</talentlabs>

CHAPTER 5

Lists & Loops

</talentlabs>

AGENDA

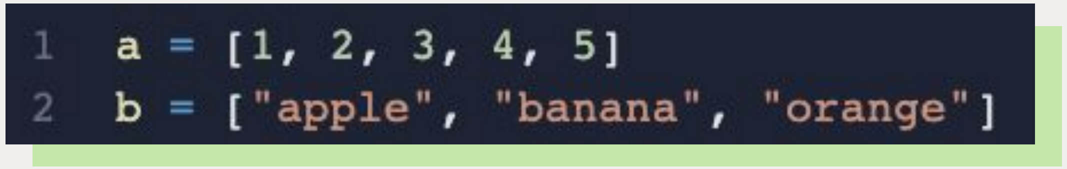
- Lists
- Loops
- While Loop
- For Loop
- While vs For
- Break & Continue

Lists

What is List?

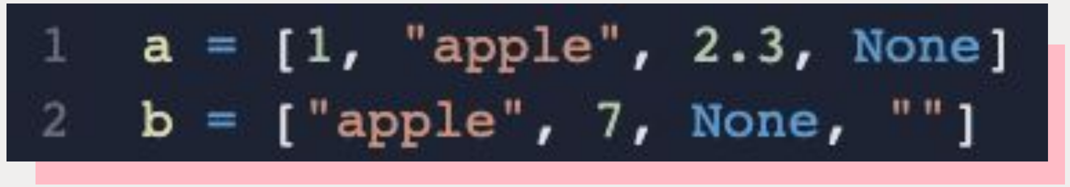
- Lists/Arrays is simply a list of value or objects (Wrapped by squared brackets)
- Technically, you could put anything in it (even if each elements are of different types)
- **Best Practice: You should try to put only same type of values/objects into the same array**

// Good Example



```
a = [1, 2, 3, 4, 5]
b = ["apple", "banana", "orange"]
```

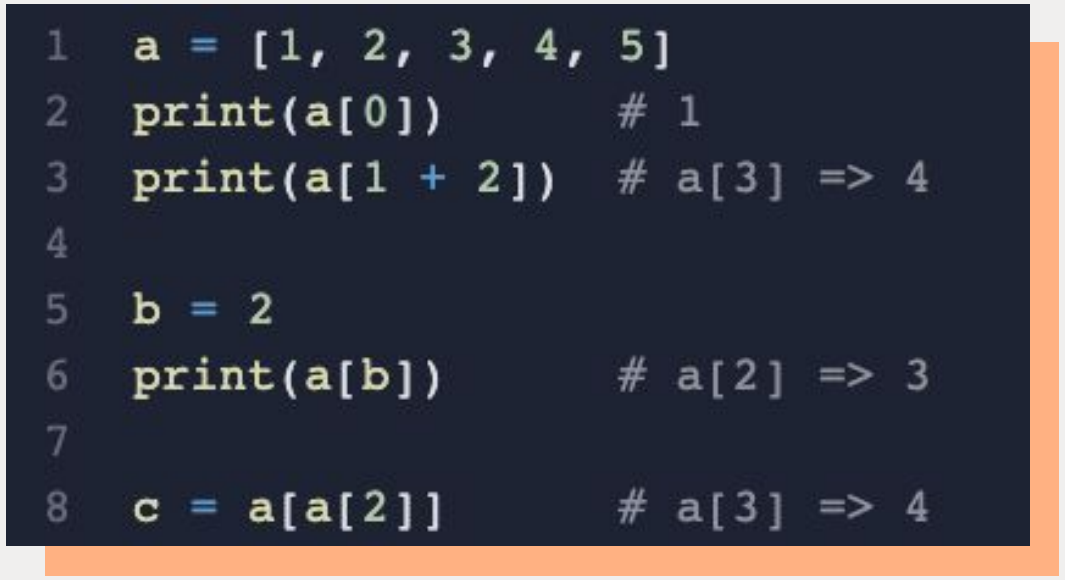
// Bad Example



```
a = [1, "apple", 2.3, None]
b = ["apple", 7, None, ""]
```

How to Access the Content in the Array?

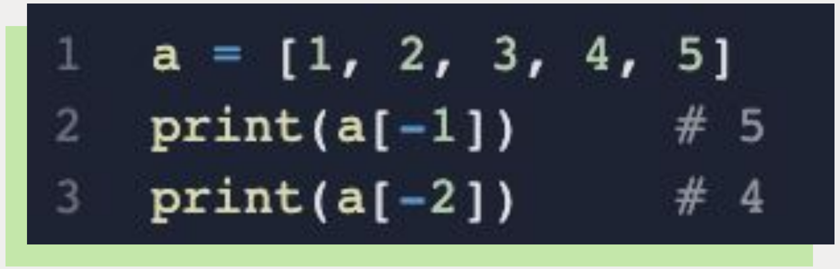
- use array_name[index]
- index start from 0
- index can also be a variable or a result of calculation



```
a = [1, 2, 3, 4, 5]
print(a[0])       # 1
print(a[1 + 2])   # a[3] => 4

b = 2
print(a[b])       # a[2] => 3

c = a[a[2]]       # a[3] => 4
```

