Lesson 8: Drawing Shapes with Turtle

In this lesson, we'll learn about how to draw different shapes using the Turtle module!

Using the same basic movement functions, we can draw a lot of different shapes using Turtle, like squares, triangles, and circles.

PART 1: Setting up the Turtle

Like before, we'll want to set up our turtle before we try to draw anything.

Type the code below into a new program:

```
import turtle
T = turtle.Turtle()
T.speed(1)
T.pencolor("red")
turtle.done()
```

Our turtle's speed can be set between 0 and 10.

1	2	3	4	5	6	7	8	9	10	0
Slowest					Medium					Fastest

Just like in our last program, we've made a turtle called T, and modified its speed.

This time, we're using a speed of 1 (slowest) so that we can see what our commands do.

We used the `T.pencolor()` function to set the pen color of the turtle to red.
This will affect the color of the line the turtle draws as it's moving.

We can change the pen color at any time, but it will only affect lines that are drawn after the pen color is changed.

You can change the turtle's pen color to any of the colors below, and more!

Black, Red, Orange, Gold, Yellow, Green, Gray, Cyan, Blue, Purple, Violet, Pink, Brown

Before we move on, you should save your program as *square.py*

PART 2: Drawing a Square

In this first program, we'll use the same movement functions as before to draw a square.

Here's a table of the movement functions, with their shortened versions included:

`T.forward(p)`	`T.fd(p)`	Moves the turtle forwards **p** pixels
`T.backward(p)`	`T.bk(p)`	Moves the turtle backwards **p** pixels
`T.left(d)`	`T.lt(d)`	Turns the turtle left by **d** degrees.
`T.right(d)`	`T.rt(d)`	Turns the turtle right by **d** degrees.

If you'd like, you can use the shortened functions instead of the longer ones used in this lesson.

We'll first add a few movement functions, and see how our turtle moves in response.

Add the following code above `turtle.done()`:

```
T.pencolor("red")
T.forward(100)
T.right(90)
turtle.done()
```

Make sure to place all other Turtle functions before `turtle.done()` !

When our turtle starts, it will be in the center of the screen, facing to the right.

With these movement commands, we're first telling our turtle to move forward 100 pixels. After that, we tell our turtle to turn right by 90 degrees, which will make it face down.

Using what we've learned so far, we should be able to draw most basic shapes.

Now add more commands to make the turtle finish drawing the square!

(Hint: You should be able to make a square by adding 6 more lines of code above the last line.)

PART 3: Creating a Triangle

Once you've finished drawing a square, we'll move on to creating a triangle.

We'll start this shape in a new file, so that we have space to draw it on the screen.

First, create a new program, go to File > Save As and name it *triangle.py* .
Then, add the following code to your new triangle program:

```
import turtle

T = turtle.Turtle()

T.speed(1)

T.fillcolor("green")

turtle.done()
```

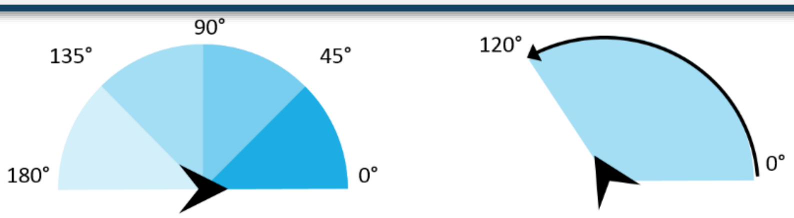
So far, our triangle program looks very similar to how we started the square program.

The main difference is that rather than setting the pen color, we set the fill color to green. This will let us fill in the space inside a shape with a color.

Next, we'll try making our turtle move in a triangle, using similar commands as before.

Add the following code above `turtle.done()`:

```
T.fillcolor("green")

T.begin_fill()

T.left(120)

T.forward(100)

T.end_fill()

turtle.done()
```



Just like before, we're using the same Turtle functions to move and draw with our turtle. This time, though, we're rotating the turtle 120 degrees instead of 90.

The turtle will only fill in a shape once we call `end_fill()`, so everything between `T.begin_fill()` and `T.end_fill()` will be part of the shape.

If you run the program now, it will only draw a black line, until you complete the program. After that, you'll see the whole triangle fill with color!

Now, try adding code above `T.end_fill()` to draw the rest of the triangle!
(Hint: You should be able to draw the rest of the triangle by adding just 4 lines.)

PART 4: Making a Circle

Once we're done drawing a triangle, we'll try drawing a circle using a `for` loop.
A `for` loop is like a while loop, but we choose the number of times the loop repeats.

First, create a new program, go to File > Save As and name it *circles.py* .
Then, add the following code to your new circle program:

```
import turtle

T = turtle.Turtle()

T.speed(1)

T.color("blue")

for i in range(36):

    T.forward(10)

    T.right(10)

turtle.done()
```

Here, we set up our turtle the same way, but this time we used `T.color()` which changes both the fill color and the pen color to the same color.

Afterwards, we added the line `for i in range(36):`

This line tells our program to use a variable `i` to count upwards until it gets to 36. Each time it counts, it will repeat the commands to move the turtle forwards and turn it right.

In programming, it's common to use a short variable name in `for` loops.
Often, programmers will name the variable `i`, which can stand for index.

Since the values we use in the `T.forward()` and `T.right()` function are small, this will cause our turtle to keep moving forward a little bit, and then turning right by 10 degrees.

There are 360° in a circle, so if we repeat the steps 36 times, our turtle will draw a circle.

Before moving on, make sure you've saved your program as *circles.py*

PART 5: Coloring the Circle

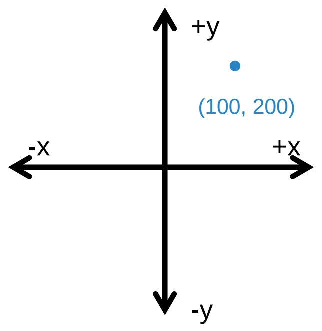
If you were to test your program now, it would only draw a blue outline of a circle.

In order to also fill in the circle, we'll need to use `begin_fill()` and `end_fill()` again.

Try adding the following code to your program:

```
T.color("blue")
T.penup()
T.goto(100,50)
T.pendown()
T.begin_fill()
for i in range(36):
    T.forward(10)
    T.right(10)
T.end_fill()
T.hideturtle()
turtle.done()
```

The `goto()` function tells our turtle to move to a certain position on the screen.



In this case, we want our circle to start in the upper-right section of the screen, at the position (100, 200)

The `pendown()` and `penup()` functions let us place or lift the turtle's pen, so that it can only draw lines when we want it to.
By telling our turtle to lift the pen up before going to the position (100, 200), we're asking it not to draw any lines until we tell it to place the pen down again.

After calling `end_fill()`, we use `T.hideturtle()` to make our turtle object invisible.

★ **Bonus - Draw 2 more circles on the screen with a different color, size and position.**