Lab03_LoopsAndIteration

September 15, 2025

A loop allows you to execute the same statement multiple times. Python has two kinds of loop structures: ***for*** loops, which iterate over the items of a sequence, and ***while*** loops, which continue to execute as long as a condition is true.

1 Content Learning Objectives

After completing this activity, students should be able to:

- Explain the syntax and purpose of a ***for*** statement
- Predict how ***range()*** works given 1, 2, or 3 arguments
- Identify the three main components of a ***while*** loop

2 Process Skill Goals

During the activity, students should make progress toward:

- Tracing the execution of ***while***/***for*** loops and predict their final output. (Critical Thinking)

3 Part 1: *for* Statements

A ***for*** loop executes the same block of code "for each item in a sequence". Create a new file named ***loops.py***, and enter the following code

[ ]:
```
print("Hello")
for x in [2,7,1]:
    print("the number is", x)
print("Goodbye")
```

1. Run the ***loops.py*** program. How many times does the indented line of code execute under the ***for*** loop?

3

2. How many times does the line of code ***NOT*** indented execute after the ***for*** loop?

1

3. Identify the value of x each time the indented line of code is executed.

a) 1st Time: 2

b) 2nd Time: 7

c) 3rd Time: 1

4. Modify the list ***[2, 7, 1]*** in the following ways, and rerun the program each time. Indicate how many times the ***for*** loop executes.

a) non-consecutive number: ***[5, -7, 0]*** 3

b) numbers decreasing in value: ***[3, 2, 1, 0]*** 4

c) all have the same value: ***[4, 4, 4]*** 3

d) single value in a list ***[8]*** 1

5. In general, what determines the number of times that the loop repeats?

The amount of numbers in the brackets.

6. What determines the value of the variable x? Explain your answer in terms of what is assigned (x = ...) each time the loop runs.

The sequence of numbers in the given index (index 0 is 2, index 1 is 7, index 2 is 1).

7. Modify a statement that assigns ***[0, 1, 2, 3, 4]*** to the variable ***numbers***. Rewrite the ***for x ...*** statement to use the variable ***numbers*** instead. Does the assignment have to come before or after the ***for*** statement?

Before

8. Add the following code at the end of your program:

```
[ ]: for c in "Hi!":
         print(c)
```

9. What is the output of this ***for*** statement?

```
H
i
!
```

10. What determined how many times ***print(c)*** was called?

Amount of characters in the parentheses.

11. Explain what a ***for*** statement does with strings.

Goes over each character and assigns each character of the string to the variable.

4 Part 2: The *range* Function

The Python ***range*** function will generate a list of numbers. The ***range*** function can take up to three numbers as arguments. Fill the table below by typing the code into a Python Shell:

Python Code	Shell Output
range(5)	range(0,5)
list(range(5))	[0, 1, 2, 3, 4,
x = range(3)	n/a
print(list(x))	[0, 1, 2]
list(range(5,10))	[5, 6, 7, 8, 9]
list(range(-3,4))	[-3, -2, -1, 0,
list(range(4,10,2))	[4, 6, 8]
for i in range(5): print(i)	0, 1, 2, 3, 4

12. Explain the differences in output between the first two lines of code (with and without the ***list*** function).

The list function gives the sequence of numbers in the range given. When list is not included, it just gives the start and end limit.

13. If the argument of the ***range*** function specifies a single number (x). What will be the first number listed? What will be the last number listed? How many numbers will be in the list?

The first number listed will be 0, while the last number listed will be (x-1). The amount of numbers in the list is x

14. In a Python Shell, use the ***range*** function to generate the sequence 0, 1, 2, 3.

```
list(range(4))
```

15. If the argument of the ***range*** function specifies two numbers (x, y). What will be the first number listed? What will be the last number listed? How many numbers will be in the list?

The first number will be x and the last number will be (y-1). The amount of numbers listed is y-x.

16. In a Python Shell, use the ***range*** function to generate the sequence 1, 2, 3, 4.

```
list(range(1,5))
```

17. If the argument of the ***range*** function specifies three numbers (x, y, z). What will be the first number listed? What does the third number represent? How many numbers will be in the list?

x is the start number, y is the end limit, and z is the value that goes in between each number to get from x to y. The amount of numbers will be (y-x)/z.

18. In a Python Shell, Use the ***range*** function to generate the sequence 1, 3, 5, 7.

```
list(range(1,9,2))
```

19. In your ***loops.py*** program, copy and paste the ***for*** loop from Part 1. Now modify the ***for*** statement so that the number of times the loop executes is determined by a variable named ***times***. How did you change the ***for*** statement?

```
times = list(range(x+1))
for x in times:
    print("the number is", x)
```

20. How would you modify the loop to print the values 0 to 5?

```
times = list(range(6))
for x in times:
   print("the number is",x)
```

21. Consider the two different types of ***for*** statements used in Part 1 and Part 2. If you wanted to execute a loop 100 times, which type of ***for*** statement would you choose and why?

The range for loop.

22. If you wanted to use each item of an existing list inside the loop, which type of ***for*** statement would you choose and why?

For items in the existing list.

5 Part 3: *while* Statements

A more general looping structure is the ***while*** statement. Add the code below to your current ***loops.py*** program:

[ ]:
```
i = 0
while i < 3:
    print("The number is", i)
    i = i + 1
print("Goodbye")
```

23. What must the value of the Boolean Expression (after the ***while***) be in order for the first print statement to execute?

True

24. Which line changes the value of ***i*** in the above code?

i = i + 1

25. What happens to the value of the variable ***i*** during the execution of the loop?

It increases by 1 in every line.

26. Explain why the loop body does not execute after it outputs "The number is 2".

i has to be less than 3. The boolean is false.

27. Reverse the order of the statements in the loop body:

```
i = 0
while i < 3
    i = i + 1
    print("The number is ", i)
```

[ ]:
```
while i < 3:
    i = i + 1
    print("The number is ", i)
```

28. How does the order impact the output displayed by the ***print*** function?

Tests previous value of i before printing the current value.

29. Does the order impact the total number of lines that are output?

No

30. Identify three different ways to modify the code so that the loop only executes twice.

i = 1, i < 2, i = i + 2

31. Describe the three parts of the ***while*** loop that control the number of times the loop executes.

Initialization, the boolean, and the iteration.

32. Comment out the statement ***i = i + 1***, and run the module. Then press Ctrl+C (hold down the Ctrl key and press C). Describe the behavior you see, and explain why it happened.

It repeats infinitely.

When writing a ***while*** loop, it's helpful to answer a few questions before you start:

- What needs to be initialized before the loop?
- What condition must be true for the loop to repeat?
- What will change so that the loop eventually ends?

33. Consider the function ***recent_precip(x)*** that prompts the user for ***x*** hourly rainfall rates and returns the sum of these values. For example, when ***recent_precip(6)*** is called, the user is asked to input six numbers that represent the rainfall rate over the last 6 hours. If the user input the numbers ***0.05, 0.03, 0.08, 0.07, 0.04, and 0.02*** the function would return the value ***0.29***. Describe the variable(s) that needs to be initialized before the loop begins.

counter = 0

34. Describe the Boolean Expression that must be true for the loop to continue.

while counter < x:

35. Describe what will need to change so that the loop will eventually end.

counter = counter + 1

36. Now list what else must happen inside the body of the loop for you to calculate the sum of the user inputs.

```
rain = float(input('enter rainfall rate: '))
total = total + rain
total = 0
```

37. Given the previous answer, are there any other variables that need to be initialized before the start of the loop?

return total