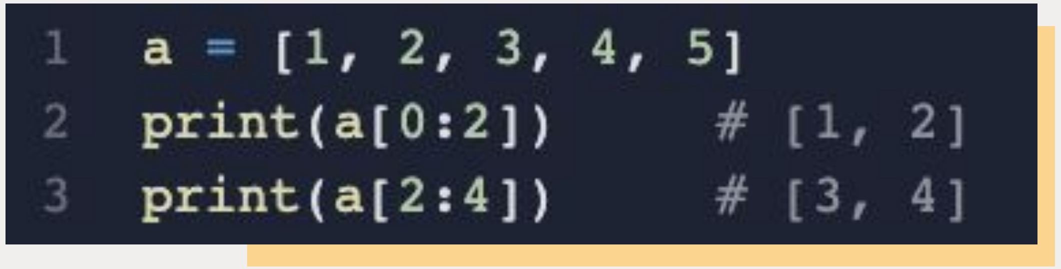
Some Convenient Lists Features

Shortcut to count from the end



```python
a = [1, 2, 3, 4, 5]
print(a[-1])      # 5
print(a[-2])      # 4
```

Getting a subset of a list



```python
a = [1, 2, 3, 4, 5]
print(a[0:2])      # [1, 2]
print(a[2:4])      # [3, 4]
```

Loops

What are Loops?

- One of the main purpose of computer is to help doing repetitive tasks for human
- Loops = repeating a piece of code multiple times
- 2 types of loops (almost universal in all programming languages)

While Loop

For Loop

When do we need Loops?

1. Loop through a list of values/objects

   Example: Lists

2. Repeat a task for multiple times

   Example: Repeat a task for 10 times

3. Do something until a certain conditions is met

   Example:

   until time is up

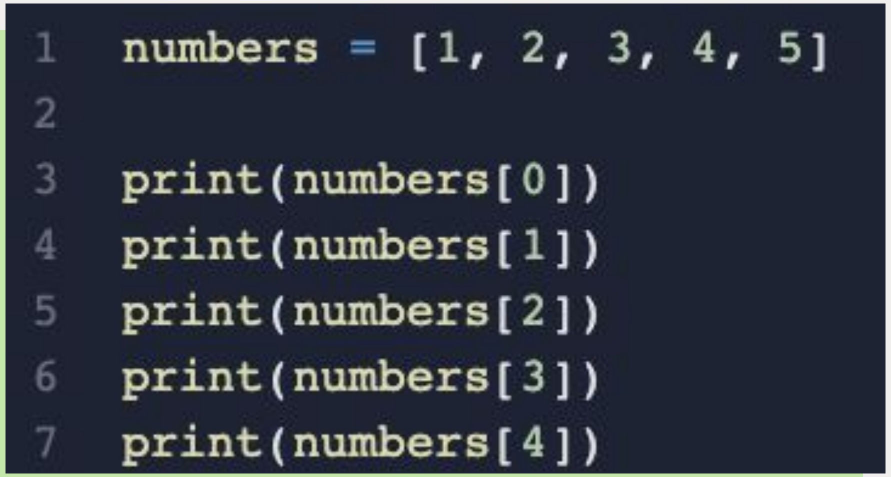
   until user finished input

   until server side has send me the results

4. Reduce the number of times we need to copy paste the code

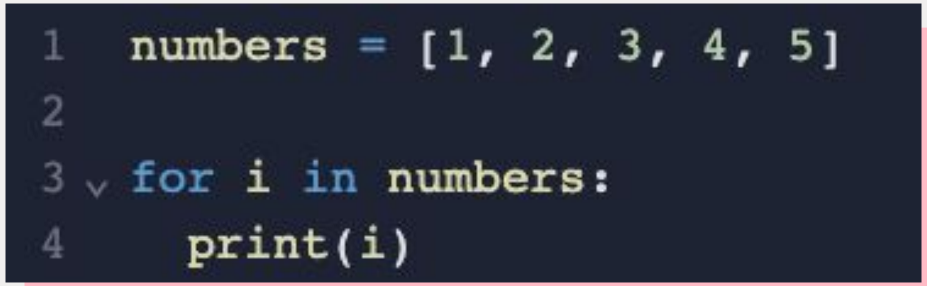
Example

Without Loop



```python
numbers = [1, 2, 3, 4, 5]

print(numbers[0])
print(numbers[1])
print(numbers[2])
print(numbers[3])
print(numbers[4])
```

With Loop



```python
numbers = [1, 2, 3, 4, 5]

for i in numbers:
  print(i)
```

