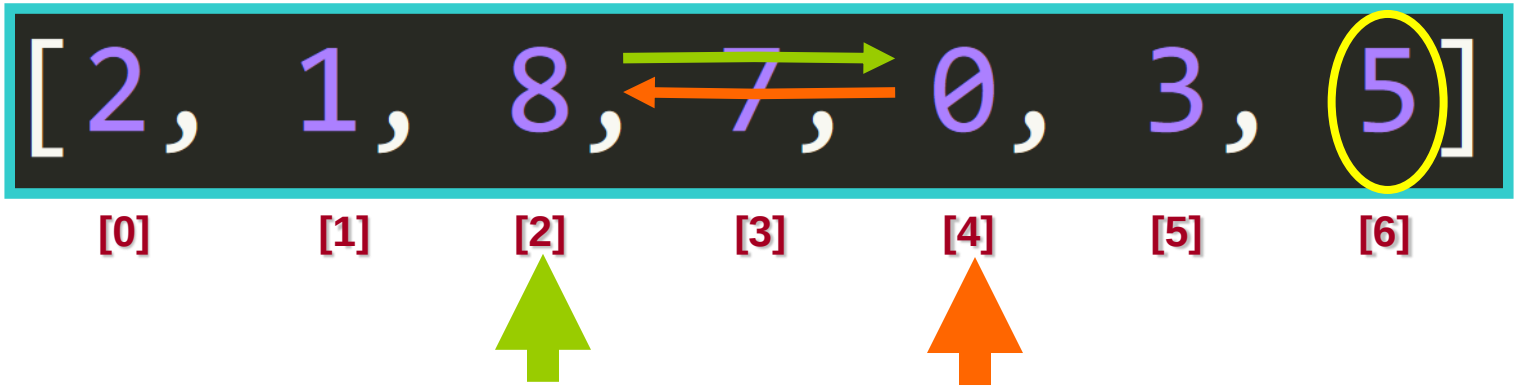


pivot = 5

i = 2

j = 4

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

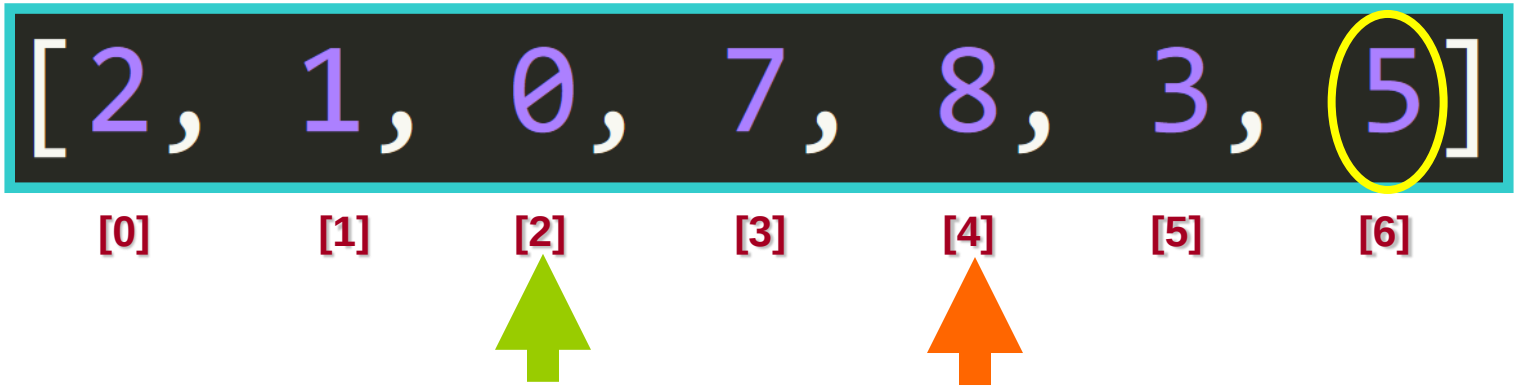


pivot = 5

i = 2

j = 4

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

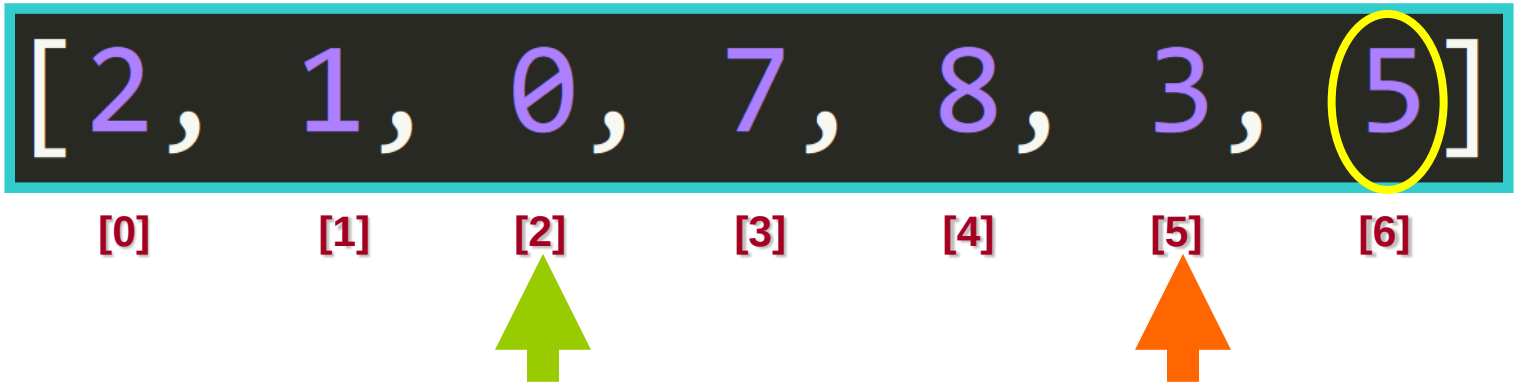


pivot = 5

i = 2

j = 4

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 5

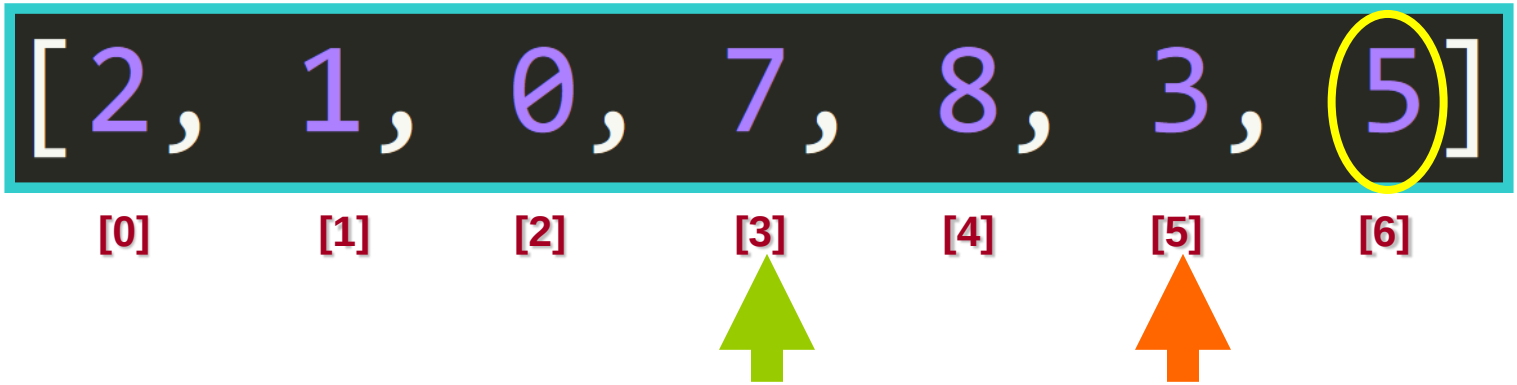
i = 2

j = 5

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

3 <= 5?

Yes!

