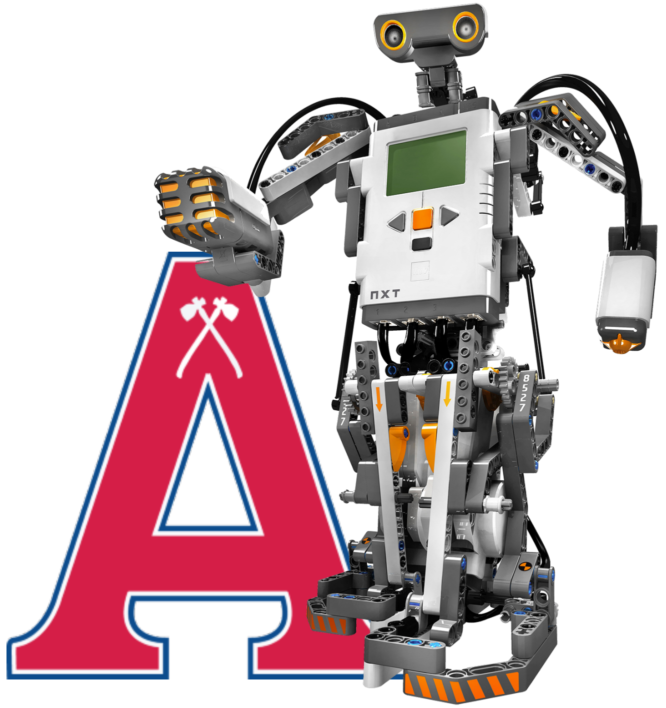
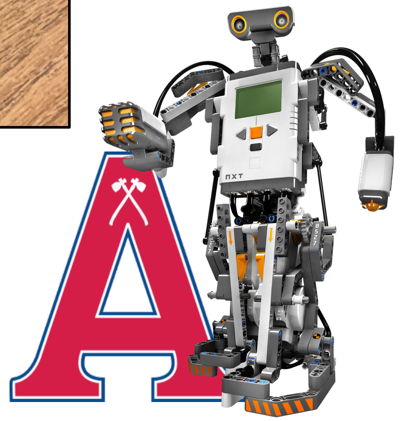
Acadia Robotics BottleSumo Workshop

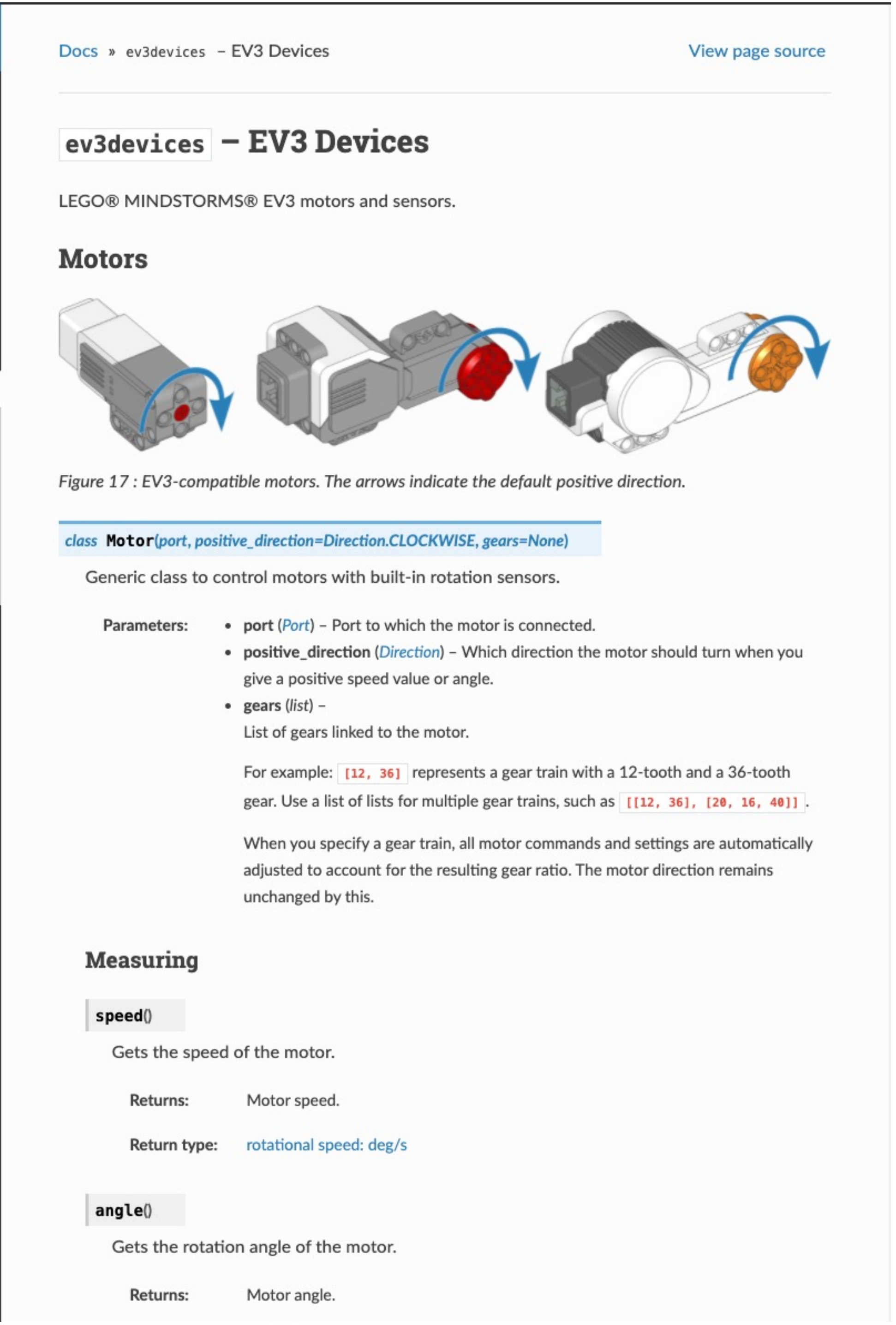
Python for the LEGO EV3



Getting Started

- First, you’ll need to follow the steps below to set up your EV3 brick and Visual Studio Code to be able to being coding your robot.
- The videos are super helpful for getting everything set up, but you’ll need a MicroSD card and your EV3 brick.
- https://education.lego.com/en-us/product-resources/mindstorms-ev3/teacher-resources/python-for-ev3





Docs » `ev3devices` – EV3 Devices   View page source

`ev3devices` – EV3 Devices

LEGO® MINDSTORMS® EV3 motors and sensors.

Motors

Figure 17 : EV3-compatible motors. The arrows indicate the default positive direction.

class **`Motor`**(*port, positive_direction=Direction.CLOCKWISE, gears=None*)

Generic class to control motors with built-in rotation sensors.

Parameters:
- **port** (*Port*) – Port to which the motor is connected.
- **positive_direction** (*Direction*) – Which direction the motor should turn when you give a positive speed value or angle.
- **gears** (*list*) – List of gears linked to the motor.

