CITY
UNIVERSITY OF LONDON
EST 1894

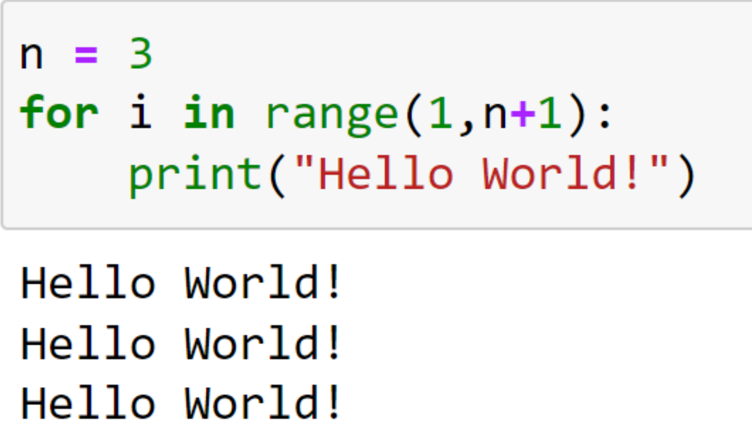
STEM Digital Academy

School of Science & Technology

www.city.ac.uk

Ordered Linear Searches

Programming and Algorithms



```
n = 3
for i in range(1,n+1):
    print("Hello World!")
```

```
Hello World!
Hello World!
Hello World!
```

Lecture by

Dr Daniil Osudin

What will we Cover?

- Ordered linear search algorithm
- Understanding the efficiency of the algorithm using big-O notation

Ordered Linear Search

- Search an ordered list for a value and return its index if the value is found.
- Ordered linear search can stop immediately when it has passed the possible position of the search value.

Ordered linear search example

Search key is: 27

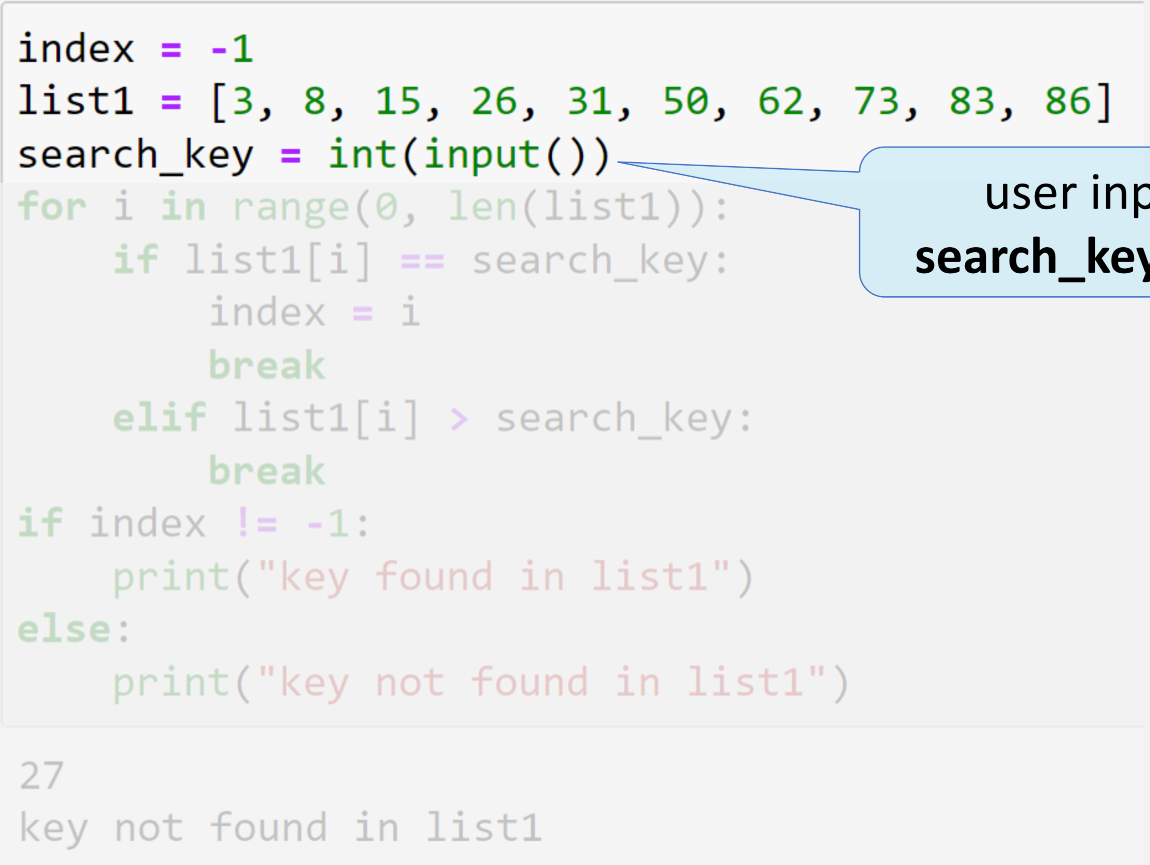
List is:

3	8	15	26	31	50	62	73	83	86

Ordered Linear Search Algorithm

```
set index = -1
input search key
for i in range len(list)
    if list[i] == key
        index = i
        break
    if list[i] > key
        break
if index != -1
    print(key found in list)
else
    print(key not found in list)
```

Ordered Linear Search Algorithm Example



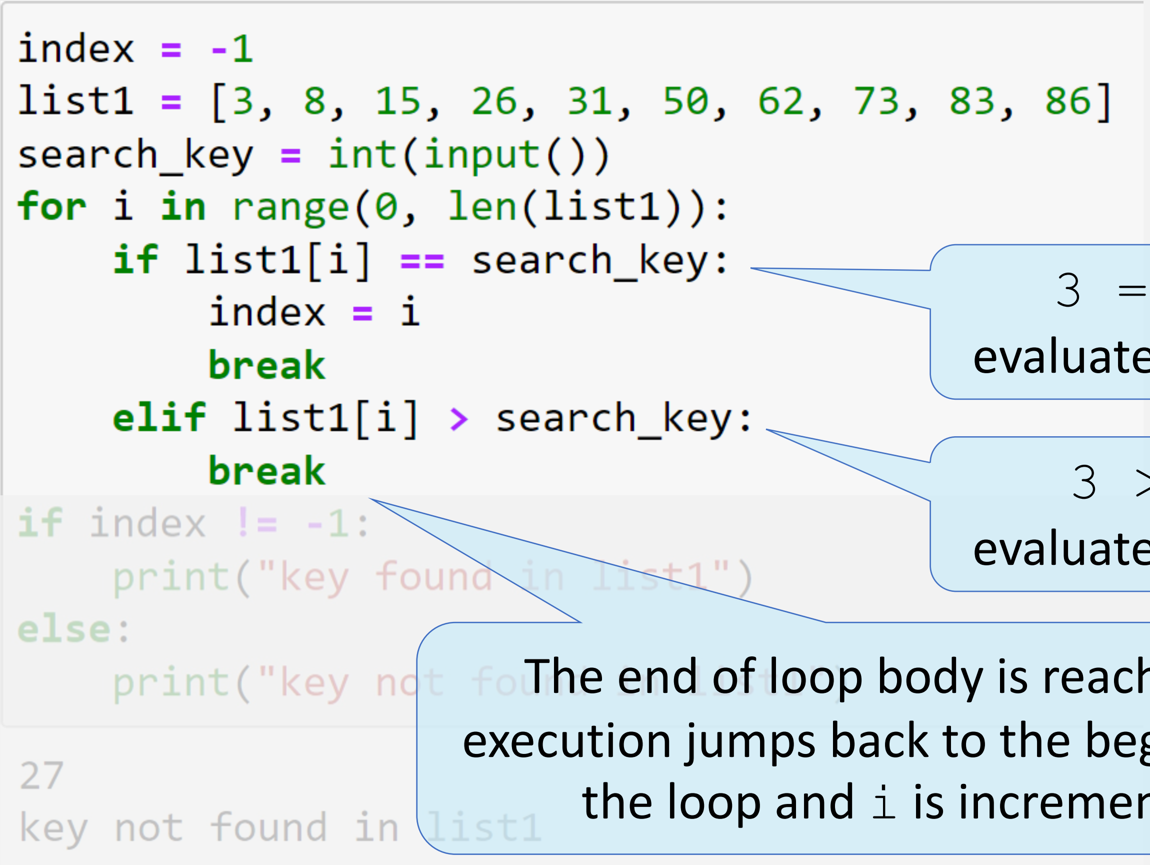
```python
index = -1
list1 = [3, 8, 15, 26, 31, 50, 62, 73, 83, 86]
search_key = int(input())
for i in range(0, len(list1)):
    if list1[i] == search_key:
        index = i
        break
    elif list1[i] > search_key:
        break
if index != -1:
    print("key found in list1")
else:
    print("key not found in list1")
```

```
27
key not found in list1
```