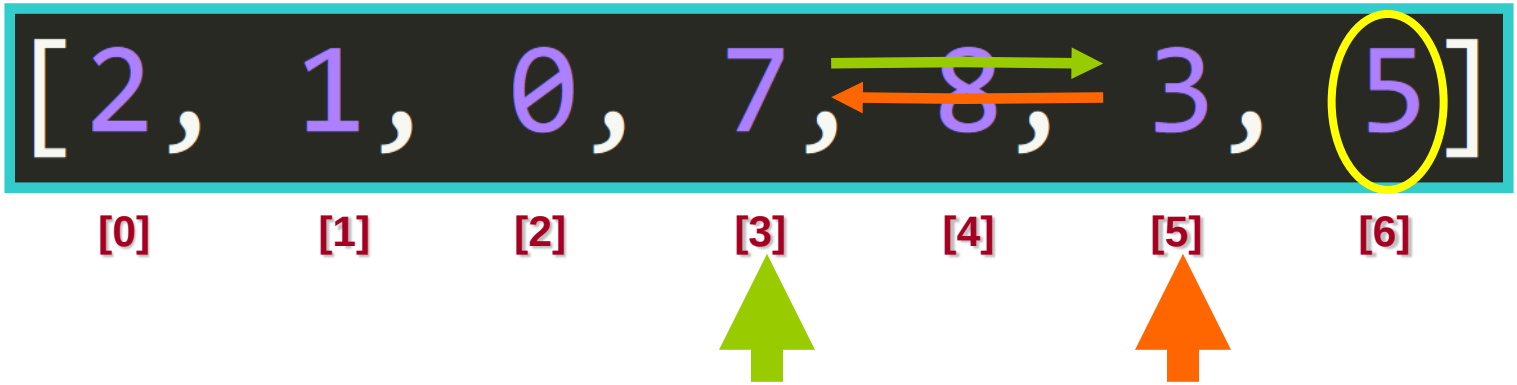


pivot = 5

i = 3

j = 5

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

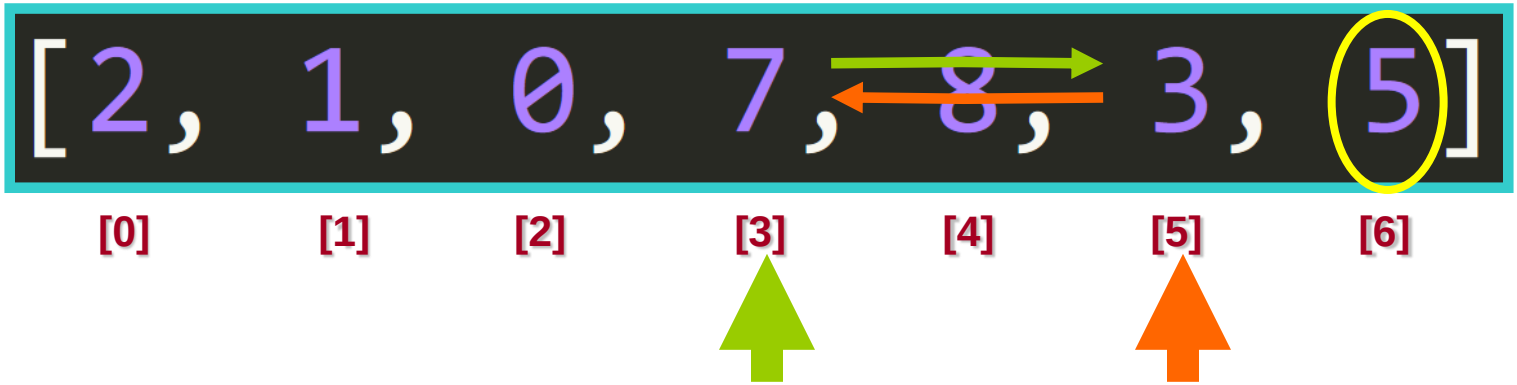


pivot = 5

i = 3

j = 5

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

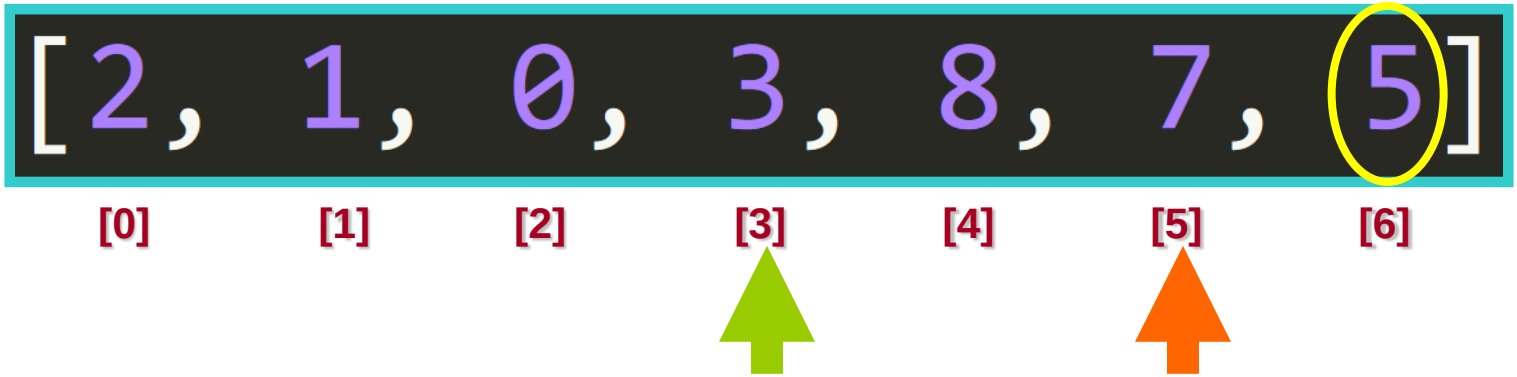


pivot = 5

i = 3

j = 5

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

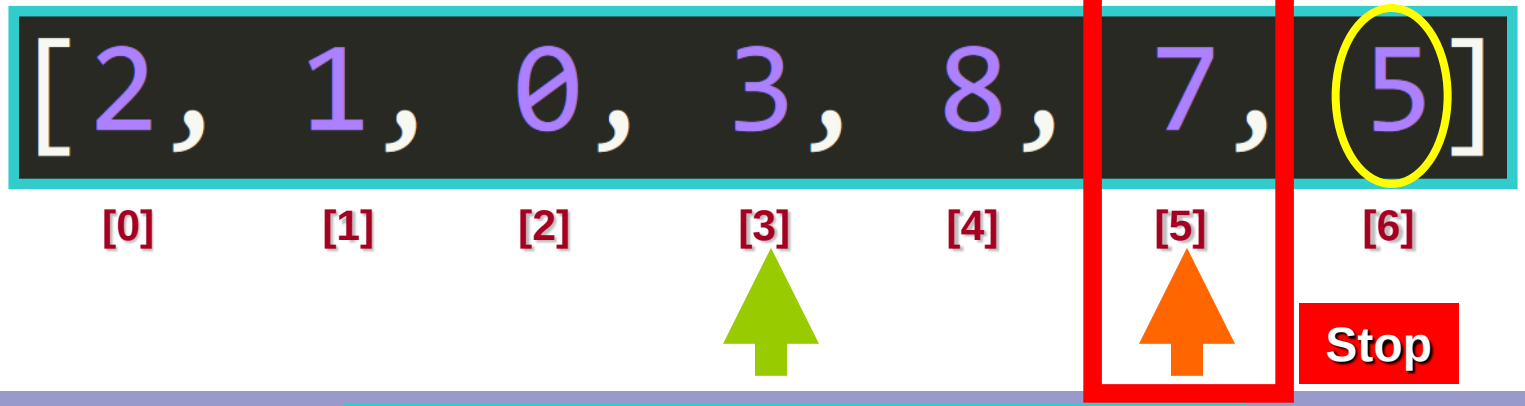


pivot = 5

i = 3

j = 5

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

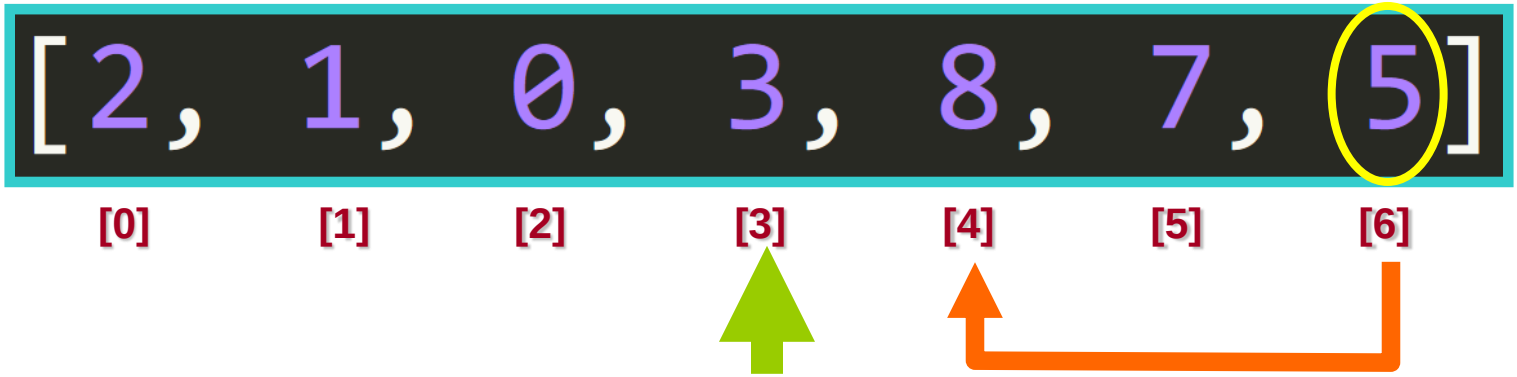


pivot = 5

i = 3

j = 5

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

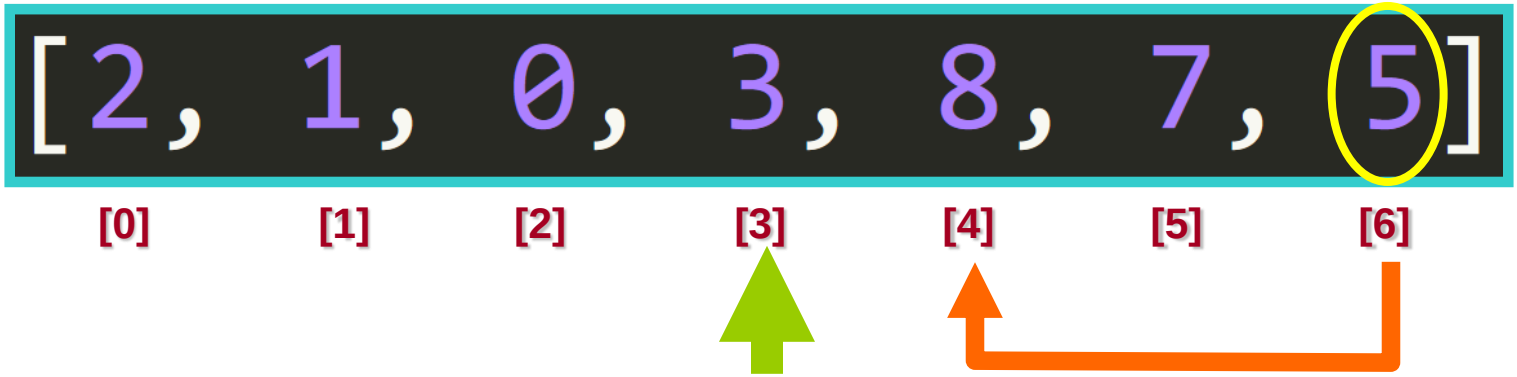


pivot = 5

i = 3

j = 5

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

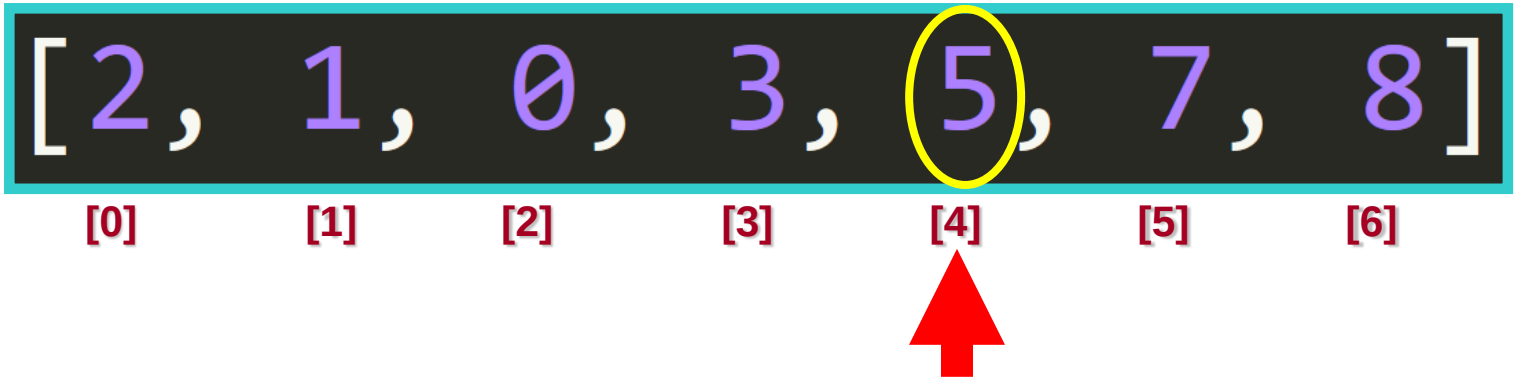


pivot = 5

i = 3

j = 5

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 5

i = 3

j = 5

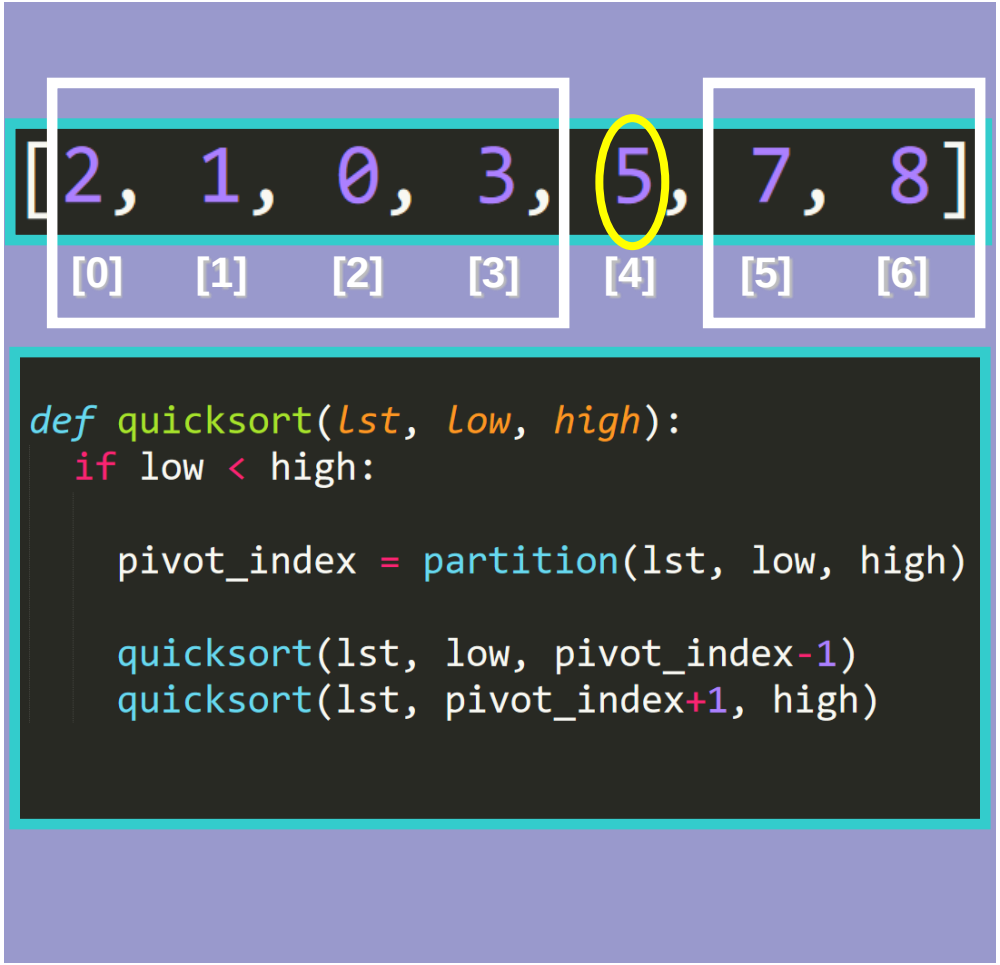
```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

return 4

```
quicksort(a, 0, len(a)-1)
```

```
[2, 1, 0, 3, 5, 7, 8]
 [0] [1] [2] [3] [4] [5] [6]
```

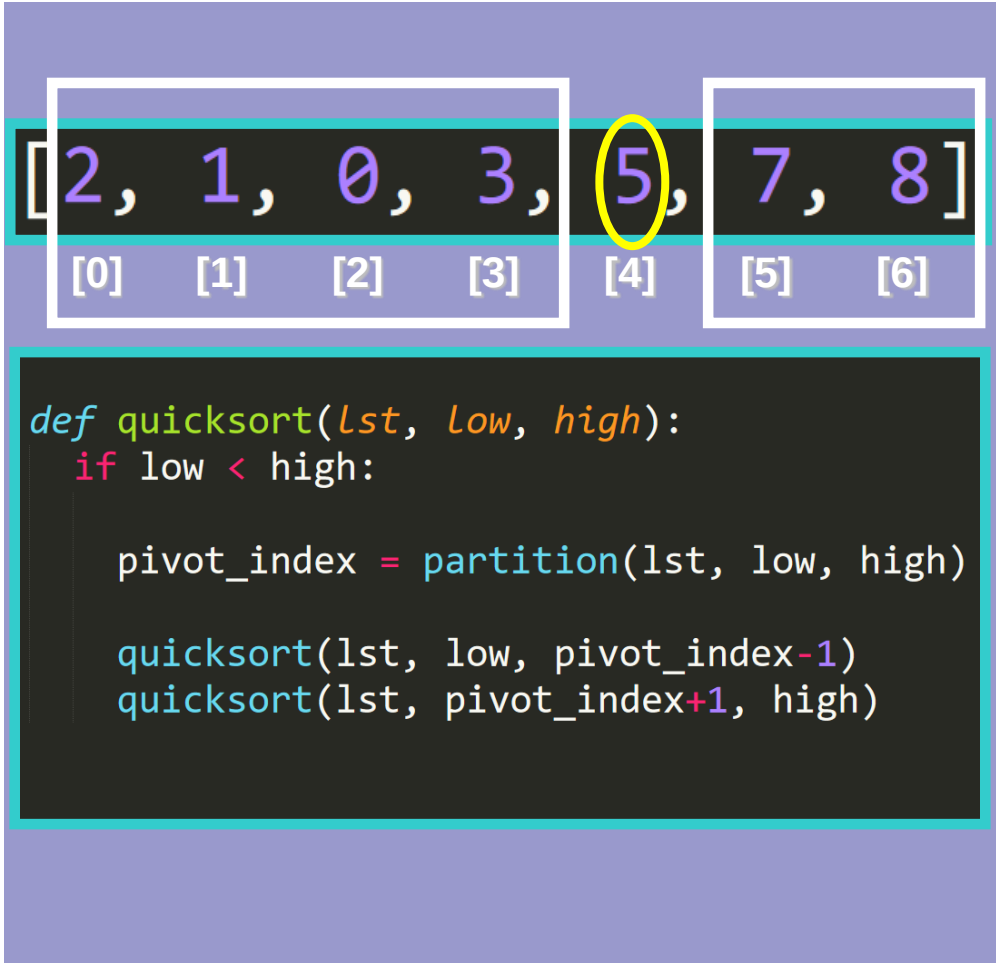
```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

    quicksort(lst, low, pivot_index-1)
    quicksort(lst, pivot_index+1, high)
```




```
quicksort(a, 0, len(a)-1)
```

↓

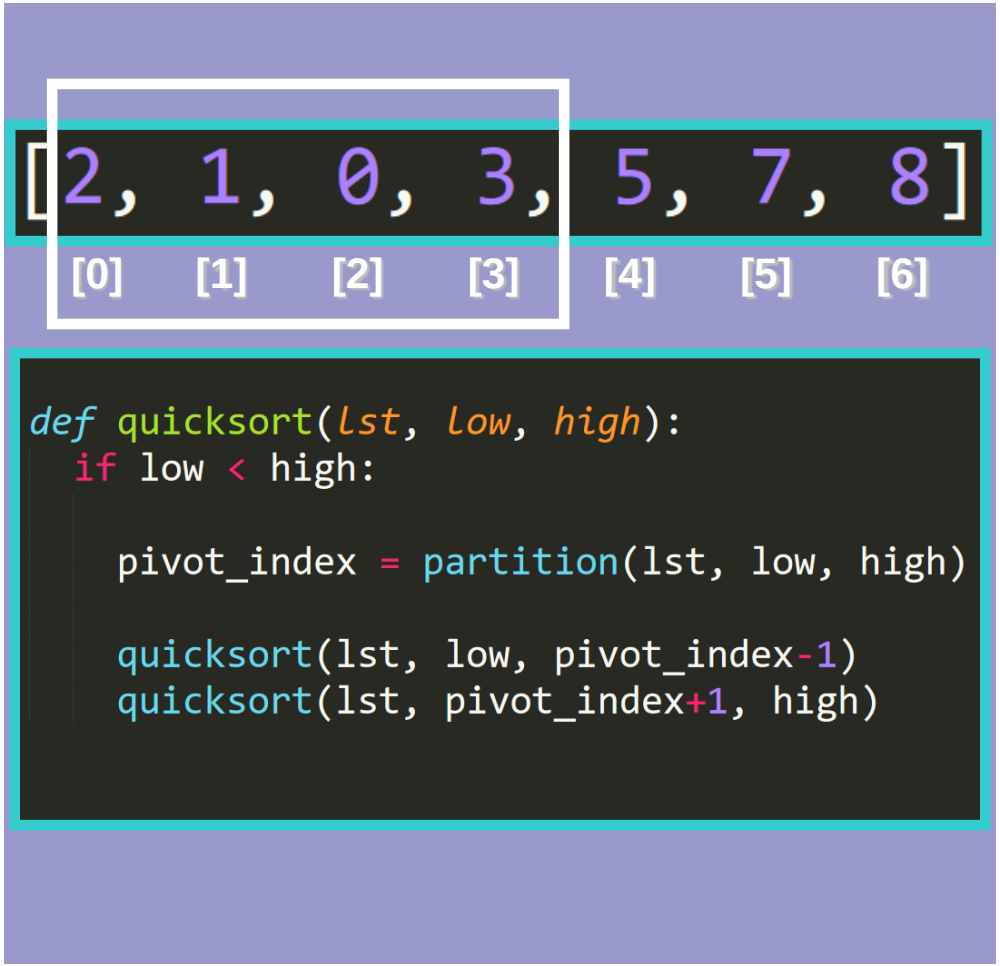
```
quicksort(lst, 0, 3)
```




