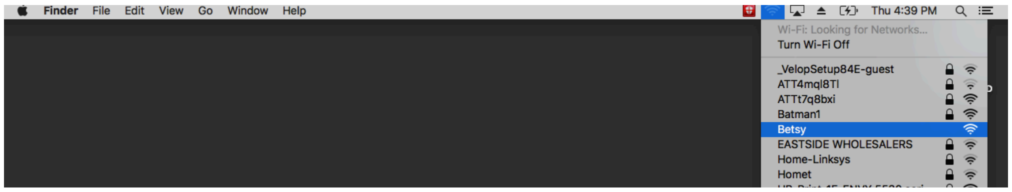
# Robots & Programming with Classes

## General Directions

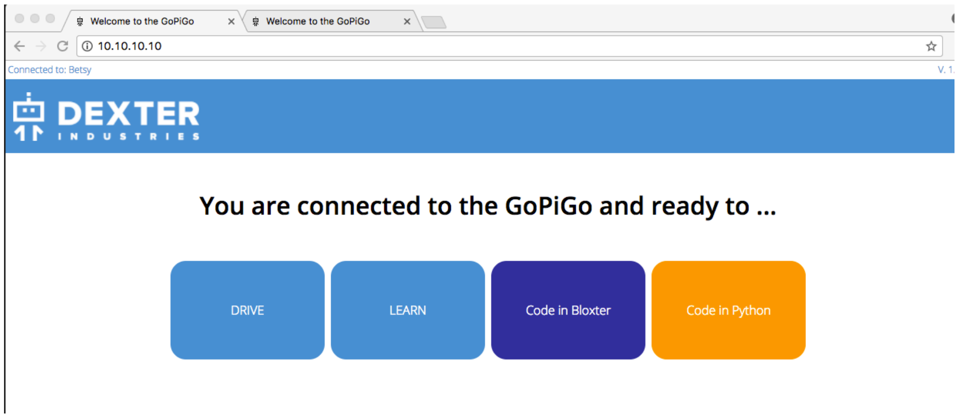
In this lab you will log on to a MacBook Pro and connect to robot. You will use the MacBook Pro to write Python code to make the robot move forward unless there is something in front of it. Otherwise the robot will move right or left or backward if need be. This lab will also reinforce programming with Object Oriented Programming concepts as you will use classes to create objects. These objects will have methods that allow you to make the robot move and allow the robot to see where it is going.

## Getting Connected to the Robot.

1. Turn on the MacBook Pro.
2. When prompted for a user select the soccer ball representing a student.
3. Click on the air waves icon, that is located at the top right of your screen.
4. Find the name of your robot and click on it.

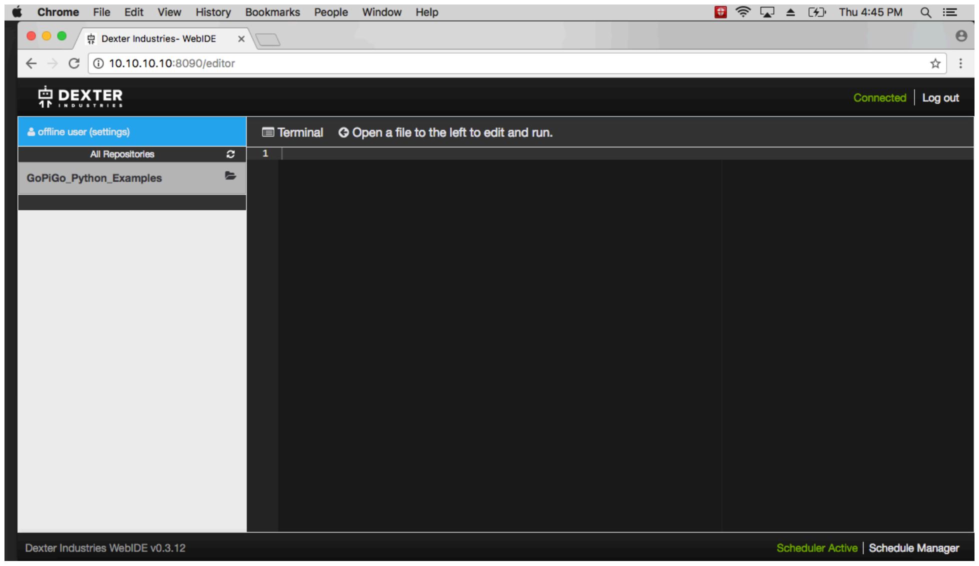


5. Open Google Chrome by clicking on the icon in the tray, located at the bottom of your screen.
6. If the browser does not find your robot, type in the address box 10.10.10.10.
7. You should see the screen below:



