Algorithm

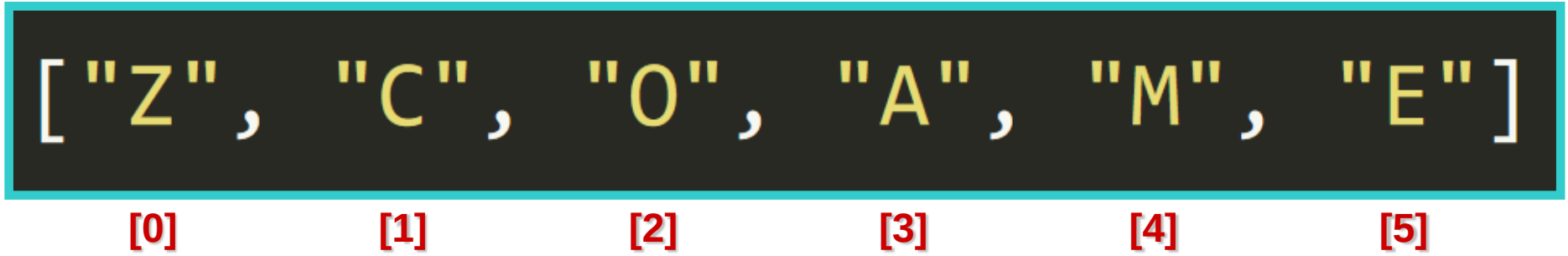
Quicksort

Example | Visual

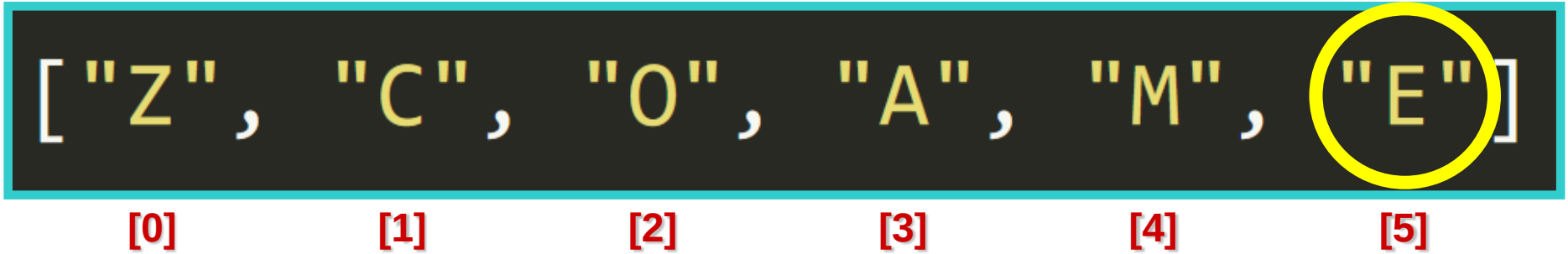
Quicksort

```
["Z", "C", "O", "A", "M", "E"]
```