```
quicksort(a, 0, len(a)-1)
```

<waiting recursive call>

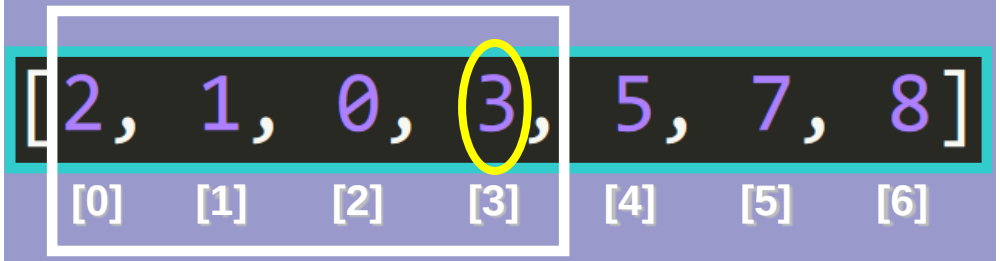
```
quicksort(lst, 0, 3)
```




```
quicksort(a, 0, len(a)-1)
```

<waiting recursive call>

```
quicksort(lst, 0, 3)
```

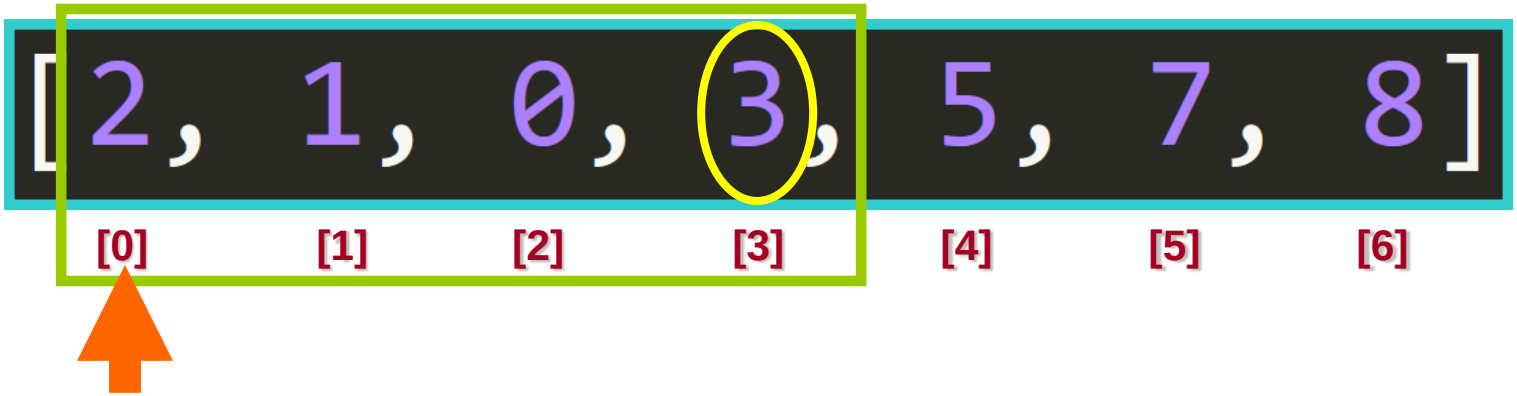



```python
def quicksort(lst, low, high):
    if low < high:

        pivot_index = partition(lst, low, high)

        quicksort(lst, low, pivot_index-1)
        quicksort(lst, pivot_index+1, high)
```

```
quicksort(a, 0, len(a)-1)
```

<waiting recursive call>

```
quicksort(lst, 0, 3)
```

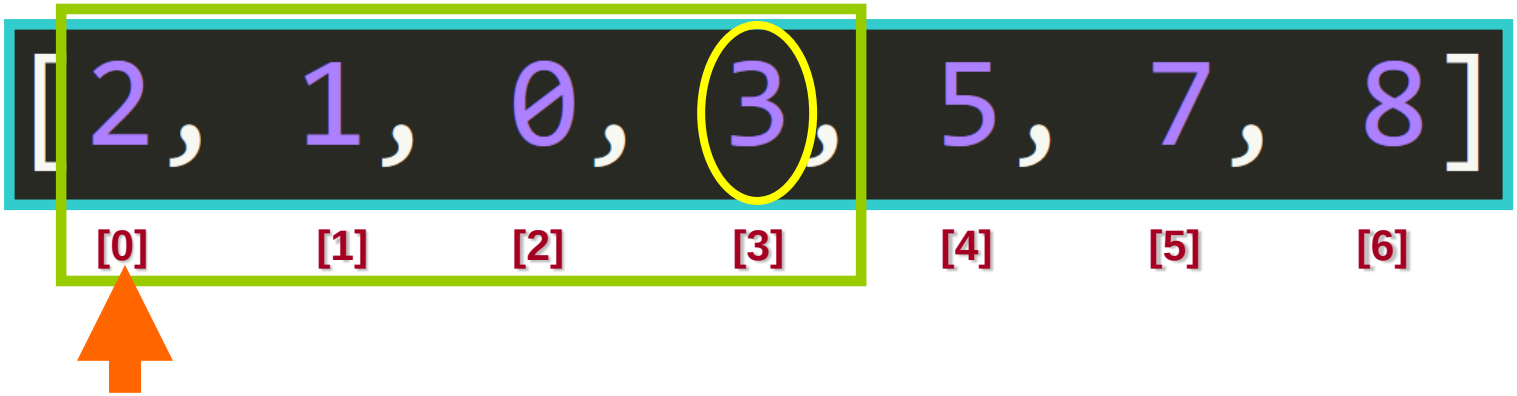


pivot = 3

i = -1

j = 0

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 3

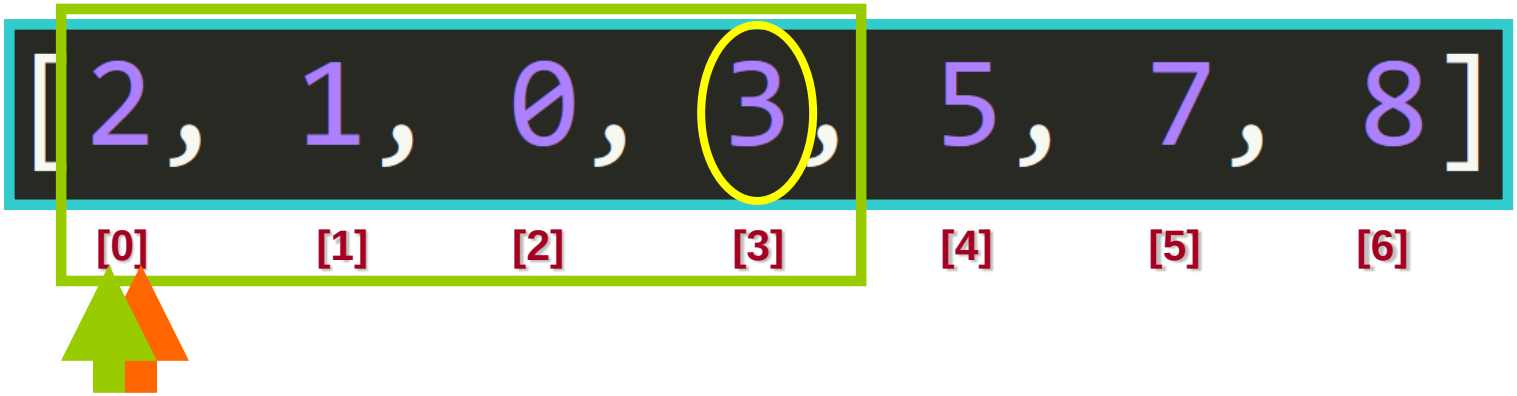
i = -1

j = 0

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

2 <= 3?

Yes!

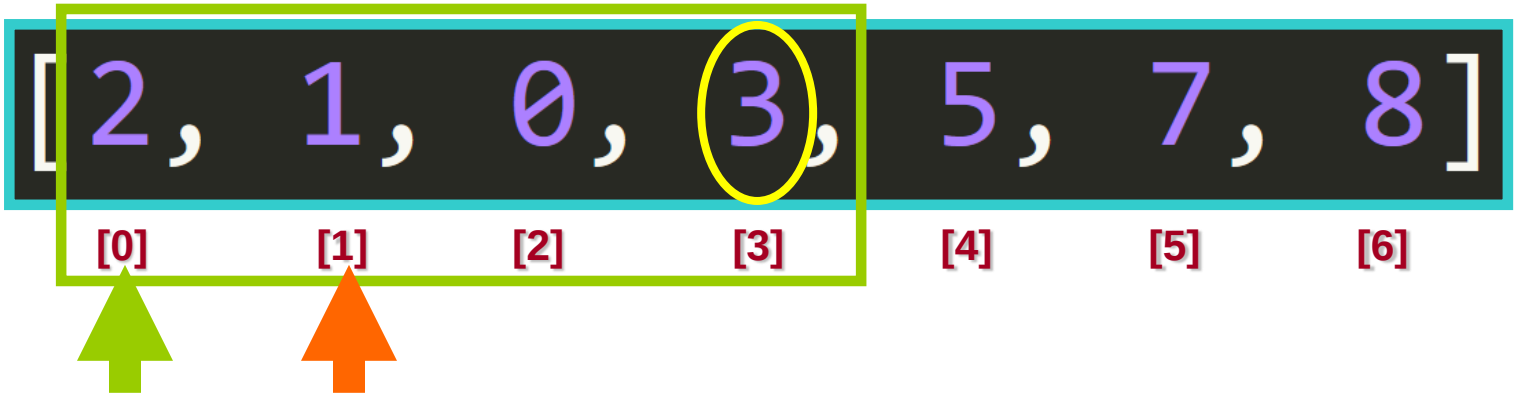


pivot = 3

i = 0

j = 0

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 3

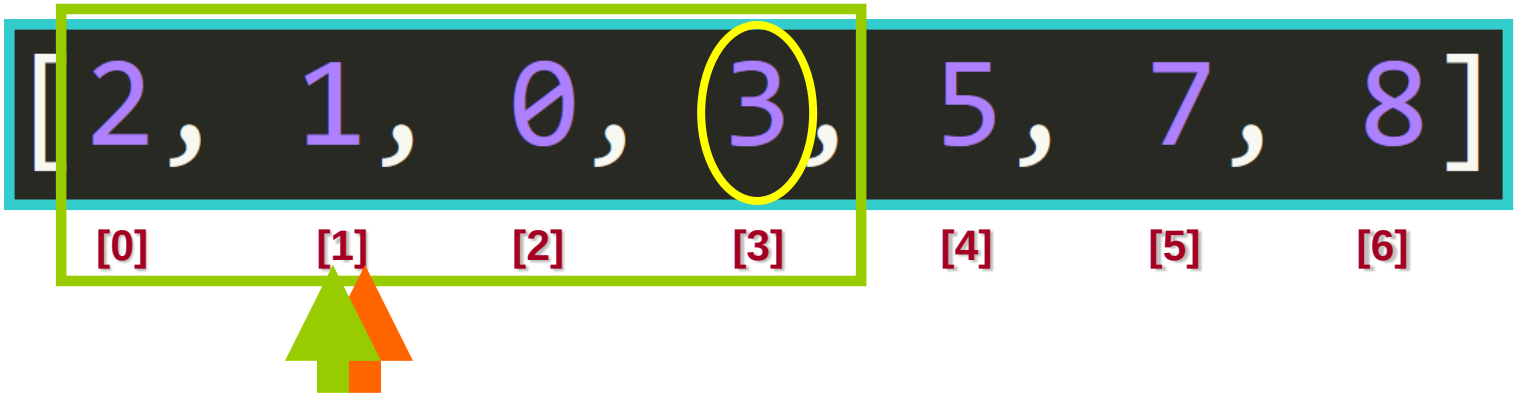
i = 0

j = 1

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

1 <= 3?

Yes!

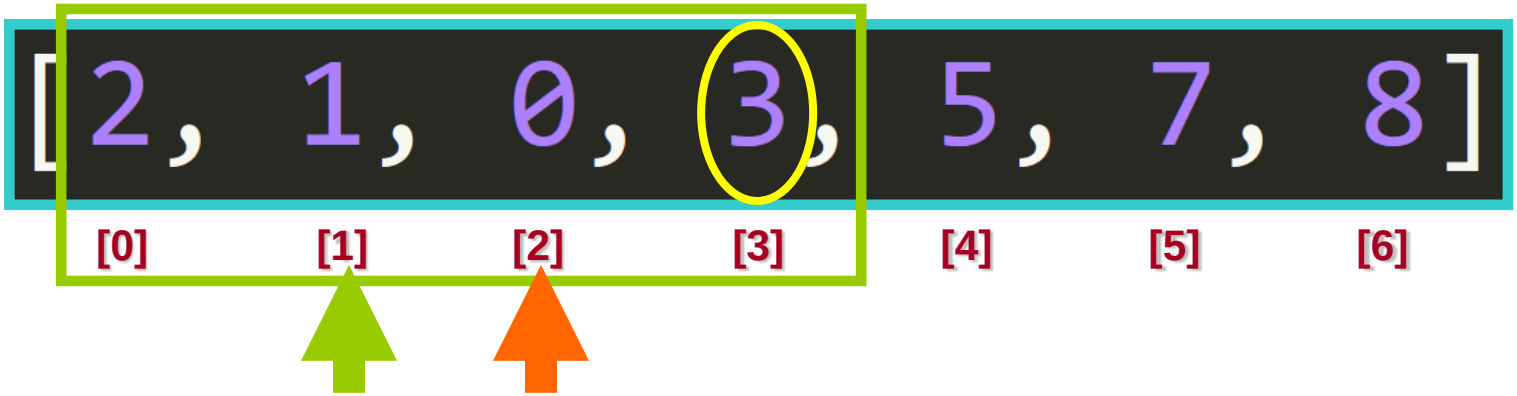


pivot = 3

i = 1

j = 1

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 3

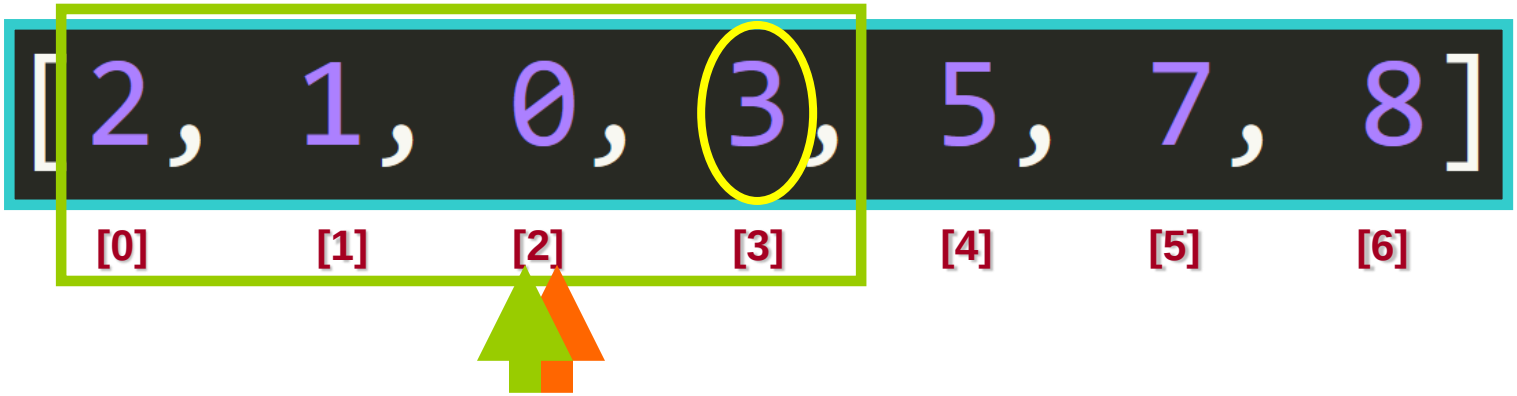
i = 1

j = 2

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

0 <= 3?

Yes!



pivot = 3

i = 1

j = 2

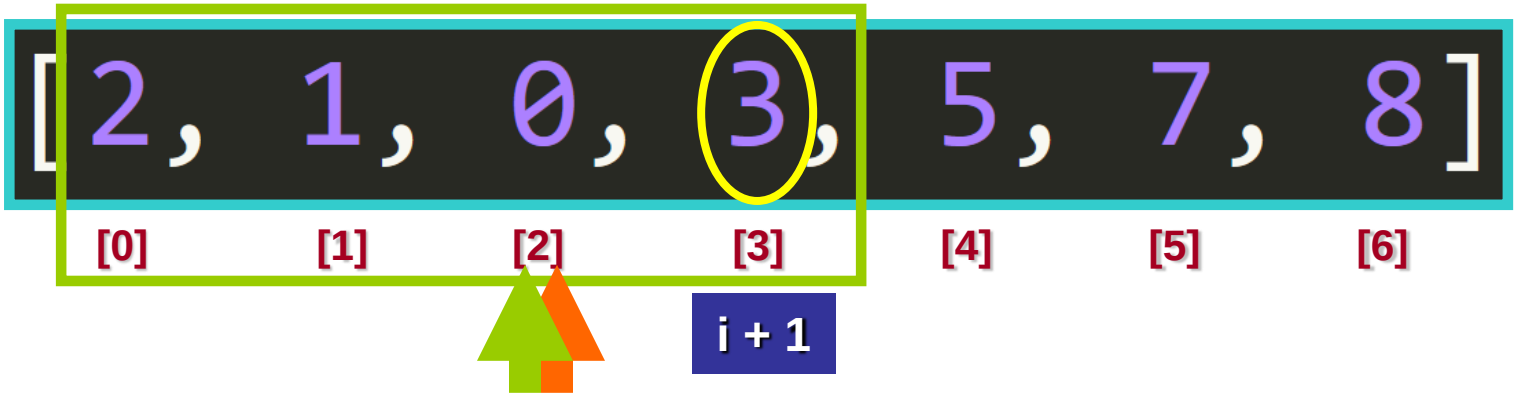
```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