You are connected to the GoPiGo and ready to ...

8. Click on the button that reads “Code in Python”.
9. You will see the screen listed below.

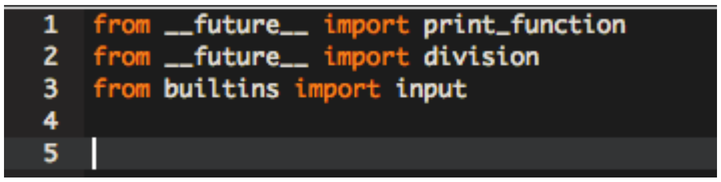


Open a file to the left to edit and run.

10. Click on GoPiGo_Python_Examples. This will be to the left of your screen.
11. Click on Create New File. In the text box enter: RobotClasses.py

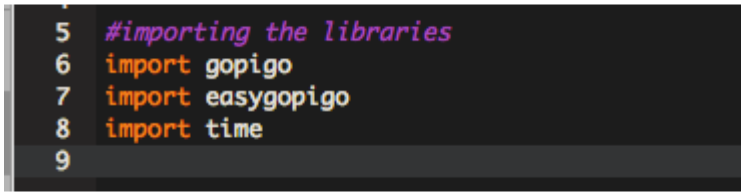
## Learning how to move the Robot

The wonderful thing about working with classes is that we can save a lot of time writing code and spend more time on using code and creating cool projects. In order to work with our robot we need to import some classes. The following code will allow us to write code that is compliant with Python version 3.x in a 2.x compiler.



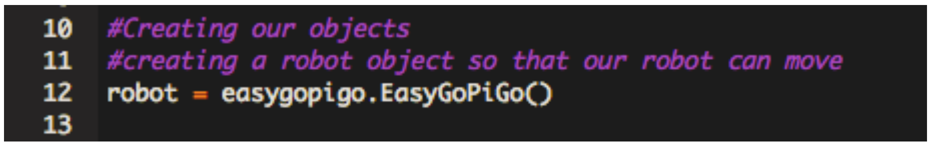
```
from __future__ import print_function
from __future__ import division
from builtins import input

```

Next type in the following code which will import the classes we will need to finish our project.



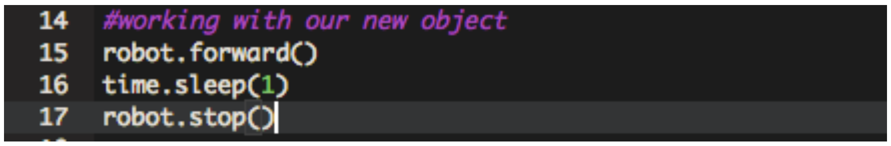
```
#importing the libraries
import gopigo
import easygopigo
import time

```

Now we can create our first object. This object will allow us to move the robot backward, forward, right and left. To create our object enter the following code:



```
#Creating our objects
#creating a robot object so that our robot can move
robot = easygopigo.EasyGoPiGo()

