Middle School Robotics Lessons

Using iRobot Create**®**

and Python™

By Scott Maxwell

Nashua Catholic Regional Junior High School

Nashua, NH

6/1/17

Table of Contents

1. Introduction
2. Lesson 1- Understanding Python
3. Lesson 2- Changing Python Code
4. Lesson 3- Understanding Robot Code
5. Lesson 4- Writing Robot Code

# Introduction

Teaching middle school computer classes today is far different than it was when I was a middle school student in the late 1980’s. As a student, I had class in the computer lab once a month. The technology was still emerging and it was exciting to enter the lab and play a game of Oregon Trail on the school’s Apple II computers. Those of us fortunate enough to have a computer at home were able to type up our reports and print them on our dot matrix printers to wow our teachers. Today, computers are an integral part of everyday life and students are more advanced by age twelve than I ever dreamed of being.

The curriculum of a computer class has to constantly evolve with the times. Not long ago, a year-long course on Microsoft Office was sufficient to meet the needs of our students. In fact, it’s the curriculum many current computer teachers likely inherited. Nashua Catholic seventh grade students now master the basics of word processing, spreadsheets and presentation software by Christmas break. They are arriving from elementary schools with a working knowledge of these programs and are ready to move to the next level. This enables us to introduce topics such as 3D design, 3D printing, digital music creation, basic coding and more to our students. Our principal, Mrs. Glenda McFadden, not only wants us to keep up with the times, she wants us to be innovators. That is where this partnership began with iRobot.

Unable to find much in the way of an existing accessible yet challenging middle school curriculum, the school began to do some exploring. With the assistance of STEM coordinator Mr. Dean Winkelmann, we were able to set up a meeting with Mr. Paul Schmitt from iRobot and discuss the possibilities of working together. From those discussions we had the framework and materials needed to get this project off the ground, and we’re very excited to share the results with you.

***Lesson 1: Understanding Python***

**Time:** 45-50 minutes

**Materials:** Computers for each student with internet access

Word processing document such as Google Docs or Word.

**Prior Knowledge:** Block Coding

|  |
| --- |
| **Objective:** *Students will take basic knowledge of block coding and understand the basic code underneath.* |

**Introducing Text**

* Direct students to repl.it or similar web site where Python code can be run in a web browser.
* Introduce the print command and explain how syntax is important. If parenthesis or quotes are missing, students.
* Have students enter the code:

print ('Hello, World!')

* Now have them run the code and see what happens.
* Instruct students to replace ‘Hello World’ with a greeting using their name. For example:

print ('Hello, my name is Scott')

* Have students run the code again to see their changes.

**Introducing Math**

* Explain basic math functions:
  + + for addition
  + - for subtraction
  + \* for multiplication
  + / for division
* Now have students enter code using the math functions:
  + print(6+2)
  + print(6-2)
  + print(6\*2)
  + print(6/2)
* Review the order of operations and have students try to figure out how they would find the average of three test grades: 97, 95, 92. Have students explain why the following code won’t work:
  + print(97+95+92/3)
* Students should deduce that an extra set of parenthesis are needed to get the desired result. The corrected code should be:
  + print((97+95+92)/3)

|  |
| --- |
| **Assessment:** *Students will create and run basic code in Python.* |

* Give students the following tasks to complete by the end of class:
  + Create a code that will print the text ‘This is my first Python code’
  + Create a code that will print the text ‘A student had grades of 85, 94, 78 and 88, what is her test average?’
  + Create a code that will calculate the student’s average using math functions

***Lesson 2: Changing Python Code***

**Time:** 45-50 minutes

**Materials:** Computers for each student with internet access

**Prior Knowledge:** Basic Python

|  |
| --- |
| **Objective:** *Students will take existing Python Code and change it to create new functions.* |

**Changing Code**

* Students will be sent Python code for a basic calculator that they will paste into repl.it or similar web Python environment.
* Explain the function of the code:

running = True

while running:

print("1 = Addition")

print("2 = Subtraction")

print("3 = Multiplication")

print("4 = Division")

print("5 = Exit program")