pivot = 3

i = 1

j = 2

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

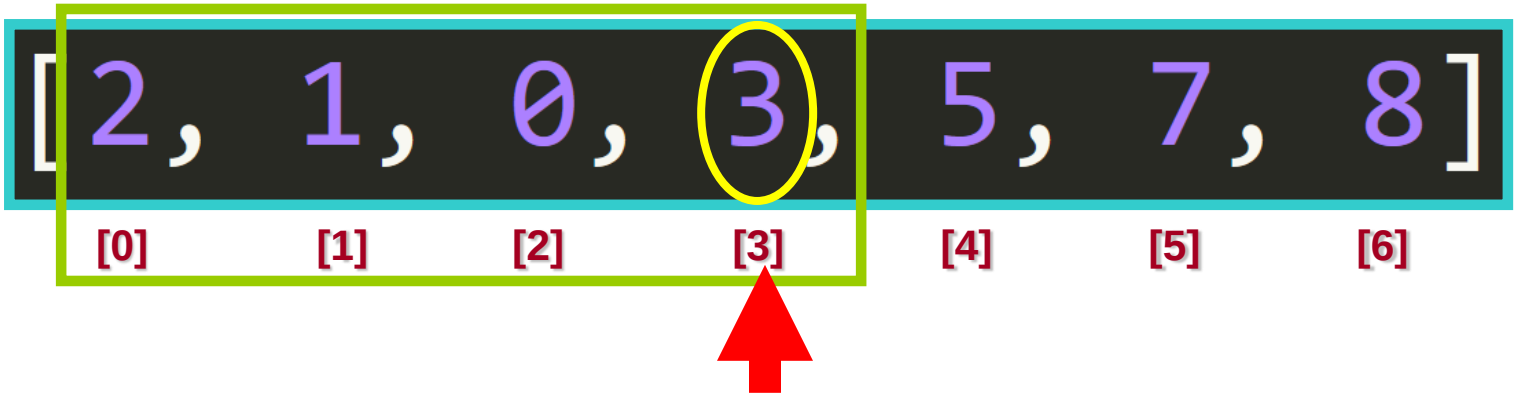


pivot = 3

i = 1

j = 2

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



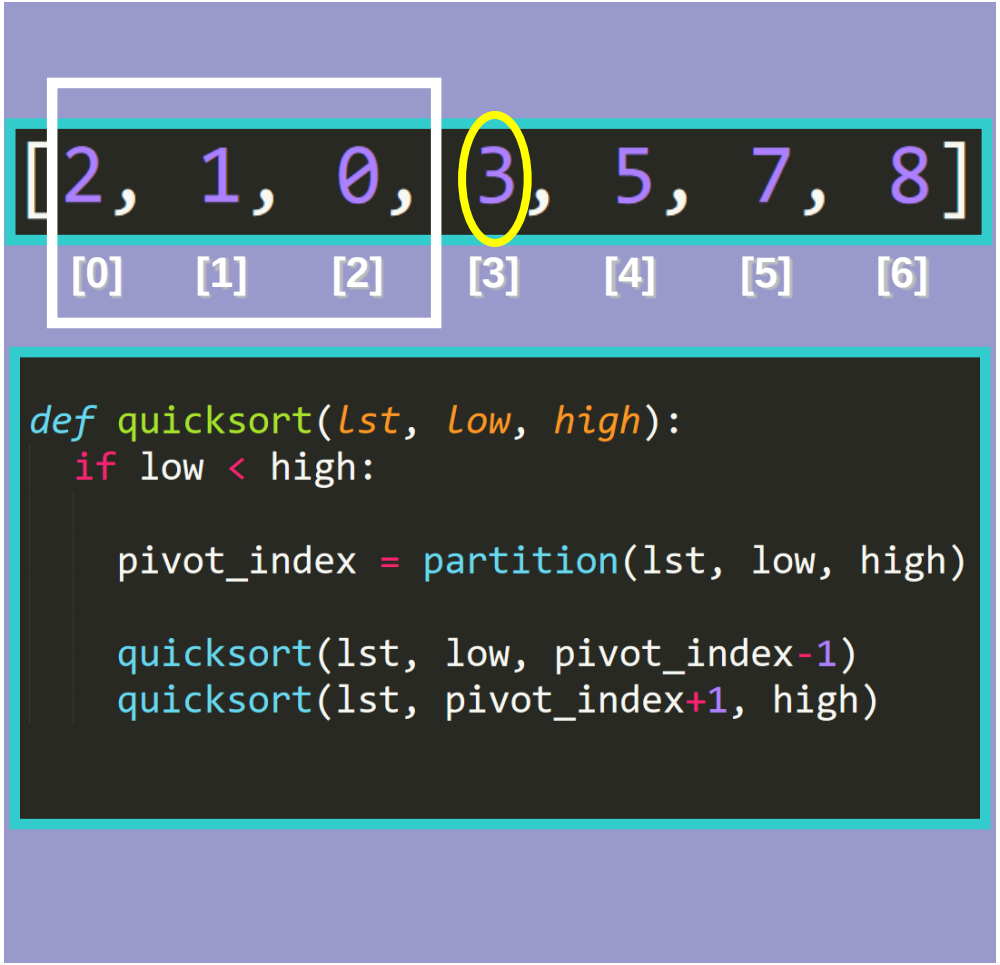
pivot = 3

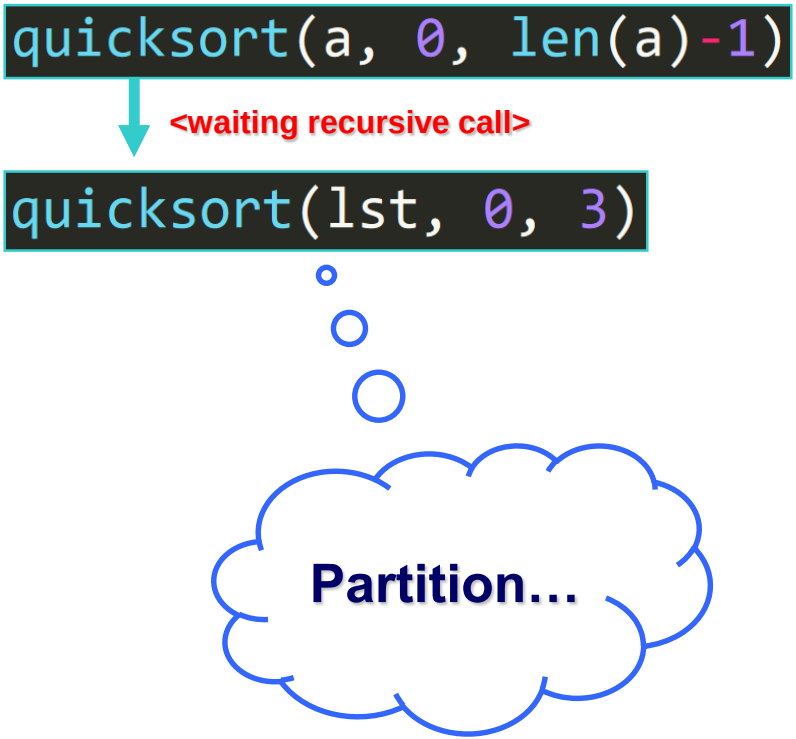
i = 1

j = 2

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

return 3


[2, 1, 0, 3, 5, 7, 8]
[0] [1] [2] [3] [4] [5] [6]
def quicksort(lst, low, high):
if low < high:
pivot_index = partition(lst, low, high)
quicksort(lst, low, pivot_index-1)
quicksort(lst, pivot_index+1, high)


quicksort(a, 0, len(a)-1)
<waiting recursive call>
quicksort(lst, 0, 3)
Partition…


```python
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

    quicksort(lst, low, pivot_index-1)
    quicksort(lst, pivot_index+1, high)
```

```
quicksort(a, 0, len(a)-1)
```

<waiting recursive call>

```
quicksort(lst, 0, 3)
```

<waiting recursive call>

```
quicksort(lst, 0, 2)
```


```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

    quicksort(lst, low, pivot_index-1)
    quicksort(lst, pivot_index+1, high)
```

```
quicksort(a, 0, len(a)-1)
```