  For example: `[12, 36]` represents a gear train with a 12-tooth and a 36-tooth gear. Use a list of lists for multiple gear trains, such as `[[12, 36], [20, 16, 40]]`.

  When you specify a gear train, all motor commands and settings are automatically adjusted to account for the resulting gear ratio. The motor direction remains unchanged by this.

Measuring

`speed`()

Gets the speed of the motor.

Returns: Motor speed.

Return type: rotational speed: deg/s

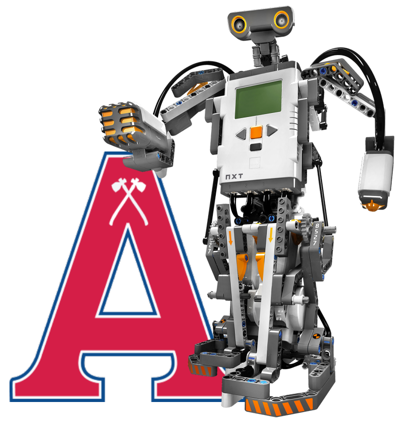
`angle`()

Gets the rotation angle of the motor.

Returns: Motor angle.

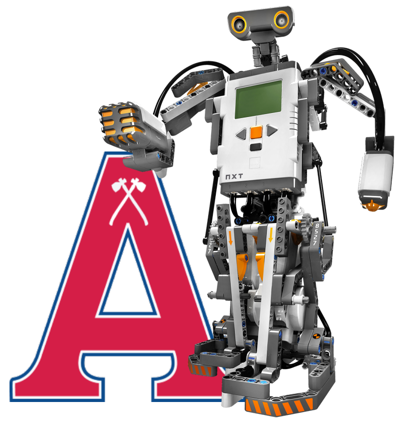
Python Commands Defined

- I found the documentation on commands linked below extremely helpful to figure out the commands I had available for each sensor/motor and how to use them.
- https://pybricks.com/ev3-micropython/ev3devices.html
- https://pybricks.com/ev3-micropython/robotics.html



Building Programming Skills

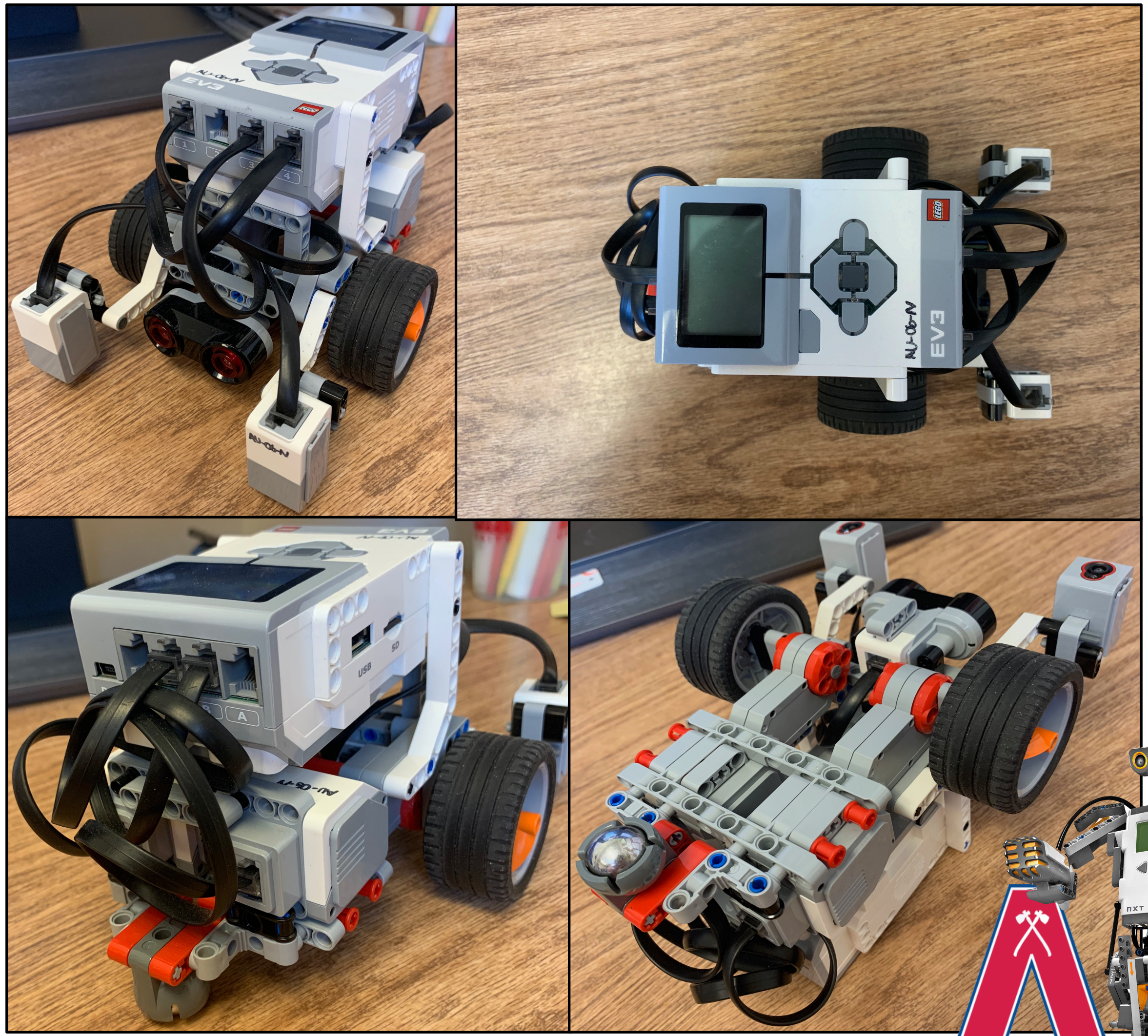
- Consider your robot design and configuration.
- Breakdown the tasks your robot needs to be able to accomplish
  - Find the edge of the table:
    - Stay on the table.
    - Find the edge with two sensors.
  - Find the bottle.
  - Push the bottle off the table.
    - Robot must stay on the table.

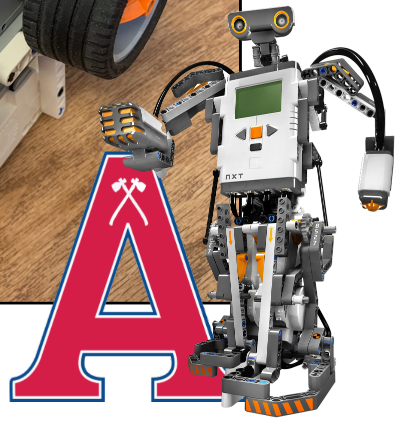


Robot Design

- Two colour/light sensors.
  (One will work but two is recommended)
- Your light/colour sensor(s) should be IN FRONT of your wheels.
- You can choose any robot design you want if it is per the robot rules provided for the event.
- For this tutorial, the motors and sensor are assumed to be in the pictured configuration.