```

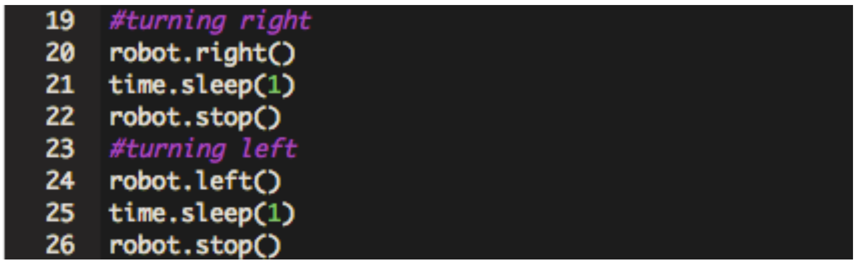
To make the robot move we need to activate certain methods within our object. Enter the code listed below to move your robot forward. The reason we need time.sleep and robot.stop is to make sure our robot stops moving forward. Before you enter this code, make sure your robot is upside down so that it does not fall off the desk.



```
#working with our new object
robot.forward()
time.sleep(1)
robot.stop()
```

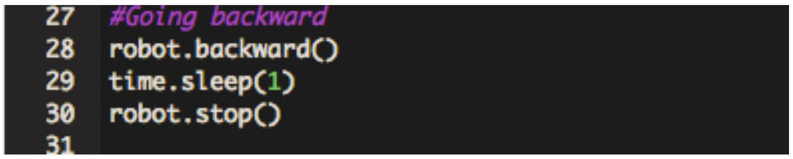
Click on the Run button at the top of your screen to run your code.

Next we will learn how to make the robot turn left and right. When turning we will use the right and left methods. We also need to specify how long we want the robot to engage in a turn. We do this with the sleep method, which is a part of the time class. Click on the Run button to test your code.



```
#turning right
robot.right()
time.sleep(1)
robot.stop()
#turning left
robot.left()
time.sleep(1)
robot.stop()
```

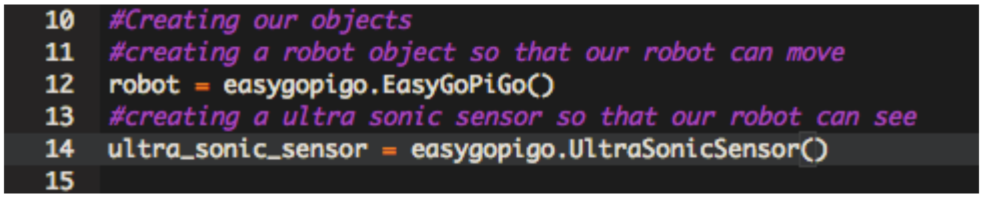
Finally we will let our robot go backward.



```
#Going backward
robot.backward()
time.sleep(1)
robot.stop()

```

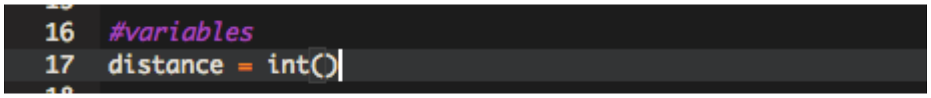
The ultra sonic sensors or eyes of our robot allow the robot to detect if there is anything in front of it. We will use these sensors to see if there is something in front of the robot so that the robot can avoid a collision. To work with the robot's ultrasonic sensor we need to create an ultra sonic object in order to utilize the methods in this specific class.

Under the creation of your robot object, type in the following code to create a new object.



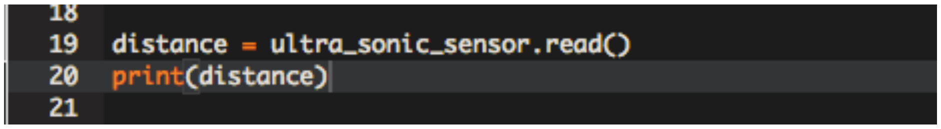
```
#Creating our objects
#creating a robot object so that our robot can move
robot = easygopigo.EasyGoPiGo()
#creating a ultra sonic sensor so that our robot can see
ultra_sonic_sensor = easygopigo.UltraSonicSensor()
```

In order to work with the ultra sonic sensor we need to measure distance. The sensor will measure the distance from the sensors or eyes to the object and return an integer. Therefore we need to create a variable to store the distance. Underneath your creation of the ultra sonic sensor object, declare a variable called distance and assign it to an integer data type.



```
#variables
distance = int()
```