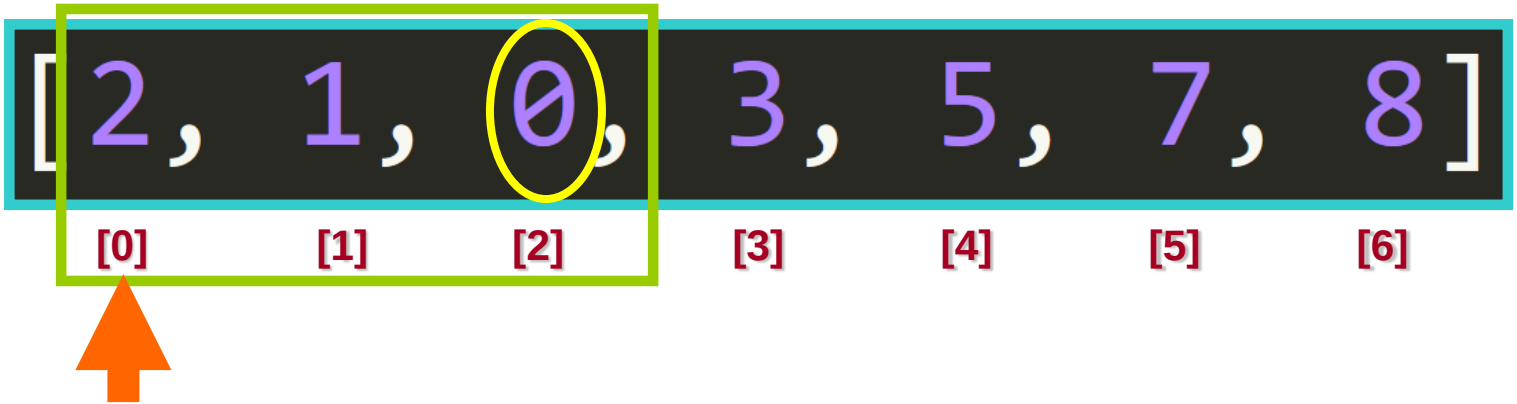
<waiting recursive call>

```
quicksort(lst, 0, 3)
```

<waiting recursive call>

```
quicksort(lst, 0, 2)
```

Partition…

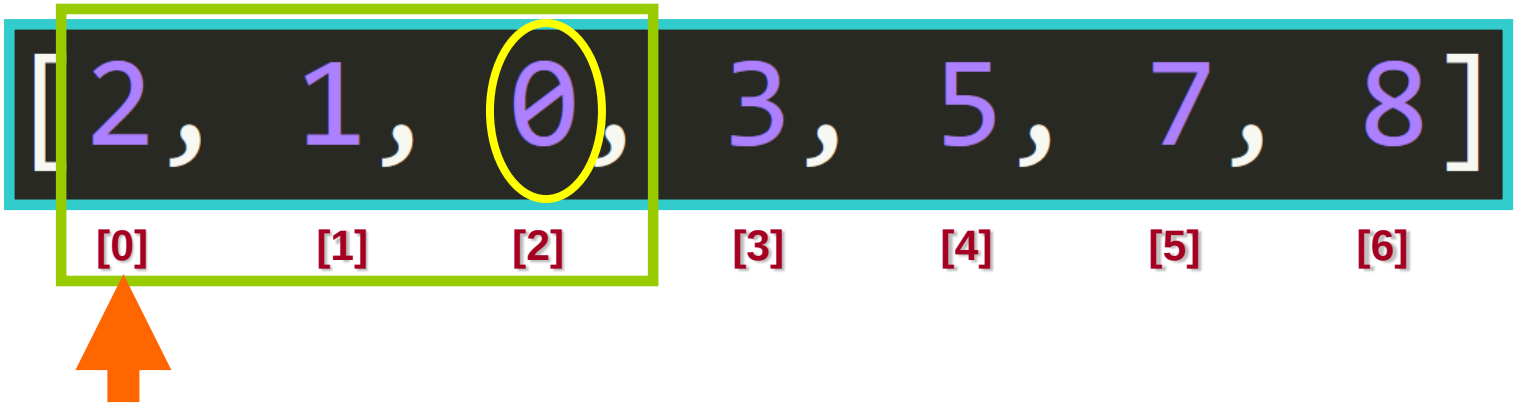


pivot = 0

i = -1

j = 0

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 0

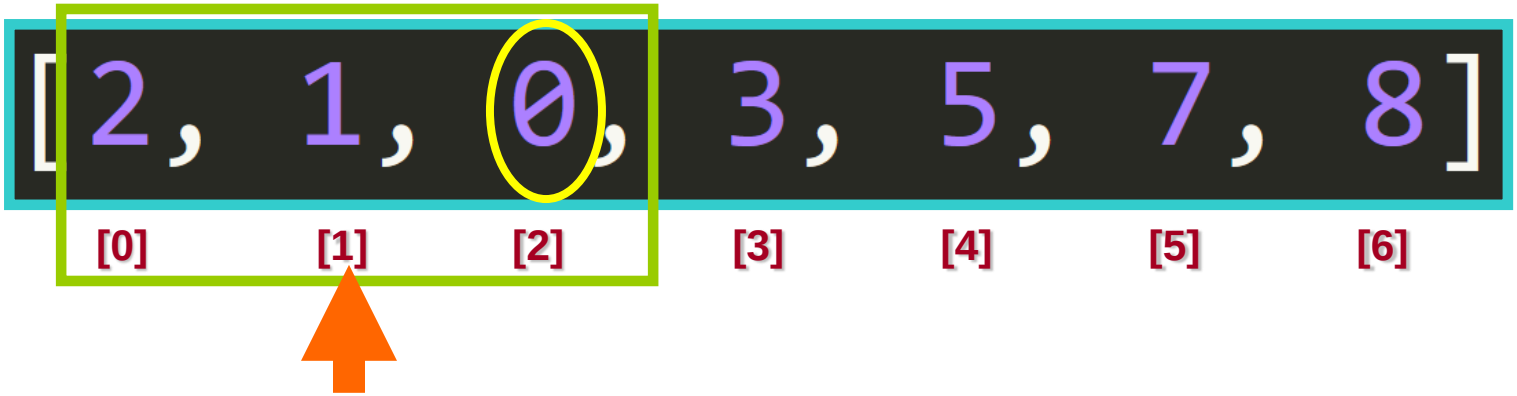
i = -1

j = 0

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

2 <= 0?

No!

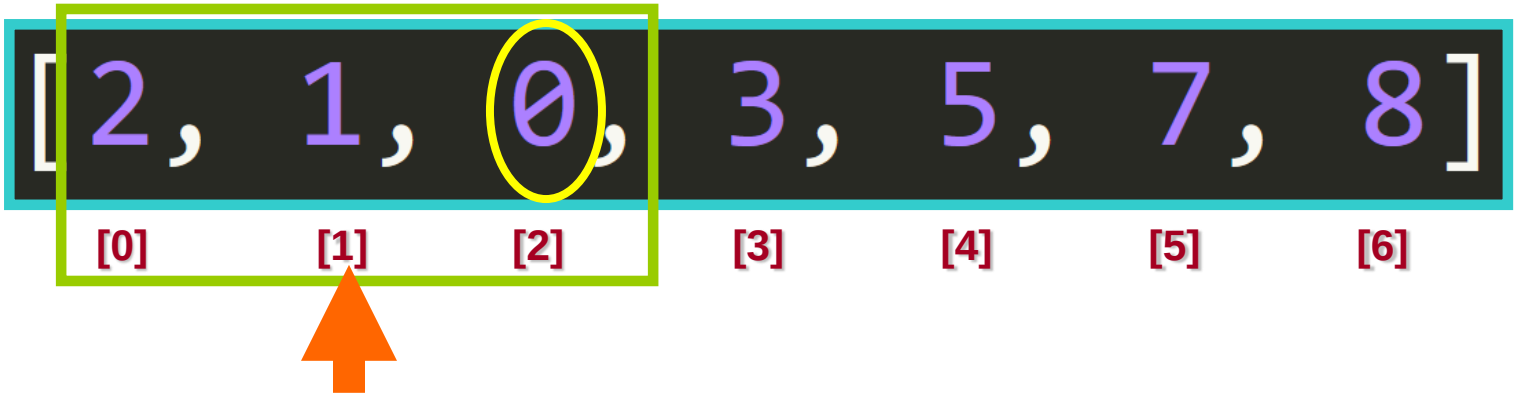


pivot = 0

i = -1

j = 1

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 0

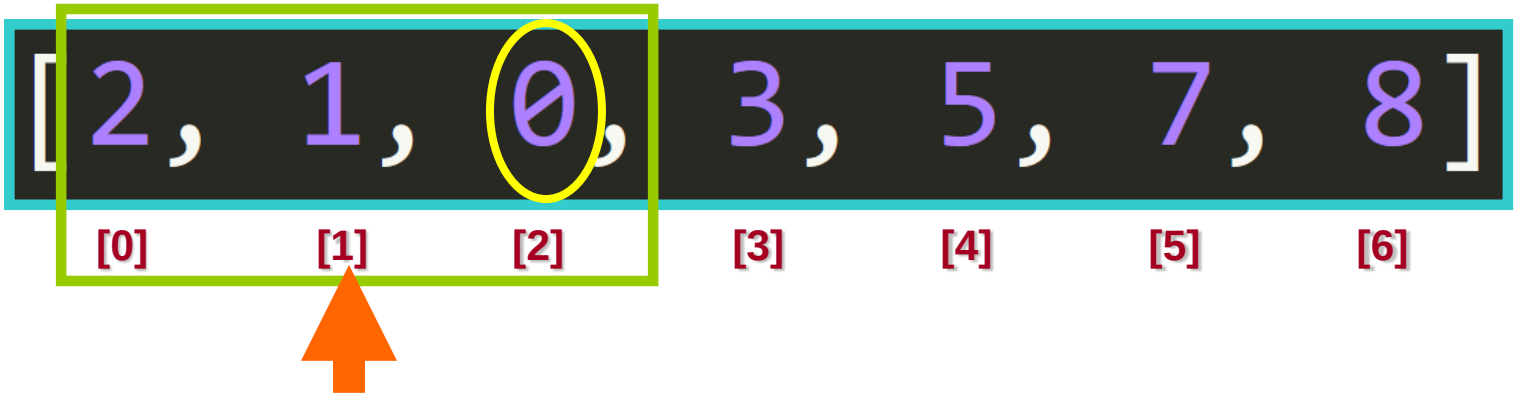
i = -1

j = 1

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

1 <= 0?

No!



pivot = 0

i = -1

j = 1

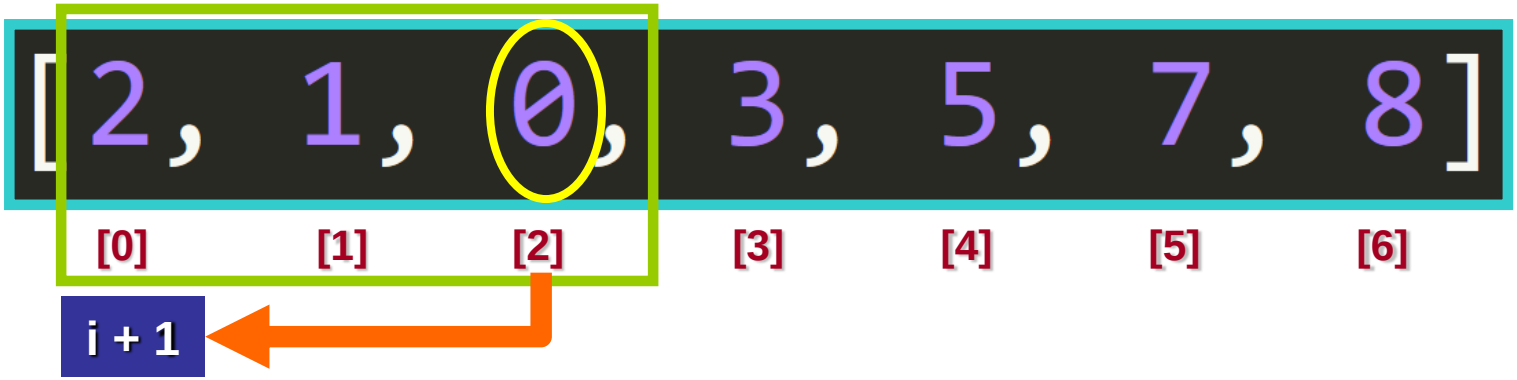
```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

pivot = 0

i = -1

j = 1

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

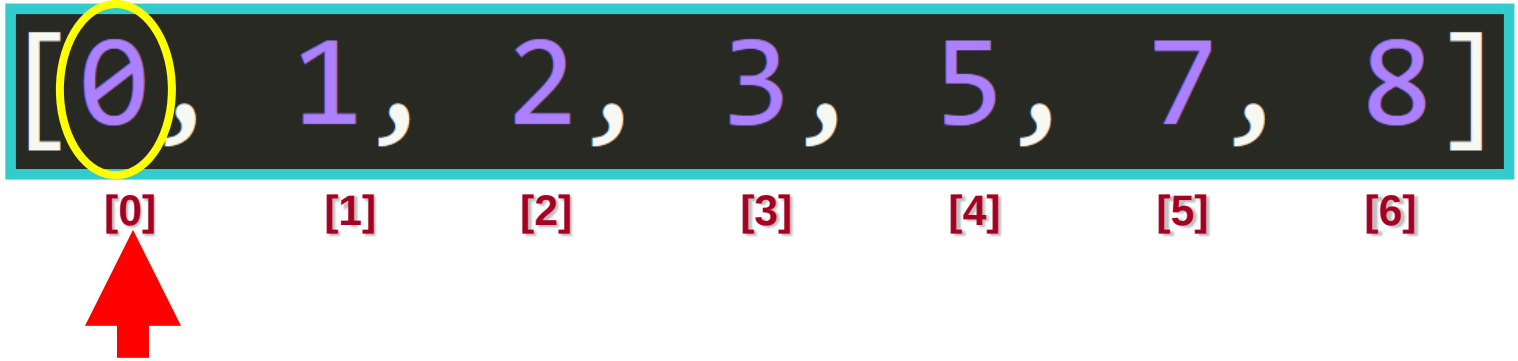


pivot = 0

i = -1

j = 1

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 0

i = -1

j = 1

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

return 0

```
[0, 1, 2, 3, 5, 7, 8]
 [0] [1] [2] [3] [4] [5] [6]
```

```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

    quicksort(lst, low, pivot_index-1)
    quicksort(lst, pivot_index+1, high)
```

```
quicksort(a, 0, len(a)-1)
```

<waiting recursive call>

```
quicksort(lst, 0, 3)
```

<waiting recursive call>

```
quicksort(lst, 0, 2)
```

0

```
[0, 1, 2, 3, 5, 7, 8]
```

[0] [1] [2] [3] [4] [5] [6]

```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

    quicksort(lst, low, pivot_index-1)
    quicksort(lst, pivot_index+1, high)
```

```
quicksort(a, 0, len(a)-1)
```

<waiting recursive call>

```
quicksort(lst, 0, 3)
```

<waiting recursive call>

```
quicksort(lst, 0, 2)
```

<waiting recursive call>

```
quicksort(lst, 0, -1)
```

Stop!