While Loop

</talentlabs>

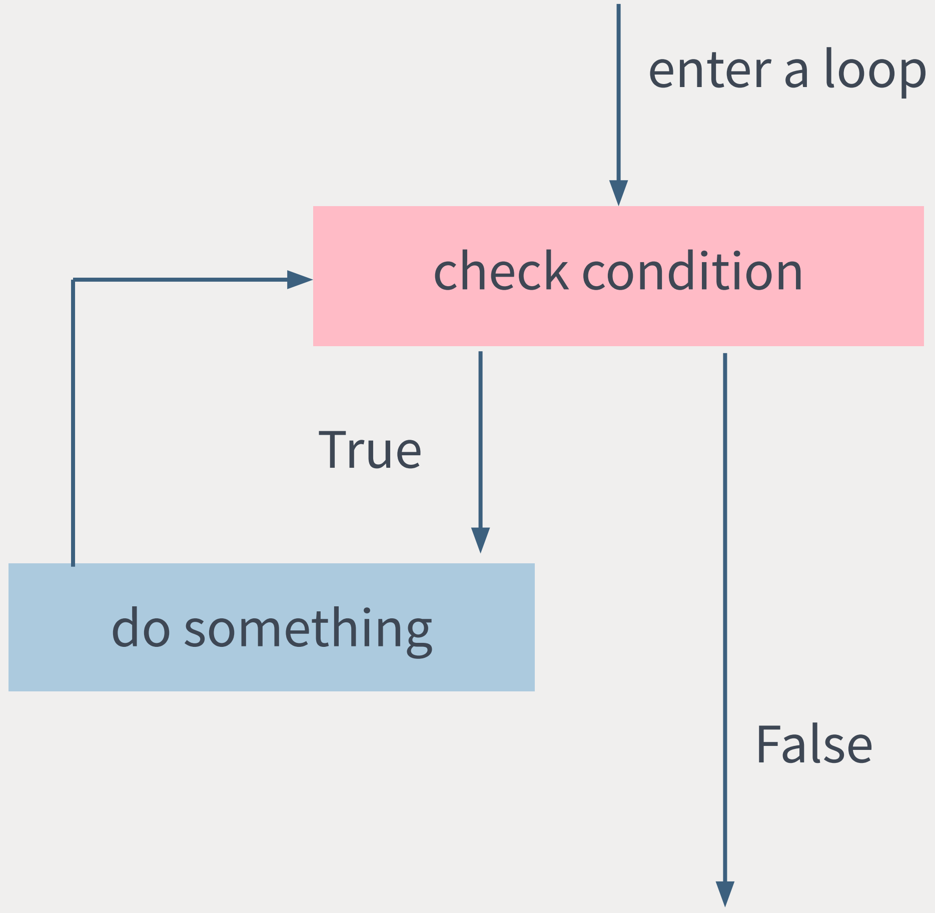
Types of While Loops

- Basic Concept: keep running the piece of code until a specific condition is **not met**
- There is only 1 type of while loop in Python (You are lucky!)