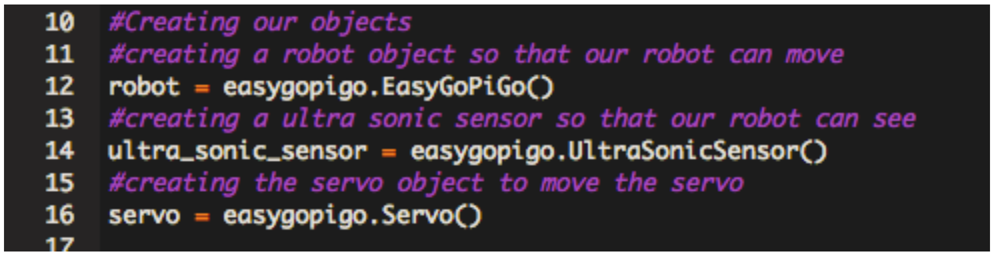
To get a measurement from our robot you will need to enter the following code. Before testing your code make sure you have an object in front of the sensors.



```

distance = ultra_sonic_sensor.read()
print(distance)
```

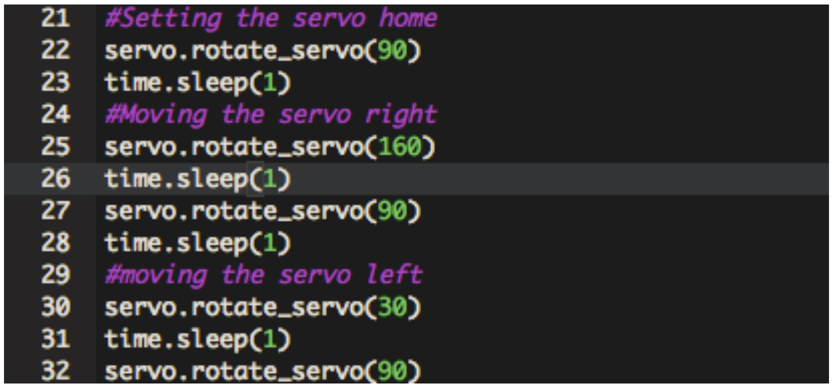
Run your code multiple times. With each test, move your object different distances away from the sensors.

In our final project we want our robot to check the direction it is heading, before committing to that direction. In order to do this we need to be able to move the sensors in that direction. To move the sensors means we need to move the servo, which is the plastic arm and small motor attached to the sensors. Moving the servo requires the creation of yet one more object. At the top of your code, where you have created the other objects create your third and final object to move the servo.



```
#Creating our objects
#creating a robot object so that our robot can move
robot = easygopigo.EasyGoPiGo()
#creating a ultra sonic sensor so that our robot can see
ultra_sonic_sensor = easygopigo.UltraSonicSensor()
#creating the servo object to move the servo
servo = easygopigo.Servo()
```

There is one method within the Servo() class that we need to move the servo and that is rotate_servo. This method does require some data. Specifically it is looking for which direction to move the servo in, and is measured in degrees. 90 degrees is center, 0 is right and 180 is left. When working with the servo we also need to use the sleep method to slow down processing. Enter the following code to move the servo.