```
[0, 1, 2, 3, 5, 7, 8]
```

[0] [1] [2] [3] [4] [5] [6]

```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

    quicksort(lst, low, pivot_index-1)
    quicksort(lst, pivot_index+1, high)
```

```
quicksort(a, 0, len(a)-1)
```

<waiting recursive call>

```
quicksort(lst, 0, 3)
```

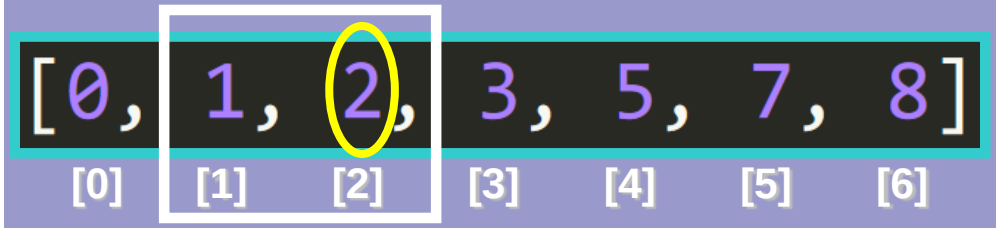
<waiting recursive call>

```
quicksort(lst, 0, 2)
```

```
quicksort(lst, 0, -1)
```

```
quicksort(lst, 1, 2)
```

Partition…




```python
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

    quicksort(lst, low, pivot_index-1)
    quicksort(lst, pivot_index+1, high)
```

```
quicksort(a, 0, len(a)-1)
```

<waiting recursive call>

```
quicksort(lst, 0, 3)
```

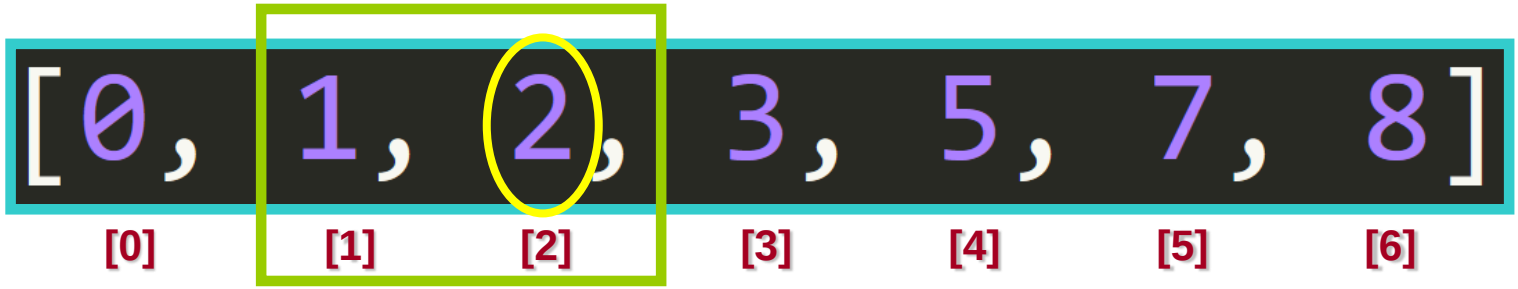
<waiting recursive call>

```
quicksort(lst, 0, 2)
```

```
quicksort(lst, 0, -1)
```

```
quicksort(lst, 1, 2)
```

Partition…

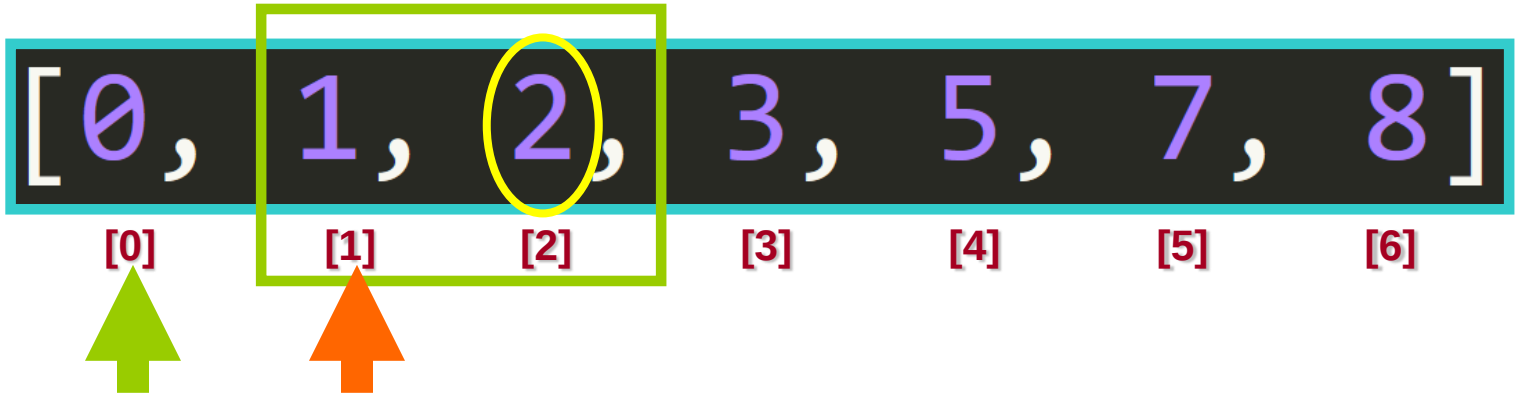


pivot = 0

i = 0

j = 1

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 0

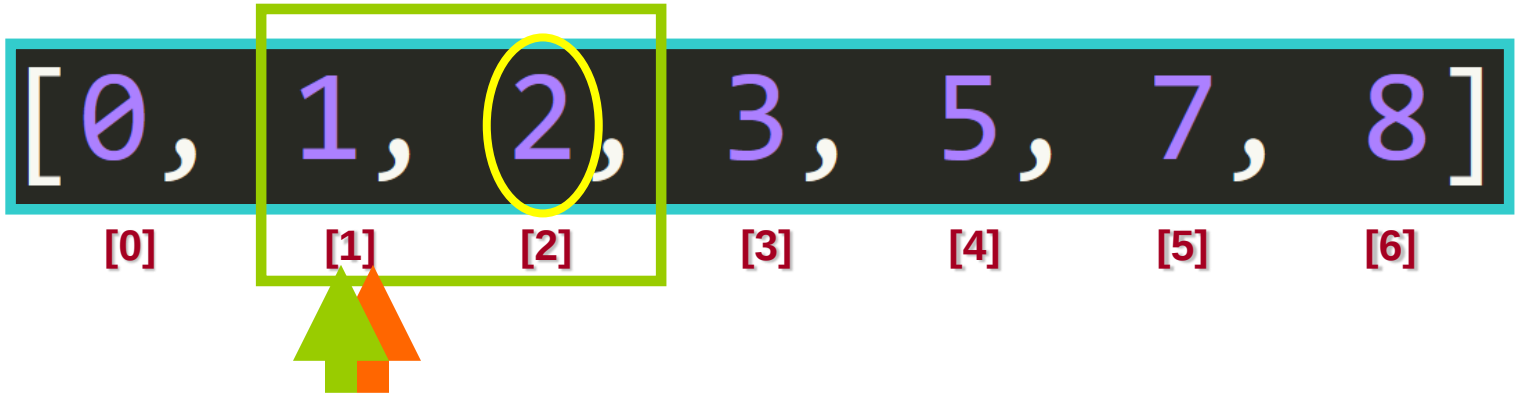
i = 0

j = 1

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

1 <= 2?

Yes!



pivot = 0

i = 0

j = 1

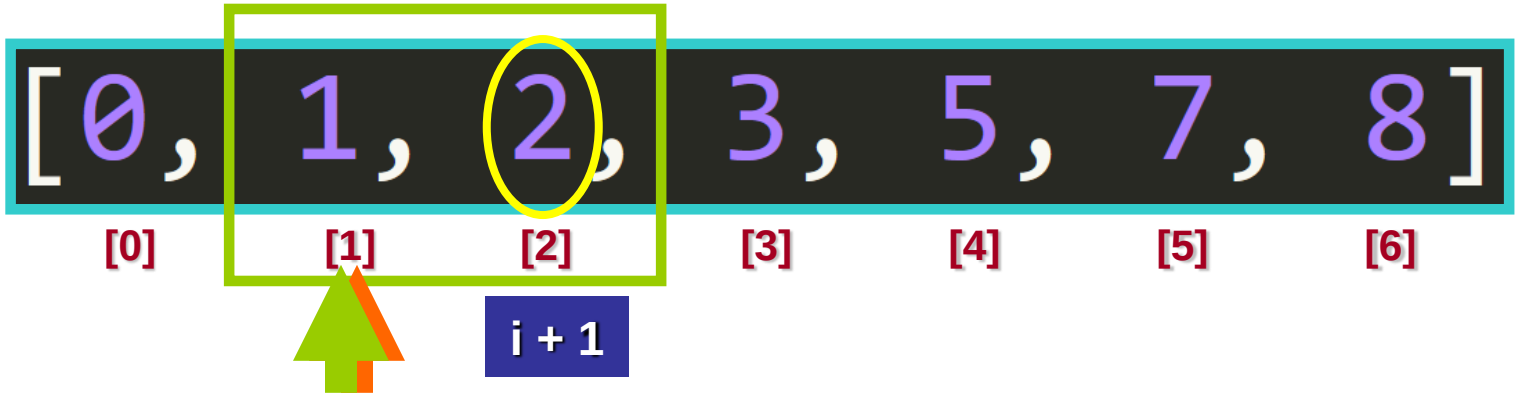
```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

pivot = 0

i = 0

j = 1

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

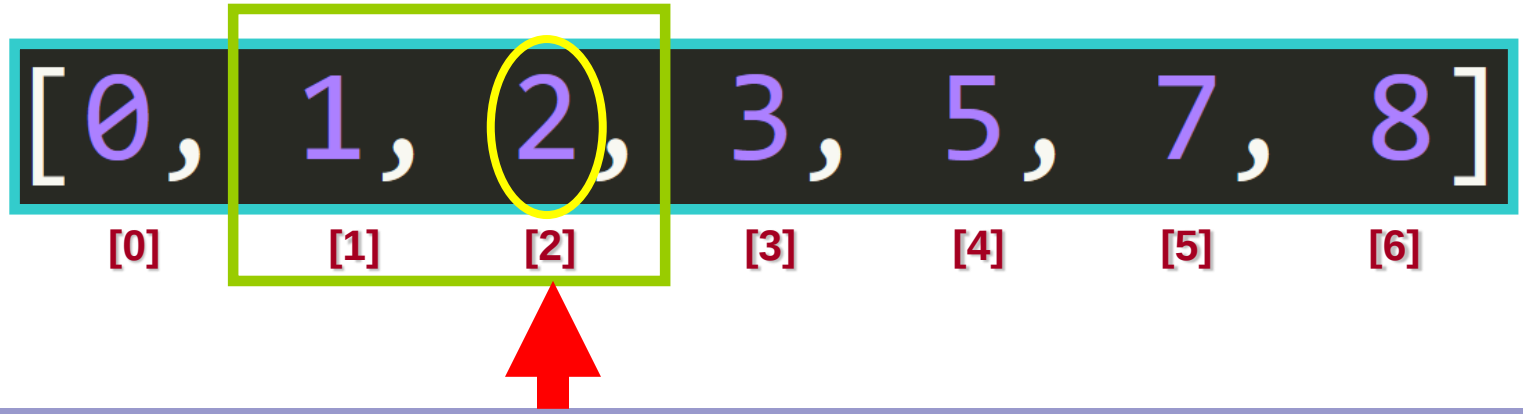


pivot = 0

i = 0

j = 1

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



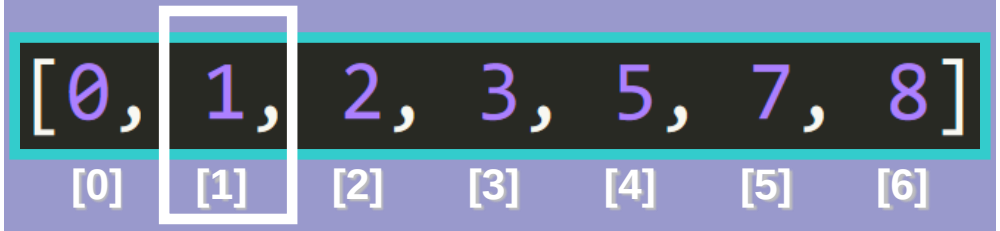
pivot = 0

i = 0

j = 1

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

return 2




```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

    quicksort(lst, low, pivot_index-1)
    quicksort(lst, pivot_index+1, high)
```

```
quicksort(a, 0, len(a)-1)
```