The Robot pictured below is 20cm x 15 cm x 12 cm (L x W x H)





Rules on robot size can be found in the BottleSumo Time Trial Rules.

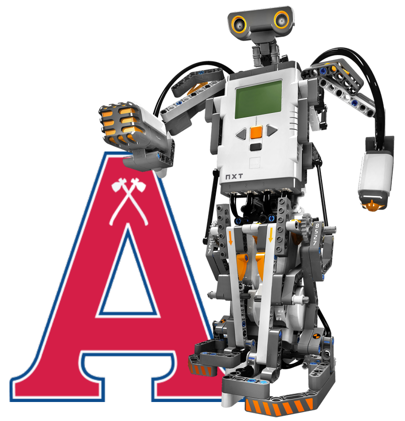
Motor and Sensor Ports

Remember to adjust your programming based on your own robot configuration.

Our robot is set up with sensors in these ports:

- Left Motor connects to Port B.
- Right Motor connects to Port C.

- Left Light/Colour Sensor connects to Port 1.
- Right Light/Colour Sensor connects to Port 3.
- Ultrasonic Sensor connects to Port 4.

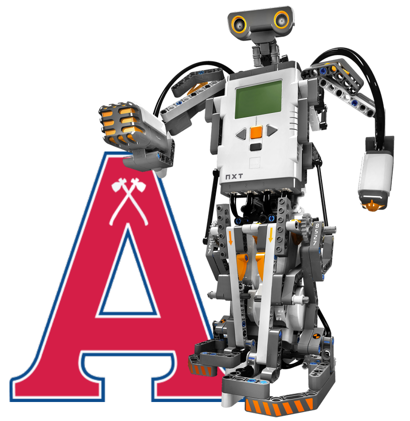


How to Use Your Sensor(s) Readings

- To monitor your sensors, you'll need to look at your EV3 brick screen.
- To get to the sensors from the home screen we'll need to go through

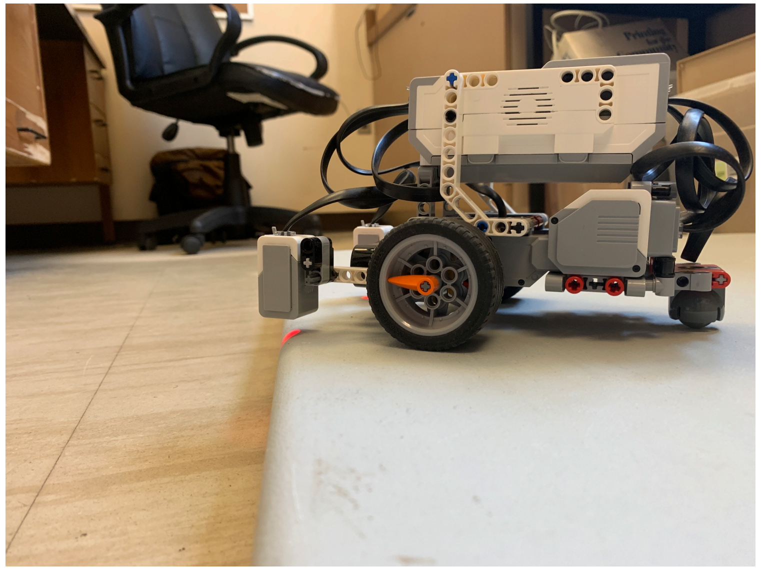
    Device Browser > Sensors

- Then we'll need to click the sensor we want to look at and select Watch Values.
- In this same location we can also see the mode of our sensors and in our case both Light/Colour sensors should be set to COL-REFLECT and our Ultrasonic sensor should be set to US-DIST-CM.



How to Use Your Sensor(s) Readings Pt.2

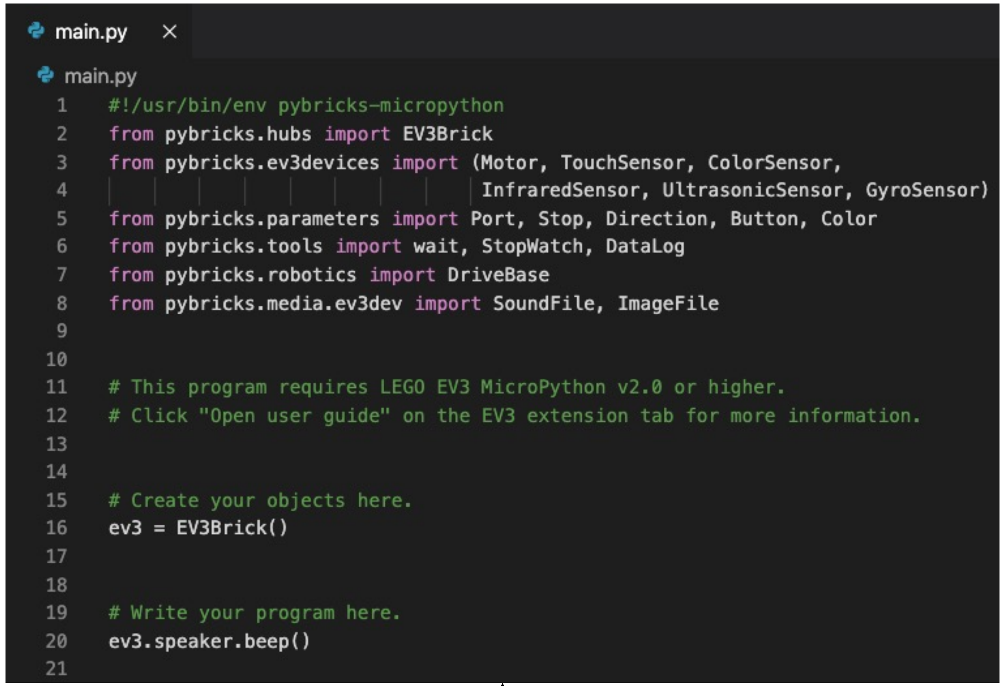
- Now we’ll need to know the threshold value for each Light/Colour Sensor. This indicates when our robot is about to drive off the table.
- To find these values you’ll have to monitor each of the sensor's outputs on your EV3 brick and find the percentage when it is fully on the table and the percentage when it is about to drive off the table.



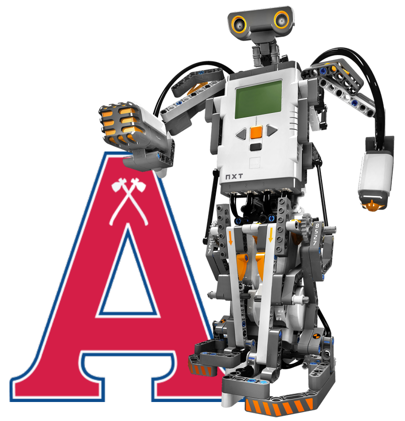
- I found this by slowly moving my Light/Colour sensor closer to the edge of the table until I saw a significant drop from the numbers I recorded when it was fully on the table.
- My threshold values were 20% for my Right Light/Colour Sensor and 10% for my Left Light/Colour Sensor.