Quicksort



Quicksort



i = -1

j = 0

Quicksort

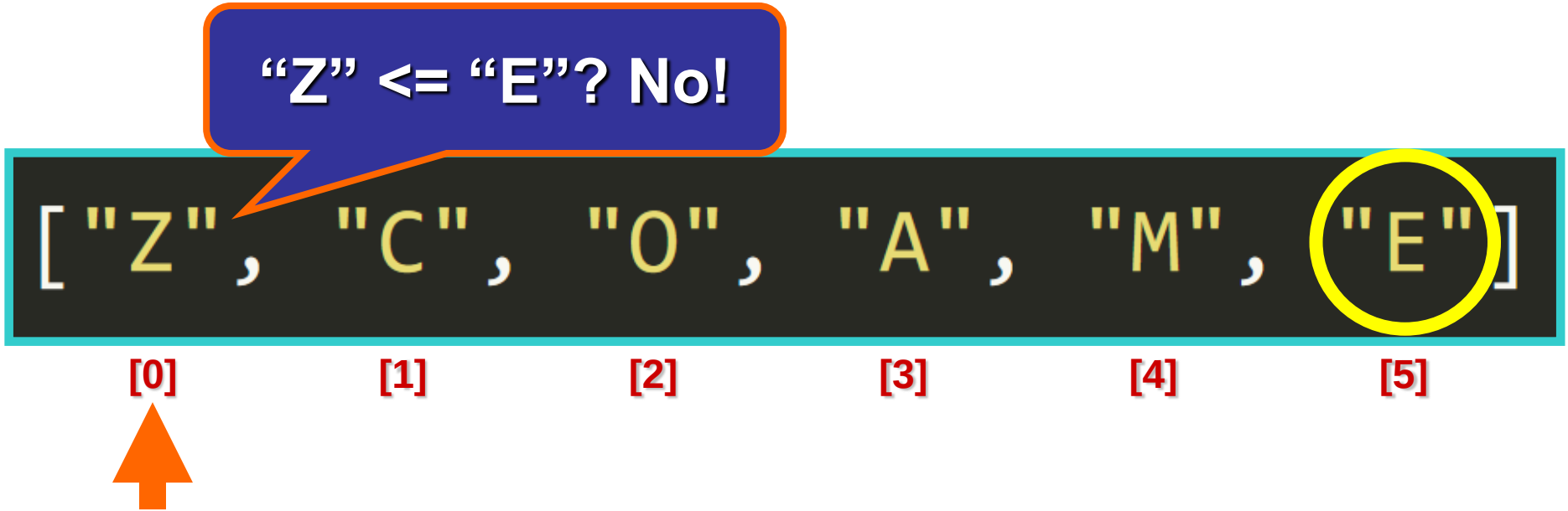
```
["Z", "C", "O", "A", "M", "E"]
```

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

i = -1

j = 0

Quicksort



Quicksort

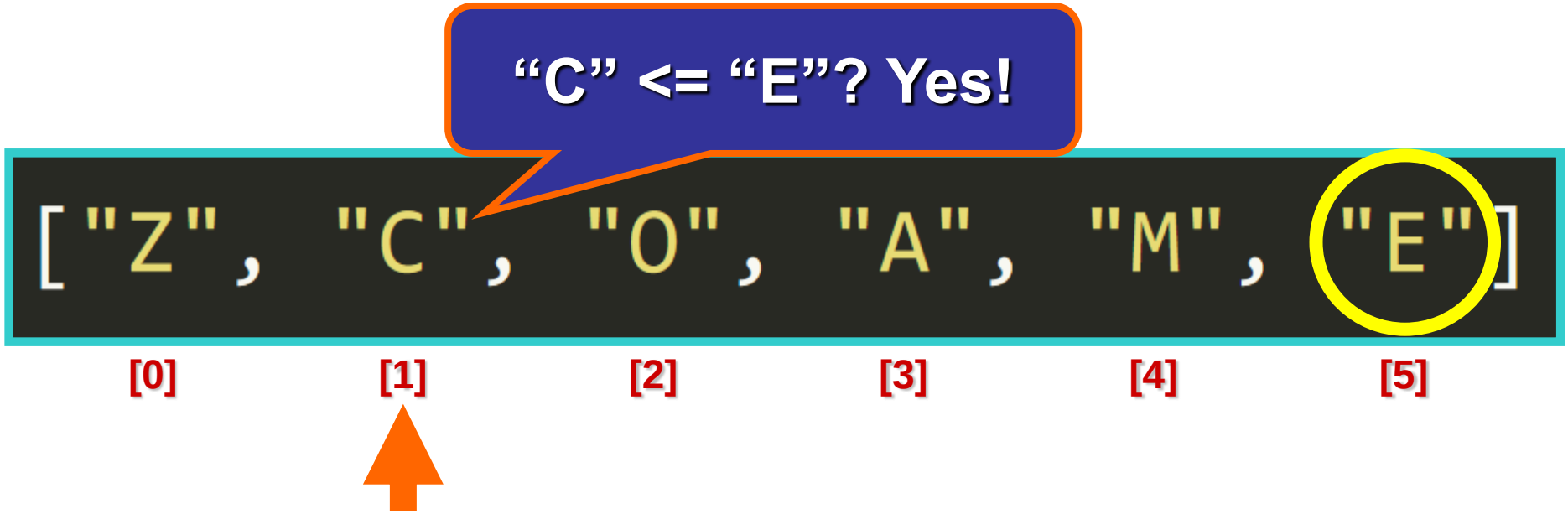
```
["Z", "C", "O", "A", "M", "E"]
```

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

i = -1

j = 1

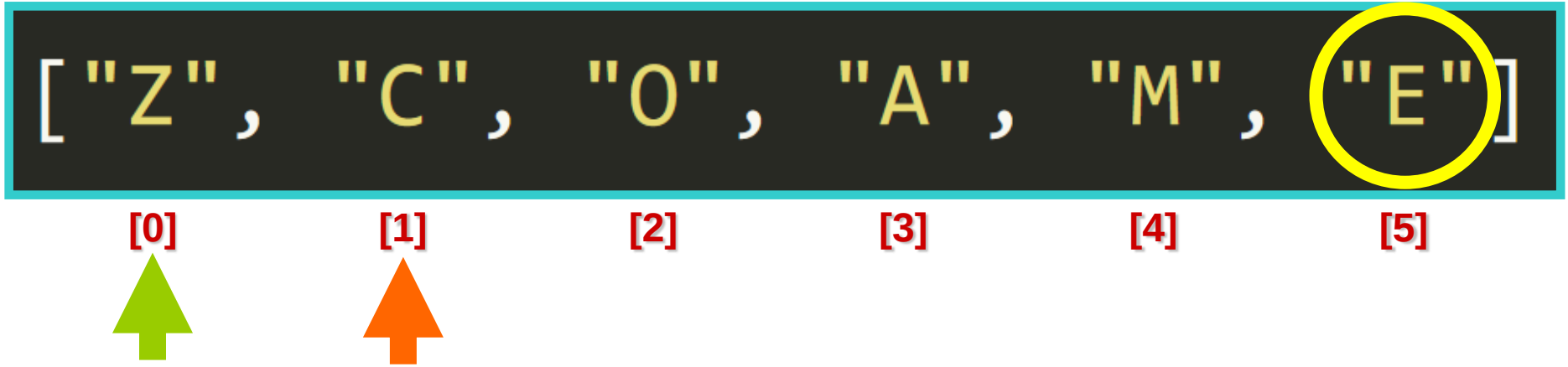
Quicksort



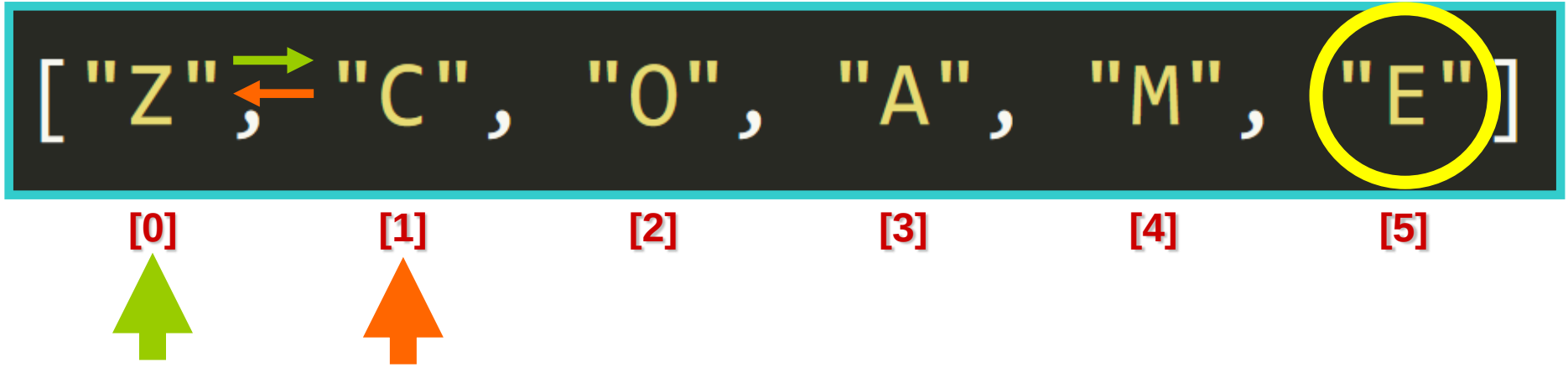
i = -1

j = 1

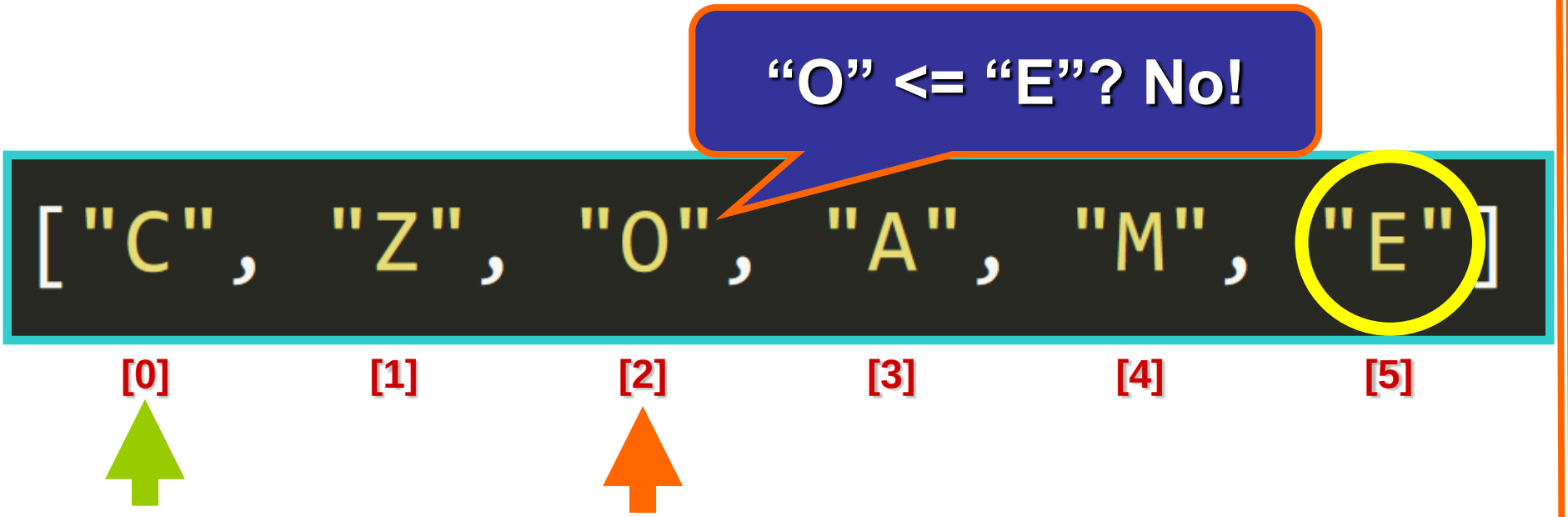
Quicksort



Quicksort



Quicksort



Quicksort

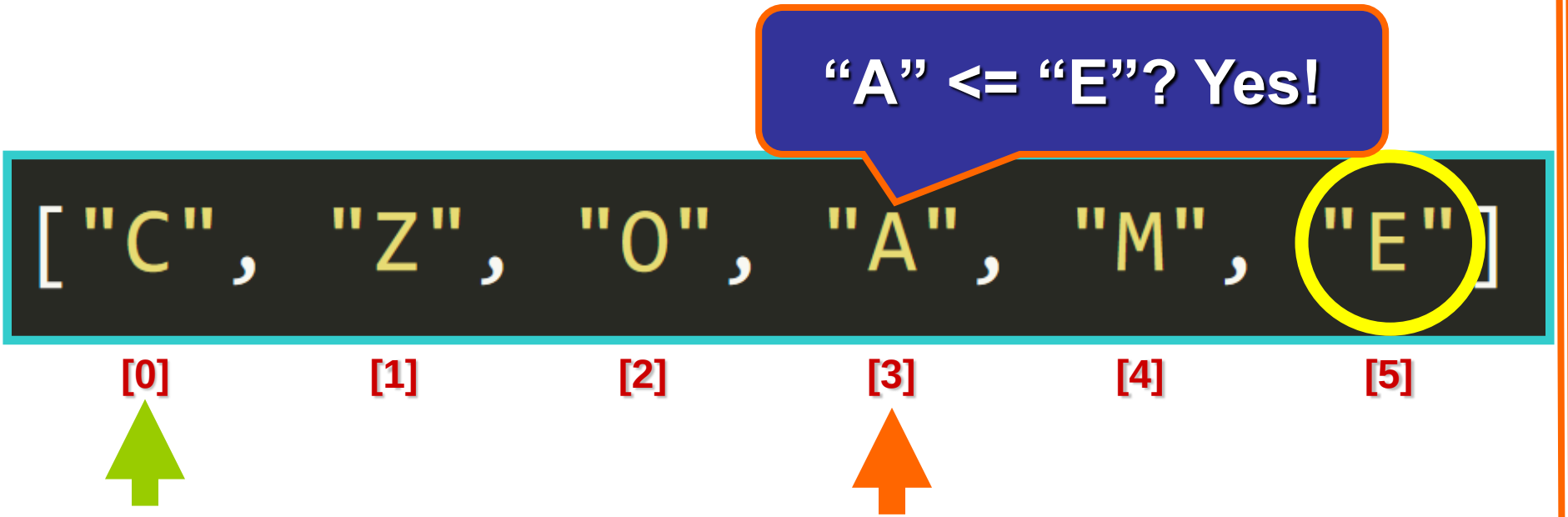
```
["C", "Z", "O", "A", "M", "E"]
```

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

i = 0

j = 3

Quicksort



Quicksort

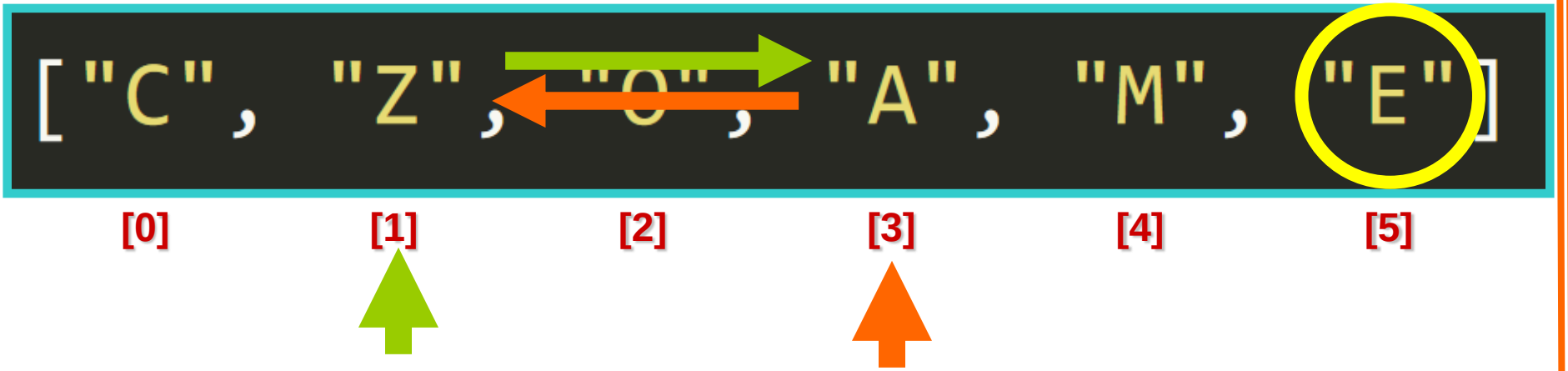
```
["C", "Z", "O", "A", "M", "E"]
```

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

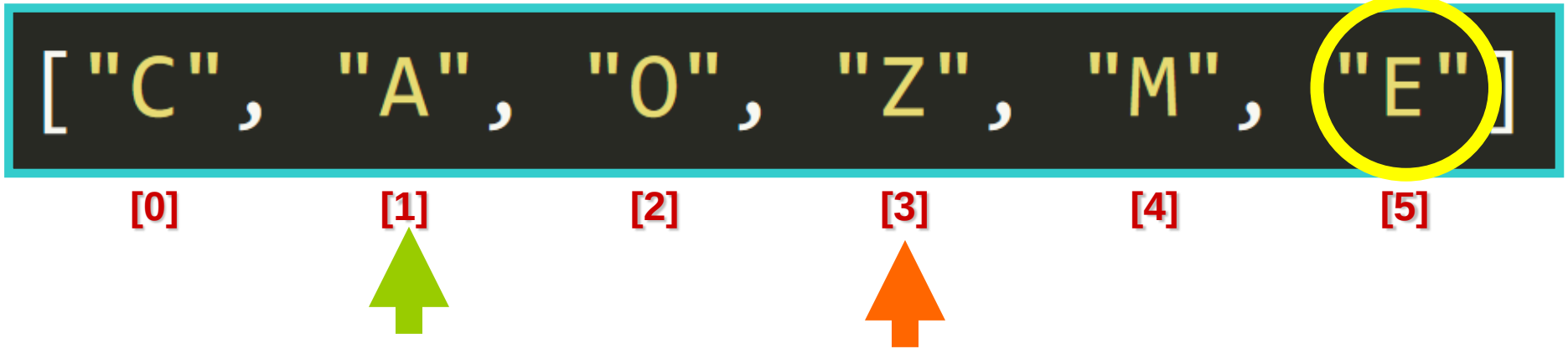
i = 1

j = 3

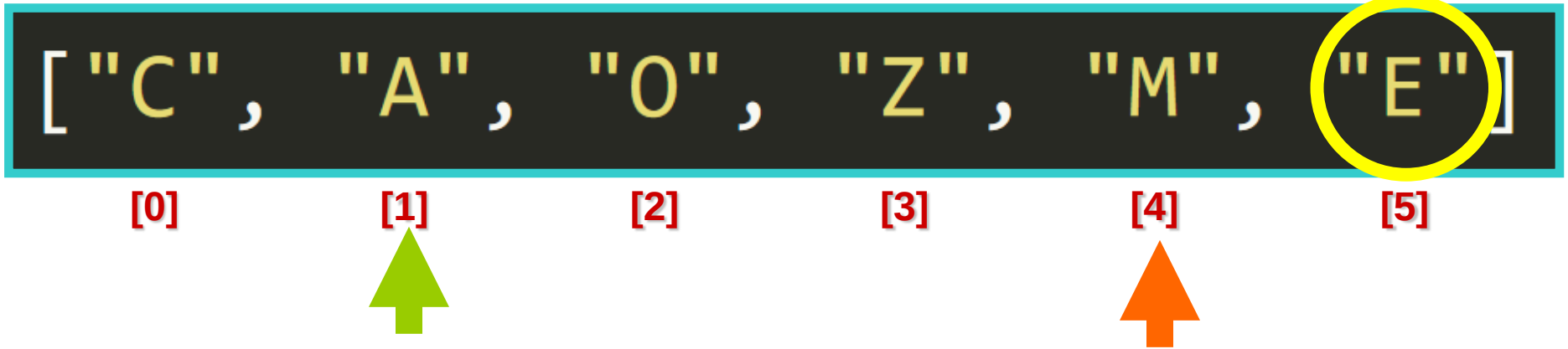
Quicksort



Quicksort



Quicksort



Quicksort

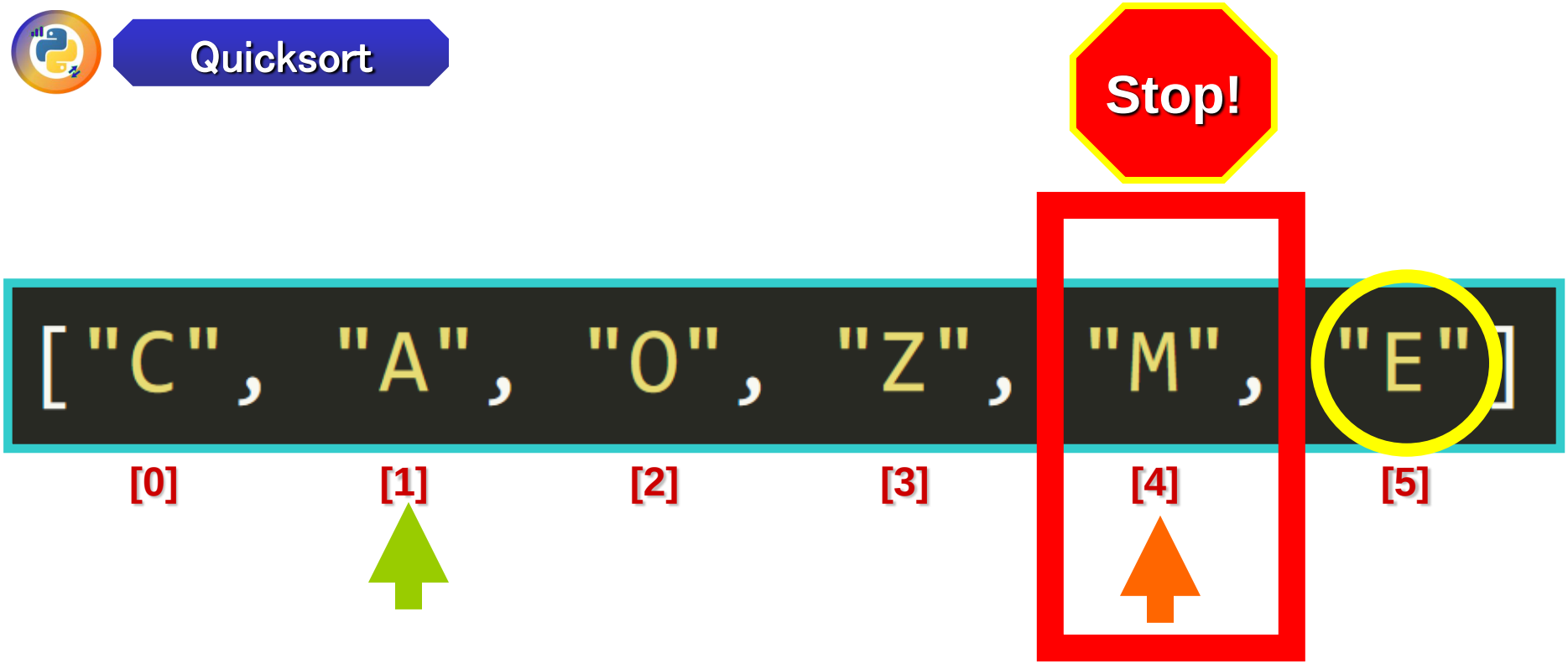
"M" <= "E"? No!

```
["C", "A", "O", "Z", "M", "E"]
```

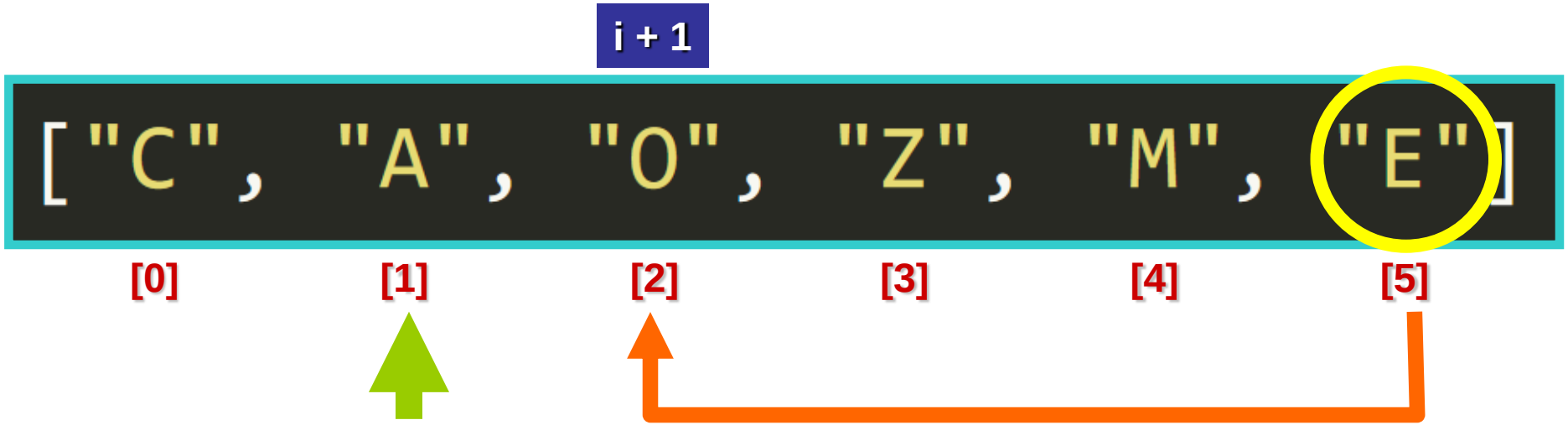
[0] [1] [2] [3] [4] [5]

i = 1

j = 4


Quicksort
Stop!
["C", "A", "O", "Z", "M", "E"]
[0]
[1]
[2]
[3]
[4]
[5]
i = 1
j = 4