user input
search_key is 27

```
search_key = 27
index = -1
i = ?
list1[?] = ?
```

Ordered Linear Search Algorithm Example



```
index = -1
list1 = [3, 8, 15, 26, 31, 50, 62, 73, 83, 86]
search_key = int(input())
for i in range(0, len(list1)):
    if list1[i] == search_key:
        index = i
        break
    elif list1[i] > search_key:
        break
if index != -1:
    print("key found in list1")
else:
    print("key not found in list1")
```

```
27
key not found in list1
```

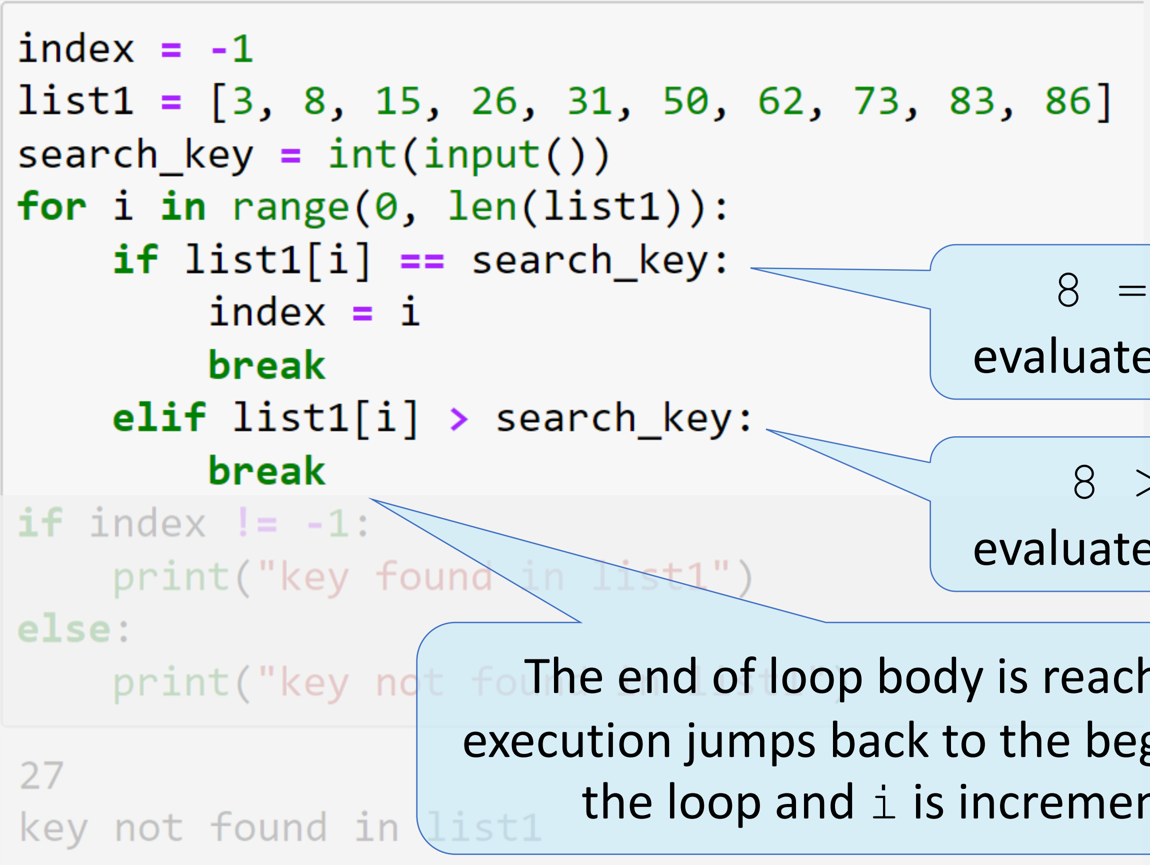
`3 == 27`
evaluates to False

`3 > 27`
evaluates to False

The end of loop body is reached, so execution jumps back to the beginning of the loop and `i` is incremented

```
search_key = 27
index = -1
i = 0
list1[0] = 3
```