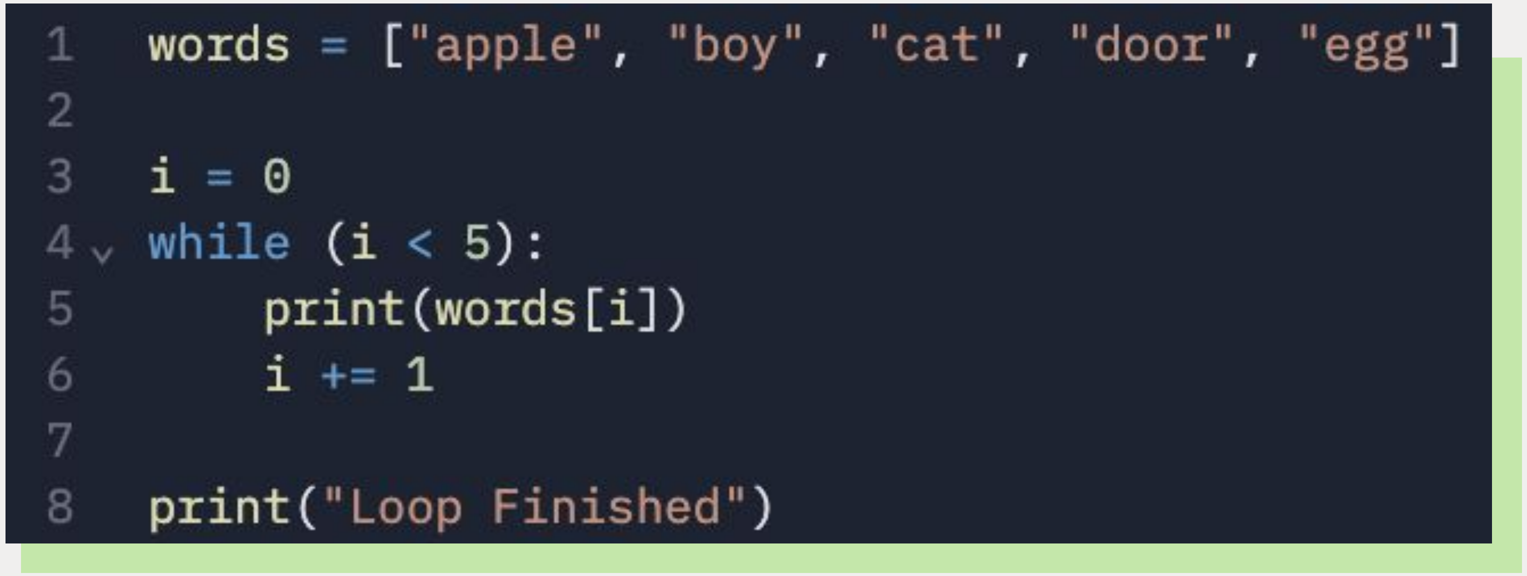
  While Loop

While

```
while condition:

    //do something
```



Example



```
words = ["apple", "boy", "cat", "door", "egg"]

i = 0
while (i < 5):
    print(words[i])
    i += 1

print("Loop Finished")
```

i values		Boolean value
0 < 5		TRUE
1 < 5		TRUE
2 < 5		TRUE
3 < 5		TRUE
4 < 5		TRUE
5 < 5	**Condition is *not met***	FALSE

