5.5. Repetition with a For Loop

Some of the programs we've seen so far are a bit tedious to type. If we want to make a repetitive pattern in our drawings, then it can take many lines of code. Thankfully, Python has a few ways for making this kind of task easier. For now you'll get a brief preview of a helpful control structure and function in Python which you will learn about later.

The control structure is called a for loop. If you've learned other programming languages then you may be familiar with what it does but the structure may be new. A for loop allows Python to execute a program in a non-linear fashion. Instead of evaluating the code line by line until it reaches the end, once the program reaches a for loop, it will tell the program to execute a set of lines repeatedly. After doing all that, the program will then continue to evaluate and execute whatever is below the for loop.

In the code below, we make use of the `range` function to specify how many times the code inside the for loop will execute. In a later chapter, we will explain exactly what the range function is doing and how it works with the for loop. For now, just try to understand what happens when the following code executes.

Save & Run 4/17/2021, 3:51:26 PM - 2 of 2 Show in CodeLens

```
1 print("This will execute first")
2
3 for _ in range(3):
4     print("This line will execute three times")
5     print("This line will also execute three times")
6
7 print("Now we are outside of the for loop!")
8
```

```
This will execute first
This line will execute three times
This line will also execute three times
This line will execute three times
This line will also execute three times
This line will execute three times
This line will also execute three times
Now we are outside of the for loop!
```

Activity: 1 -- ActiveCode (ac3_5_1)

There are a few things to notice here for when you use this later on. First, is that the two print statements on line 4 and 5 are executed three times, but we don't print line 4 three times and then print line 5 three times. Instead, we print line 4, then line 5. Once that is done the for loop iterates, or brings the program back to the beginning of the for loop, and continues to print out lines 4 and 5 again until it has printed them both a total of three times.

Second, these lines were printed the same number of times as is inside the `range` function. If we wanted to print them more or fewer times, then we would just need to change the number inside of the parentheses on line 3.

Finally, the indentation is important here. All of the statements that were printed out multiple times were indented under the for loop. Once we stopped indenting those lines, then the program was outside of the for loop and it would continue to execute linearly. If you'd like to watch the execution, checkout the code above in codelens!

Now it's time to combine this with the Turtle module. We can do a lot of cool stuff if we combine these two things! Below is code to do just that. Try to predict what the program will do before running it.

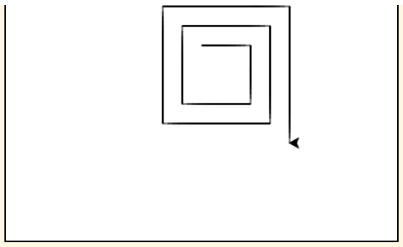
Save & Run 4/17/2021, 3:52:00 PM - 2 of 2 Show in CodeLens

```
import turtle
wn = turtle.Screen()

elan = turtle.Turtle()

distance = 50
for _ in range(10):
    elan.forward(distance)
    elan.right(90)
    distance = distance + 10

```



Activity: 2 -- ActiveCode (ac3_5_2)

Note

Try it out yourself in the space below. What can you make?

Save & Run 5/13/2021, 7:30:49 PM - 3 of 3 Show in CodeLens

```
import turtle
wn = turtle.Screen()

```

Activity: 3 -- ActiveCode (ac3_5_3)

You have attempted 4 of 3 activities on this page

5.4. Object Oriented Concepts">

ect Oriented Concepts">

✔ Completed. Well Done!

5.6. A Few More turtle Methods and Observations">

| Back to top