<waiting recursive call>

```
quicksort(lst, 0, 3)
```

<waiting recursive call>

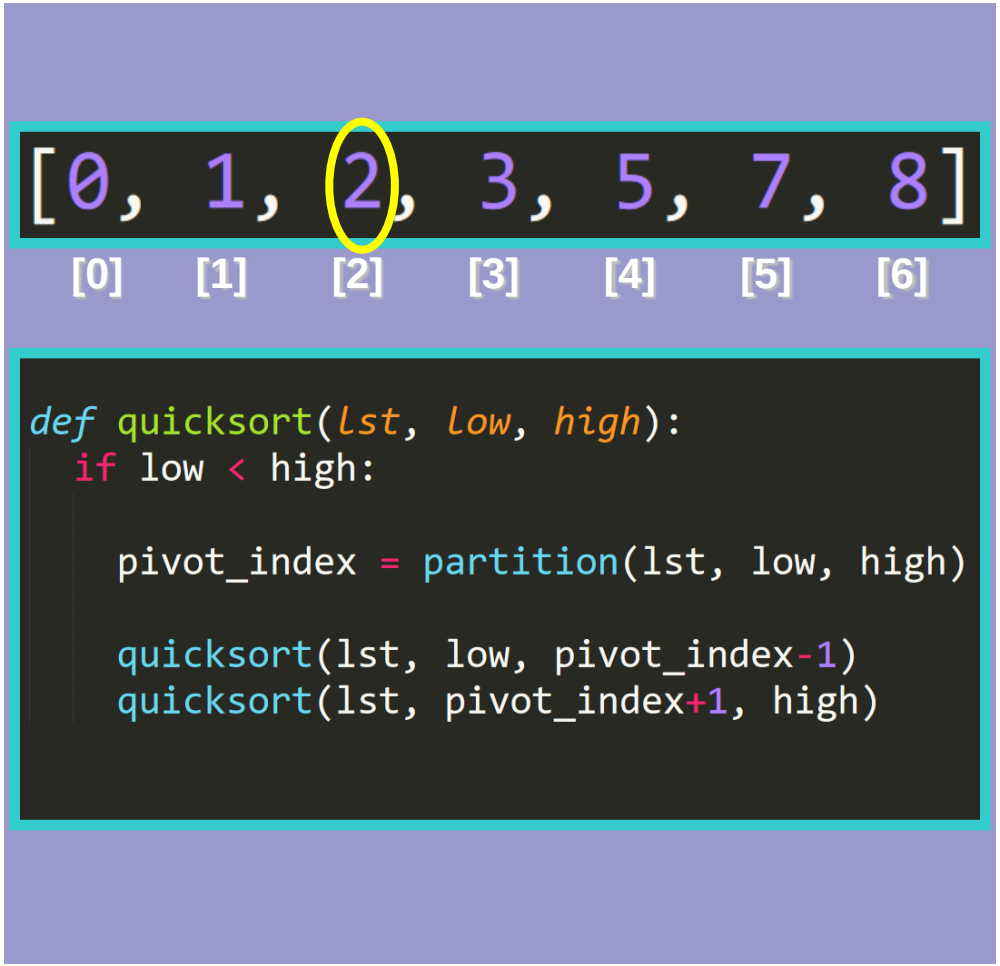
```
quicksort(lst, 0, 2)
```

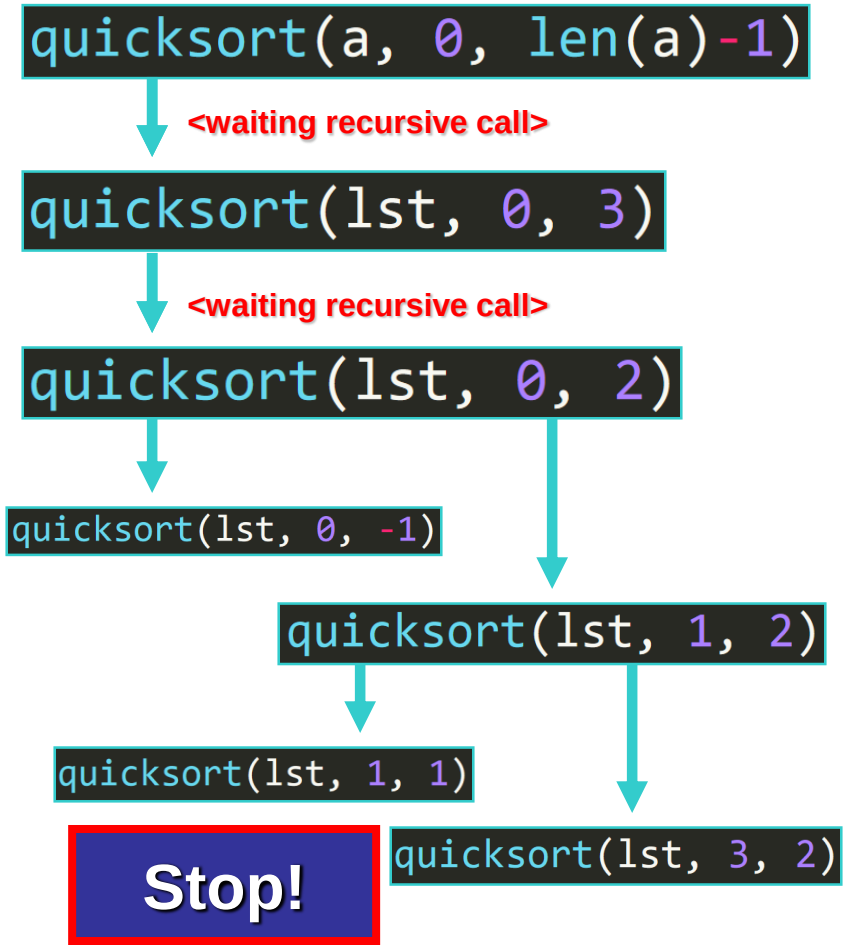
```
quicksort(lst, 0, -1)
```

```
quicksort(lst, 1, 2)
```

```
quicksort(lst, 1, 1)
```

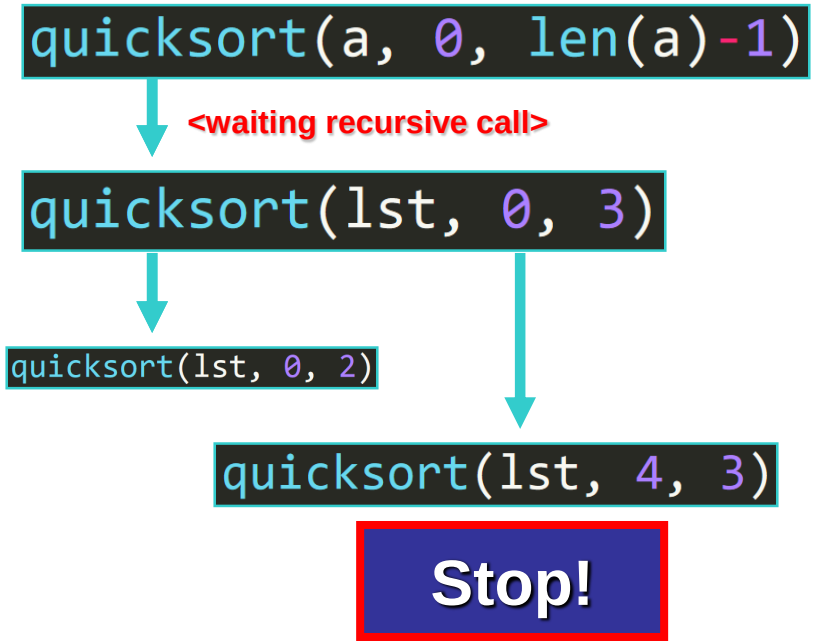
Stop!


[0, 1, 2, 3, 5, 7, 8]
[0] [1] [2] [3] [4] [5] [6]
def quicksort(lst, low, high):
if low < high:
pivot_index = partition(lst, low, high)
quicksort(lst, low, pivot_index-1)
quicksort(lst, pivot_index+1, high)


quicksort(a, 0, len(a)-1)
<waiting recursive call>
quicksort(lst, 0, 3)
<waiting recursive call>
quicksort(lst, 0, 2)
quicksort(lst, 0, -1)
quicksort(lst, 1, 2)
quicksort(lst, 1, 1)
Stop!
quicksort(lst, 3, 2)

```
[0, 1, 2, 3, 5, 7, 8]
 [0] [1] [2] [3] [4] [5] [6]
```

```python
def quicksort(lst, low, high):
    if low < high:

        pivot_index = partition(lst, low, high)

        quicksort(lst, low, pivot_index-1)
        quicksort(lst, pivot_index+1, high)
```



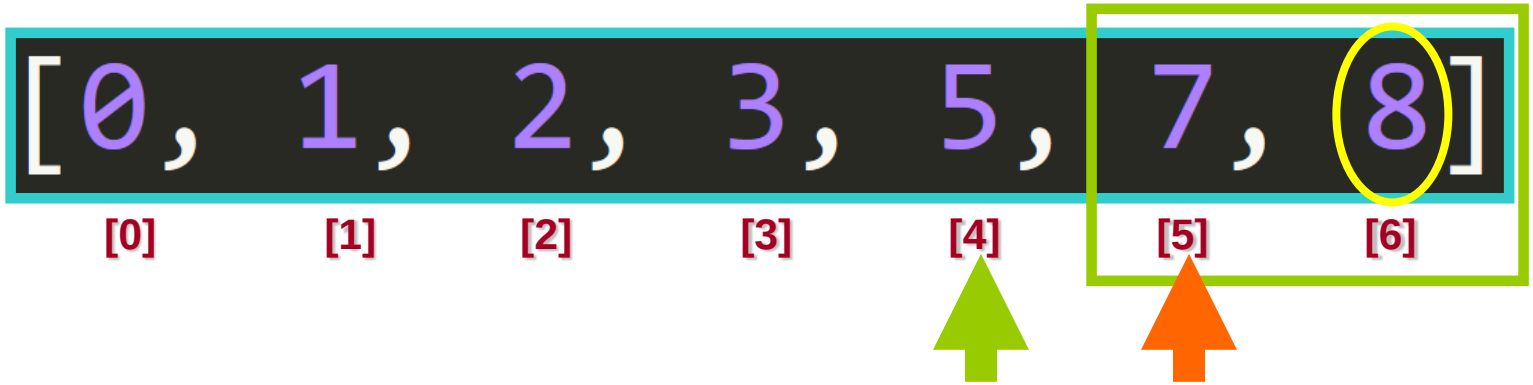
```
[0, 1, 2, 3, 5, 7, 8]
 [0] [1] [2] [3] [4] [5] [6]
```

```python
def quicksort(lst, low, high):
    if low < high:

        pivot_index = partition(lst, low, high)

        quicksort(lst, low, pivot_index-1)
        quicksort(lst, pivot_index+1, high)
```

```
quicksort(a, 0, len(a)-1)
```

```
quicksort(lst, 0, 3)
```

```
quicksort(lst, 5, 6)
```

