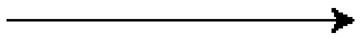
# Lesson 7: The Turtle Module

In this lesson, you'll learn how to use Python's built-in *Turtle* graphics system!

Turtle is a module in Python that lets us make and move a turtle object on the screen. While the object moves, it draws a line behind it, tracing the path that it took with a pen.



Here's what it looks like after the object draws a line while moving to the right!

The module was named turtle in tribute to a graphical turtle that appeared in the drawing mode of an early education-based programming language called Logo. In Python, the default object will look like a triangle (instead of a turtle), and will draw a black line.

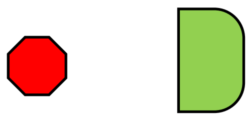
************************************************************************************************

## PART 1: Prep by Creating and Saving a New File

**IMPORTANT**: Save your file now with the name ***Score*** since the program will be a goal-scoring program. Do not name your program file ***turtle*** when you save it. If you name it ***turtle***, your program will not work since it is the same name as the Python module it will use.

## PART 2: Place a Background Image in Your Work Folder

We will use a pre-made background image in our file. The image will be of an obstacle and goal, similar to that shown below.



The point of the game will be to maneuver the "turtle" from a start position to the left of the octagon and into the goal in as few commands as possible.

Locate the file **goal.gif** in the Backgrounds folder of the Python Images folder in the Student Files folder. **Right-click to copy** the goal.gif file and then **paste** it into your folder. It will need to be with your Python Score program to be incorporated into your program when you run it.

## PART 3: Creating a Turtle

Before we can draw with a turtle, we have to create the turtle, and adjust its settings.

**Type the code below into a new program:**

```
import turtle

T = turtle.Turtle()

turtle.done()
```

Python is *case sensitive*, so upper and lower case letters are different.
Make sure to capitalize everything properly! ( `turtle` vs. `Turtle` )

To use a function contained in a module, we can use this format: `module.function()`
The `turtle` module contains a function `Turtle()`, which makes a new turtle object.

The `Turtle()` function will then *return* the new turtle object, which means the function sends the value (the turtle object) back to where the function was called in the program.

`turtle.done()` tells Python not to wait for the turtle to draw again, so that we can close the window without it freezing.

When a function returns a value, we can save it in a variable with the form:
`variable = function()` like in the example: `number = random.randint(1,3)`
Using this form, we've created a new turtle object, and stored it in the variable `T`.

## PART 4: Setting a Background Image

Next, we'll add a background image to the screen. This will make our program into a game where you have to move the turtle past an obstacle and into a goal to win!

In order to make a background, we first need to get the screen that our turtle is using. We'll do this using a function that belongs to our turtle, called `getscreen()`.
`getscreen()` returns a screen object, which we'll store in a variable called `screen`.

Afterwards, we'll use the `bgpic()` function of the screen, which lets us set an image.

**Try typing the following code above the line `turtle.done()`:**

```
T = turtle.Turtle()

screen = T.getscreen()

screen.bgpic("goal.gif")             # Use the file suggested by

turtle.done()                        # your instructor.
```

Just like how we called a function belonging to the turtle module using `turtle.Turtle()`, we can call a function belonging to any other object by using the format: `object.function()`

## PART 5: Adjusting the Turtle's Settings

After we've set the background image, we can start changing our turtle's settings.

**Try adding the following code to your program above `turtle.done()`:**

```
screen.bgpic("goal.gif")

T.shape("turtle")

T.speed(1)

turtle.done()
```

Any time we call a function using the format `T.function()`, it will affect our turtle `T`.
In this case, we wanted to change the shape of our `turtle` object `T` to look like a turtle.

After that, we changed the speed of `T` to `1`, which is the slowest speed.
This way we can see what our turtle will do with each command afterwards.

## PART 6: Moving the Turtle

Now that we've changed the shape and speed settings for our turtle, we want to move it past the obstacle in our image, and towards the goal.