Starting to Program

- When you start your program, you'll need to start by creating a project through VS Code.
- To do this click on the LEGO MINDSTORMS EV3 MicroPython extension in VS Code and select "Create a new project"
- This will create a main.py file for you to do most of your coding which looks like this:

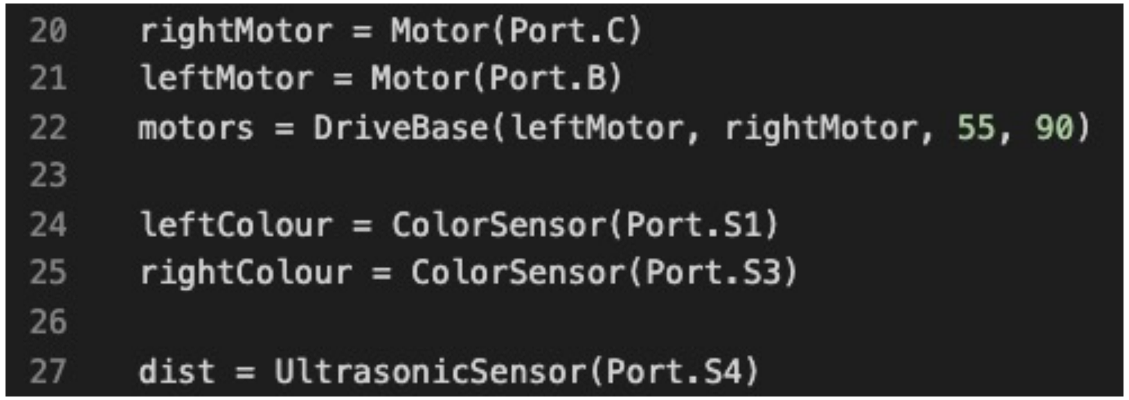


```python
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile


# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.


# Create your objects here.
ev3 = EV3Brick()


# Write your program here.
ev3.speaker.beep()
```



Starting to Program Pt.2

- Now that you've created a project, you'll need to initialize all your sensors and motors so we can start doing stuff with them.
- Referring the documentation mentioned earlier will help you at this point.
- I initialized mine as follows:

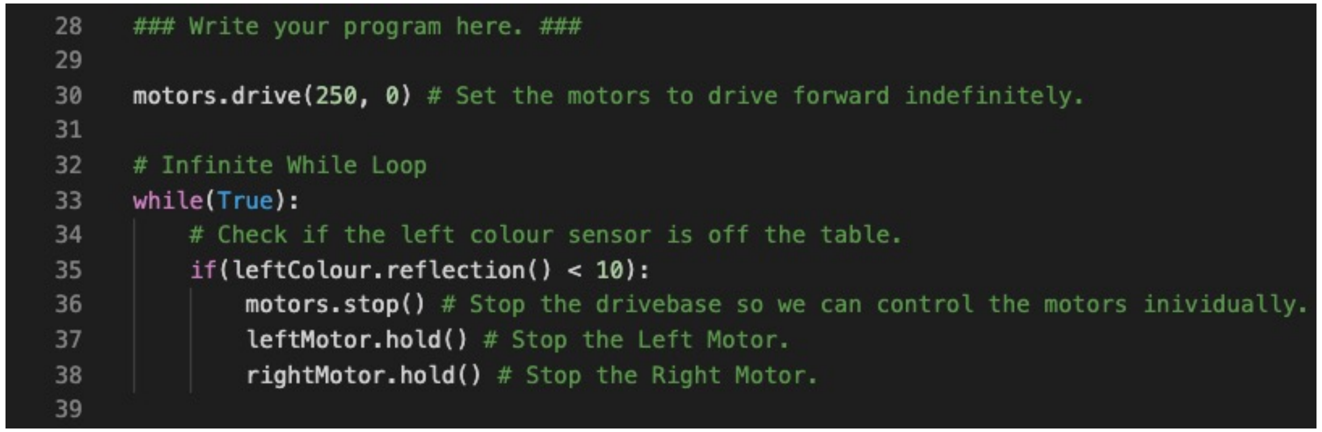


```
rightMotor = Motor(Port.C)
leftMotor = Motor(Port.B)
motors = DriveBase(leftMotor, rightMotor, 55, 90)

leftColour = ColorSensor(Port.S1)
rightColour = ColorSensor(Port.S3)

dist = UltrasonicSensor(Port.S4)
```

Look back to the documentation to see why I have the numbers 55 and 90 in my DriveBase as this is something you will have to figure out for your robot.

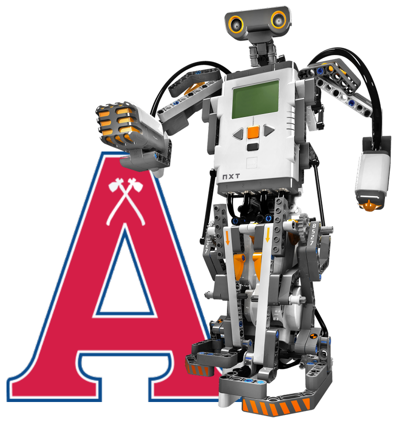
Finding the Table Edge with One Colour Sensor

- Your robot needs to stay on the table for the duration of the time trial.
- You must program the robot to find the edge of the table and STOP.
- To do this, you must continuously check the sensor to see if the light/colour threshold we found earlier is reached. This means we've found the edge of the table.
- This should look something like this:



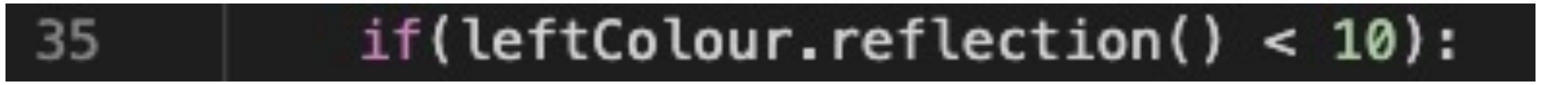
```
### Write your program here. ###

motors.drive(250, 0) # Set the motors to drive forward indefinitely.

# Infinite While Loop
while(True):
    # Check if the left colour sensor is off the table.
    if(leftColour.reflection() < 10):
        motors.stop() # Stop the drivebase so we can control the motors inividually.
        leftMotor.hold() # Stop the Left Motor.
        rightMotor.hold() # Stop the Right Motor.

```

In this example, you'll notice that we stop the motors and then brake them individually. This is because the stop() method lets the motors spin freely but causes the drive base to stop controlling them. This then allows us to access each motor individually and tell them to hold their current position.