For Loop

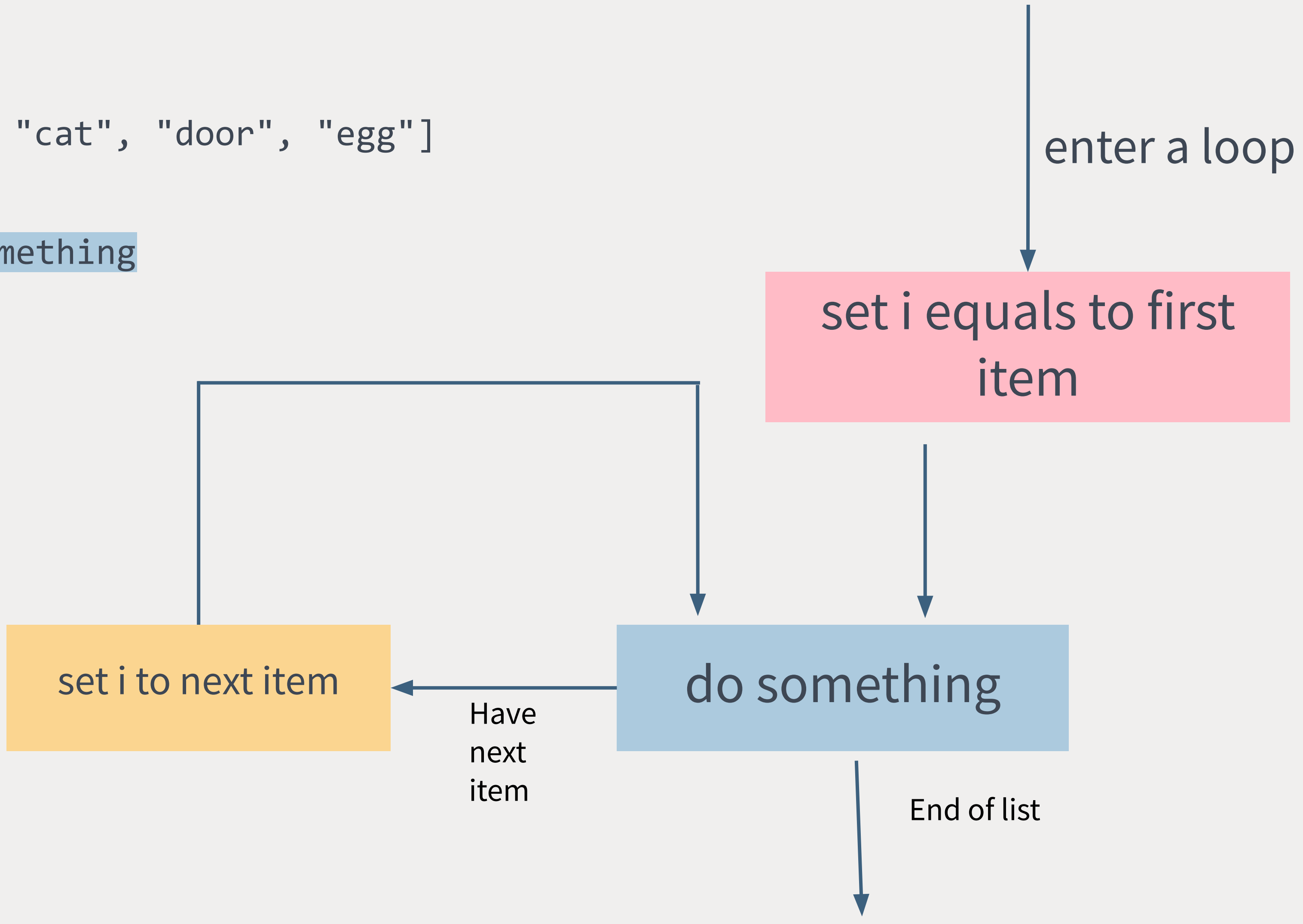
Types of For Loops

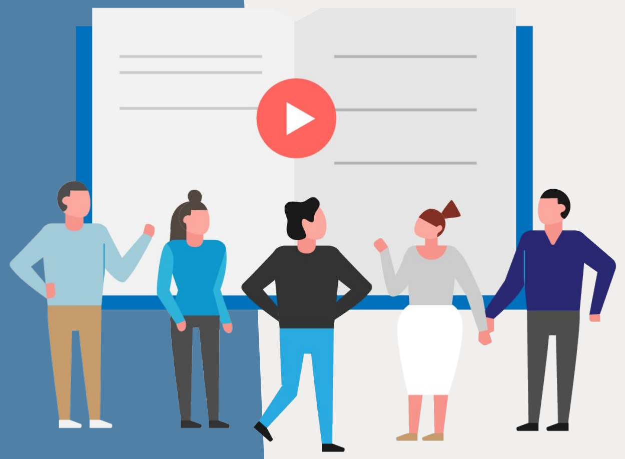
- The basic concept: for n times, execute this piece of code
- There is only 1 type of “for loop” in Python (You are lucky again!)

    for/in - loop through an iterable (i.e. list or range)

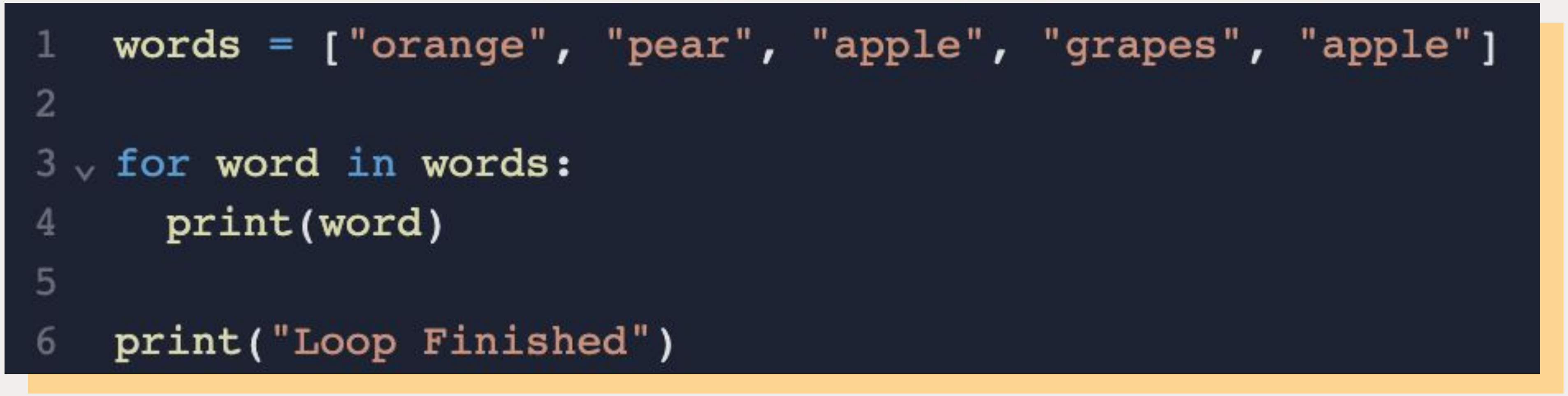
For

```
words = ["apple", "boy", "cat", "door", "egg"]
for i in words:

    // Loop Body - do something

}
```