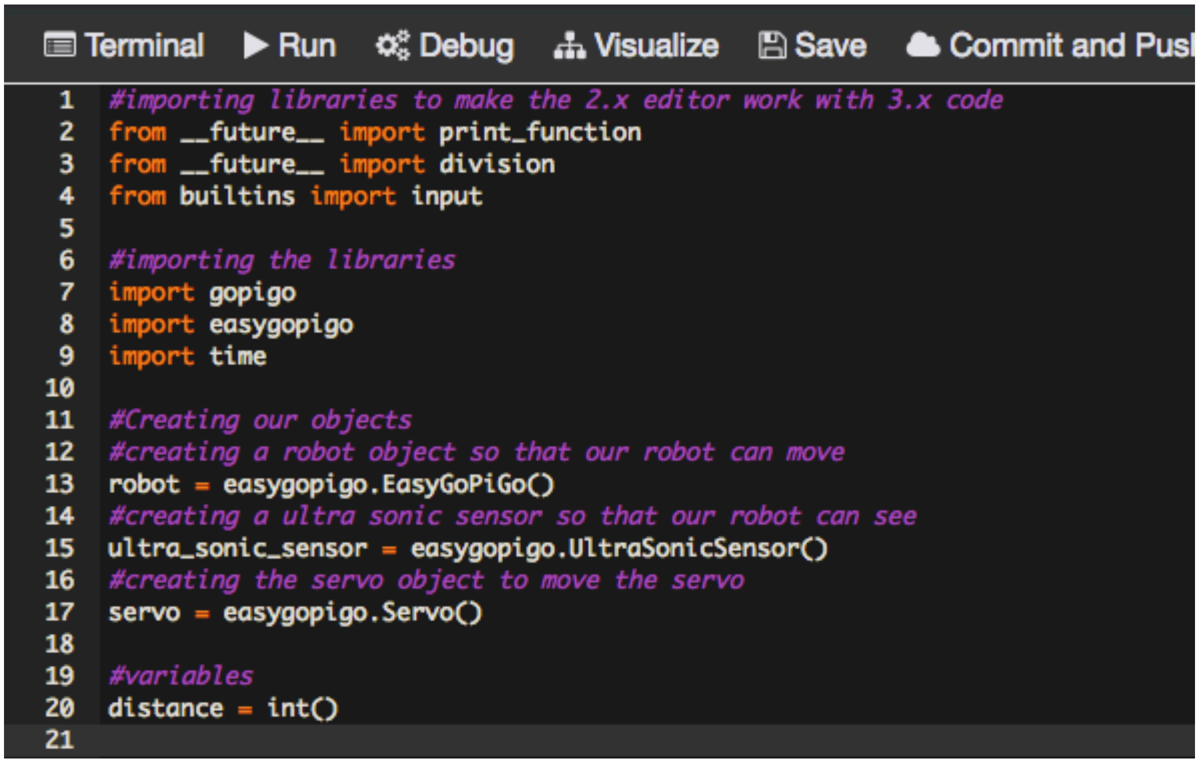
```
#Setting the servo home
servo.rotate_servo(90)
time.sleep(1)
#Moving the servo right
servo.rotate_servo(160)
time.sleep(1)
servo.rotate_servo(90)
time.sleep(1)
#moving the servo left
servo.rotate_servo(30)
time.sleep(1)
servo.rotate_servo(90)
```

## Moving the Robot

In this section of the lab you will write code to allow the robot to move forward. But before it moves forward it will need to check to make sure there is nothing in its way. If something is in its way, the servo will move to the right and check if there is something to the right. If there is nothing there the robot will turn right and move forward. If there is something to the right of the robot, the servo will move to the left and check the direction. If there is nothing there it will turn left and move forward. However, if there is something there, the robot will move backward. Here is an outline of what our robot will do:

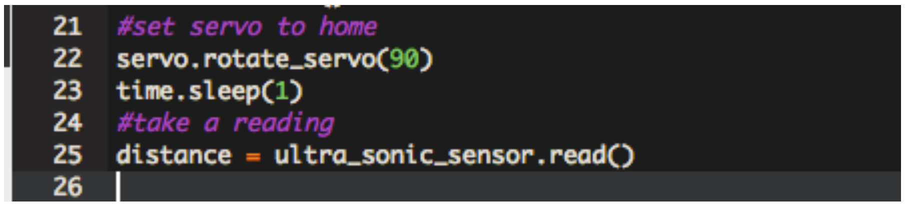
```
Robot check forward
if object detected
     rotate servo right
     get reading
     rotate servo to home
     if something detected
          rotate serve left
          get reading
          rotate servo to home
          if something detected
               move robot backward
          else
               make left turn
               move robot forward
          end if
     else
          make right turn
          move robot forward
     end if
else
     move robot forward
end if
```

