Chapter 4 - Loops

Loops

For Loops

Learning Objectives - For Loop

- **Explain for loop syntax (especially the whitespace)**
- **Identify the relationship between patterns, loops, and output**

For Loops

For Loop Syntax

Before you can start writing a loop, you need to be able to spot the pattern. Let's take something simple:

```
print("Hello")
print("Hello")
print("Hello")
print("Hello")
print("Hello")
```

The pattern is `print("Hello")`, and it is repeated five times. Since we know that the loops needs to run exactly five times, a for loop is the way to go. Here is how you write a for loop that repeats five times. Use the code visualizer to see how a for loop works.

```
for i in range(5):
    print("Hello")
```

Code Visualizer

Like **conditionals**, for loops are code blocks. Instead of a **boolean** statement, you use the `range` function followed by a `:`. All of the code that will be repeated needs to be indented.

Understanding `range`

The expression `range(5)` will return a series of five numbers, but it is important to understand how Python does this. Enter the code below and run it.

```
for i in range(5):
    print("Loop #" + str(i))
```

Code Visualizer

The loop ran five times, but the variable `i` did not start with 1. Instead it started with 0. Python, like most programming languages, starts counting with 0. Python will continue counting up to, but not including 5.

challenge

What happens if you:

- Change the print statement to `print("Loop #" + str(i + 1))`?
- Change the range statement to `range(1,6):` and the print statement to `print("Loop #" + str(i))`?
- Change the range statement to `range(0, 10, 2):`?
- Change the range statement to `range(10, 0, -1):`?

Code Visualizer

▼ **The 3rd Number in `range`**

The range statement normally works with two numbers, where it starts counting and where it ends. The two examples above show that the range statement can take a third number. This number tells `range` the amount to increment. Adding a `2` will mean that `range` counts by 2. Add a negative number and `range` will count down. In this case, be sure that the first number is larger than the second.

Turtle Graphics

Before continuing with loops, we are going to learn how to create graphical output with the **turtle library**. Like a pencil on paper, the turtle object leaves a line as it moves around the screen.

Turtle Syntax

You need to import the turtle library as the first line of your code and end your code by calling `mainloop()`.

```
import turtle

# All of your turtle commands
# go in this space here.

turtle.mainloop()
```

The next step is to create a turtle object to move around the screen.

```
import turtle

t = turtle.Turtle() # create a turtle called t

# All of your turtle commands
# go in this space here.

turtle.mainloop()
```

Here are some basic commands to use with the turtle library.

Command	Parameter	Description
`t.forward(n)`	Where `n` represents the number of pixels	Move the turtle forward
`t.backward(n)`	Where `n` represents the number of pixels	Move the turtle backward
`t.rt(d)`	Where `d` represents the number of degrees	Turn the turtle to the right
`t.lt(d)`	Where `d` represents the number of degrees	Turn the turtle to the left

Go ahead and get comfortable creating and moving a turtle around the screen before we start drawing with loops.

definition

Turtle Output

Click the button below to run your code. You may have noticed that there is no `print` command used with turtle objects, so the output of your program does not appear as you would expect. Look for the tab that reads **Preview https/...** and click on it. You should see your turtle drawing there. Close the window with the turtle output to stop your program.

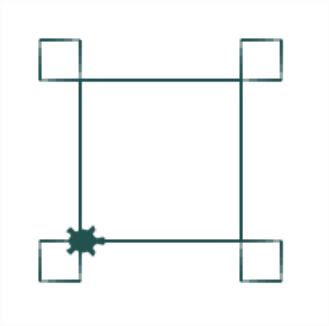
Turtle Coding - For Loop

Using your knowledge of for loops, try and recreate the images you see below. Remember, click on the tab that reads **Preview https/...** to see your output. Close the window with the turtle output to stop your program.

▼ **Customize your turtle**

- `t.color('red')` - Takes a string for the color
- `t.shape('turtle')` - Takes one of the following strings `'turtle'`, `'circle'`, `'square'`, `'arrow'`, `'classic'`, or `'triangle'`.
- `t.pensize(4)` - Takes a positive number
- `t.speed(1)` - Takes a number in the range 0..10

Challenge 1



Turtle Challenge 1

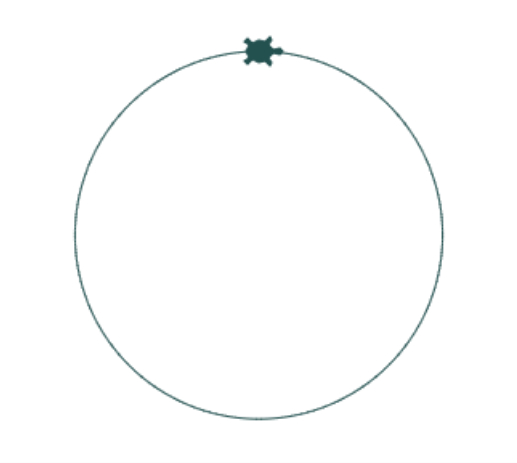
▼ **Hint**

The trick is to find the pattern and then repeat it four times. The pattern is to go forward and then make a smaller square (with right turns) at the end. The pattern should look something like this:

▼ **Hint 2**

If you are stil stuck, use these lines of code to see if it will help you complete the pattern: `t.forward(80)`, `t.rt(90)`, `t.forward(20)`.

Challenge 2



Turtle Challenge 2

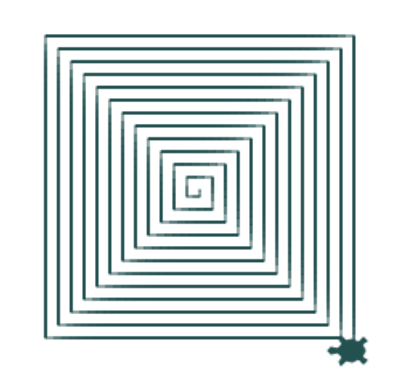
▼ **Hint**

Since a circle has 360 degrees, you will need a loop that repeats 360 times. Be careful about how far the turtle walks. The circle can get very big, very quickly.

▼ **Hint 2**

If you are stil stuck, use these lines of code to see if it will help you complete the pattern: `t.rt(1)`.

Challenge 3



Turtle Challenge 3

▼ **Hint**

The pattern here is to move forward and make a right turn. The trick is, the amount to move forward needs to get bigger as the loop advances. Think of some operators that you can use to make the loop variable get a little bit bigger each iteration.

▼ **Hint 2**

If you are still stuck, multiply the value of `i` by another number to

generate the distance the turtle needs to move forward.