Example

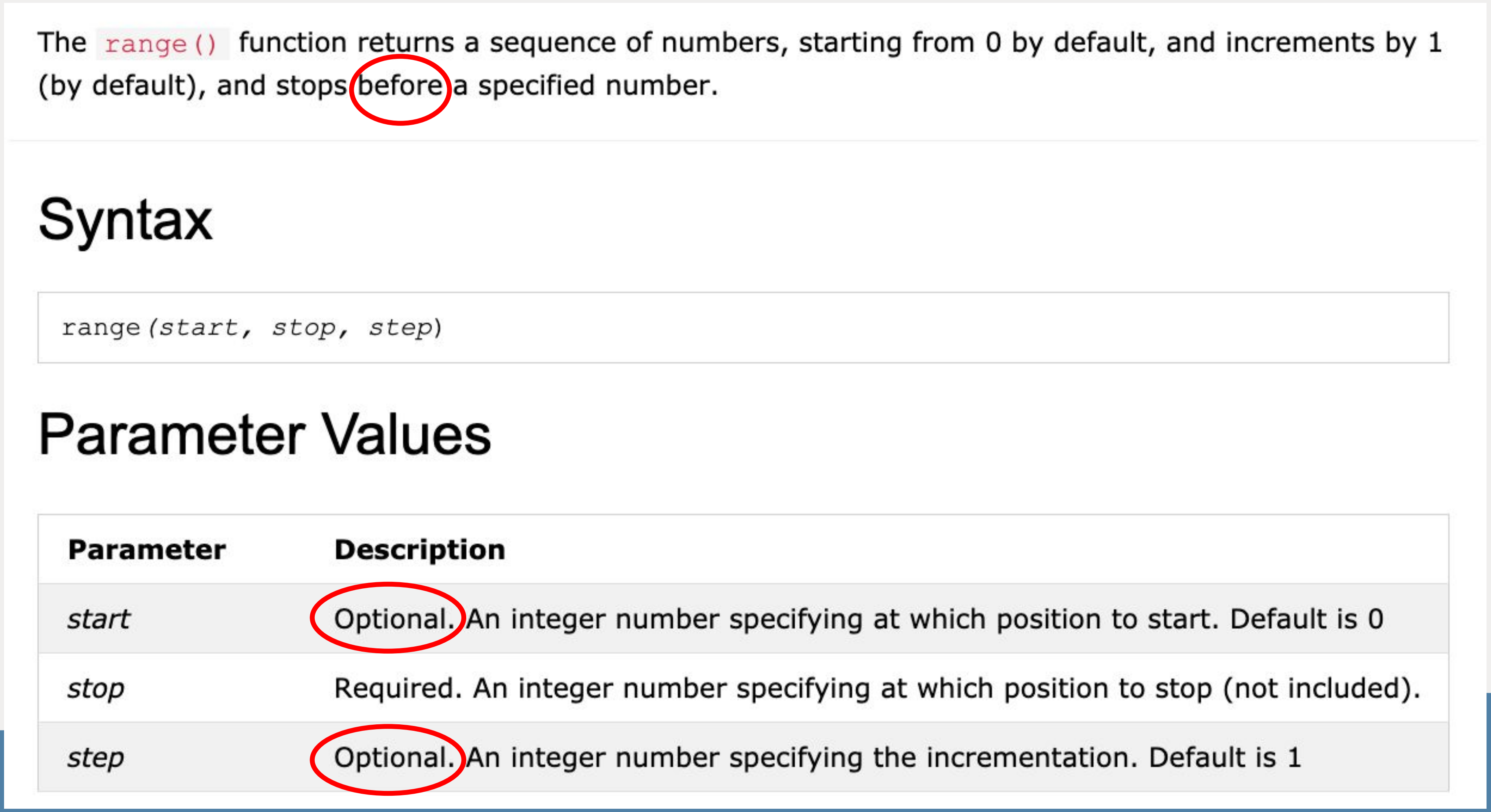


```python
words = ["orange", "pear", "apple", "grapes", "apple"]

for word in words:
  print(word)

print("Loop Finished")
```

Range

- Sometimes, we are not looping through a list, but a range.
- Range is a sequence of numbers
- You create a range by using the range() function