Potential Problems at this Point

- If the robot stops too soon or too late try adjusting the light threshold



```
    if(leftColour.reflection() < 10):
```

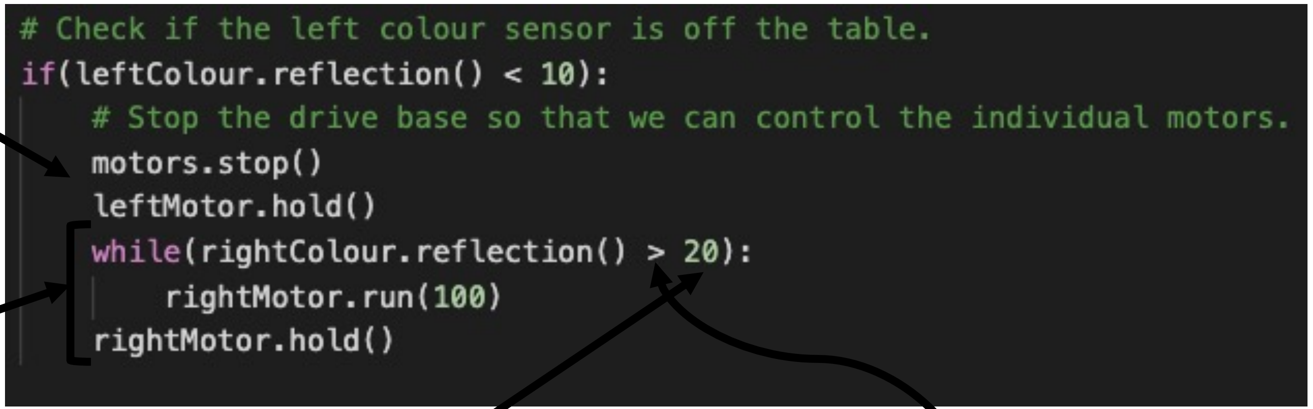
- As of right now the robot is only looking for the edge of the table with the left sensor.

What happens if the right side of the robot goes off first?

Square the Robot to the Table Edge

Two Colour Sensor Method

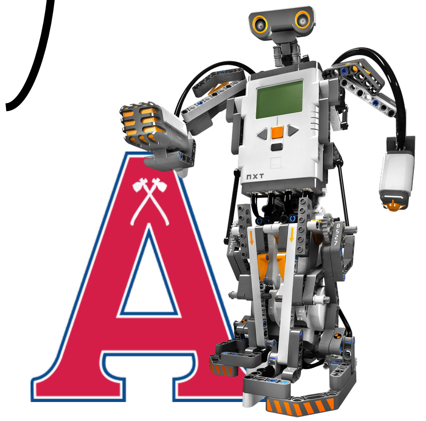
- The robot will stop the left motor when the left sensor goes off the table.
- The right motor will continue until the right sensor goes off the table, then the right motor will stop.
- The robot is now square with the table.



```
# Check if the left colour sensor is off the table.
if(leftColour.reflection() < 10):
    # Stop the drive base so that we can control the individual motors.
    motors.stop()
    leftMotor.hold()
    while(rightColour.reflection() > 20):
        rightMotor.run(100)
    rightMotor.hold()
```

Remember when using the right Light/Colour Sensor (to use the threshold value.)

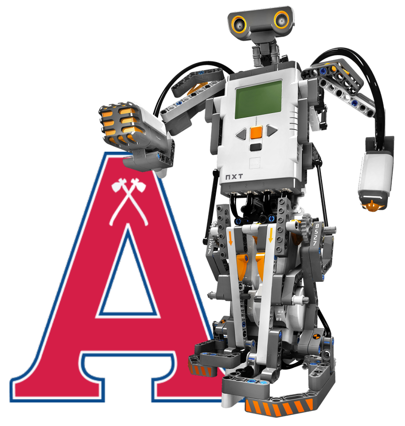
Also, Notice how we use a greater than sign in the **while(rightColour.reflection() > 20)** This is because we want to keep looping until we hit the threshold



The Right Colour Sensor

- Right now, the code will square the robot to the table if the left sensor goes off the table, but it does nothing if the right sensor goes off the table first.
- Solution: Duplicate the code we just wrote and change it to do the same thing for the right sensor!
- Make sure to change your threshold value to correspond to each colour sensor as both sensors may have different thresholds!

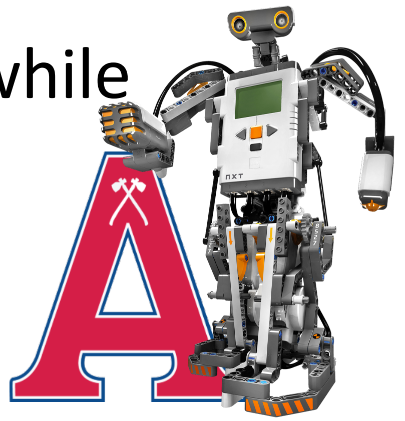
Our robot should square to the edge of the table on both sides but now what do we do?



Using Both Colour Sensors

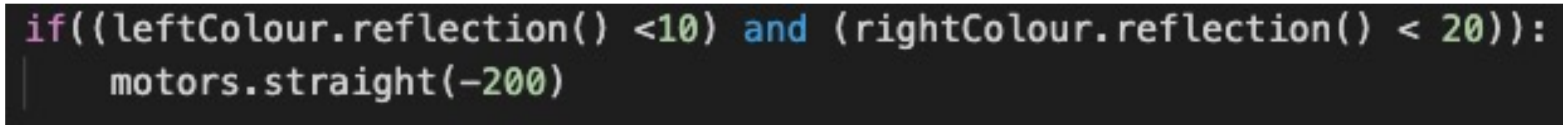
- Now that we've squared to the edge of the table we need to keep searching for the bottle.
- To do this we'll have to back away from the edge of the table and turn to start going another direction.
- We only want this to happen once the robot is squared to the table meaning both colour sensors are off the table.
  - We'll need an if statement to check this!

- The code for this part should be the first thing inside of your "while loop" above the other two if statements we already wrote.

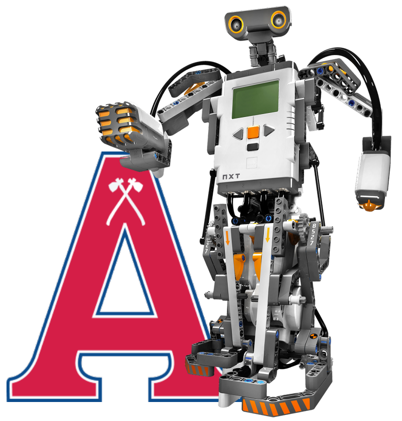


Both Colour Sensors – Reversing

- For this we'll have to check if sensor1 < sensor1_threshold AND if sensor2 < sensor2_threshold.
- Assuming this is true, we'll want to move backwards. This can be done using drivebase.straight(distance {mm})
    - Give straight a negative distance in mm for the robot to make it reverse that distance.
    - A good distance for this is -200 which is 20 cm in reverse.



```
if((leftColour.reflection() <10) and (rightColour.reflection() < 20)):
    motors.straight(-200)
```

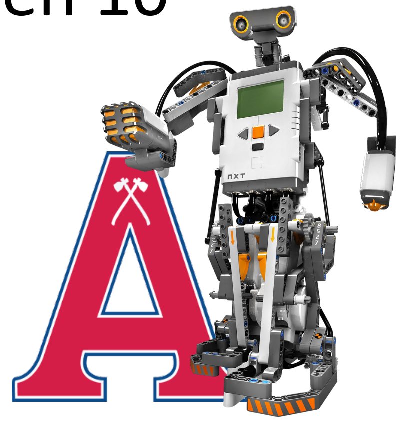


Both Colour Sensors – Turning

- For this we’ll want to turn some amount to continue trying to find the bottle.
- This can be done using drivebase.turn(degrees)
- I recommend using some wide angle such as 135 degrees to start so you can cover a wide range of the table. For example, if you use 90 degrees, you’ll cover the same rectangle over and over.
- For mine I used the python library random to generate a random integer between 1 and 27 to make a turn of somewhere between 10 and 270 degrees.