Quicksort

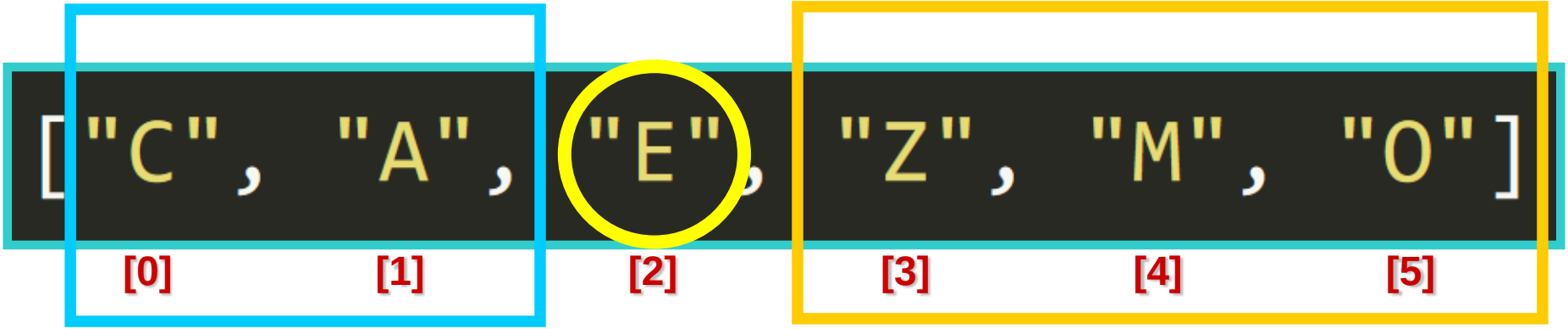


Quicksort

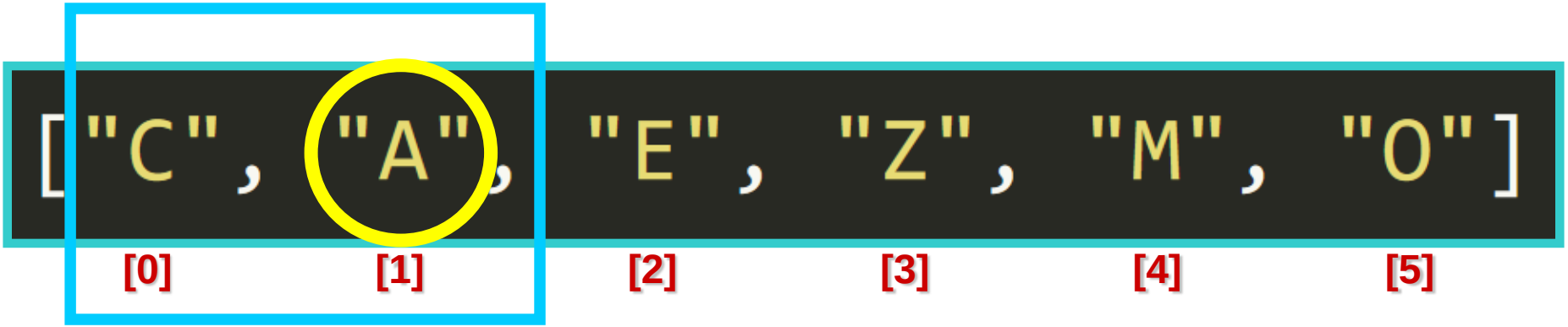
i = 1

j = 4

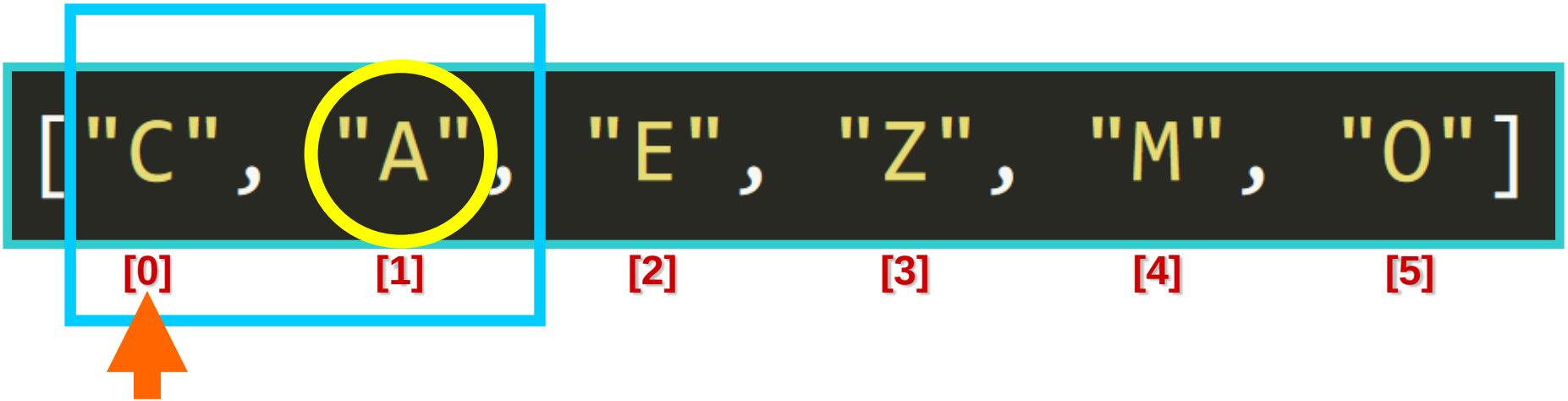
Quicksort



Quicksort



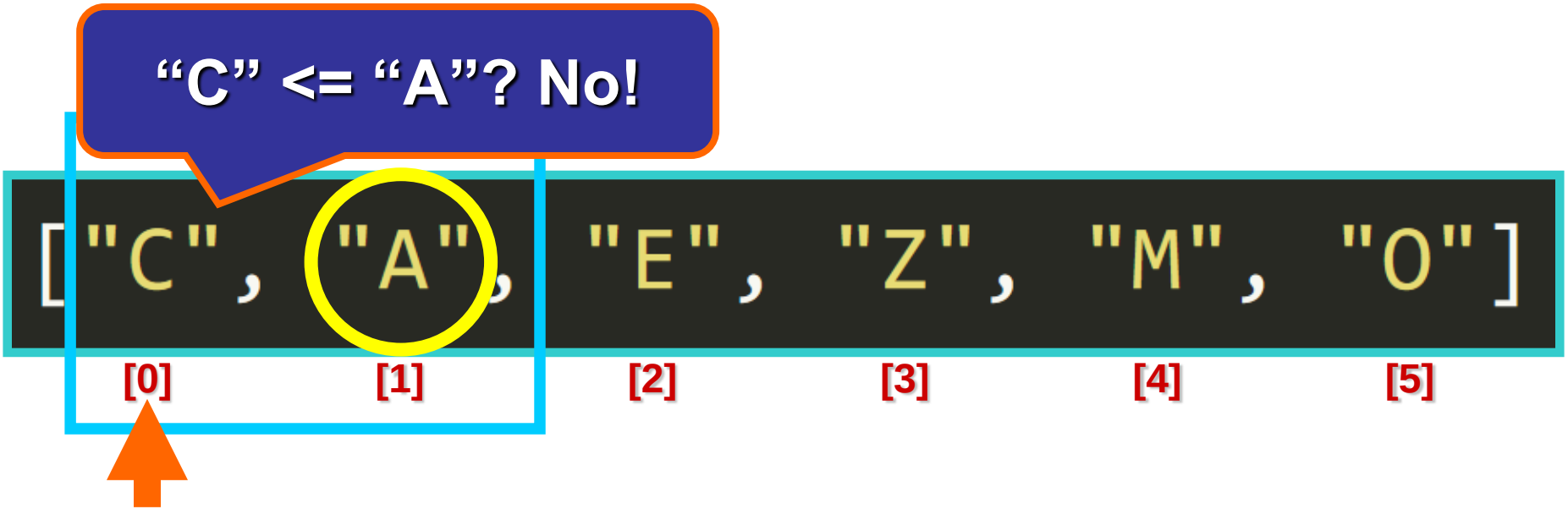
Quicksort



i = -1

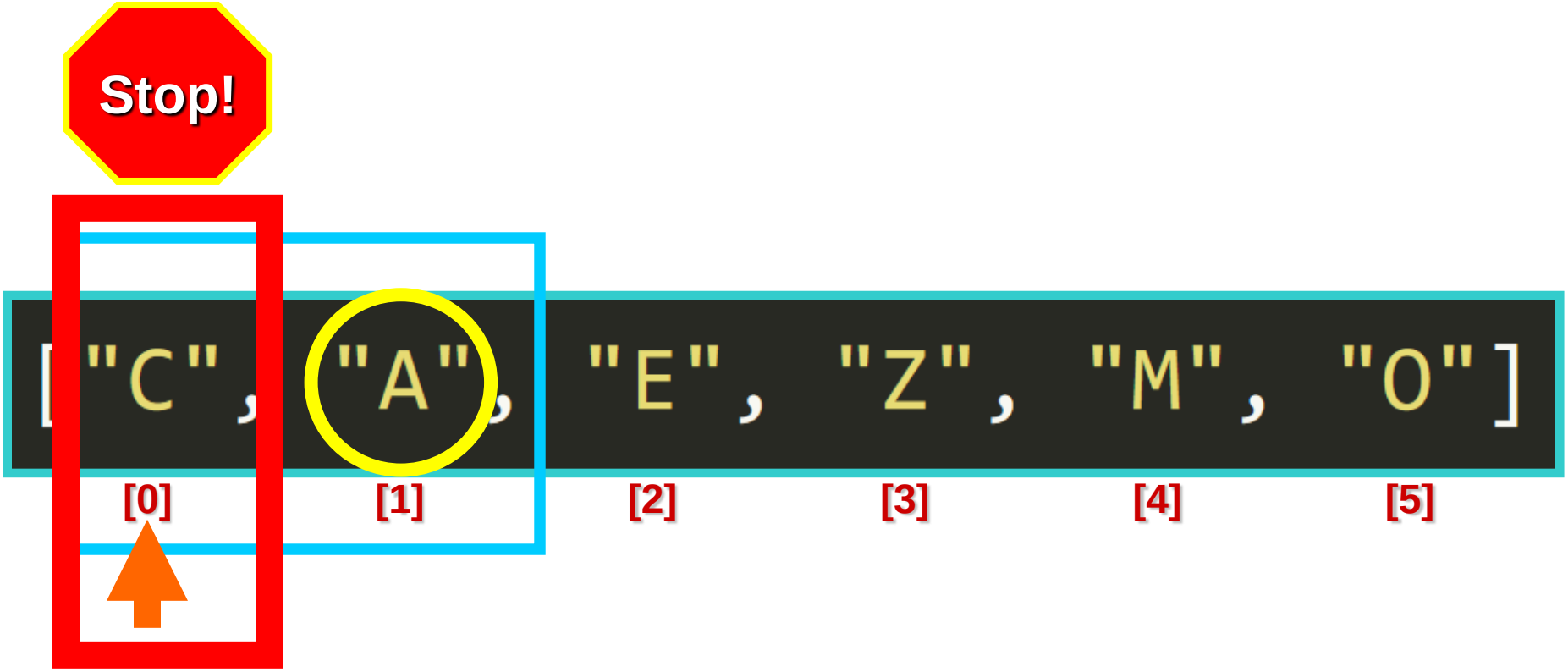
j = 0

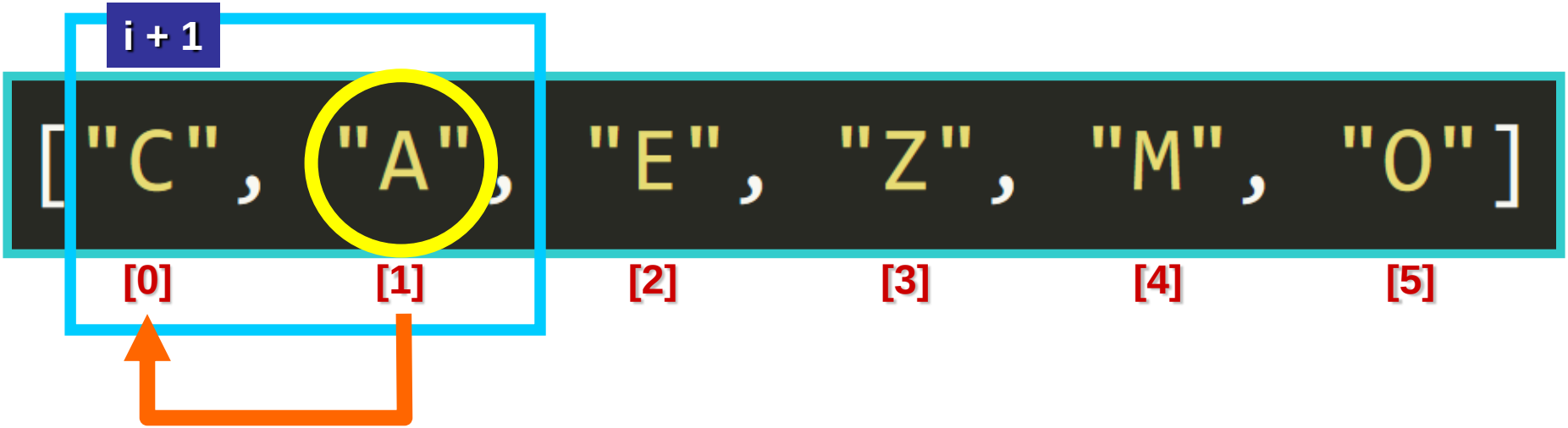
Quicksort



i = -1

j = 0


Stop!
["C", "A", "E", "Z", "M", "O"]
[0]
[1]
[2]
[3]
[4]
[5]
i = -1
j = 0



i = -1

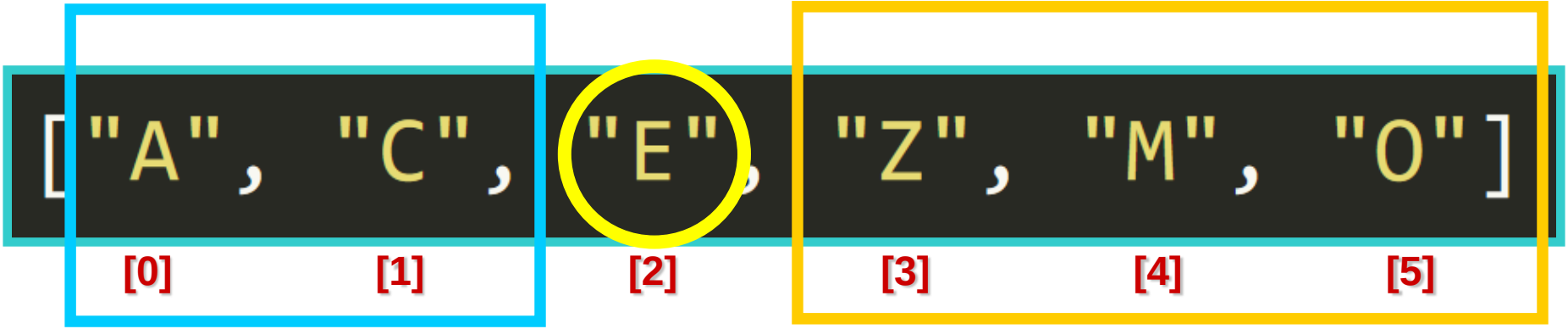
j = 0

i = -1

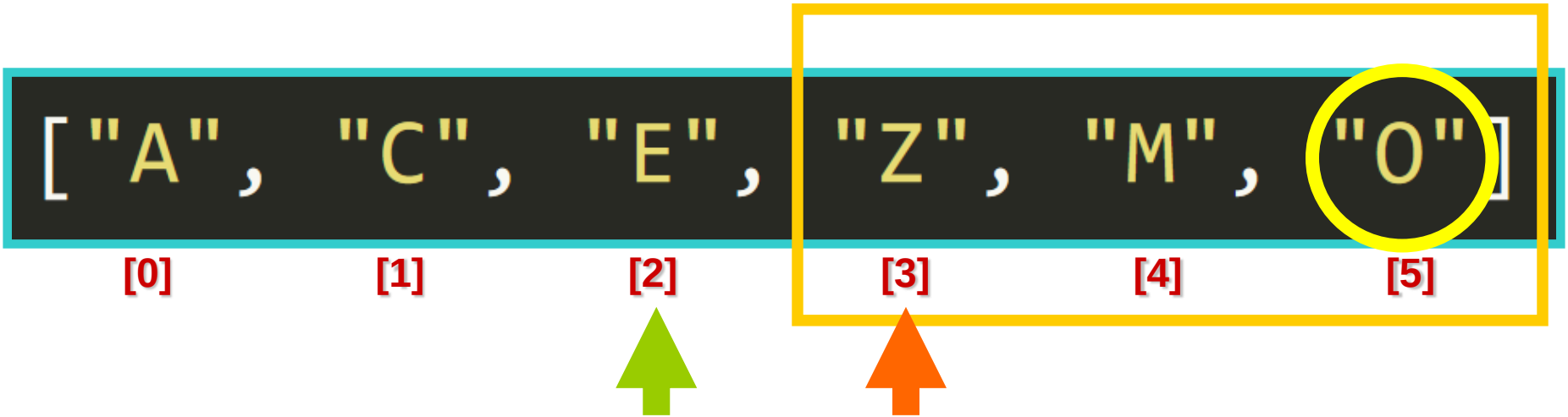
j = 0

A Few Steps Ago...