When working on something this large it is best to break it up in pieces and test as you go. Go back to your code and copy and paste your code to Text Wrangler or TextEdit.
Only keep the following code:



```python
#importing libraries to make the 2.x editor work with 3.x code
from __future__ import print_function
from __future__ import division
from builtins import input

#importing the libraries
import gopigo
import easygopigo
import time

#Creating our objects
#creating a robot object so that our robot can move
robot = easygopigo.EasyGoPiGo()
#creating a ultra sonic sensor so that our robot can see
ultra_sonic_sensor = easygopigo.UltraSonicSensor()
#creating the servo object to move the servo
servo = easygopigo.Servo()

#variables
distance = int()

```

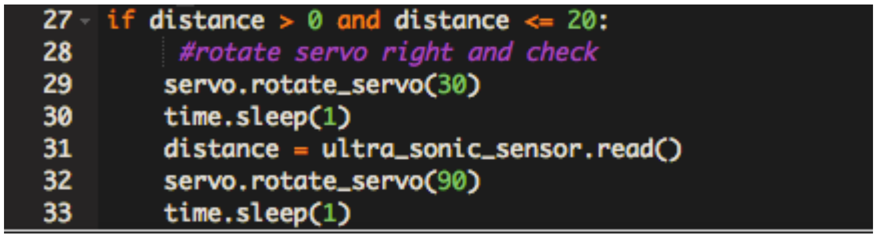
Next set the servo to home and take a reading


```
#set servo to home
servo.rotate_servo(90)
time.sleep(1)
#take a reading
distance = ultra_sonic_sensor.read()

```

We will work on the outer IF structure first and then test. The ultrasonic sensor will take a reading and return back a measurement of when it encounters the first object. This is delivered in centimeters. If we simply wrote this outer IF structure as:

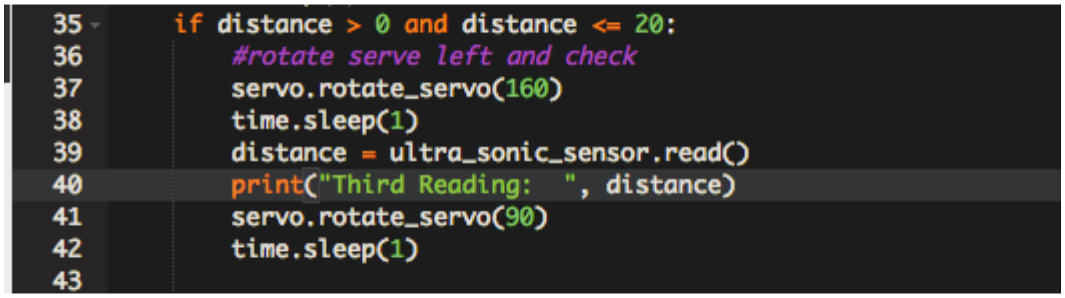
```
if distance > 0
```

what we will find is our robot never moving forward and only moving backward. The reason for this is the sensors will sense something in its path even if it is 300 centimeters away. Our robot does not need 300 centimeters to turn right or left, therefore we will limit our distance to the following:


```
if distance > 0 and distance <= 20:
    #rotate servo right and check
    servo.rotate_servo(30)
    time.sleep(1)
    distance = ultra_sonic_sensor.read()
    servo.rotate_servo(90)
    time.sleep(1)
```

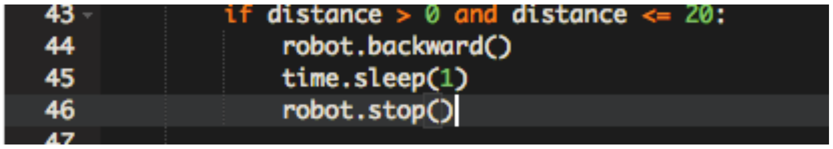
Before testing your code, place your hand in front of the sensors. You should see in the output window a number greater than 0 and less than 20. The servo should then rotate to the right and take another reading. Include a print statement underneath the reading to see what the servo returns. Run your code multiple times testing if the robot senses an object in its path either in front or to the right.

Next we will check the left side of the robot. Enter the following code being careful that this IF structure is indented inside the other IF structure.