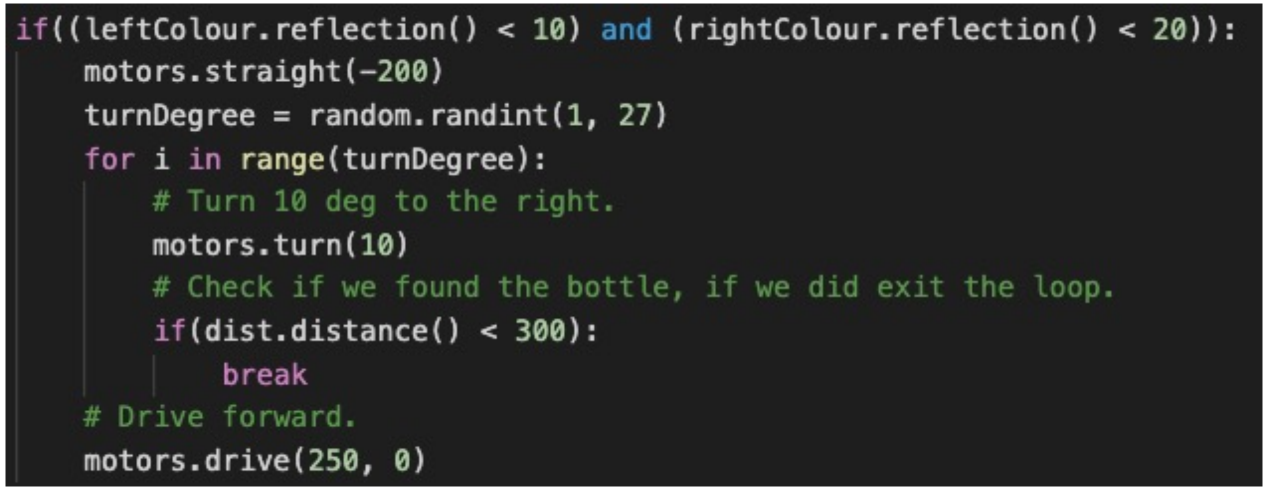
If you want to use my method of generating a random number refer to the next slide and this documentation: https://docs.python.org/3/library/random.html#functions-for-integers

Keep in mind this require doing an ’import random’

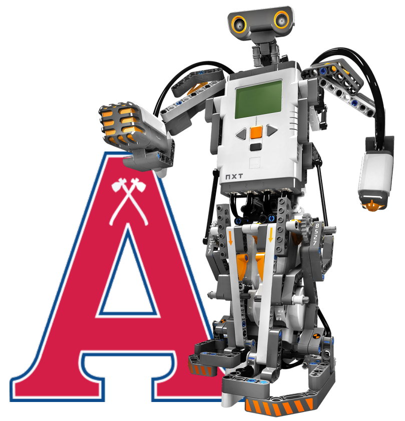


Both Colour Sensors – Turning Pt. 2

- For turning, I would also recommend looping a small turn degree multiple times so you can check for the bottle throughout your turn.
- For mine I used a “for loop” which used the random integer I generated as the number of runs it would do.
- In each run of the “for loop” I had my robot turn 10 degrees and check to see if the bottle was in front of the robot to get maximum coverage of the table.
  - If the bottle was in front of the robot, I used a break statement to break out of the loop and started driving forward.

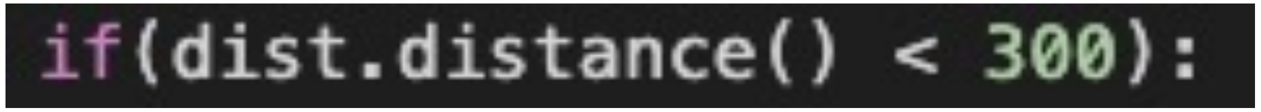


```
if((leftColour.reflection() < 10) and (rightColour.reflection() < 20)):
    motors.straight(-200)
    turnDegree = random.randint(1, 27)
    for i in range(turnDegree):
        # Turn 10 deg to the right.
        motors.turn(10)
        # Check if we found the bottle, if we did exit the loop.
        if(dist.distance() < 300):
            break
    # Drive forward.
    motors.drive(250, 0)
```

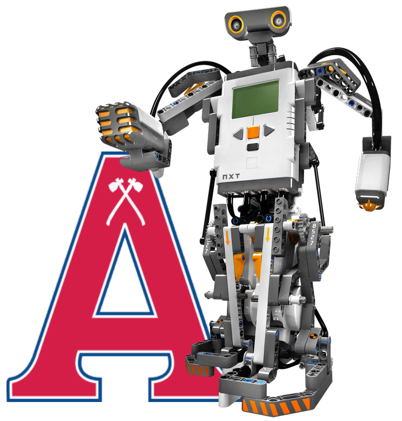


Finding the Bottle – Pt. 1

- I glossed over this in the last slide, but we are using our ultrasonic sensor to make our robot look for the bottle.
- To do this we use the UltrasonicSensor.distance() to get the distance of things in front of our robot.
  - This just returns the distance of anything in front of our robot up to 2 meters away.
  - We don't want our robot to be looking this far away for the bottle.
  - I chose to only determine that the bottle was found if the sensor detects an object 30 cm away.
    - The distance() function returns the distance in mm so 300 mm = 30 cm.