Quicksort



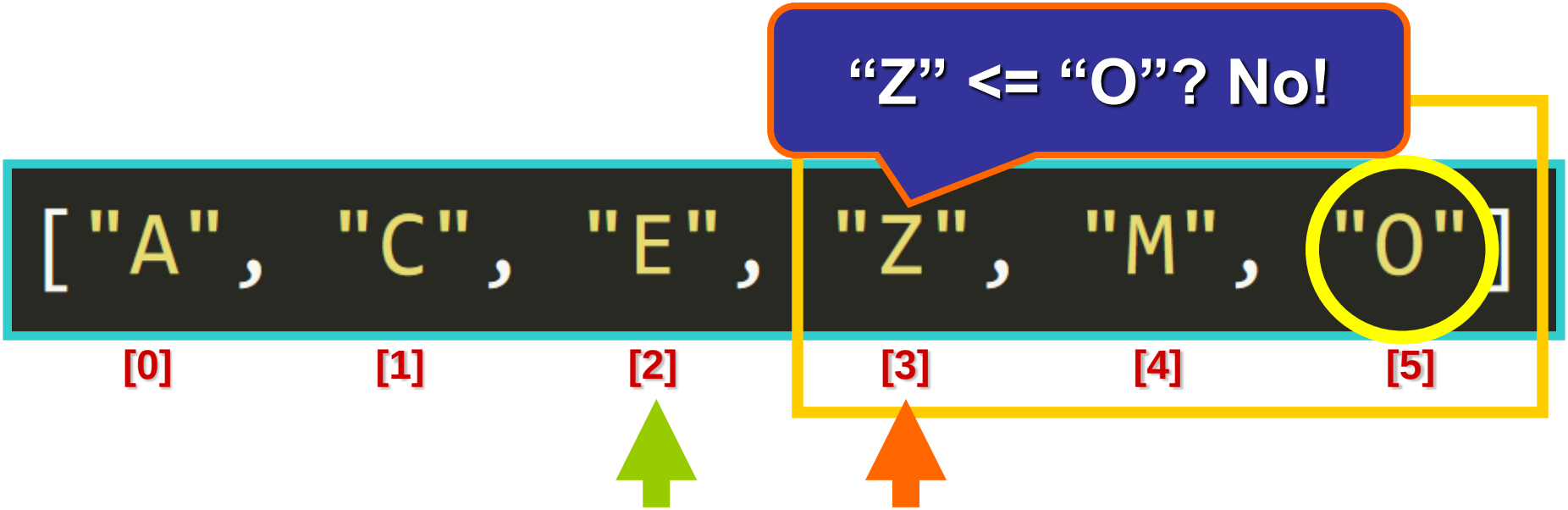
Quicksort



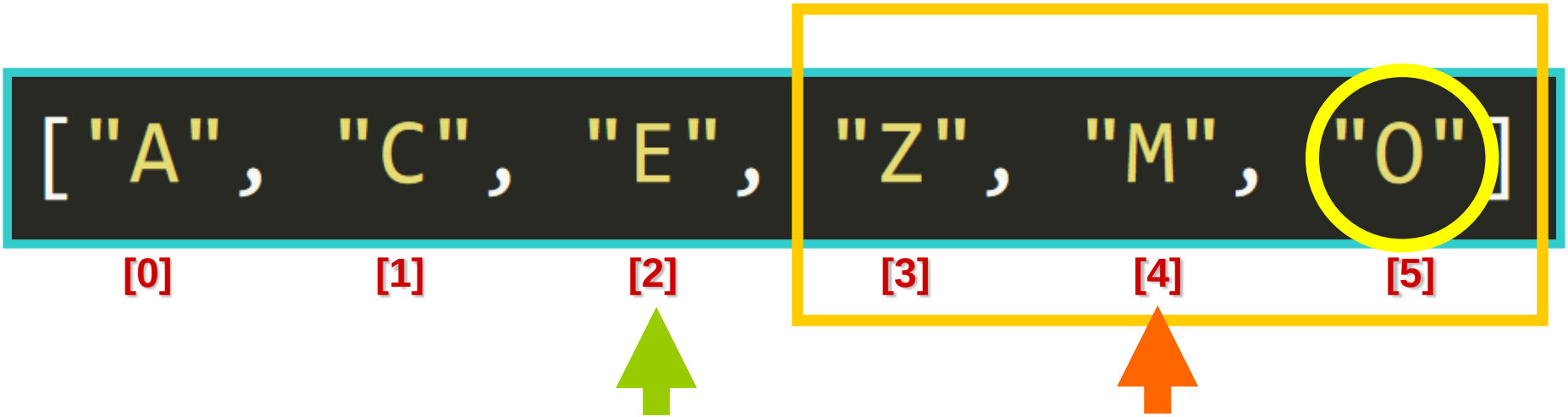
i = 2

j = 3

Quicksort



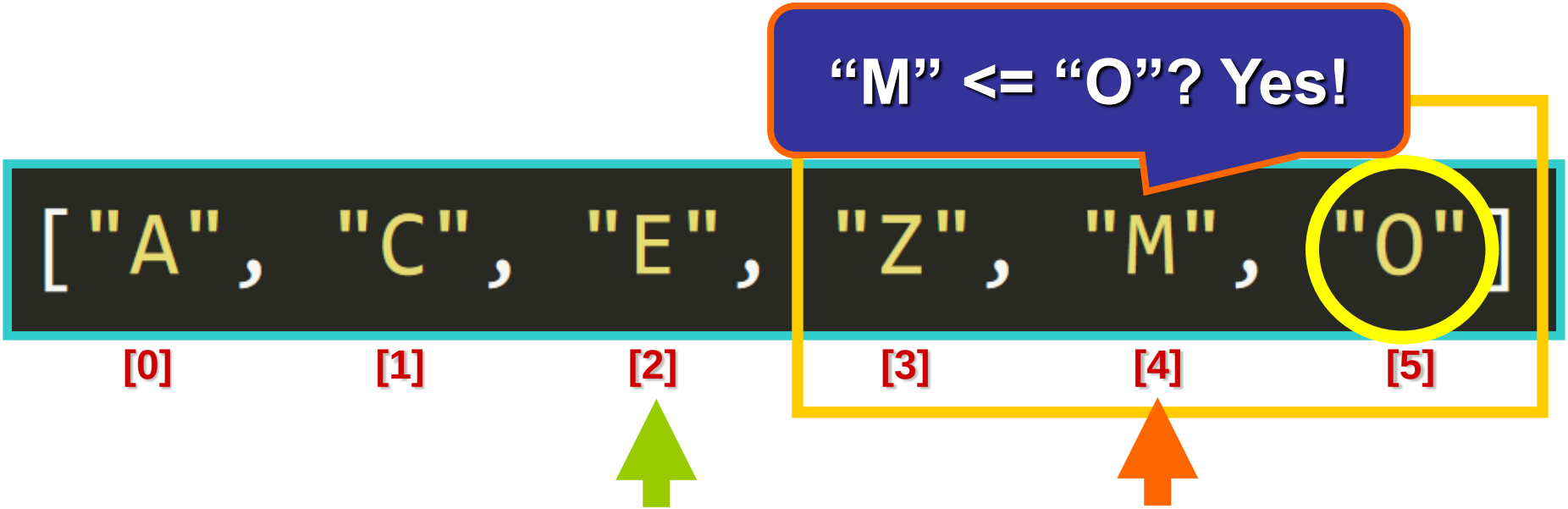
Quicksort



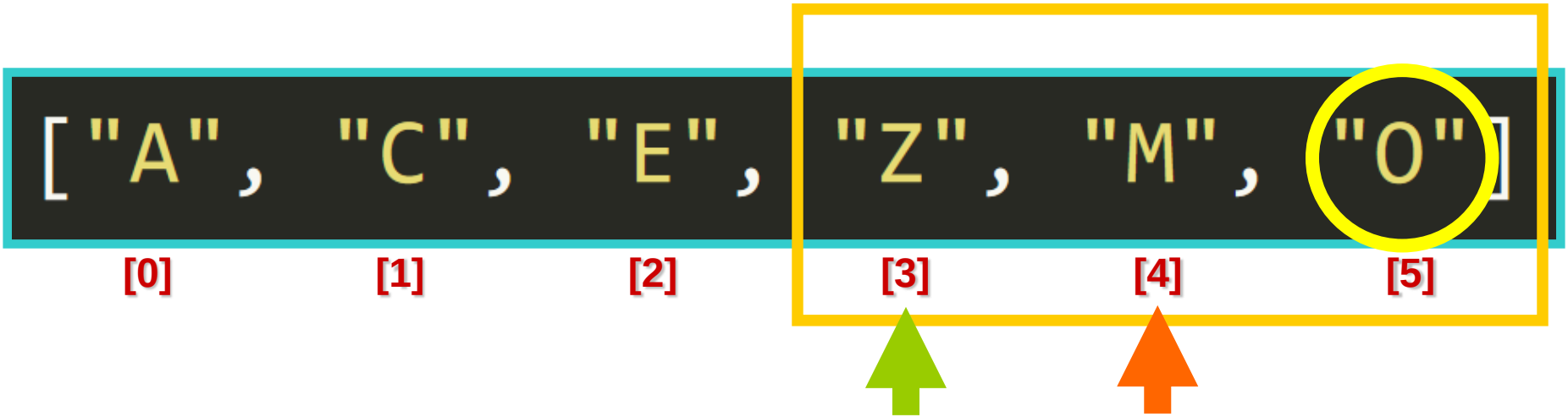
i = 2

j = 4

Quicksort



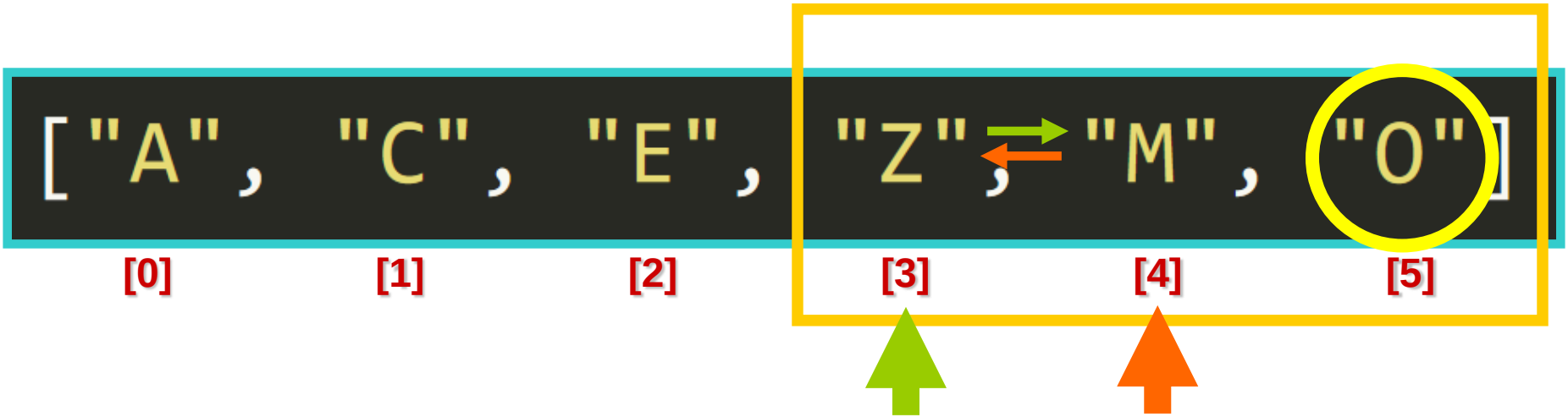
Quicksort



i = 3

j = 4

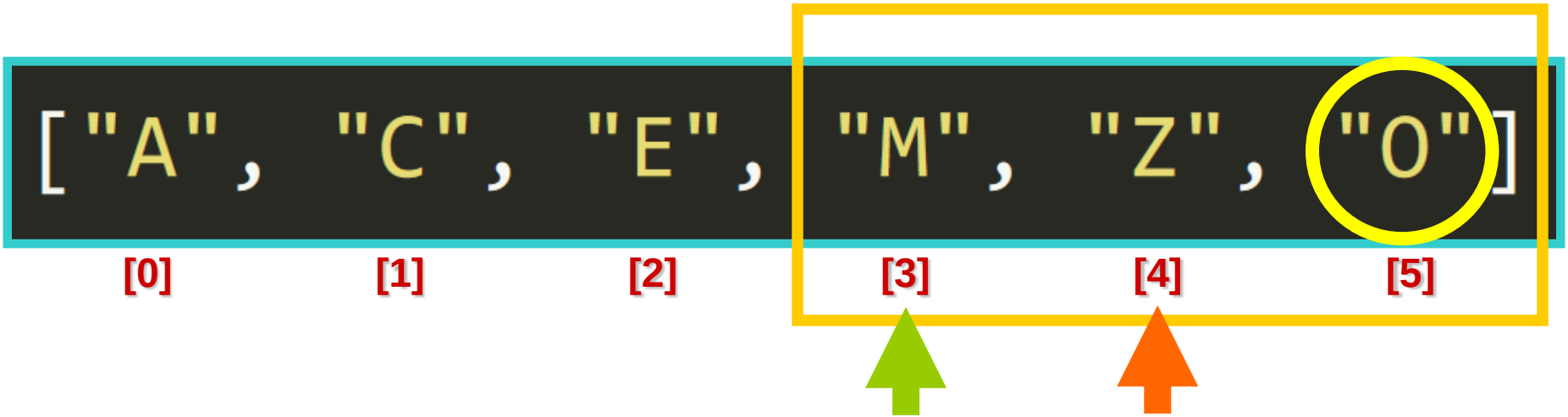
Quicksort



i = 3

j = 4

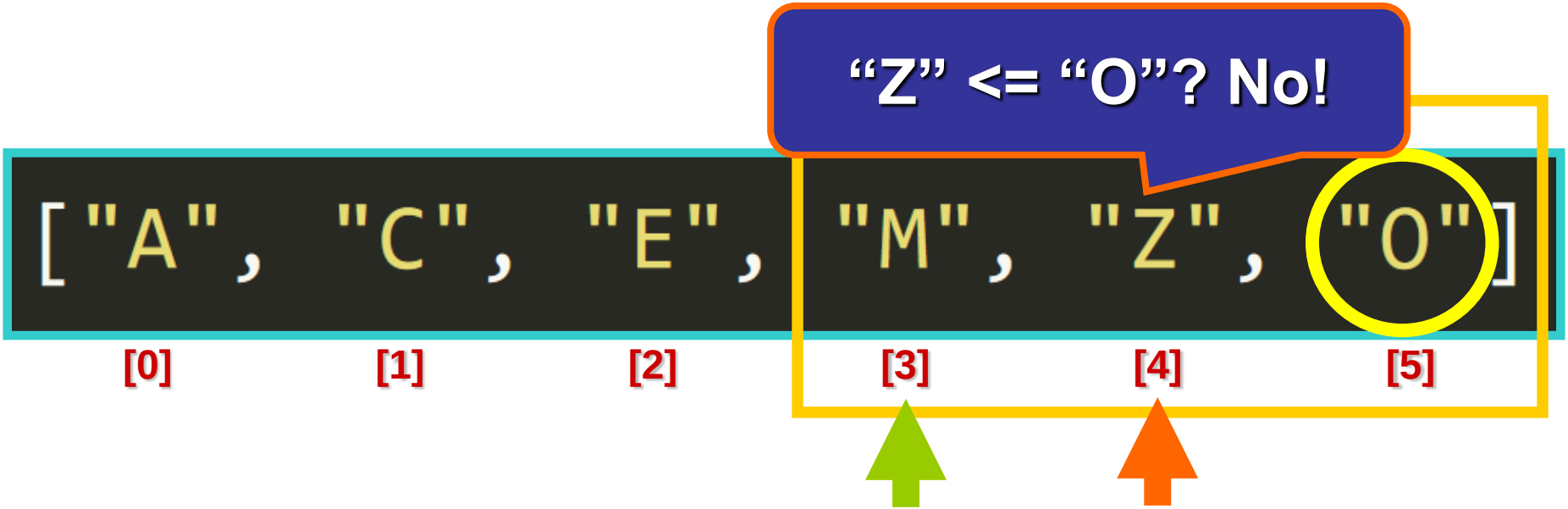
Quicksort

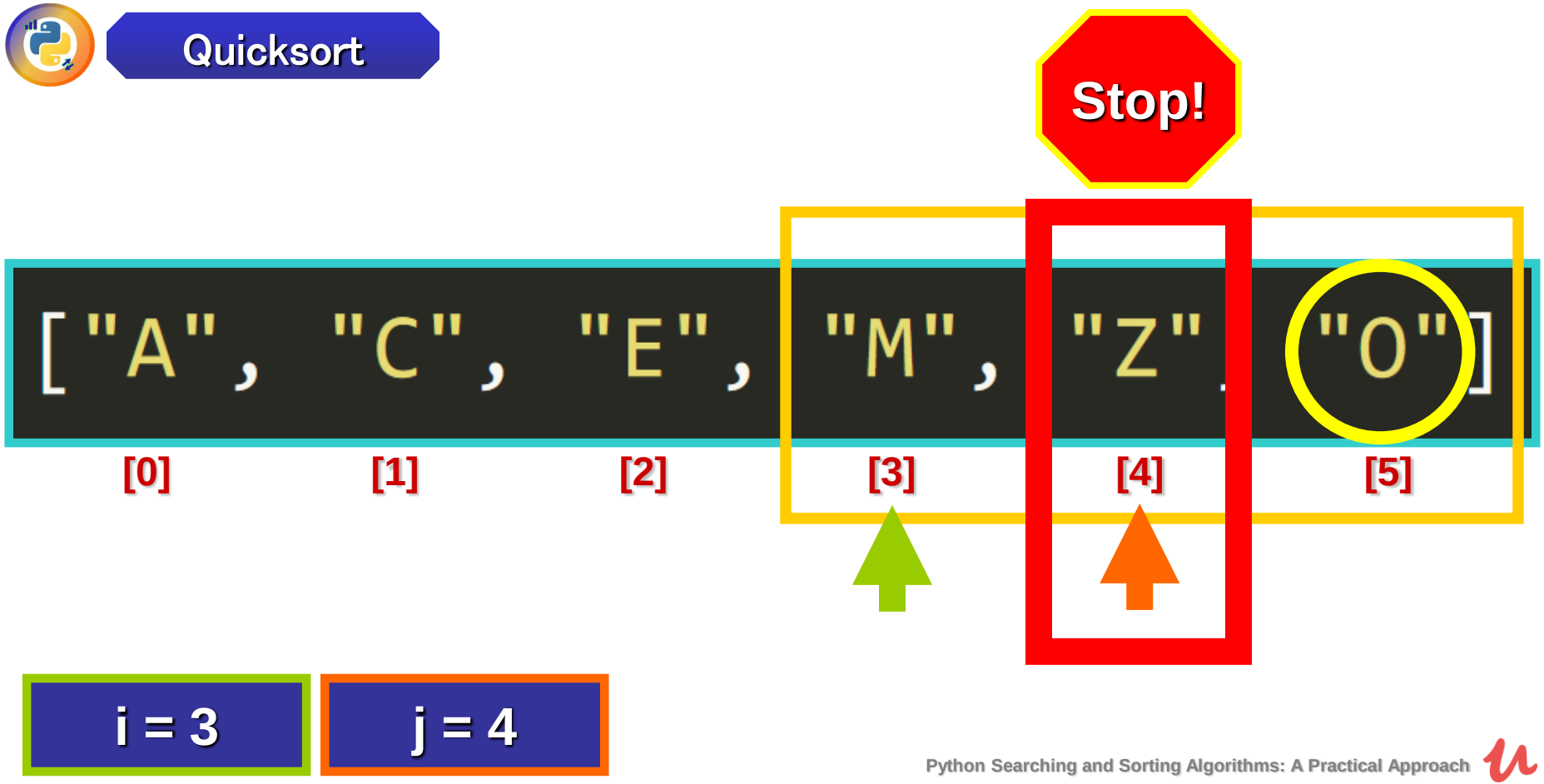


i = 3

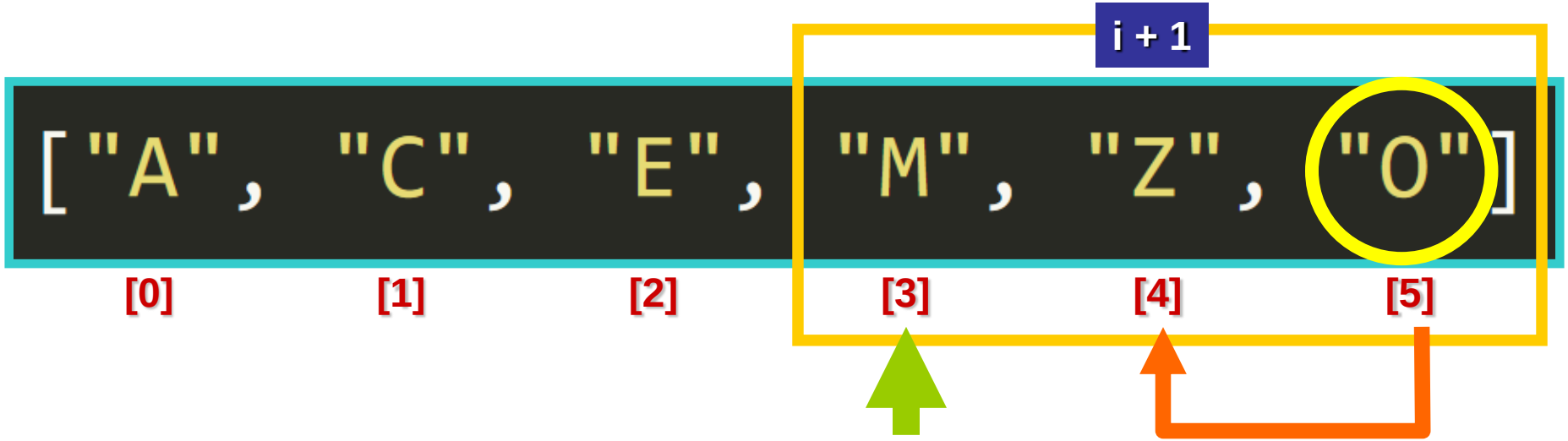
j = 4

Quicksort




Quicksort
Stop!
["A", "C", "E", "M", "Z", "O"]
[0]
[1]
[2]
[3]
[4]
[5]
i = 3
j = 4

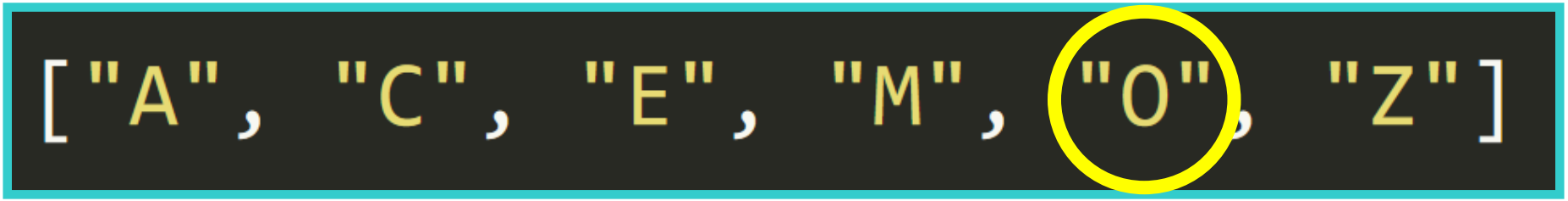
Quicksort



i = 3

j = 4

Quicksort



i = 3

j = 4

Quicksort

```
["A", "C", "E", "M", "O", "Z"]
```

Sorted!

Time to Code!

Algorithm

Quicksort

Example | Code

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

```
["Z", "C", "O", "A", "M", "E"]
```

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

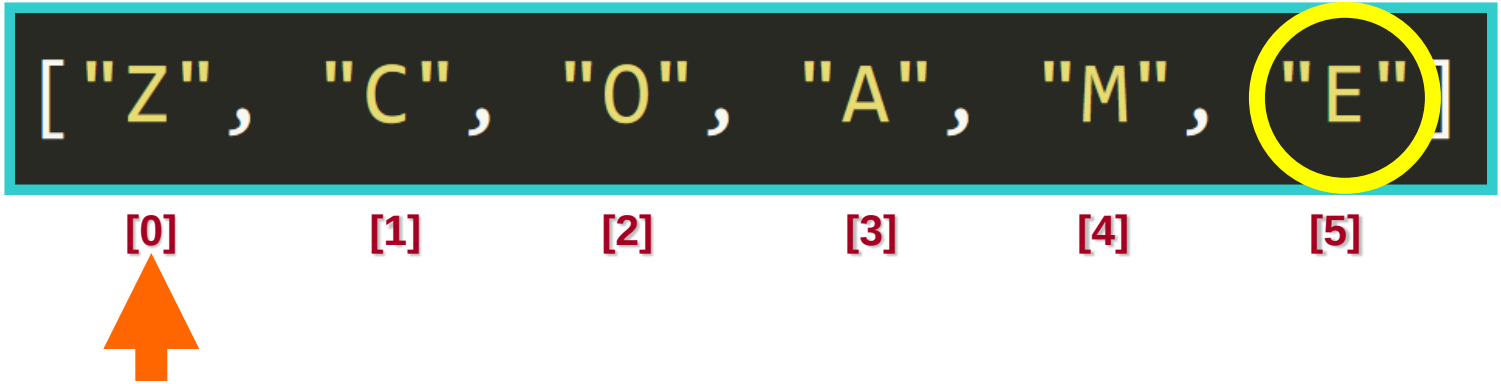
```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

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

```
["Z", "C", "O", "A", "M", "E"]
```

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