Range () Function



The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

Syntax

```
range(start, stop, step)
```

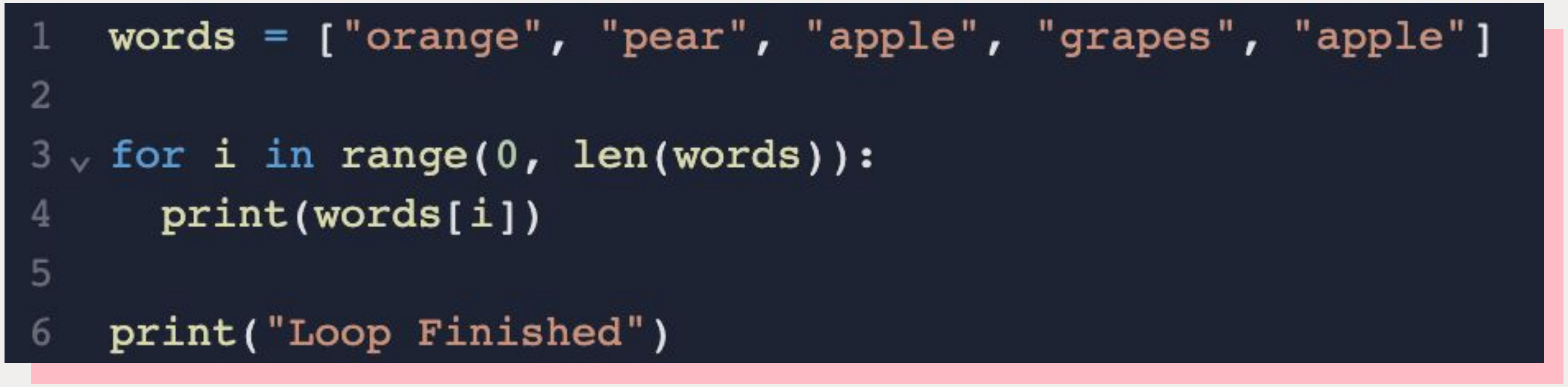
Parameter Values

Parameter	Description
start	Optional. An integer number specifying at which position to start. Default is 0
stop	Required. An integer number specifying at which position to stop (not included).
step	Optional. An integer number specifying the incrementation. Default is 1

Len () Function

- The len() function returns the number of items in an object.
- When the object is a string, the len() function returns the number of characters in the string.

Example



```python
words = ["orange", "pear", "apple", "grapes", "apple"]

for i in range(0, len(words)):
  print(words[i])

print("Loop Finished")
```

While vs For

</talentlabs>

Example

```python
numbers = [1, 2, 3, 4, 5]

for i in numbers:
    print(i)


print("Loop Finished")
```

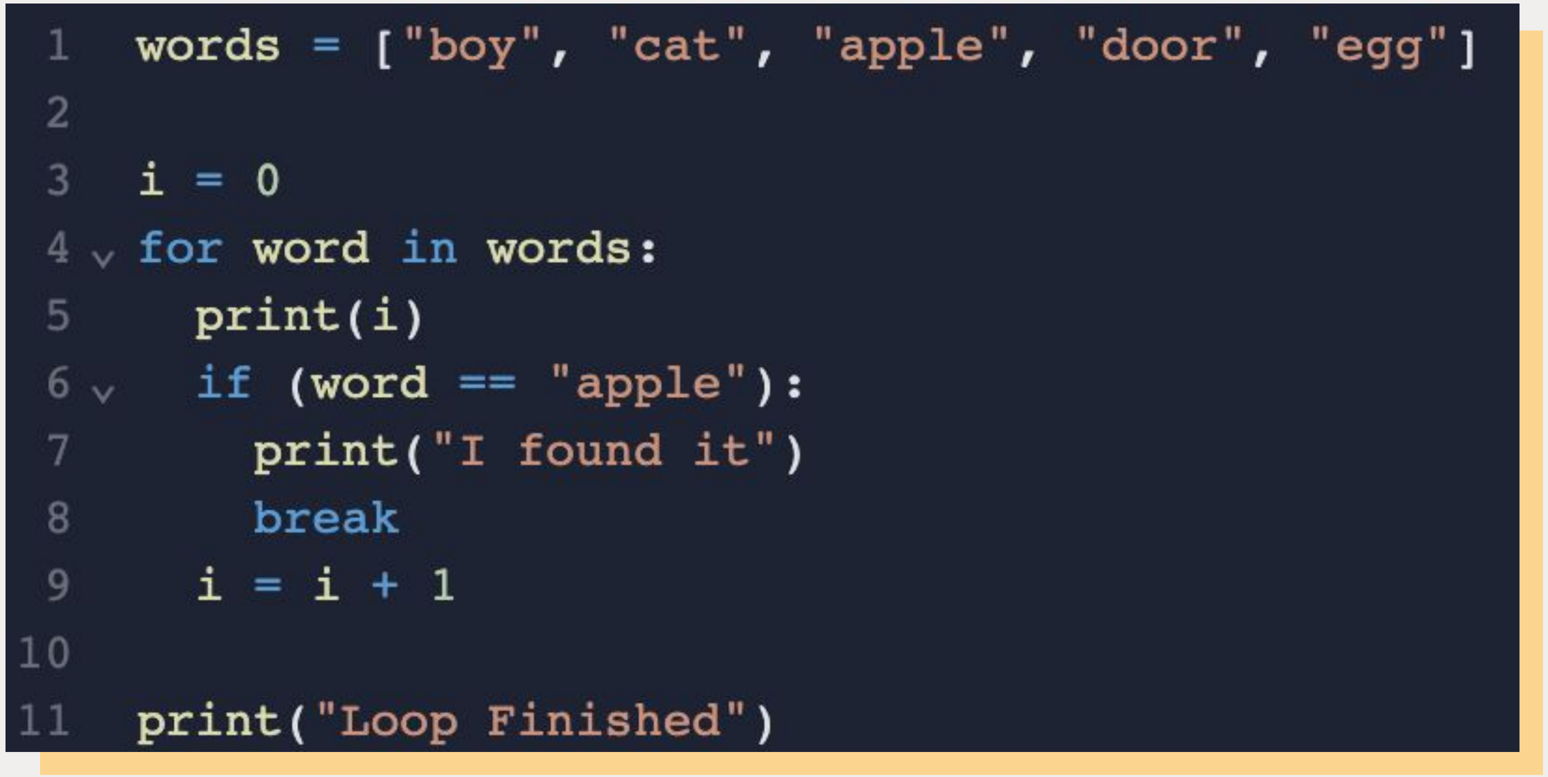
```python
numbers = [1, 2, 3, 4, 5]

i = 0
while i < 5:
    print(numbers[i])
    i = i + 1

print("Loop Finished")
```