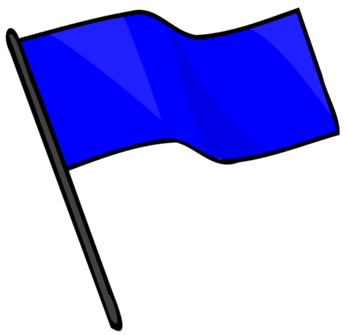


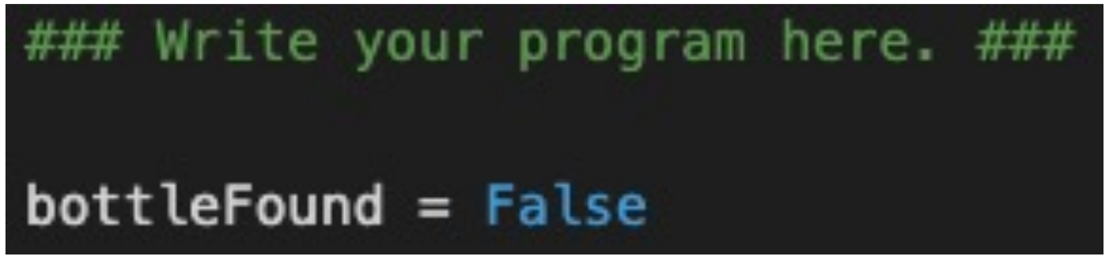
```
if(dist.distance() < 300):
```



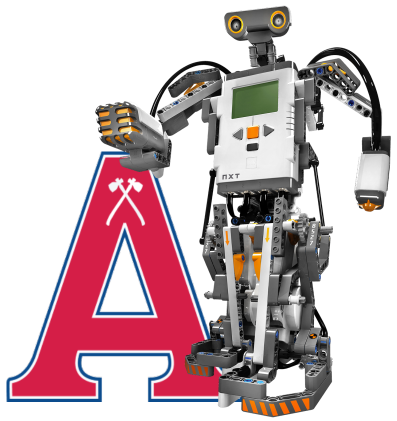
Finding the Bottle – Pt. 2

- Before we get into the rest of the code on finding the bottle go back to the beginning of your program. We're going to create a flag variable for if the bottle is found which will be useful later.
- A flag variable is a variable that's value changes when something occurs to let the program know in other parts that something has occurred.
- This should look something like this:





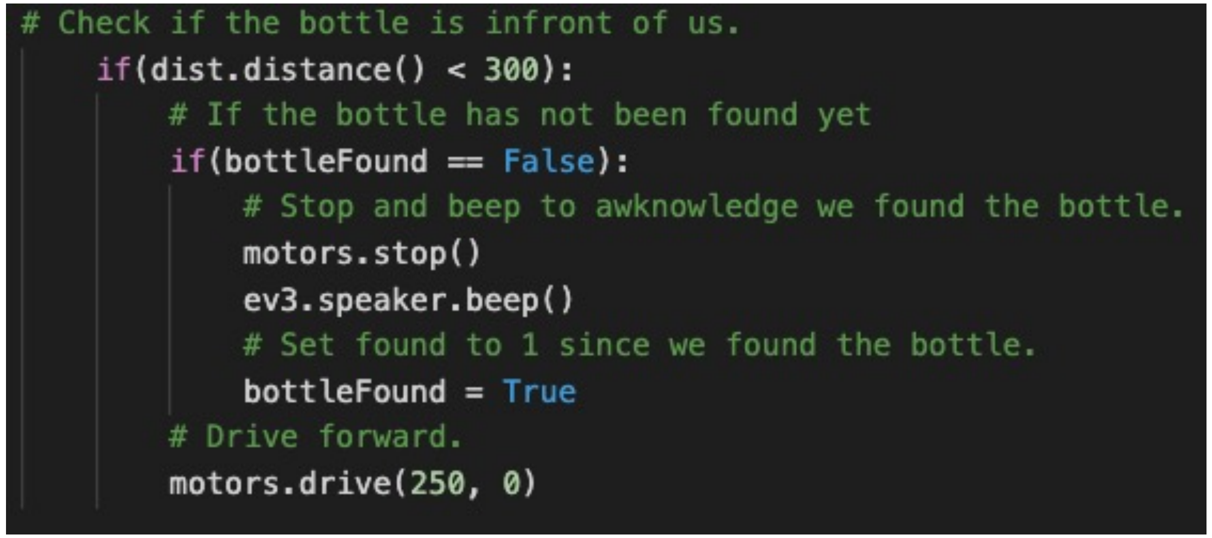
```
### Write your program here. ###

bottleFound = False
```



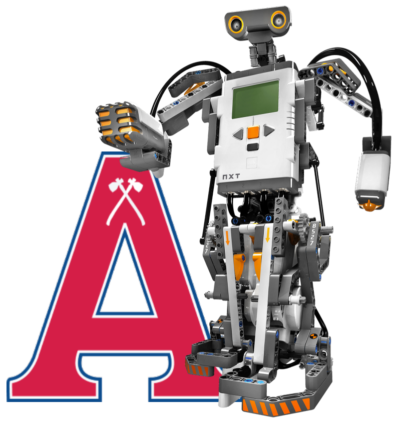
Finding the Bottle – Pt. 3

- Now that we have our flag made, we first want to check if the flag is still false once our our sensor finds something less than 30 cm away from the robot.
  - If it does set the flag to True.
  - I had my robot stop and beep to indicate that it found the bottle to me.
  - Then have your robot drive forward so that it pushes the bottle off of the table.

Are we done now?

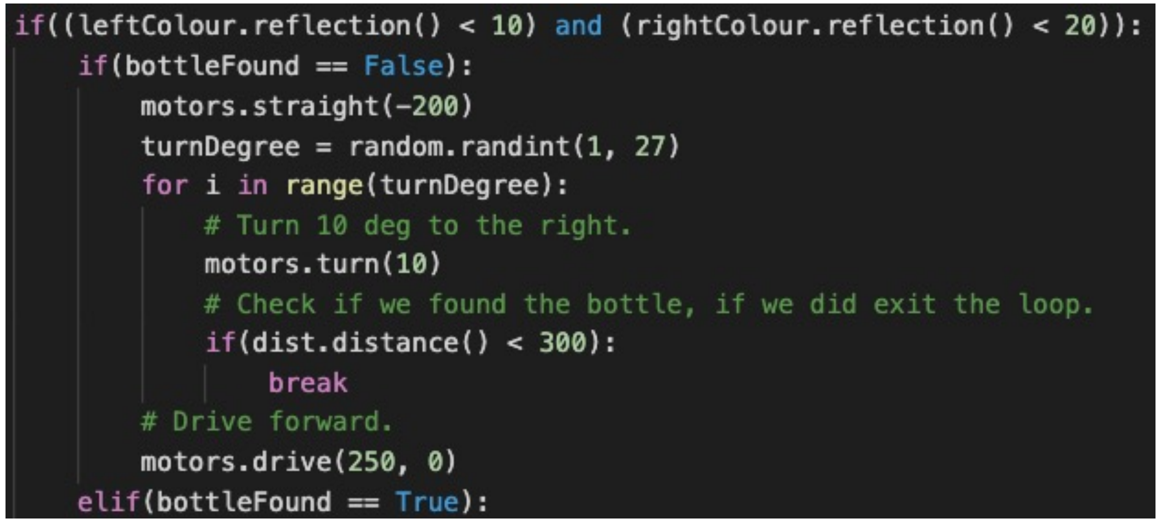


```
# Check if the bottle is infront of us.
    if(dist.distance() < 300):
        # If the bottle has not been found yet
        if(bottleFound == False):
            # Stop and beep to awknowledge we found the bottle.
            motors.stop()
            ev3.speaker.beep()
            # Set found to 1 since we found the bottle.
            bottleFound = True
        # Drive forward.
        motors.drive(250, 0)
```