```
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)
```

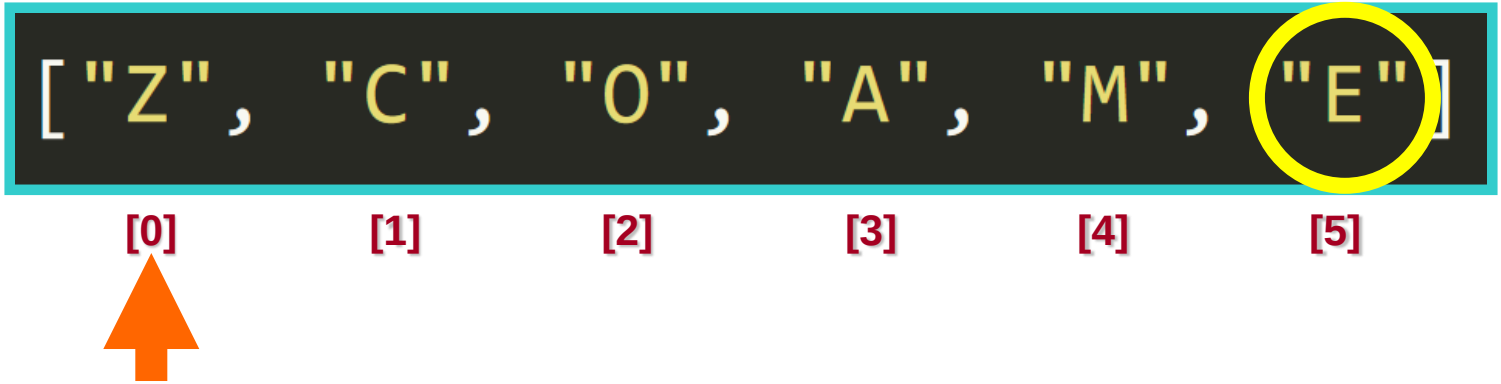


pivot = “E”

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 = “E”

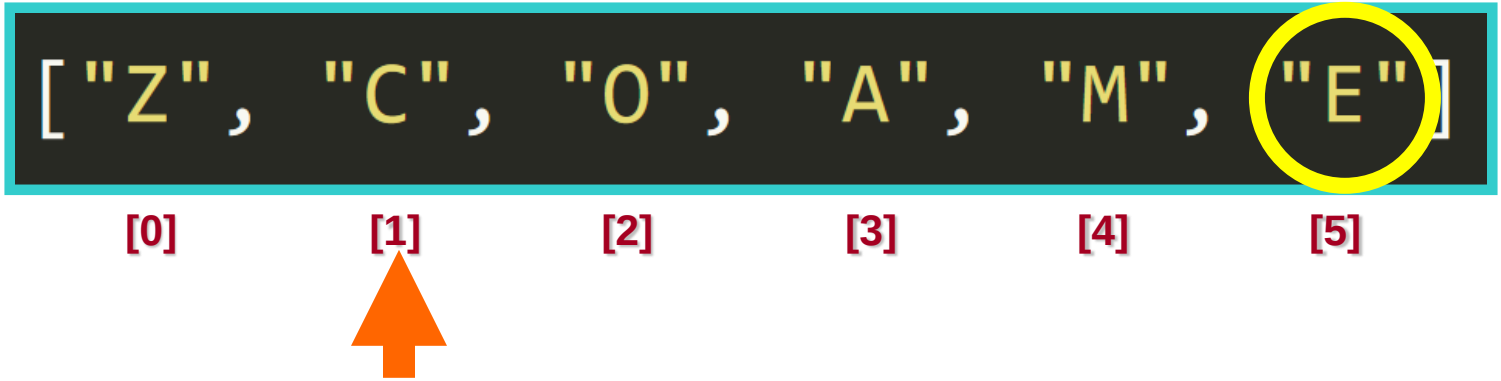
i = -1

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
```

“Z” <= “E”?

No!

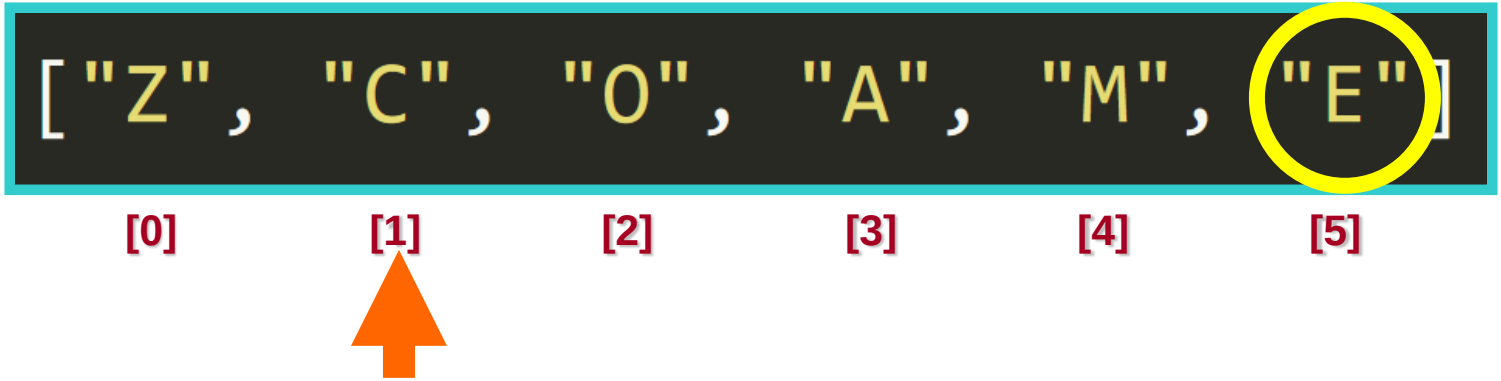


pivot = “E”

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
```



pivot = “E”

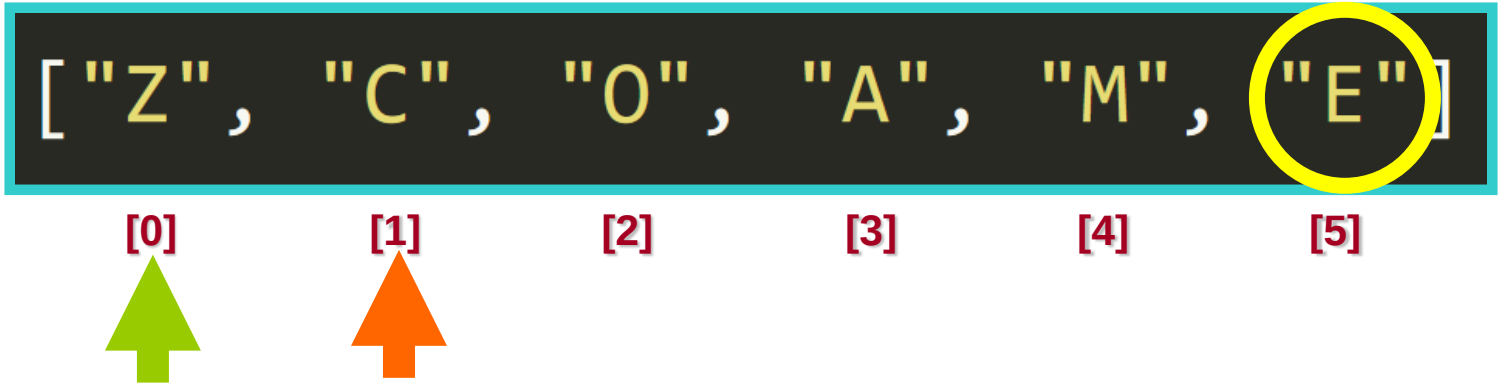
i = -1

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
```

“C” <= “E”?

Yes!

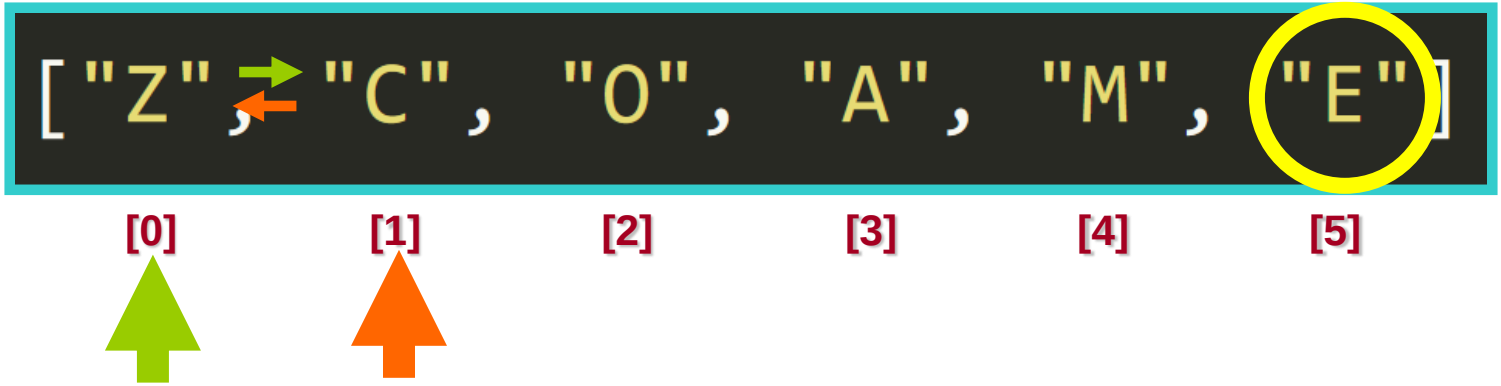


pivot = “E”

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
```

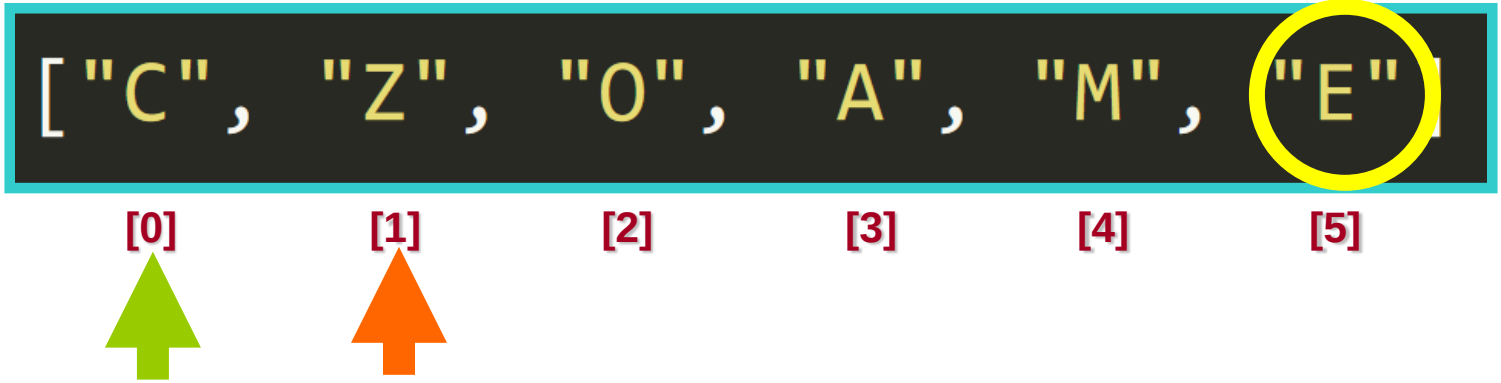


pivot = “E”

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
```

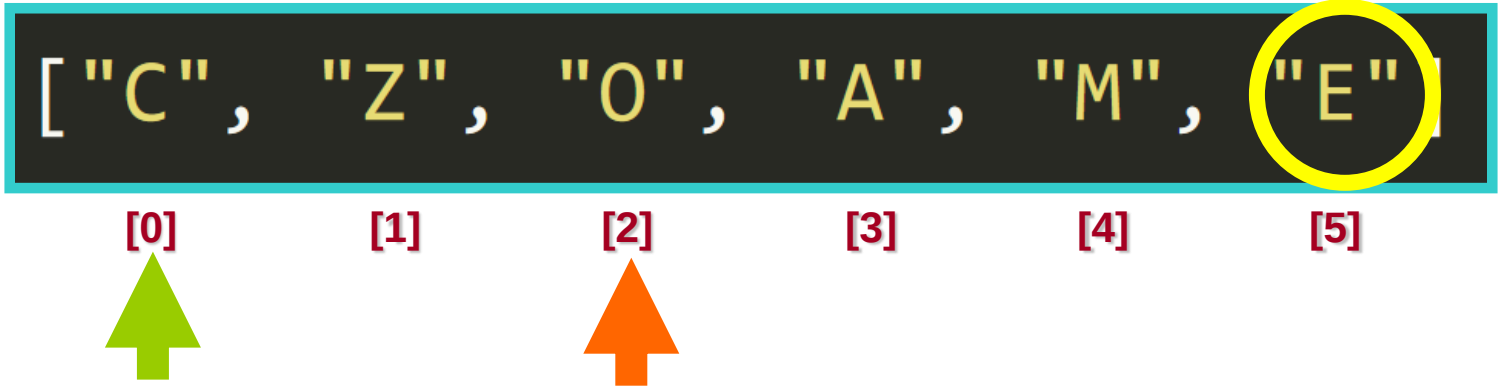


pivot = “E”

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 = “E”

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
```

pivot = “E”

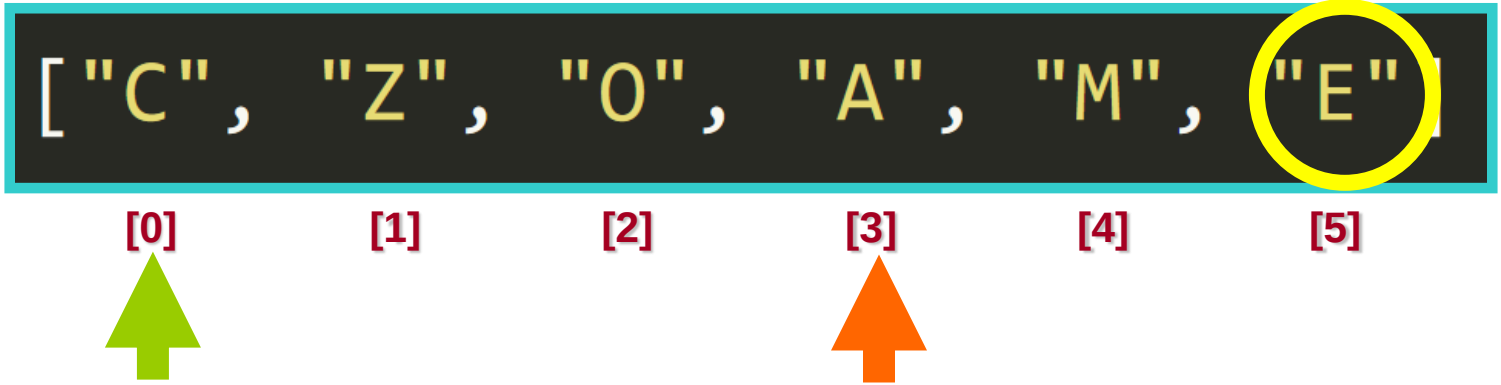
i = 0

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
```

“O” <= “E”?

No!

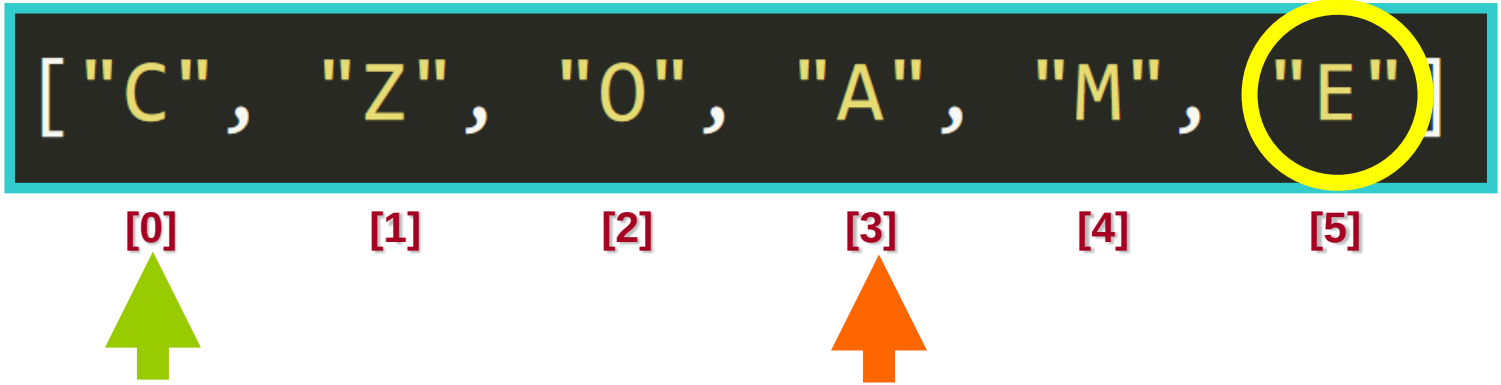


pivot = “E”

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
```



pivot = “E”

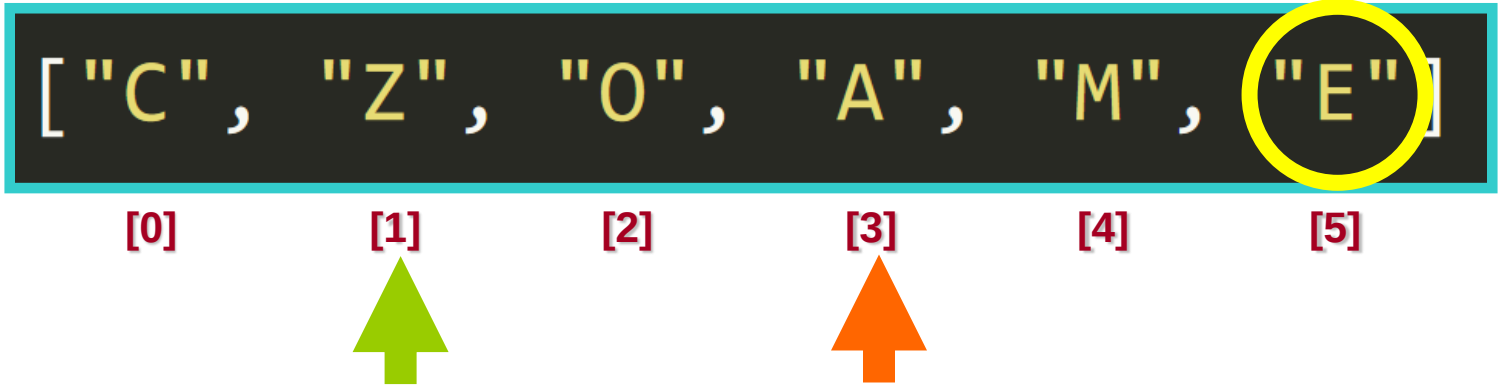
i = 0

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
```

“A” <= “E”?

Yes!