Ordered Linear Search Algorithm Example



```
index = -1
list1 = [3, 8, 15, 26, 31, 50, 62, 73, 83, 86]
search_key = int(input())
for i in range(0, len(list1)):
    if list1[i] == search_key:
        index = i
        break
    elif list1[i] > search_key:
        break
if index != -1:
    print("key found in list1")
else:
    print("key not found in list1")
```

```
27
key not found in list1
```

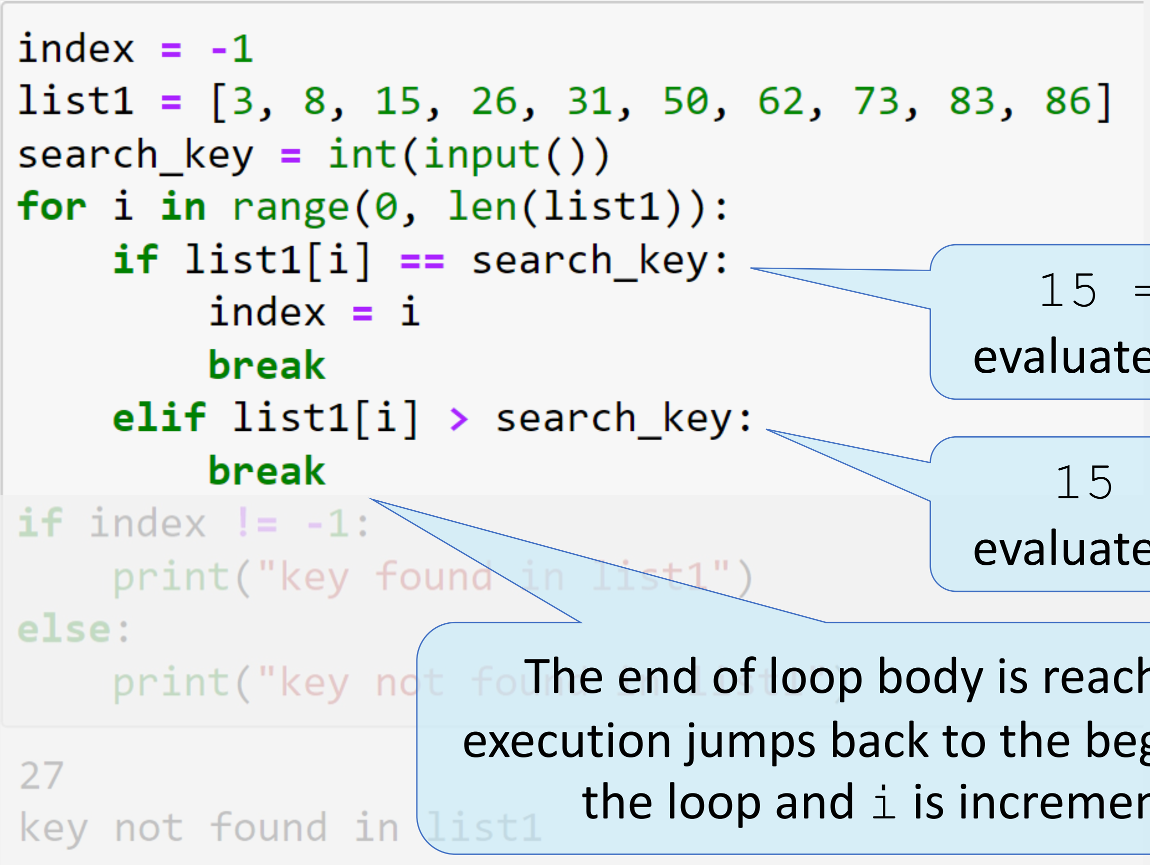
`8 == 27`
evaluates to False

`8 > 27`
evaluates to False

The end of loop body is reached, so execution jumps back to the beginning of the loop and `i` is incremented

```
search_key = 27
index = -1
i = 1
list1[1] = 8
```