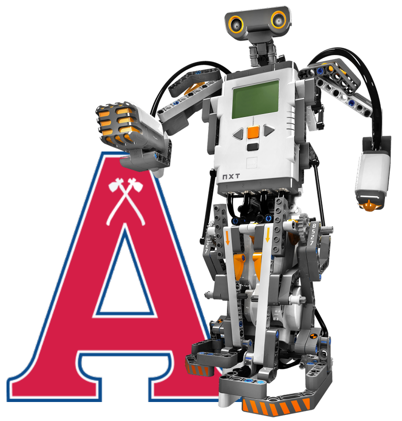


Finding the Bottle – Pt. 4

- Now we need to create a condition where the program will stop.
- Go back to your if statement where we check both light/colour sensors.
  - We're going to put everything we had inside of that if statement inside of a second if statement that occurs if the bottle has not been found yet.
  - It should look something like this now:

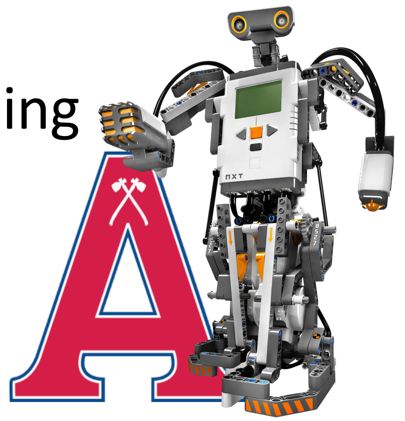


```
if((leftColour.reflection() < 10) and (rightColour.reflection() < 20)):
    if(bottleFound == False):
        motors.straight(-200)
        turnDegree = random.randint(1, 27)
        for i in range(turnDegree):
            # Turn 10 deg to the right.
            motors.turn(10)
            # Check if we found the bottle, if we did exit the loop.
            if(dist.distance() < 300):
                break
        # Drive forward.
        motors.drive(250, 0)
    elif(bottleFound == True):
```



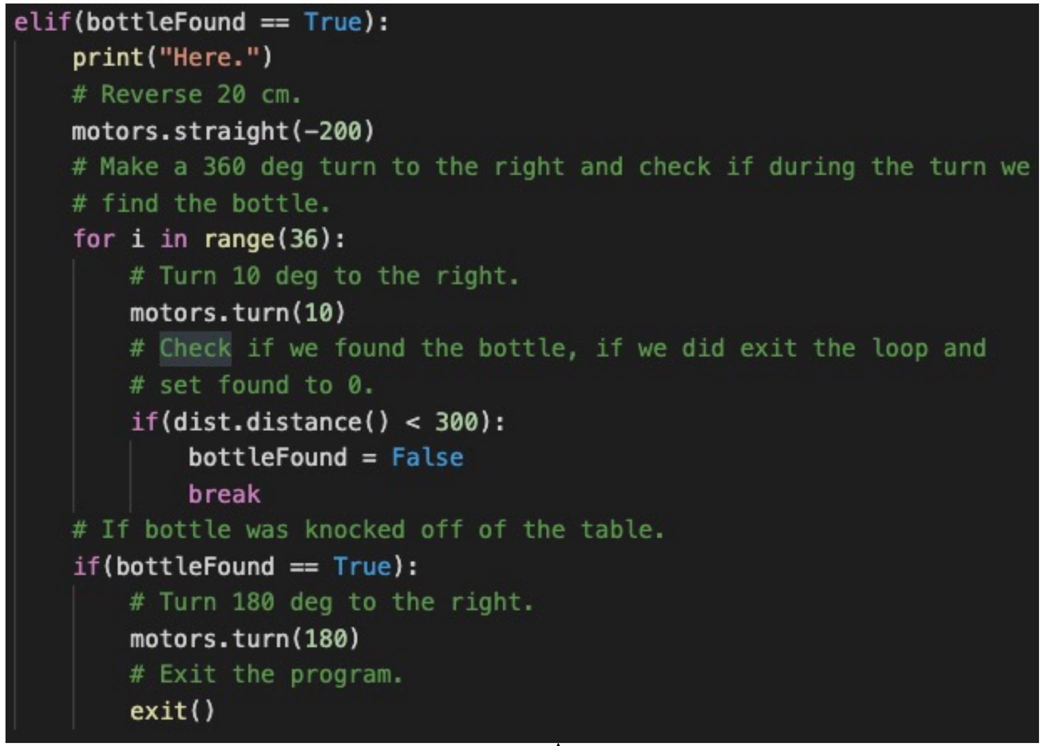
Finding the Bottle – Pt. 5

- When the robot pushes the bottle off, you'll notice that both sensors will go off the table and the robot will square itself to the table.
  - If we've already found the bottle and eliminated it, the program can end now.
    - But what if we missed the bottle?
- Let's have the robot back up again so it's no longer on the edge.
- Then we should check to make sure we actually eliminated the bottle.
  - To do this let's have our robot do the same method of turning as before but we'll force it to do a 360-degree turn checking every 10 degrees to make sure the bottle isn't still in its view.
  - If the bottle is still in view set our flag to False and break out of the turning loop.
    - This will allow our found bottle code to re-execute, and our robot will attempt to eliminate the bottle again.

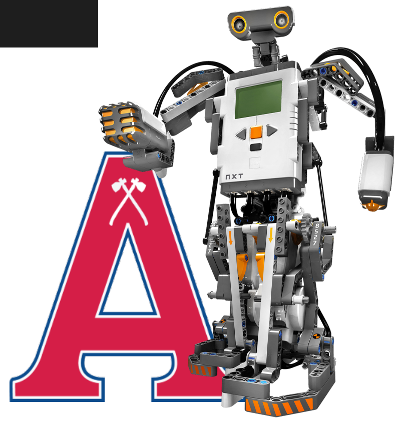


Finding the Bottle – Pt. 6

- When the loop ends or is broken, we'll do another check on our flag to see if the loop changed its value.
- If the flag is still set to True, then we can exit the program as the bottle has been eliminated.
- At this point the code should look like this:



```
elif(bottleFound == True):
    print("Here.")
    # Reverse 20 cm.
    motors.straight(-200)
    # Make a 360 deg turn to the right and check if during the turn we
    # find the bottle.
    for i in range(36):
        # Turn 10 deg to the right.
        motors.turn(10)
        # Check if we found the bottle, if we did exit the loop and
        # set found to 0.
        if(dist.distance() < 300):
            bottleFound = False
            break
    # If bottle was knocked off of the table.
    if(bottleFound == True):
        # Turn 180 deg to the right.
        motors.turn(180)
        # Exit the program.
        exit()
```



This Concludes Our Workshop!

- At this point your code should be functional all though I'm sure you'll find ways to make improvements!
- Thank you for completing this workshop put together by Acadia Robotics.