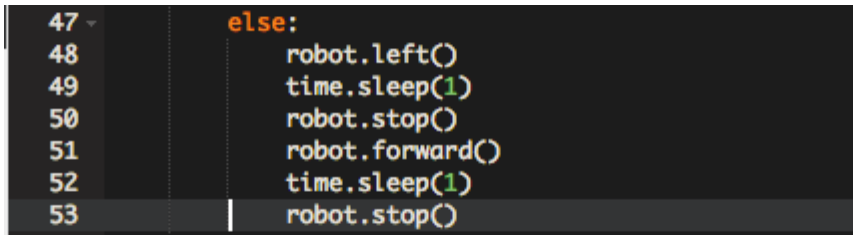

```
    if distance > 0 and distance <= 20:
        #rotate serve left and check
        servo.rotate_servo(160)
        time.sleep(1)
        distance = ultra_sonic_sensor.read()
        print("Third Reading:   ", distance)
        servo.rotate_servo(90)
        time.sleep(1)

```

Once you have entered your code, run it multiple times, placing your hand in front of the sensor to make sure it is getting accurate readings. This IF structure will move the servo to the left, take a reading and then rotate the servo back to home.

Now to get the robot to move. Enter the following code to have the robot move backwards if it detects something in front of it, to the right and to the left. After you enter the code, make sure you test it to make sure your robot moves backward when there is something close to the center. You can easily do this by cupping your hand in front of the sensor and moving your hand when the servo arm moves.

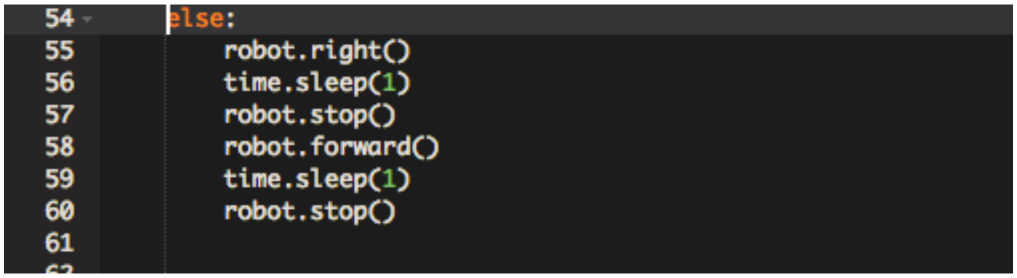

```
        if distance > 0 and distance <= 20:
            robot.backward()
            time.sleep(1)
            robot.stop()
```

Continuing on we need to make our robot turn to avoid an obstruction to its right. After entering in the code, make sure you test it.



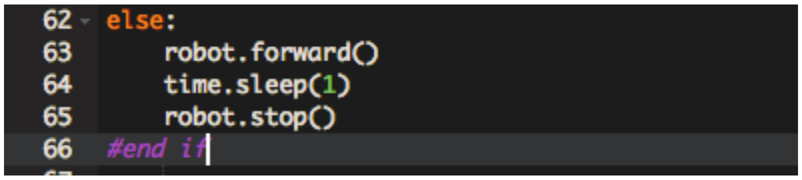
```
        else:
            robot.left()
            time.sleep(1)
            robot.stop()
            robot.forward()
            time.sleep(1)
            robot.stop()
```

Now to avoid an obstruction in front of the robot. Make sure you test.



```
    else:
        robot.right()
        time.sleep(1)
        robot.stop()
        robot.forward()
        time.sleep(1)
        robot.stop()

```

Finally to get your robot to move forward if there are no obstructions.



```
else:
    robot.forward()
    time.sleep(1)
    robot.stop()
#end if
```

Be very careful when typing your code in to preserve the indenting.

If there is time allowing enclose this IF structure in a loop to allow your robot to roam around the room.

When you are done working with your robot, sign out of the editor and go back to the main screen. Click on Drive Robot. Click on the File menu and select Shut down. Click on all buttons and dialog boxes confirming the shut down.

## Deliverable

1. Download the Extra Credit worksheet from the Makerspace website.
2. Print out the worksheet and fill in the appropriate information.
3. Sign the form.
4. Have the lab aid on duty sign the form.
5. Turn it in to your instructor.