Ordered Linear Search Algorithm Example



```
index = -1
list1 = [3, 8, 15, 26, 31, 50, 62, 73, 83, 86]
search_key = int(input())
for i in range(0, len(list1)):
    if list1[i] == search_key:
        index = i
        break
    elif list1[i] > search_key:
        break
if index != -1:
    print("key found in list1")
else:
    print("key not found in list1")
```

```
27
key not found in list1
```

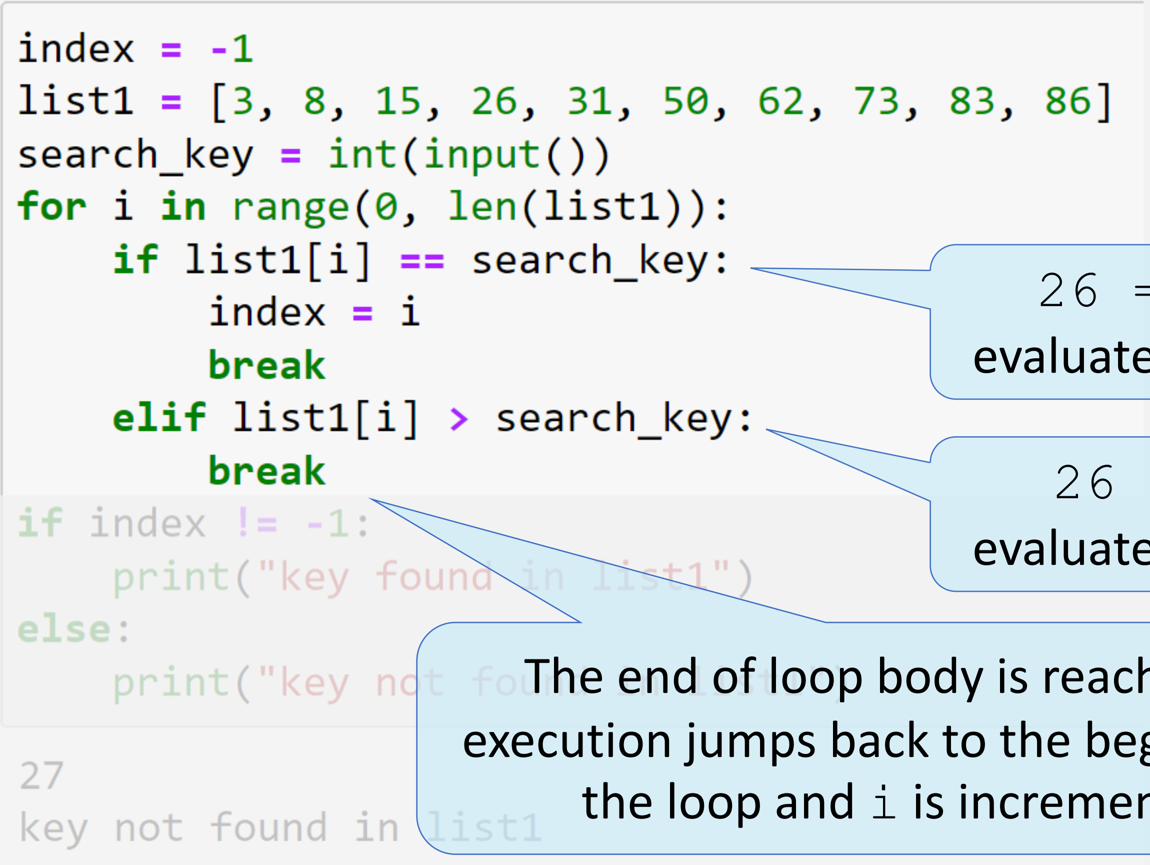
`15 == 27`
evaluates to False

`15 > 27`
evaluates to False

The end of loop body is reached, so execution jumps back to the beginning of the loop and `i` is incremented

```
search_key = 27
index = -1
i = 2
list1[2] = 15
```