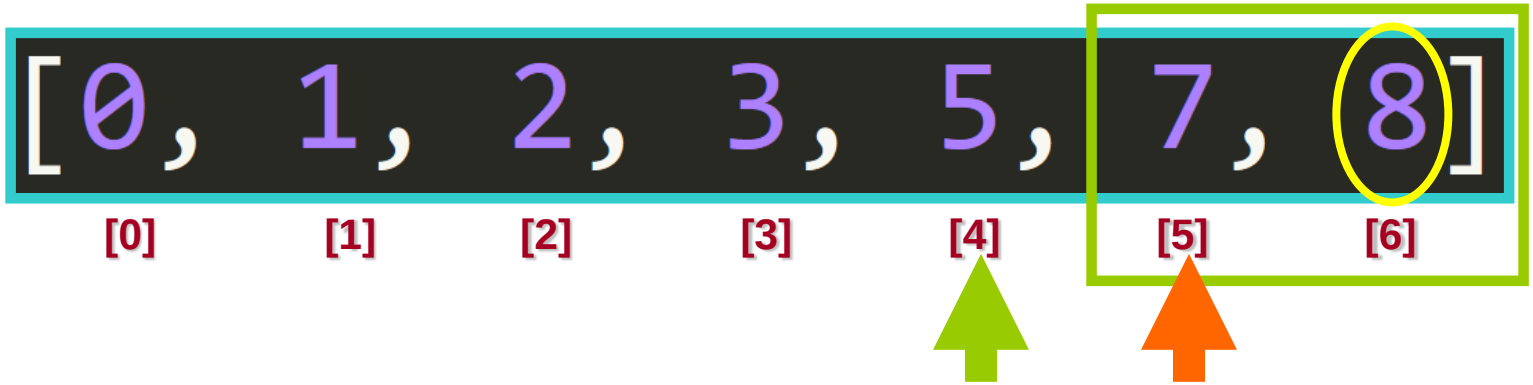


pivot = 8

i = 4

j = 5

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 8

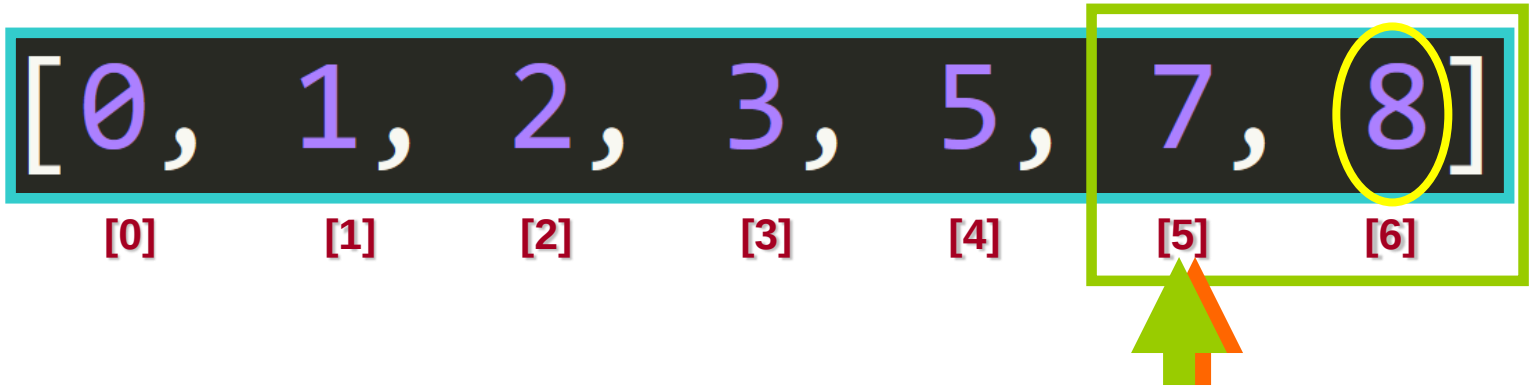
i = 4

j = 5

```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

7 <= 8?

Yes!

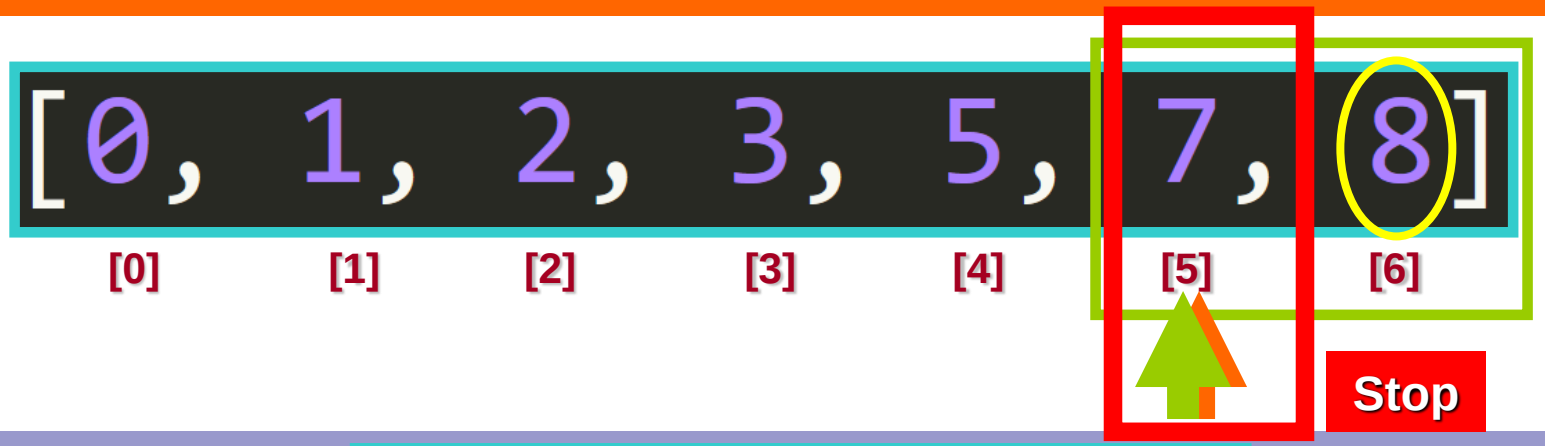


pivot = 8

i = 5

j = 5

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

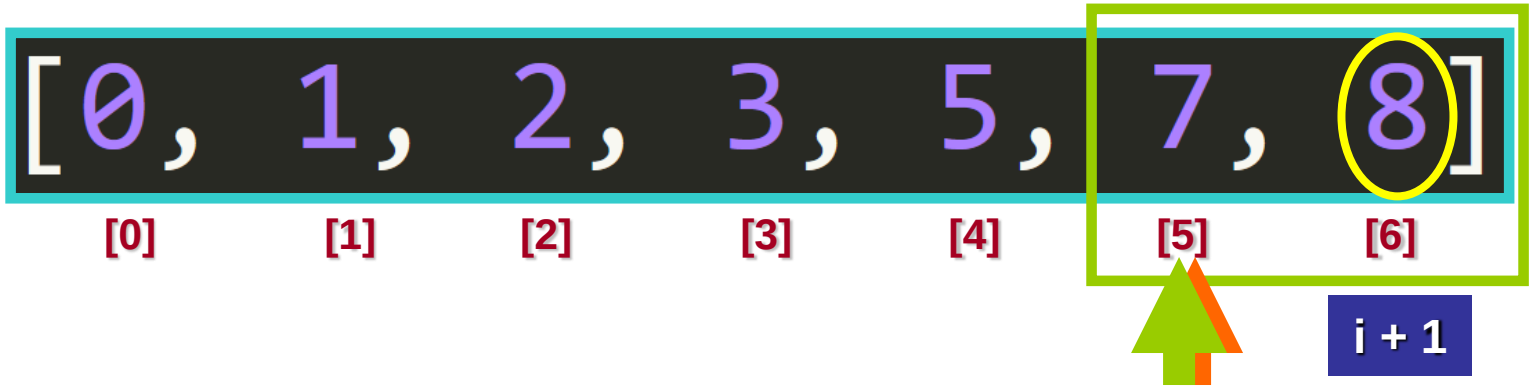


pivot = 8

i = 5

j = 5

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

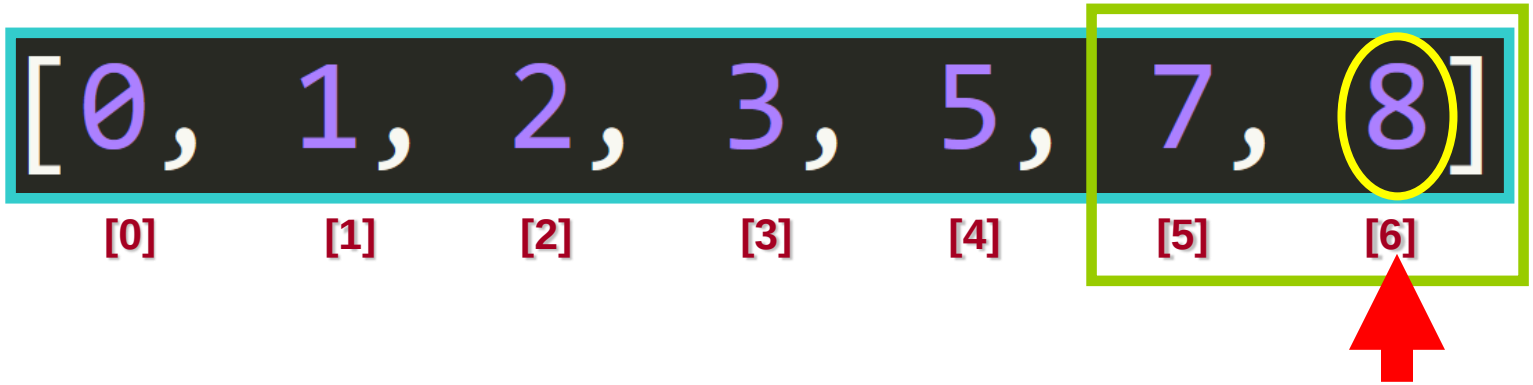


pivot = 8

i = 5

j = 5

```python
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```



pivot = 8

i = 5

j = 5

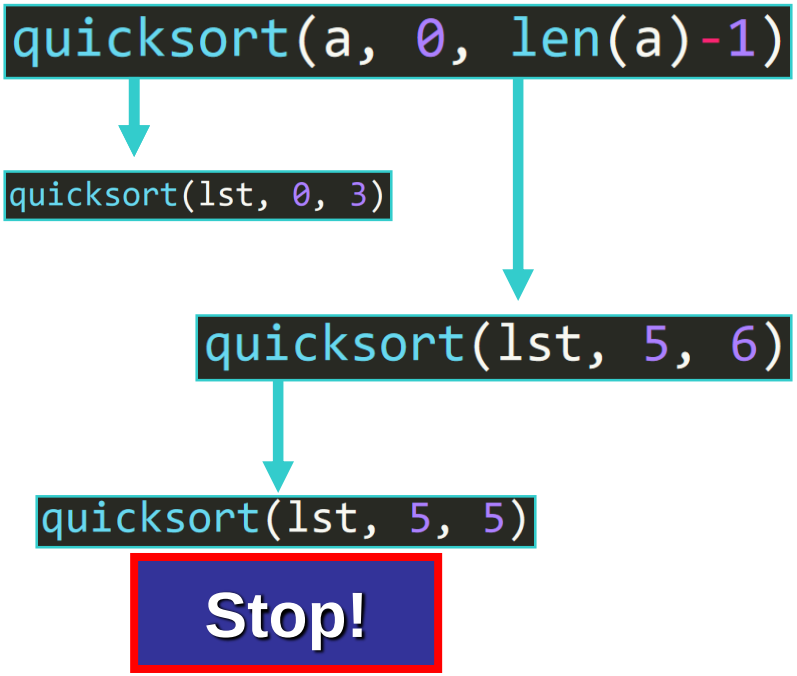
```
def partition(lst, low, high):
    pivot = lst[high]

    i = low - 1

    for j in range(low, high):
        if lst[j] <= pivot:
            i += 1
            lst[i], lst[j] = lst[j], lst[i]

    lst[i+1], lst[high] = lst[high], lst[i+1]
    return i+1
```

return 6

```
[0, 1, 2, 3, 5, 7, 8]
```

[0] [1] [2] [3] [4] [5] [6]

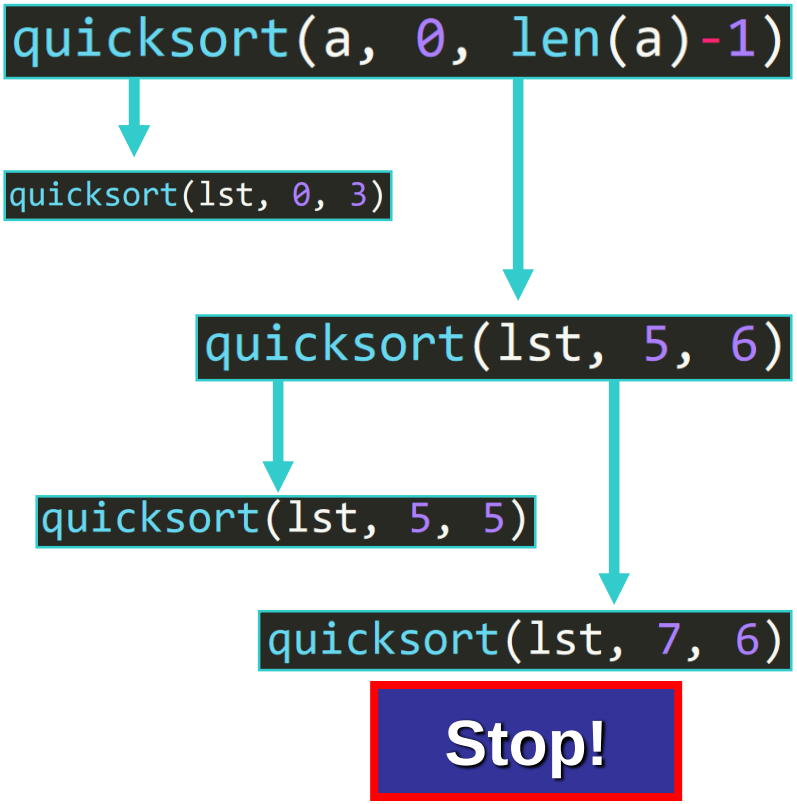
```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

    quicksort(lst, low, pivot_index-1)
    quicksort(lst, pivot_index+1, high)
```



```
[0, 1, 2, 3, 5, 7, 8]
```

[0] [1] [2] [3] [4] [5] [6]

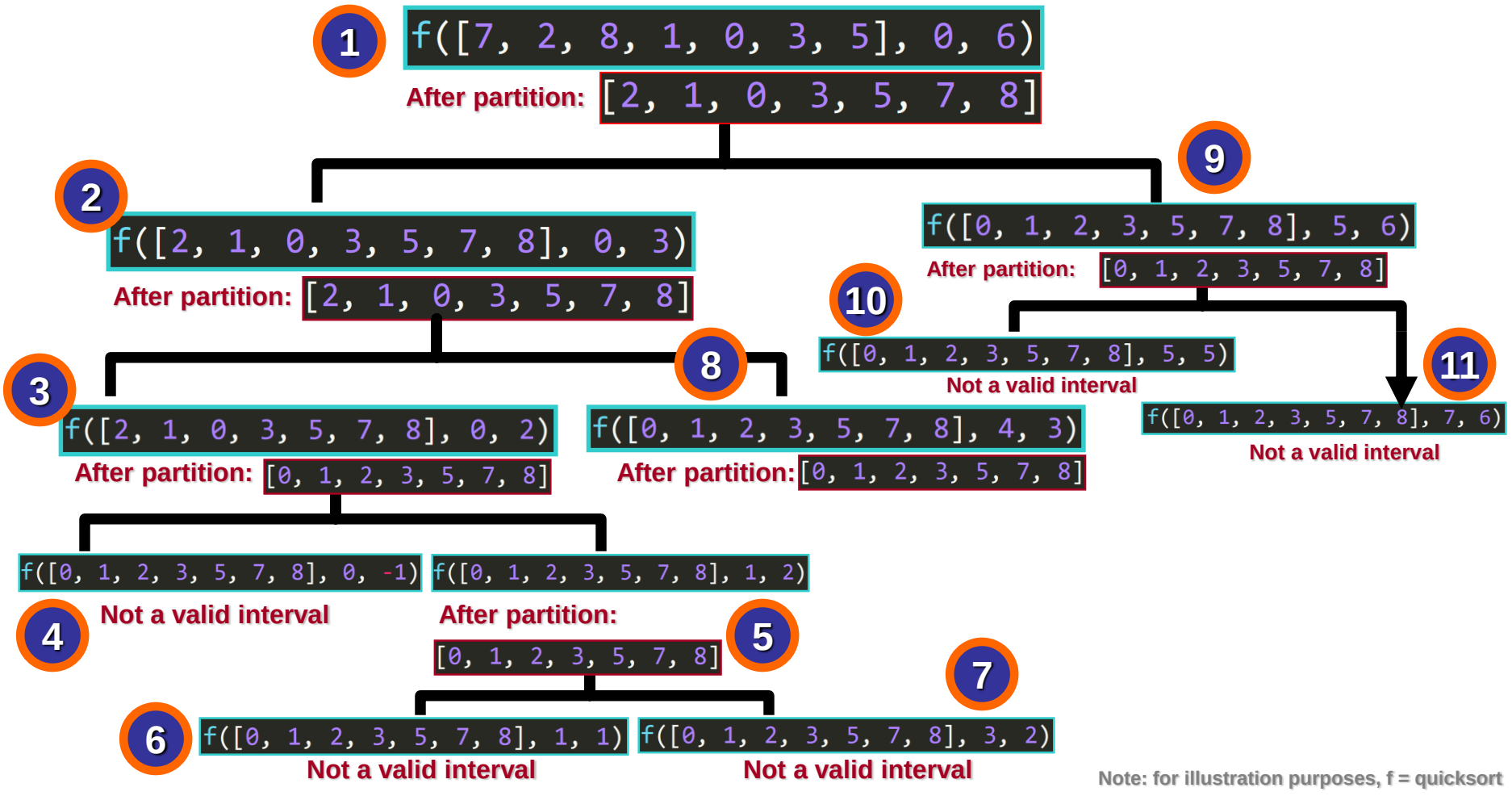
```python
def quicksort(lst, low, high):
    if low < high:

        pivot_index = partition(lst, low, high)

        quicksort(lst, low, pivot_index-1)
        quicksort(lst, pivot_index+1, high)
```



Quicksort

```
[0, 1, 2, 3, 5, 7, 8]
```

Sorted!


1
f([7, 2, 8, 1, 0, 3, 5], 0, 6)
After partition: [2, 1, 0, 3, 5, 7, 8]
2
f([2, 1, 0, 3, 5, 7, 8], 0, 3)
After partition: [2, 1, 0, 3, 5, 7, 8]
9
f([0, 1, 2, 3, 5, 7, 8], 5, 6)
After partition: [0, 1, 2, 3, 5, 7, 8]
10
f([0, 1, 2, 3, 5, 7, 8], 5, 5)
Not a valid interval
11
f([0, 1, 2, 3, 5, 7, 8], 7, 6)
Not a valid interval
3
f([2, 1, 0, 3, 5, 7, 8], 0, 2)
After partition: [0, 1, 2, 3, 5, 7, 8]
8
f([0, 1, 2, 3, 5, 7, 8], 4, 3)
After partition: [0, 1, 2, 3, 5, 7, 8]
f([0, 1, 2, 3, 5, 7, 8], 0, -1)
4
Not a valid interval
f([0, 1, 2, 3, 5, 7, 8], 1, 2)
After partition:
[0, 1, 2, 3, 5, 7, 8]
5
6
f([0, 1, 2, 3, 5, 7, 8], 1, 1)
Not a valid interval
7
f([0, 1, 2, 3, 5, 7, 8], 3, 2)
Not a valid interval
Note: for illustration purposes, f = quicksort

To the Details