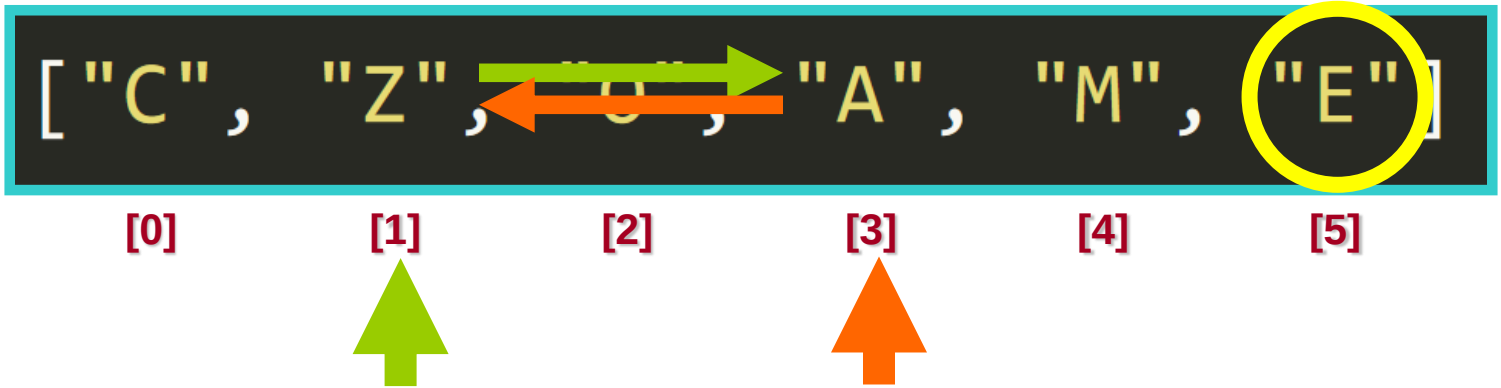


pivot = “E”

i = 1

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
```

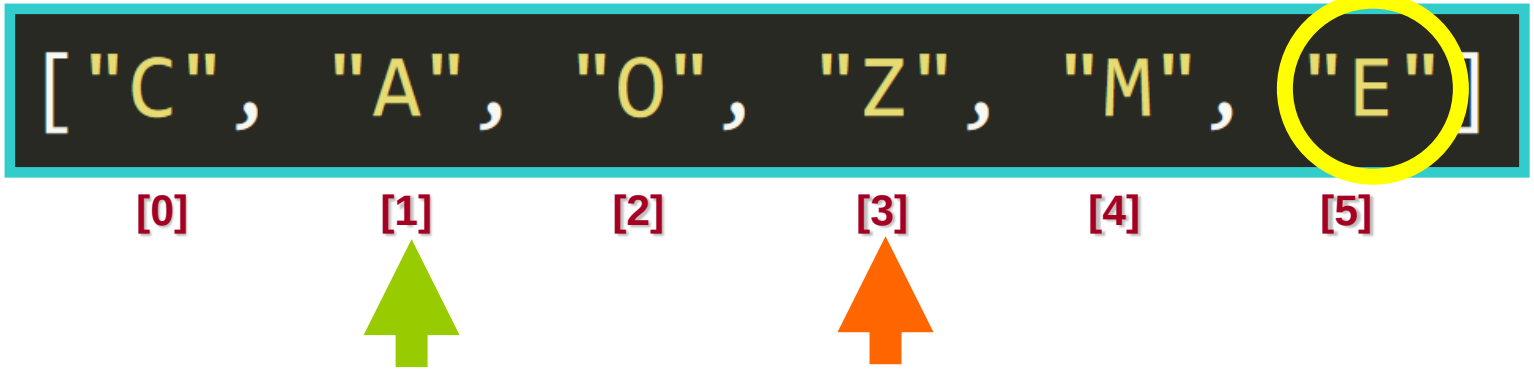


pivot = “E”

i = 1

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
```

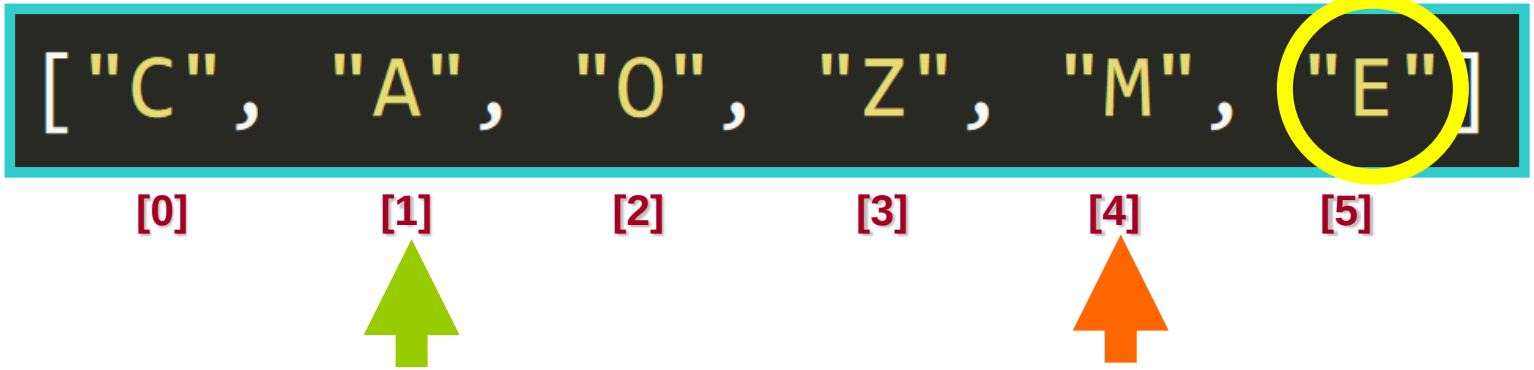


pivot = “E”

i = 1

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
```

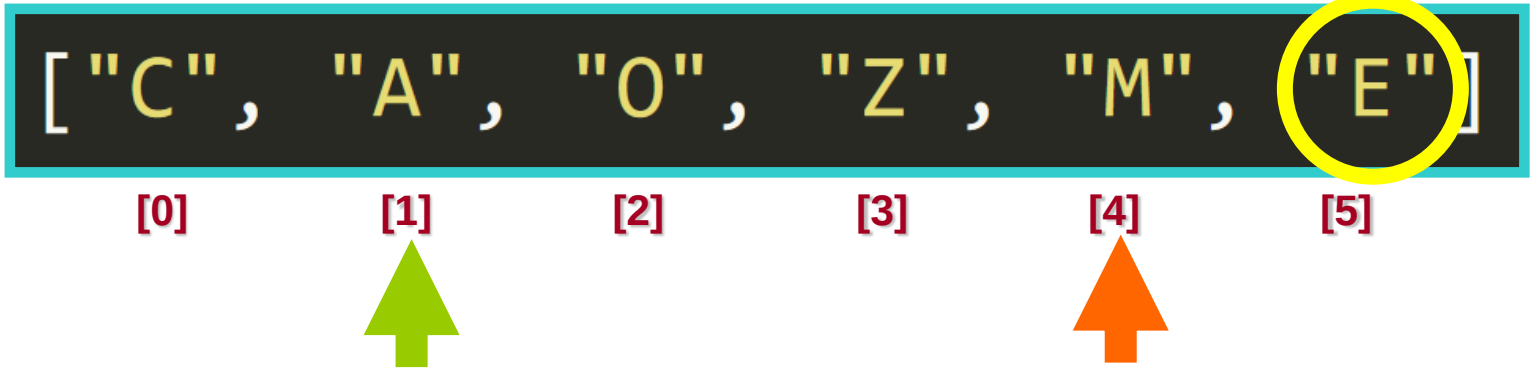


pivot = “E”

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
```



pivot = “E”

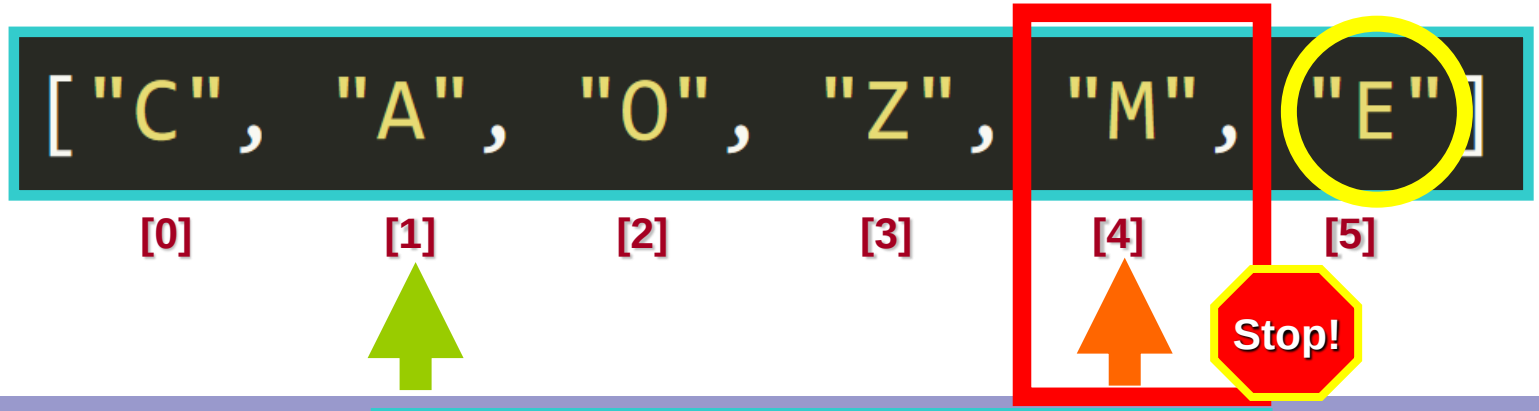
i = 1

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
```

“M” <= “E”?

No!

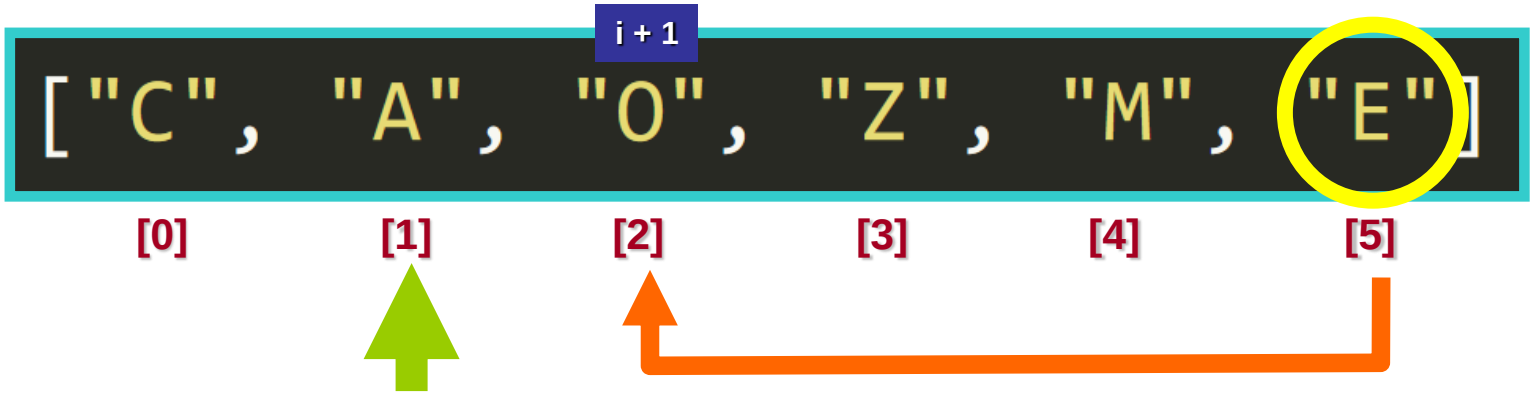


pivot = “E”

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
```



pivot = “E”

i = 1

j = 4

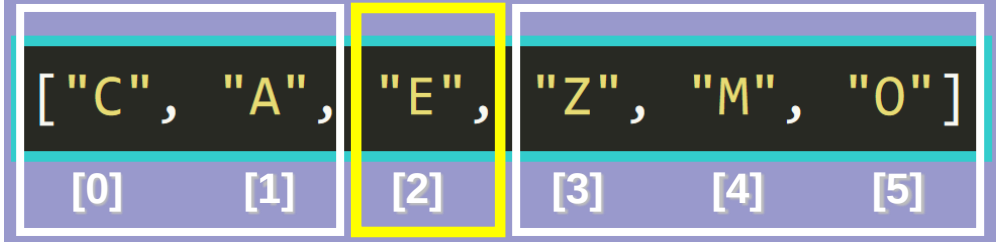
```
def partition(lst, low, high):
    pivot = lst[high]

    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
```

pivot = “E”

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
```



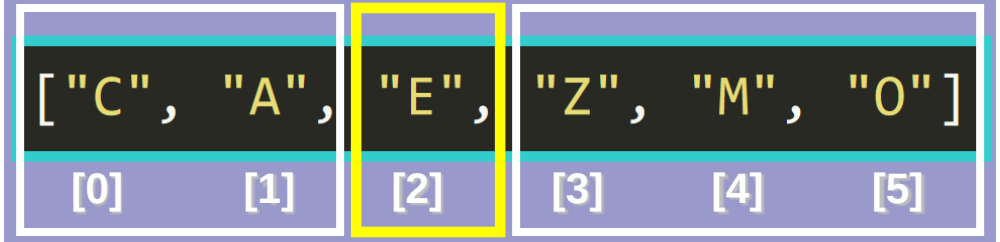

```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

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

2

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

<waiting recursive call>

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

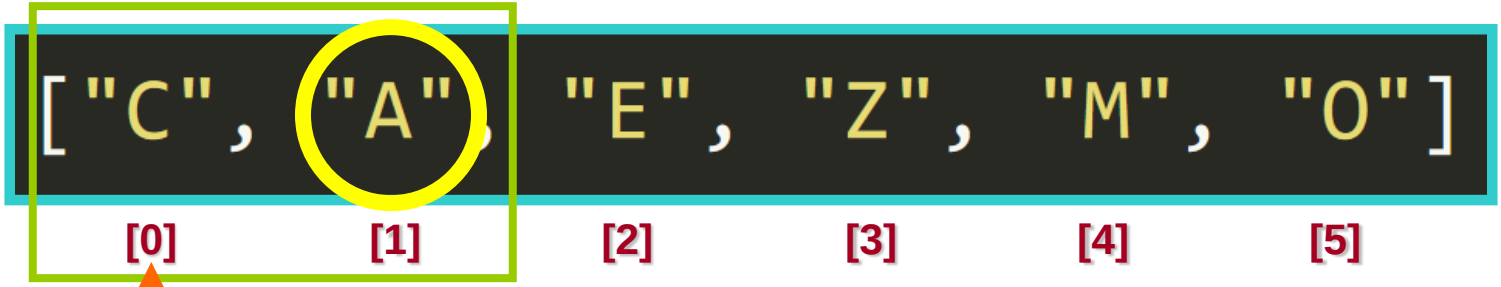



```python
def quicksort(lst, low, high):
    if low < high:

        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, 1)
```



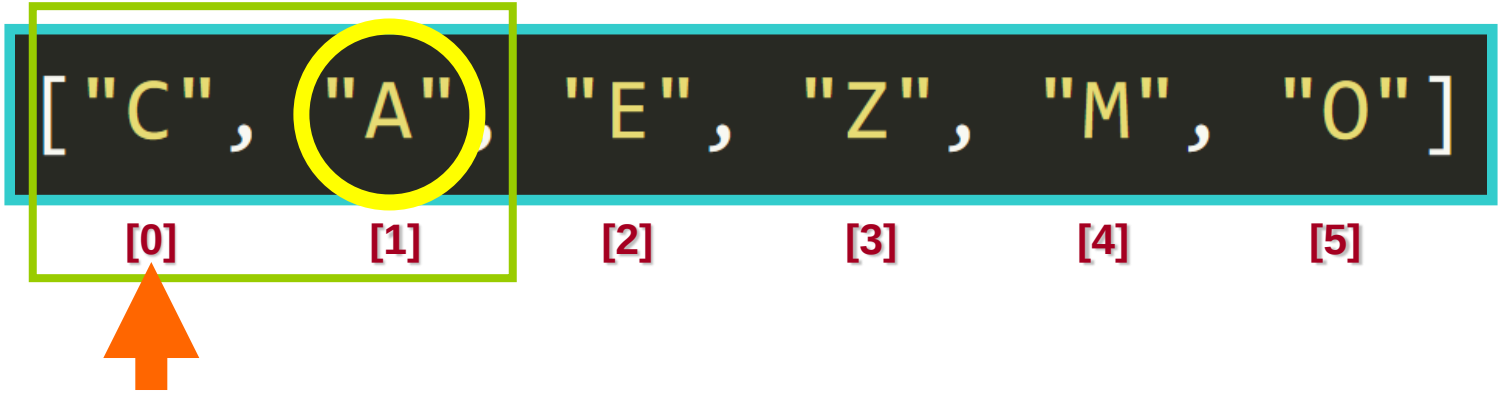

```
partition(lst, 0, 1)
```

pivot = “A”

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 = “A”