Ordered Linear Search Algorithm Example



```
index = -1
list1 = [3, 8, 15, 26, 31, 50, 62, 73, 83, 86]
search_key = int(input())
for i in range(0, len(list1)):
    if list1[i] == search_key:
        index = i
        break
    elif list1[i] > search_key:
        break
if index != -1:
    print("key found in list1")
else:
    print("key not found in list1")
```

```
27
key not found in list1
```

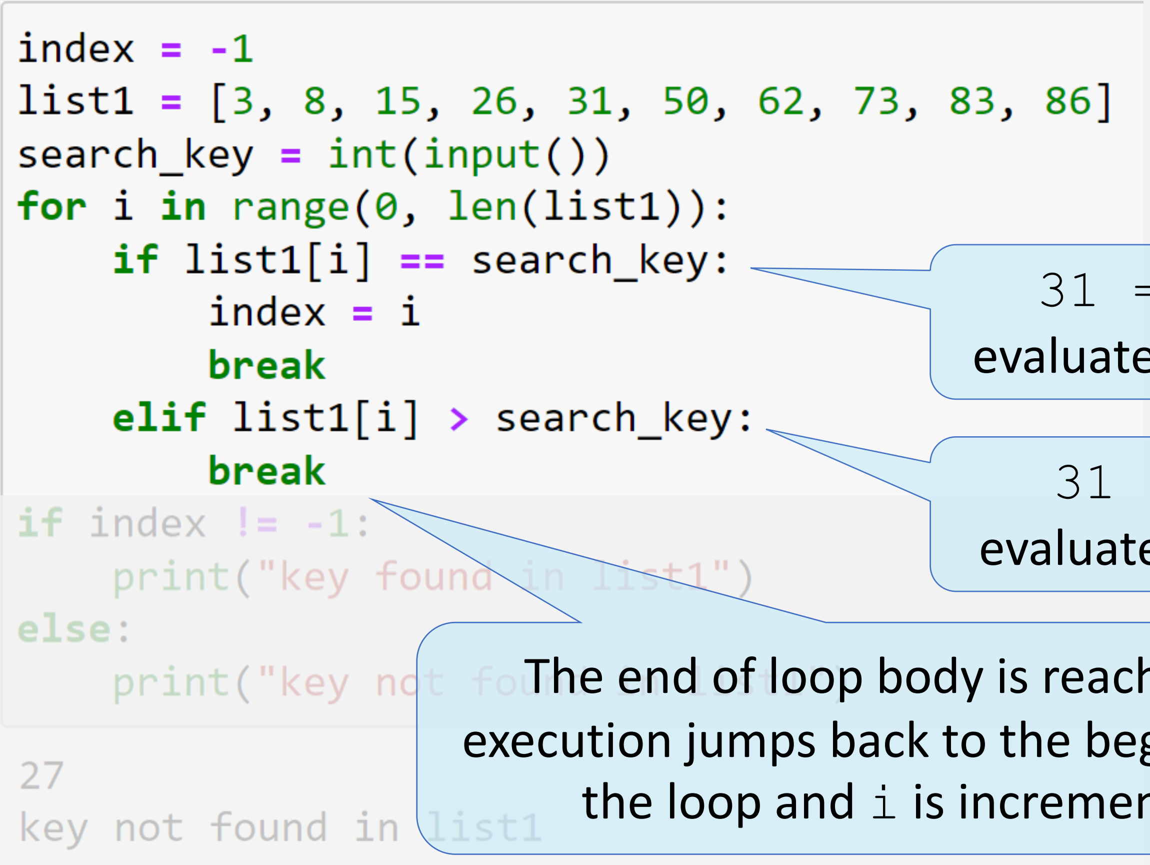
`26 == 27` evaluates to False

`26 > 27` evaluates to False

The end of loop body is reached, so execution jumps back to the beginning of the loop and `i` is incremented

```
search_key = 27
index = -1
i = 3
list1[3] = 26
```

Ordered Linear Search Algorithm Example



```
index = -1
list1 = [3, 8, 15, 26, 31, 50, 62, 73, 83, 86]
search_key = int(input())
for i in range(0, len(list1)):
    if list1[i] == search_key:
        index = i
        break
    elif list1[i] > search_key:
        break
if index != -1:
    print("key found in list1")
else:
    print("key not found in list1")
```

```
27
key not found in list1
```

