Loops in Python

Estimated time needed: **20** minutes

Objectives

After completing this lab you will be able to:

- work with the loop statements in Python, including for-loop and while-loop.

Sometimes, you might want to repeat a given operation many times. Repeated executions like this are performed by loops. We will look at two types of loops, for loops and while loops.

Before we discuss loops lets discuss the range object. It is helpful to think of the range object as an ordered list. For now, let's look at the simplest case. If we would like to generate an object that contains elements ordered from 0 to 2 we simply use the following command:

```python
# Use the range

range(3)
```

NOTE: While in Python 2.x it returned a list as seen in video lessons, in 3.x it returns a range object.

The for loop enables you to execute a code block multiple times. For example, you would use this if you would like to print out every element in a list.
Let's try to use a for loop to print all the years presented in the list dates:

This can be done as follows:

```python
# For loop example

dates = [1982,1980,1973]
N = len(dates)

for i in range(N):
    print(dates[i])
```

The code in the indent is executed N times, each time the value of i is increased by 1 for every execution. The statement executed is to print out the value in the list at index i as shown here:

In this example we can print out a sequence of numbers from 0 to 7:

```python
# Example of for loop

for i in range(0, 8):
    print(i)
```

In Python we can directly access the elements in the list as follows:

```python
# Exmaple of for loop, loop through list

for year in dates:
    print(year)
```

For each iteration, the value of the variable year behaves like the value of dates[i] in the first example:

We can change the elements in a list:

```python
# Use for loop to change the elements in list

squares = ['red', 'yellow', 'green', 'purple', 'blue']

for i in range(0, 5):
    print("Before square ", i, 'is',  squares[i])
    squares[i] = 'white'
    print("After square ", i, 'is',  squares[i])
```

We can access the index and the elements of a list as follows:

```python
# Loop through the list and iterate on both index and element value

squares=['red', 'yellow', 'green', 'purple', 'blue']

for i, square in enumerate(squares):
    print(i, square)
```

As you can see, the for loop is used for a controlled flow of repetition. However, what if we don't know when we want to stop the loop? What if we want to keep executing a code block until a certain condition is met? The while loop exists as a tool for repeated execution based on a condition. The code block will keep being executed until the given logical condition returns a **False** boolean value.

Here's how a while loop works:

1. First, you specify a condition that the loop will check before each iteration (repetition) of the code block.
2. If the condition is initially true, the code block is executed.
3. After executing the code block, the condition is checked again.
4. If the condition is still true, the code block is executed again.

5. Steps 3 and 4 repeat until the condition becomes false.
6. Once the condition becomes false, the loop stops, and the program continues with the next line of code after the loop.

Here's an example of a while loop that prints numbers from 1 to 5:

```
count = 1
while count <= 5:
    print(count)
    count += 1
```

In this example, the condition **count <= 5** is checked before each iteration. As long as count is less than or equal to 5, the code block inside the loop is executed. After each iteration, the value of count is incremented by 1 using count += 1. Once count reaches 6, the condition becomes false, and the loop stops.

Let’s say we would like to iterate through list dates and stop at the year 1973, then print out the number of iterations. This can be done with the following block of code:

```
# While Loop Example

dates = [1982, 1980, 1973, 2000]

i = 0
year = dates[0]

while(year != 1973):
    print(year)
    i = i + 1
    year = dates[i]


print("It took ", i ,"repetitions to get out of loop.")
```

A while loop iterates merely until the condition in the argument is not met, as shown in the following figure:

The main difference between a while loop and a for loop in Python is how they control the flow of execution and handle iterations.

Key point of While Loop:

1. A while loop repeatedly executes a block of code as long as a given condition is true.
2. It does not have a fixed number of iterations but continues executing until the condition becomes false.
3. The condition is checked before each iteration, and if it's false initially, the code block is skipped entirely.

4. The condition is typically based on a variable or expression that can change during the execution of the loop.
5. It provides more flexibility in terms of controlling the loop's execution based on dynamic conditions.

Key point of For Loop:

1. A for loop iterates over a sequence (such as a list, string, or range) or any object that supports iteration.
2. It has a predefined number of iterations based on the length of the sequence or the number of items to iterate over.
3. It automatically handles the iteration and does not require maintaining a separate variable for tracking the iteration count.
4. It simplifies the code by encapsulating the iteration logic within the loop itself.
5. It is commonly used when you know the exact number of iterations or need to iterate over each item in a collection.

Write a for loop the prints out all the element between -5 and 5 using the range function.

```python
# Write your code below and press Shift+Enter to execute
```

Print the elements of the following list: Genres=['rock', 'R&B', 'Soundtrack', 'R&B', 'soul', 'pop'] Make sure you follow Python conventions.

```python
# Write your code below and press Shift+Enter to execute
```

Write a for loop that prints out the following list: squares=['red', 'yellow', 'green', 'purple', 'blue']

```python
# Write your code below and press Shift+Enter to execute
```

Write a while loop to display the values of the Rating of an album playlist stored in the list PlayListRatings. If the score is less than 6, exit the loop. The list PlayListRatings is given by: PlayListRatings = [10, 9.5, 10, 8, 7.5, 5, 10, 10]

```python
# Write your code below and press Shift+Enter to execute
```

Write a while loop to copy the strings 'orange' of the list squares to the list new_squares. Stop and exit the loop if the value on the list is not 'orange':

```python
# Write your code below and press Shift+Enter to execute

squares = ['orange', 'orange', 'purple', 'blue ', 'orange']
new_squares = []
```

Some real-life problems!

Your little brother has just learned multiplication tables in school. Today he has learned tables of 6 and 7. Help him memorise both the tables by printing them using for loop.

```
# Write your code here
```

The following is a list of animals in a National Zoo. Animals = ["lion", "giraffe", "gorilla", "parrots", "crocodile","deer", "swan"]

Your brother needs to write an essay on the animals whose names are made of 7 letters. Help him find those animals through a while loop and create a separate list of such animals.

```
# Write your code here


Animals = ["lion", "giraffe", "gorilla", "parrots",
"crocodile","deer", "swan"]
New = []
i=0
while i<len(Animals):
    j=Animals[i]
    if(len(j)==7):
        New.append(j)
    i=i+1
print(New)
```

Congratulations, you have completed lab on loops

Author

Joseph Santarcangelo

Other contributors

Mavis Zhou

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-05-10	2.4	Shreya	Added real-life practise problems
2022-05-12	2.3	Amrutha	Changed the markdown solution
2022-02-24	2.2	Hema	Changed the markdown solution
2022-01-10	2.1	Malika	Removed the readme for GitShare

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-08-26	2.0	Lavanya	Moved lab to course repo in GitLab