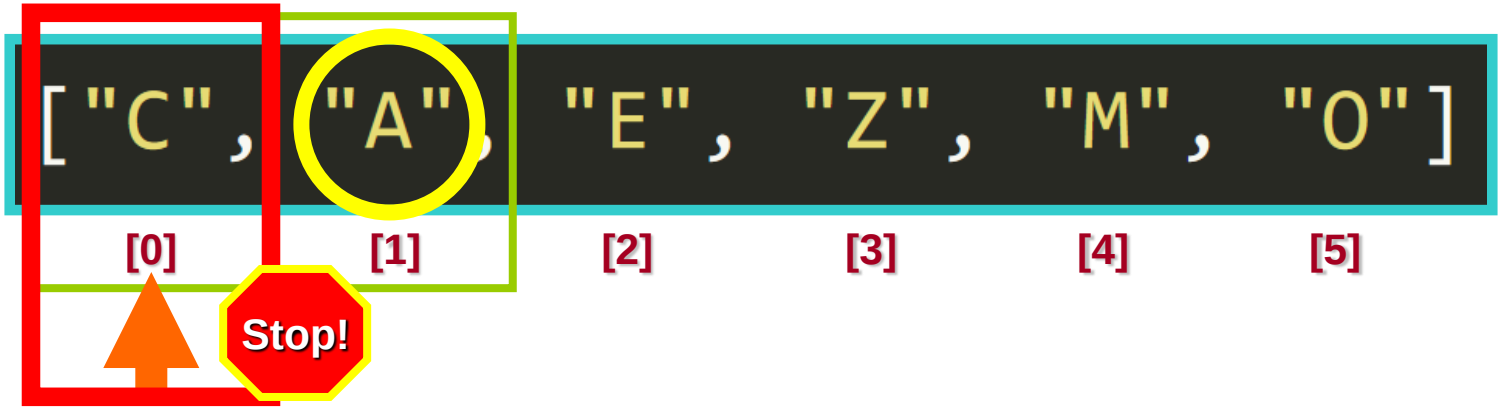
i = -1

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
```

“C” <= “A”?

No!

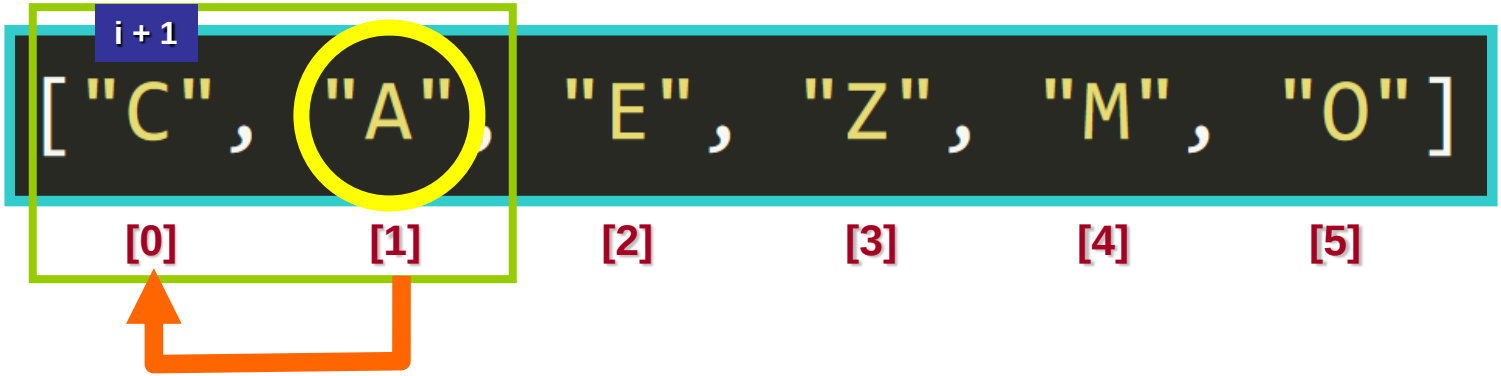


pivot = “A”

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
```

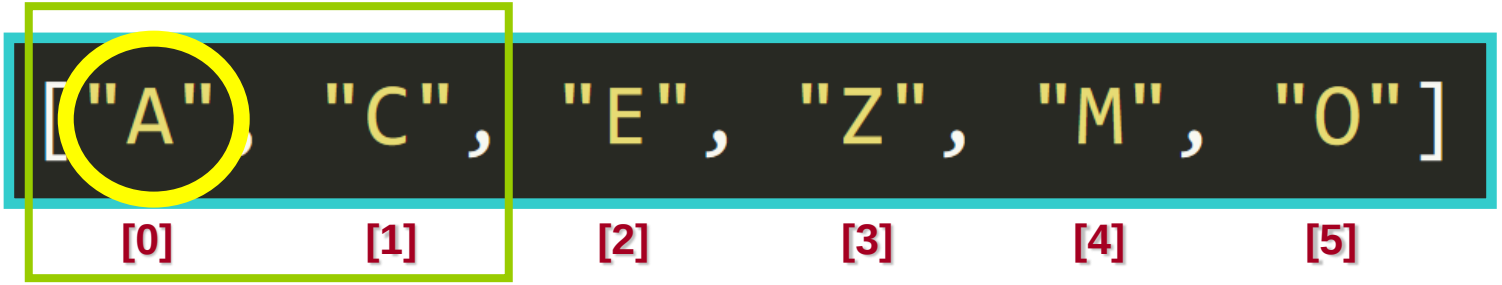


pivot = “A”

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
```

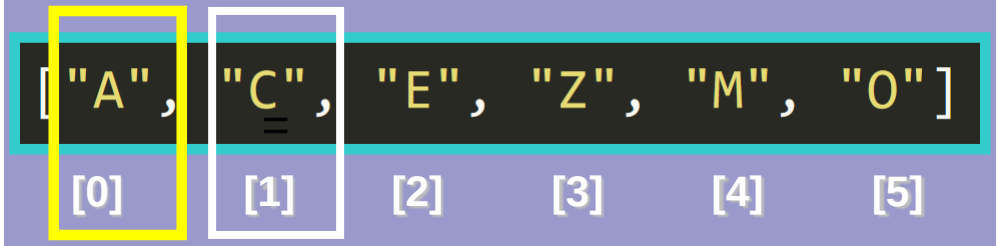


pivot = “A”

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
```




```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)
```

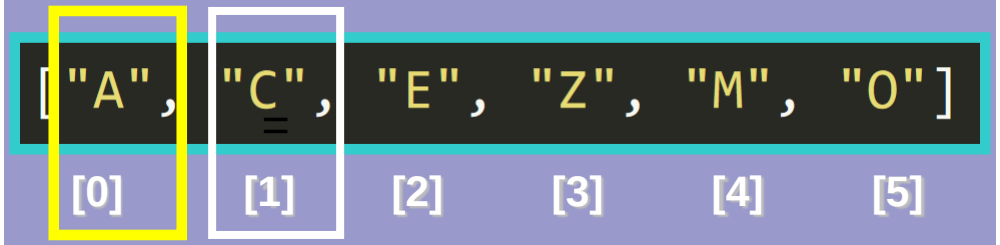
```
quicksort(a, 0, len(a)-1)
```

<waiting recursive call>

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

<waiting recursive call>

0




```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, 1)
```

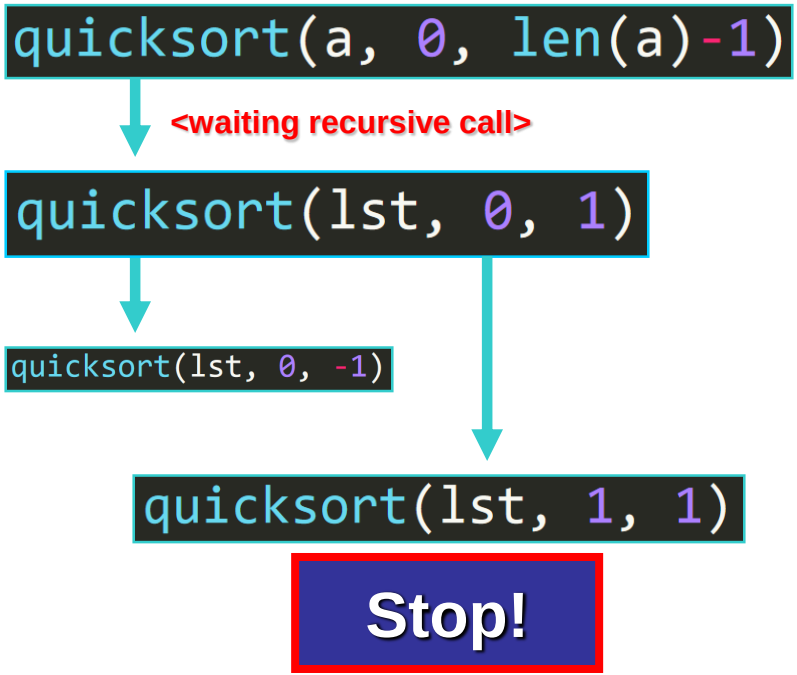
<waiting recursive call>

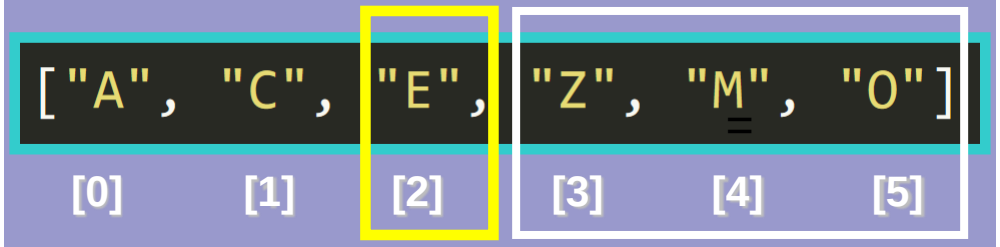
```
quicksort(lst, 0, -1)
```

Stop!

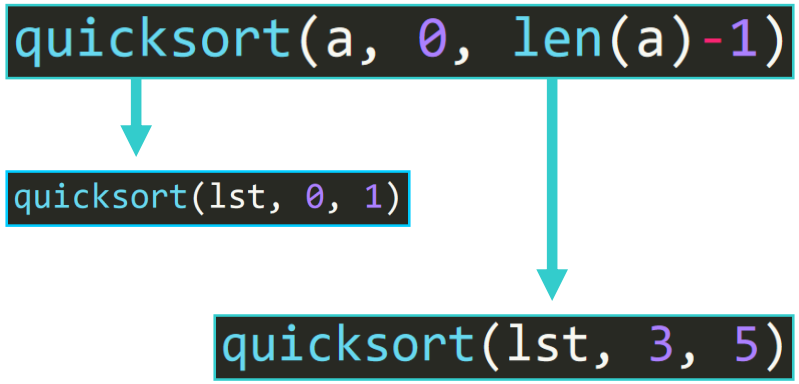
```
["A", "C", "E", "Z", "M", "O"]
 [0]   [1]   [2]   [3]   [4]   [5]
```

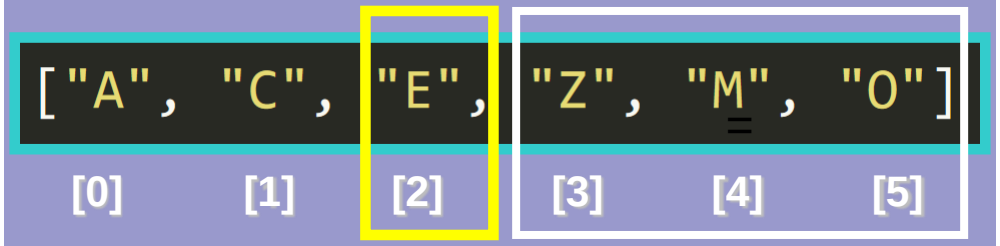
```python
def quicksort(lst, low, high):
    if low < high:

        pivot_index = partition(lst, low, high)

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






```
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)
```



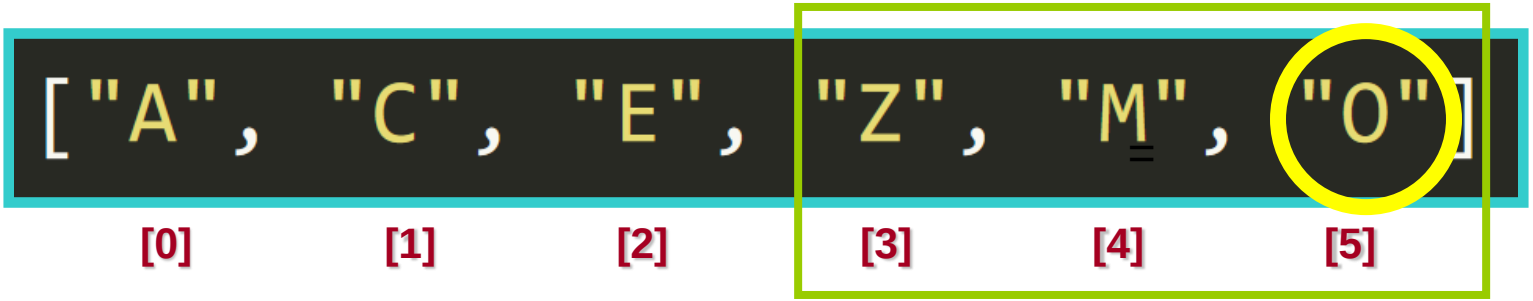



```python
def quicksort(lst, low, high):
  if low < high:

    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, 1)
```

```
quicksort(lst, 3, 5)
```



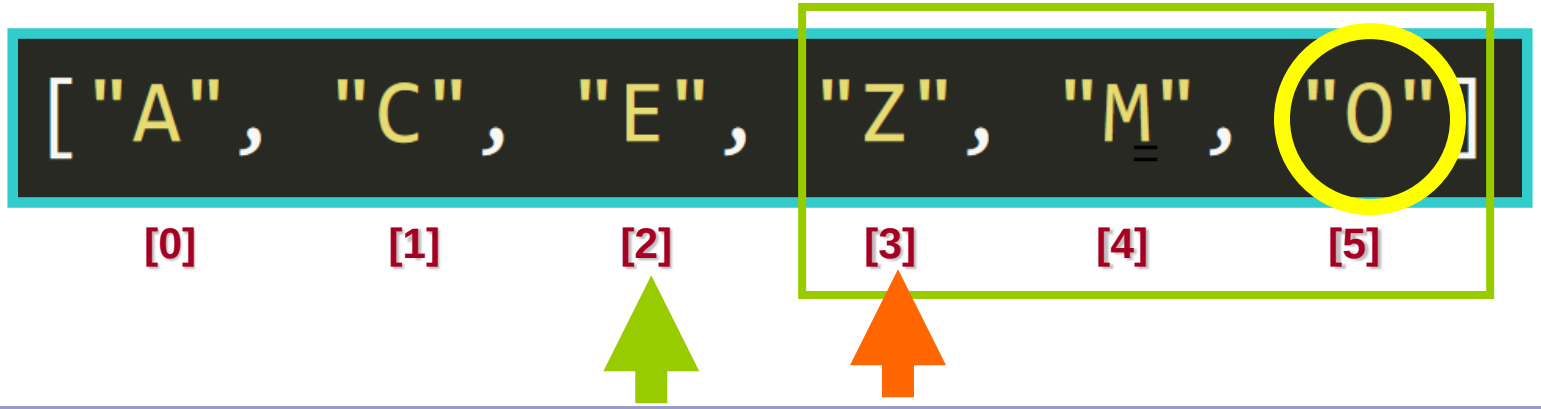

```
partition(lst, 3, 5)
```

pivot = “O”

i = 2

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
```

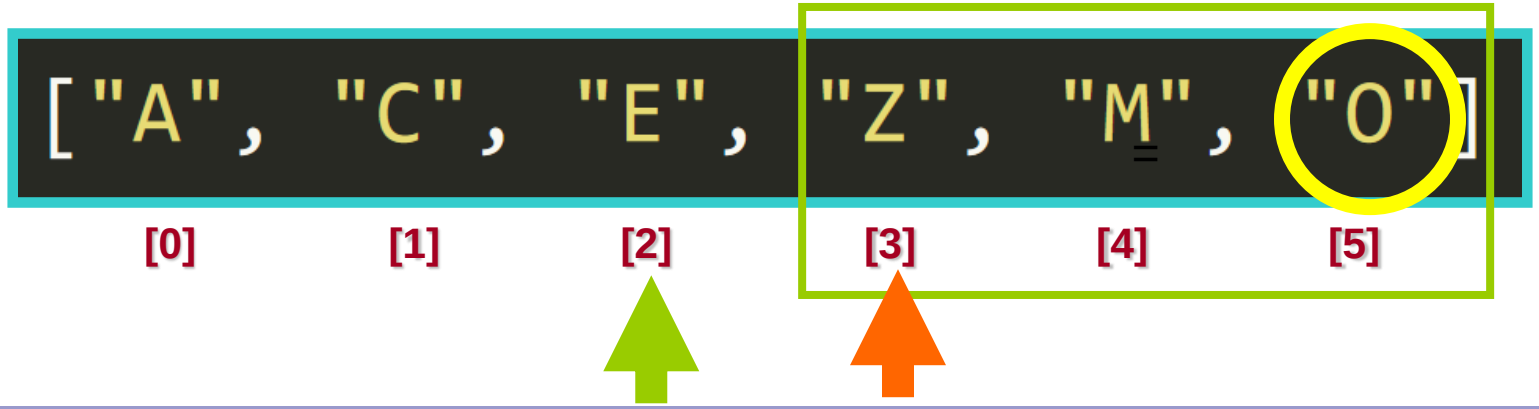


pivot = “O”

i = 2

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
```



pivot = “O”

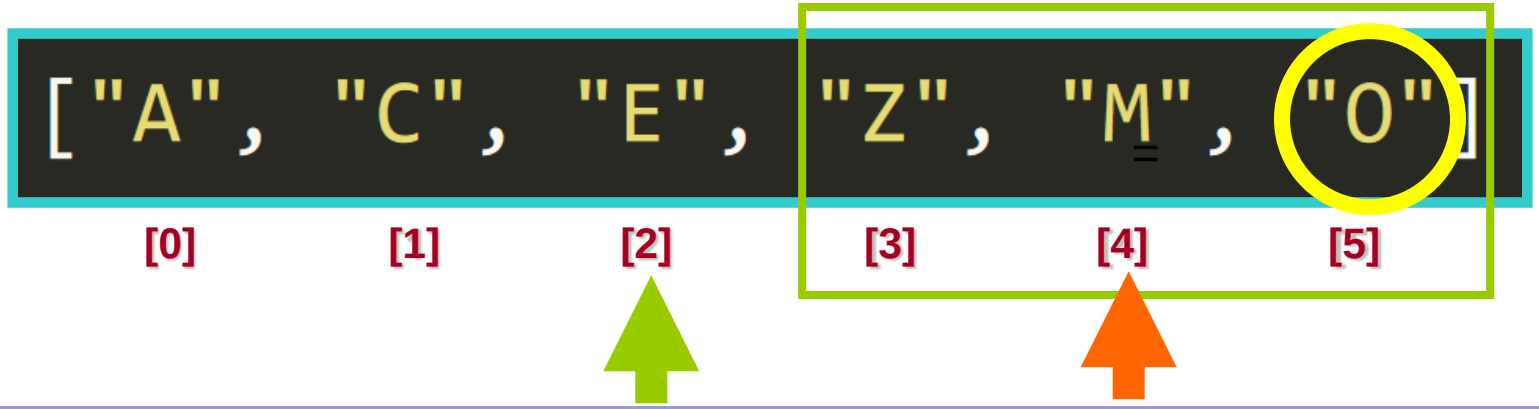
i = 2

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
```

“Z” <= “O”?