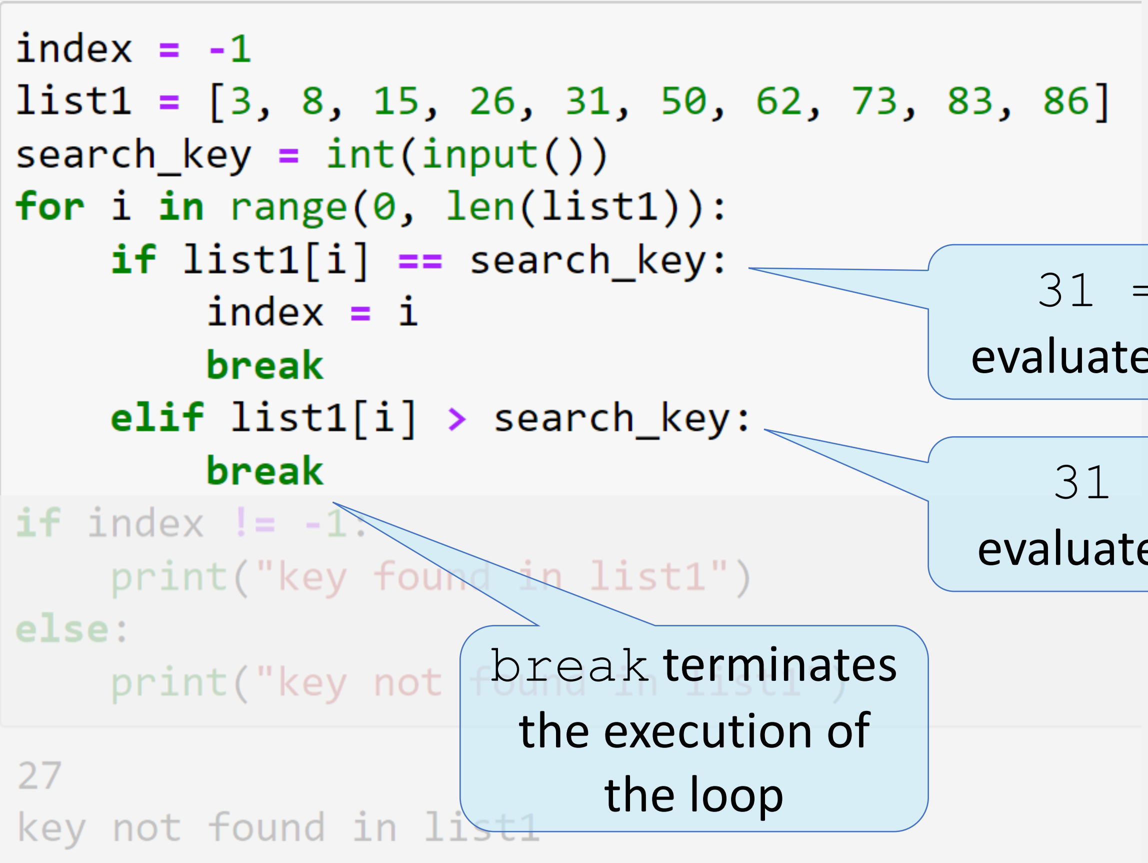
`31 == 27`
evaluates to False

`31 > 27`
evaluates to True

The end of loop body is reached, so execution jumps back to the beginning of the loop and `i` is incremented

```
search_key = 27
index = -1
i = 4
list1[4] = 31
```

Ordered Linear Search Algorithm Example



```
index = -1
list1 = [3, 8, 15, 26, 31, 50, 62, 73, 83, 86]
search_key = int(input())
for i in range(0, len(list1)):
    if list1[i] == search_key:
        index = i
        break
    elif list1[i] > search_key:
        break
if index != -1:
    print("key found in list1")
else:
    print("key not found in list1")
```

```
27
key not found in list1
```