Break and Continue

Ending the Loop Before It Ends

- Sometimes, you want the loop to stop before it ends naturally
- Consider a scenario:
  - I want to loop through a list of words, and see if the list contains the word “apple”
  - I found that the word is at the 3rd position, do I still need to continue?
- You can use the “break” keyword to end the loop **immediately** (ending the whole loop, no further rounds/iterations)

Breaking a Loop

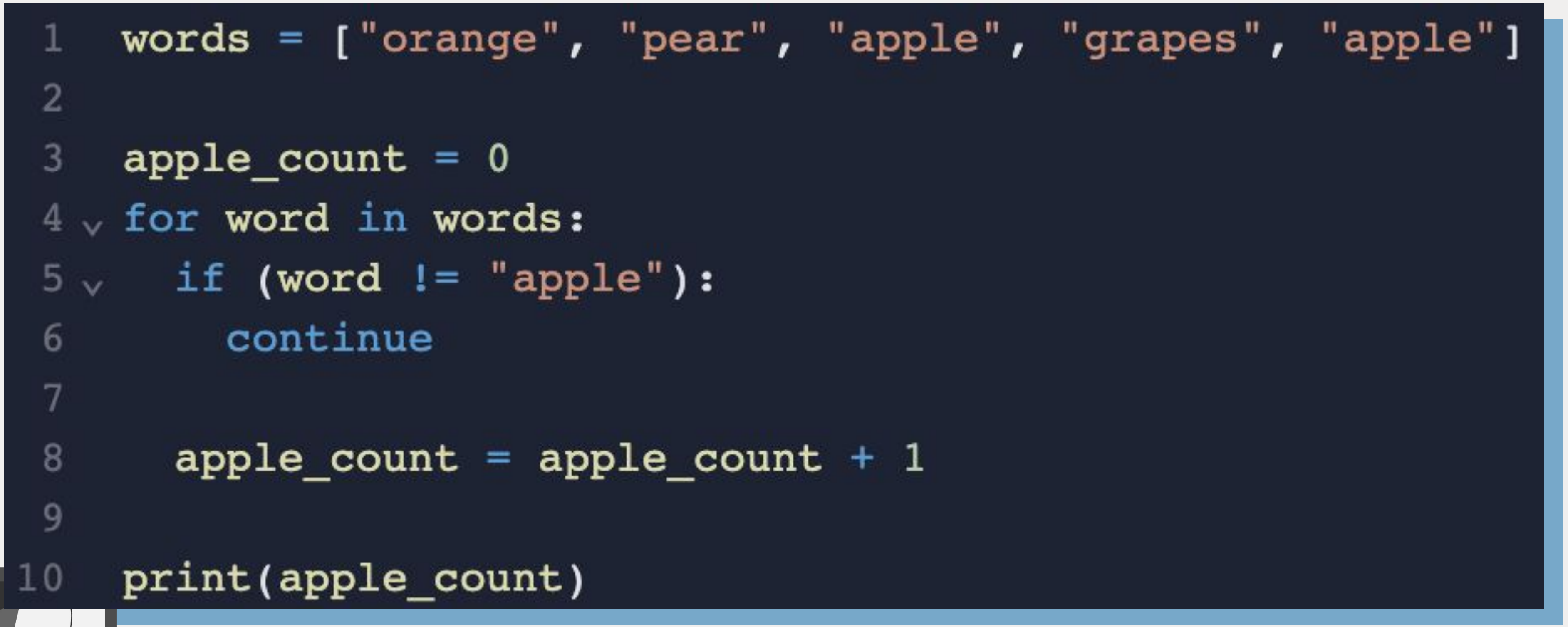


```
words = ["boy", "cat", "apple", "door", "egg"]

i = 0
for word in words:
  print(i)
  if (word == "apple"):
    print("I found it")
    break
  i = i + 1

print("Loop Finished")
```

Skipping an Iteration

- Sometimes, you want to skip one iteration
- Consider a scenario:
  - I want to loop through a list of fruits, count the number of “apple” in the list
  - For those that are not “apple”, there is nothing that I need to do
  - For those that are “apple”, I need to increase the count
- You can use the “continue” keyword to **end the current round/iteration** (but the loop will still go on for the next round)

Skipping an Iteration



```python
words = ["orange", "pear", "apple", "grapes", "apple"]

apple_count = 0
for word in words:
  if (word != "apple"):
    continue

  apple_count = apple_count + 1

print(apple_count)
```