No!



pivot = “O”

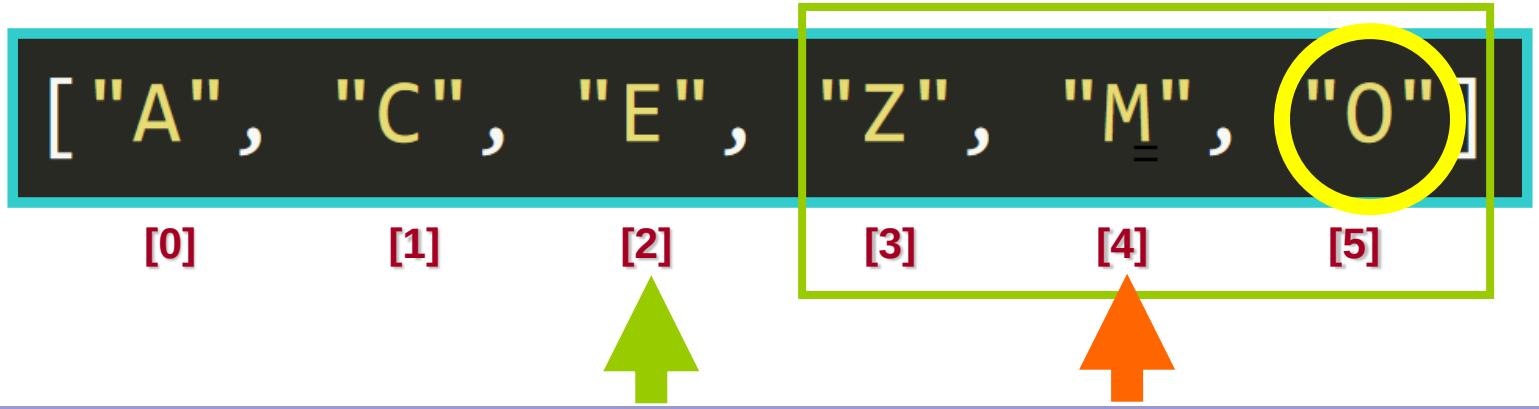
i = 2

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
```

“Z” <= “O”?

No!



pivot = “O”

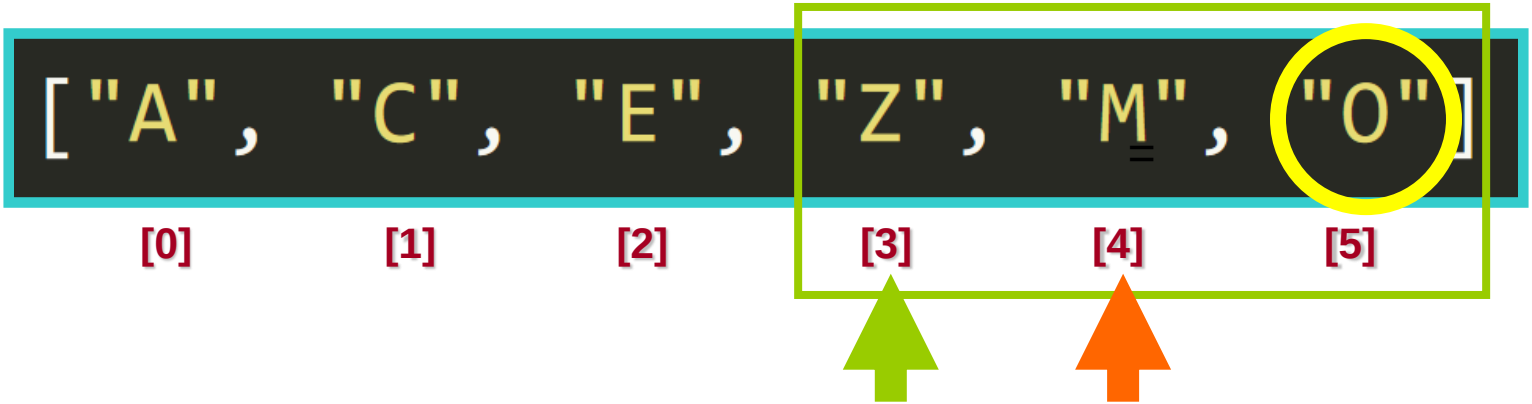
i = 2

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
```

“M” <= “O”?

Yes!

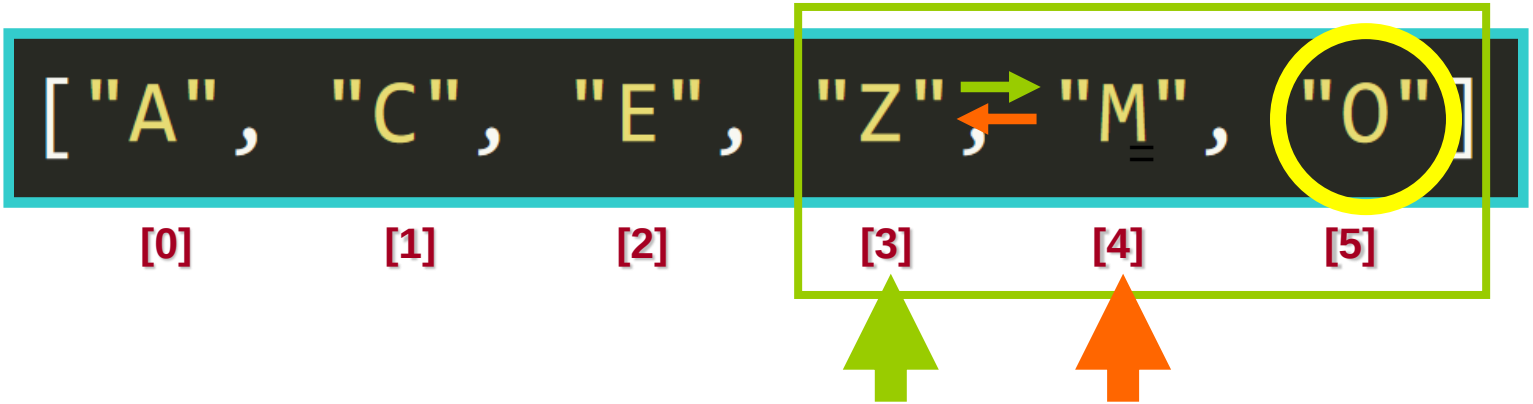


pivot = “O”

i = 3

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
```

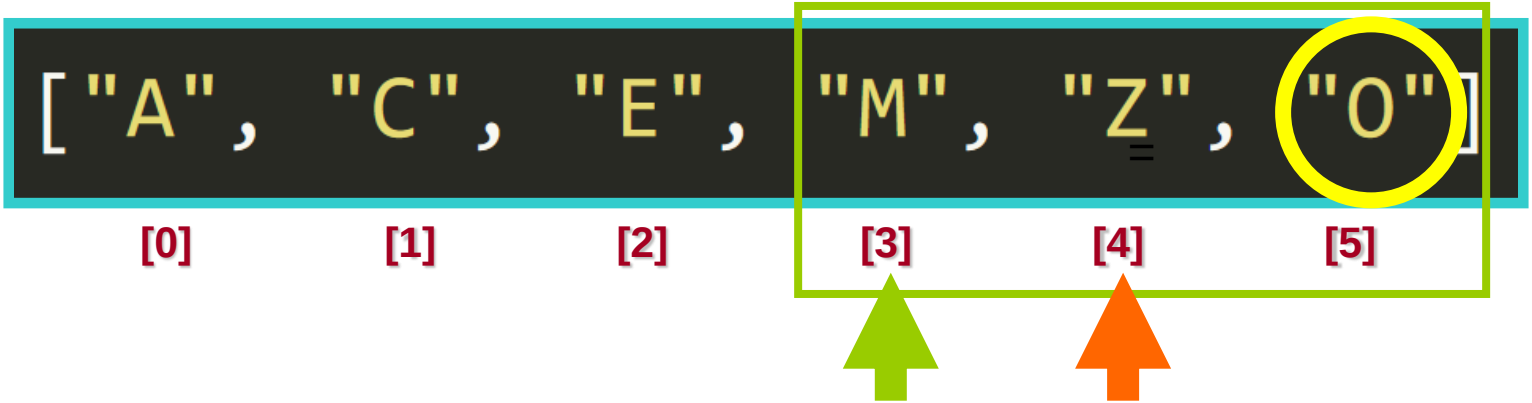


pivot = “O”

i = 3

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
```

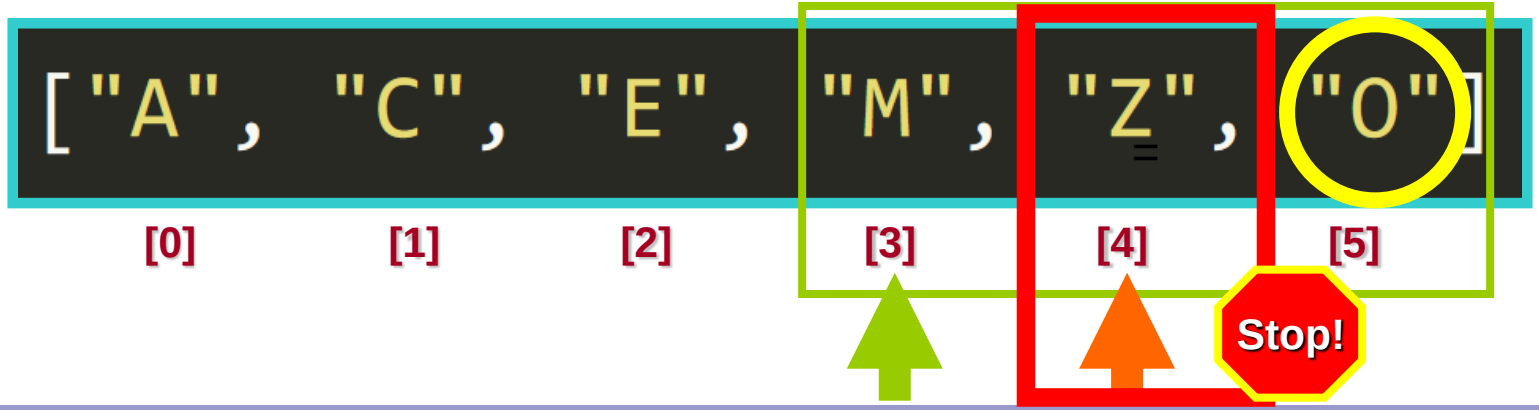


pivot = “O”

i = 3

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
```

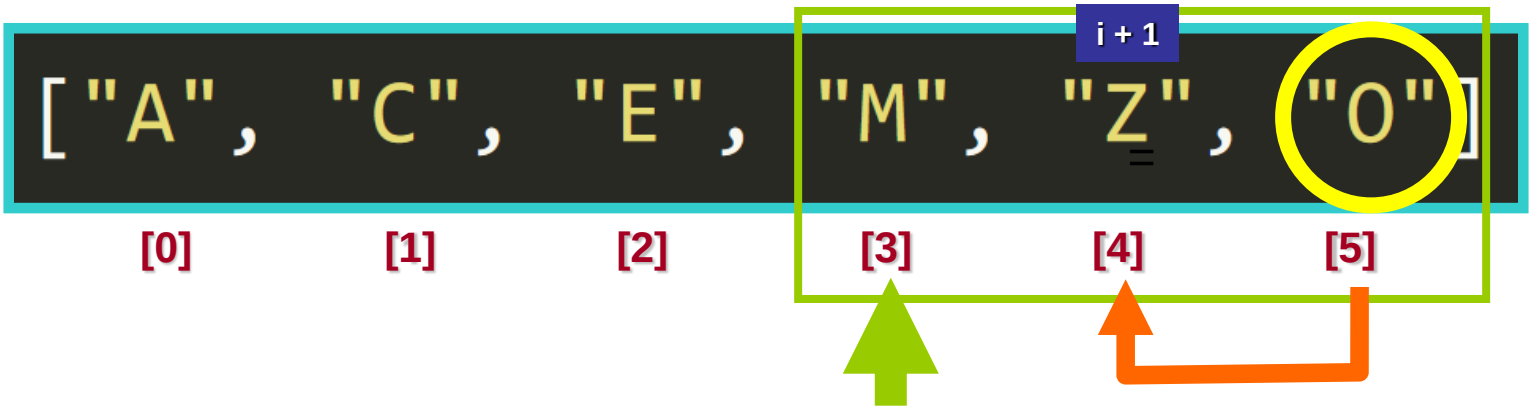


pivot = “O”

i = 3

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
```

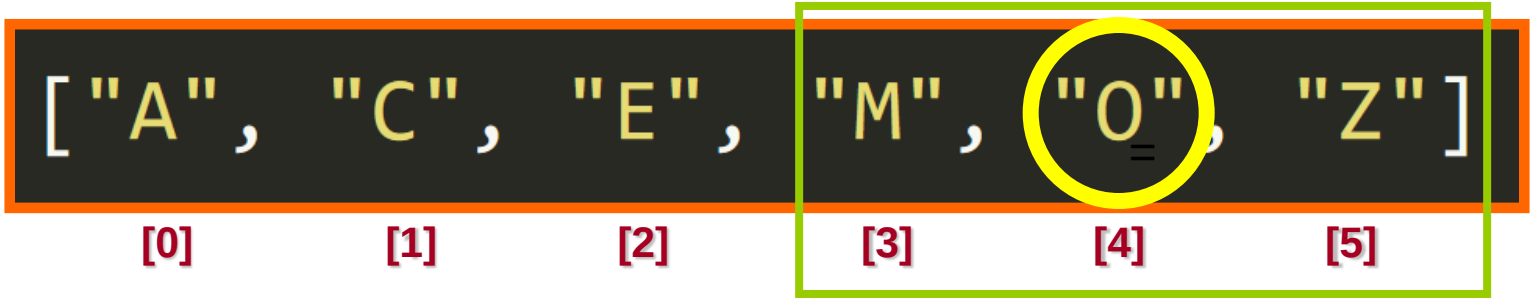


pivot = “O”

i = 3

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
```



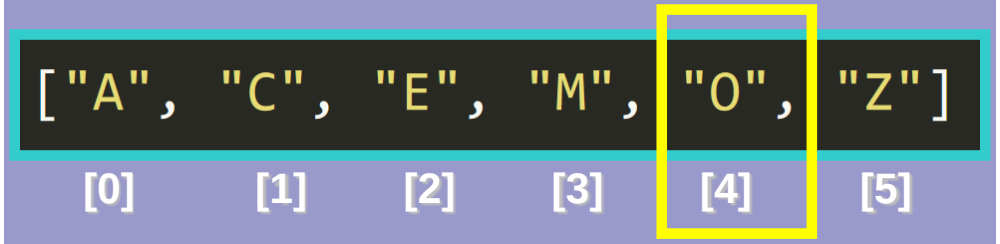
pivot = “O”

i = 3

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
```

Return 4



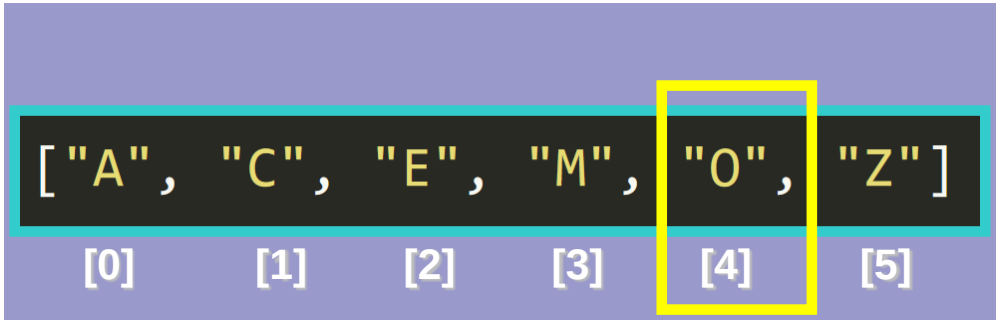

```
def quicksort(lst, low, high):
  if low < high:

    pivot_index = partition(lst, low, high)

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

4

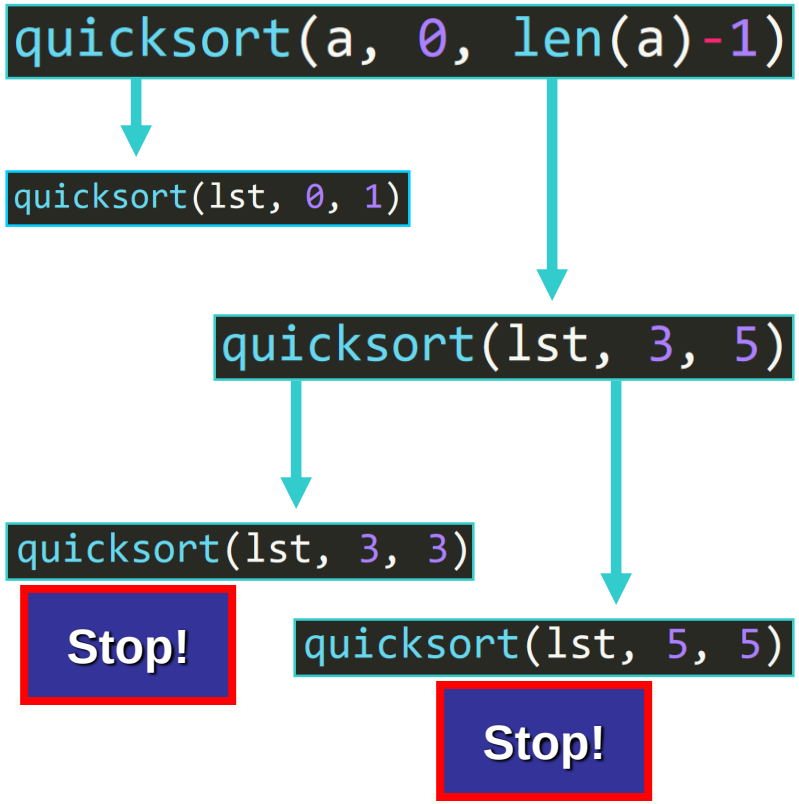
```
quicksort(a, 0, len(a)-1)
```

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

```
quicksort(lst, 3, 5)
```




```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", "C", "E", "M", "O", "Z"]
```

Time to Practice!