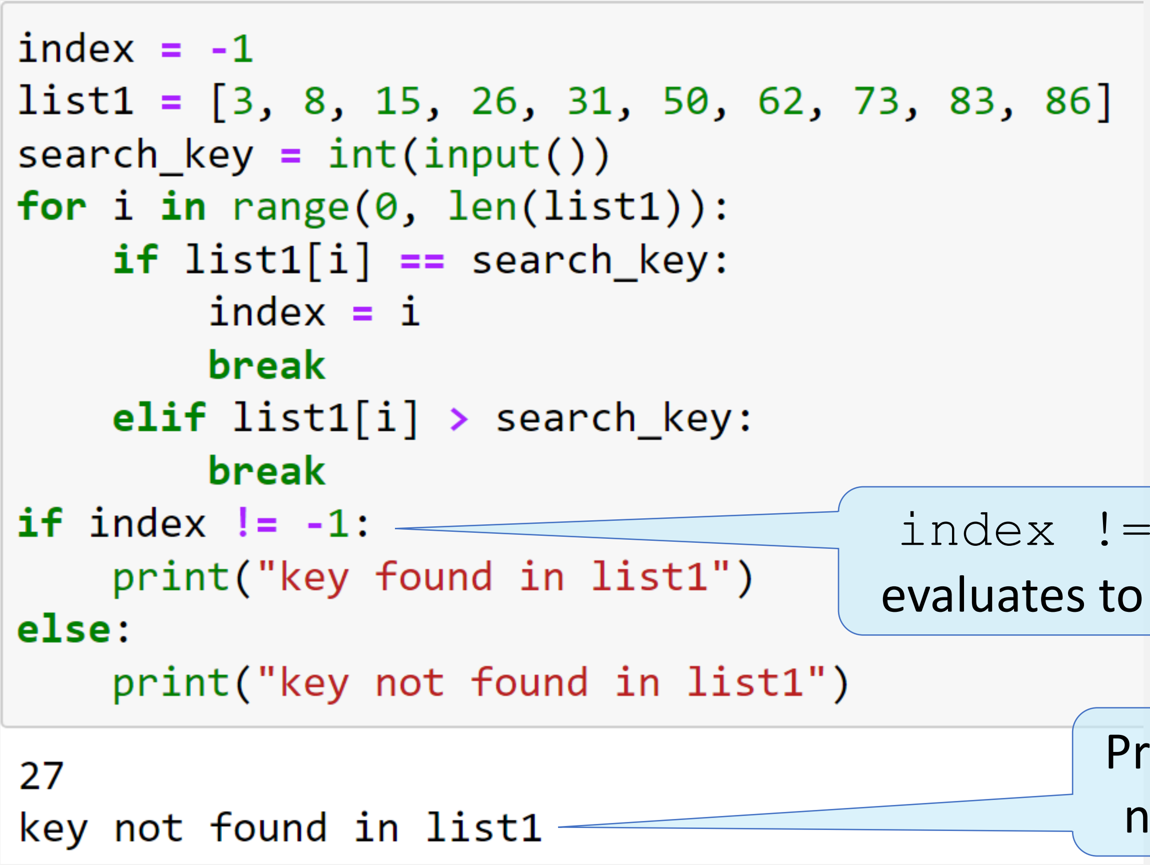
31 == 27
evaluates to False

31 > 27
evaluates to True

`break` terminates the execution of the loop

```
search_key = 27
index = -1
i = 4
list1[4] = 31
```

Ordered Linear Search Algorithm Example



```
index = -1
list1 = [3, 8, 15, 26, 31, 50, 62, 73, 83, 86]
search_key = int(input())
for i in range(0, len(list1)):
    if list1[i] == search_key:
        index = i
        break
    elif list1[i] > search_key:
        break
if index != -1:
    print("key found in list1")
else:
    print("key not found in list1")
```

```
27
key not found in list1
```

`index != -1` evaluates to False

Print that the value is not found in the list

```
search_key = 27
index = -1
i = 4
list1[4] = 31
```

Ordered Linear Search Properties

- Reasonable algorithm for short and medium size lists
- Simple and easy to implement
- Some prior data processing required
  - List has to be ordered
- More efficient when the key has no match in the list

Analysis

- What is the big-O for ordered linear search?
- If key is in the last position, the worst case scenario would still require **n** comparisons
- The big-O notation for ordered linear search is also **O(n)**

Try It Yourself

Write a program in python environment that takes a alphabetically sorted string and a character as an input and finds whether the character is found within the string using an ordered linear search