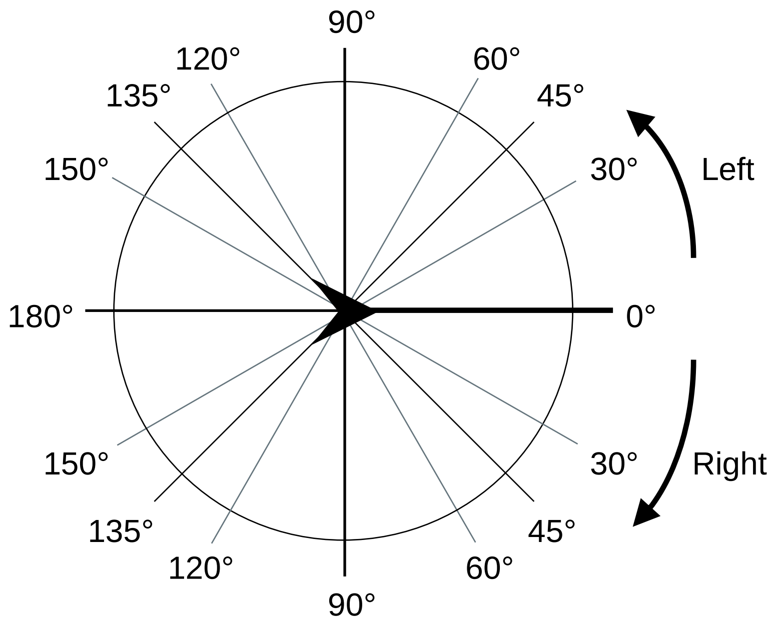
In order to do this, we'll give our turtle instructions, like turning and moving forwards.

Here are some instructions we can give our turtle in order to move it around the screen:

| | |
|---|---|
| `T.forward(`**`p`**`)` | Moves the turtle forwards, in the direction it's facing by **p** pixels. |
| `T.backward(`**`p`**`)` | Moves the turtle backwards from the direction it's facing by **p** pixels. |
| `T.left(`**`d`**`)` | Turns the turtle left by **d** degrees. |
| `T.right(`**`d`**`)` | Turns the turtle right by **d** degrees. |

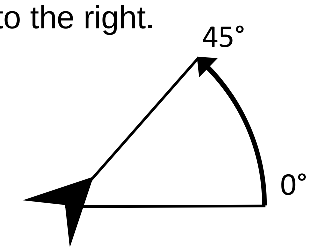
You could also use the shortened versions,
`T.fd(p), T.bk(p), T.lt(d), and T.rt(d)`

For example, to move forwards 30 pixels, we could use: `T.forward(30)` or `T.fd(30)`
To turn left by 45 degrees, we could use: `T.left(45)` or `T.lt(45)`



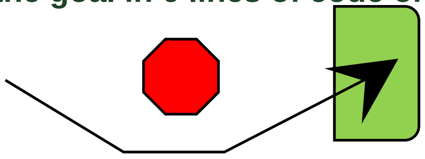
T starts facing to the right

After calling `T.left(45)` our turtle is facing up and to the right.



**Try placing movement instructions for the turtle to avoid the obstacle. See if you can get your turtle to the goal in 6 lines of code or fewer!**

**Here's a hint:**



```
T.speed(1)
T.right(30)
T.forward(100)
T.left(30)
T.forward(100)
T.left(30)
T.forward(100)
turtle.done()
```

Using code like this should make your turtle move around the obstacle on the screen, and let it reach the goal.

## PART 7: Stopping the Turtle

Once our turtle is done drawing, we need to let our program know, so that it doesn't wait for more turtle functions. In order to do that, we've added the `turtle.done()` function.

(This will also let us close the window without it freezing)

**Make sure the last line of your program is `turtle.done()`:**

```
T.forward(100)

turtle.done()   # You've already added this; just double check
                # that it's at the bottom of your program
```

The `done()` function belongs to the `turtle` module, rather than our turtle itself, so we'll call it with `turtle.done()`.

**Now, your turtle should be able to pass the obstacle and reach the goal!**