This part of the code prompts the user to select a function by typing in the corresponding number.

cmd = int(input("Enter number : "))

if cmd == 1:

print("Addition")

first = int(input("Enter first number :"))

second = int(input("Enter second number :"))

result = first + second

print(result)

Remind students of the math operators (+ - \* /). This creates a variable named ‘cmd’ that takes the user input. If the user entered ‘1’ for addition, it prints ‘Addition’ to confirm the user’s selection. Then it prompts the user for the first and second number the user wishes to add together. The variables ‘first’ and ‘second’ are created from the user input from the prompts. It then adds the two numbers together and prints the result.

The remaining code performs the functions for subtraction, multiplication and division.

elif cmd == 2:

print("Subtraction")

first = int(input("Enter first number :"))

second = int(input("Enter second number :"))

result = first - second

print(result)

elif cmd == 3:

print("Multiplication")

first = int(input("Enter first number :"))

second = int(input("Enter second number :"))

result = first \* second

print( result)

elif cmd == 4:

print("Division")

first = int(input("Enter first number :"))

second = int(input("Enter second number :"))

result = first / second

print( result)

The final part of the code ends the program if the user elects to quit.

elif cmd == 5:

print("Quit!")

running = False

|  |
| --- |
| **Assessment:** *Students will take existing Python Code and change it to create new functions.* |

Give students the task of creating a Distance, Speed and Time Calculator from the existing calculator code.

Students will be given the formulas:

*d =*

*s=*

*t=*

Student final code should look like the following:

running = True

while running:

print("1 = Solve for Distance")

print("2 = Solve for Speed")

print("3 = Solve for Time")

print("4 = Exit")

cmd = int(input("Enter number : "))

if cmd == 1:

print("Solve for Distance")

first = int(input("Enter Speed :"))

second = int(input("Enter Time:"))

result = first \* second

print(result)

elif cmd == 2:

print("Solve for Speed")

first = int(input("Enter Distance:"))

second = int(input("Enter Time:"))

result = first / second

print(result)

elif cmd == 3:

print("Solve for Time")

first = int(input("Enter Distance :"))

second = int(input("Enter Speed :"))

result = first / second

print(result)

elif cmd == 4:

print("Quit!")

running = False

***Lesson 3: Understanding Robot Code***

**Time:** 45-50 minutes

**Materials:** Computers for each student with internet access, iRobot Create Kit including Robot, Cable, Raspberry Pi and USB Drive

**Prior Knowledge:** Basic Python

|  |
| --- |
| **Objective:** *Students will learn about Python code specifically made for the Create and see a demonstrator of the robot.* |

**Understanding the Code**

There is a lot of code in the MyPythonRobotController.py file. Much of it the students don’t need to concern themselves with. A general overview showing that the code begins by importing functions, then defines a series of variables for use later is all that is necessary.

Point students to the part of the code that states:

#-------------Start My robot program------------

Now we’re going to get into the two functions that are most important to the lesson: DriveDirect and Sleep.

The DriveDirect function controls the two wheels of the iRobot Create. Therefore, there will be two numbers in parenthesis separated by a comma. The first number controls the left wheel, the second controls the right. The speed of the wheels are measured in cm/sec and can range from 0-50 cm/s.

The code will look like this:

DriveDirect(25,25)

*To check for understanding, ask students what will happen when this code is run. Will the robot go left, right, or straight?*

The robot will go straight at 25 cm/sec.

The problem is, we haven’t told it how long to go straight. That’s where the time.sleep function comes in. Though you might think the function stops the robot or puts it to sleep, it’s actually how long the previous command will run until moving on to the next command.

So, let’s look at these commands.

DriveDirect(25,25)

time.sleep(10)

*Ask students what will happen when this code is run, and how far it will go.*

The robot will drive at 25 cm/s for 10 seconds.

Since we know distance = speed x time, it will travel 250 cm or 2.5 m.

Now the robot needs to stop.

*Ask students how they would stop the robot using the functions that they know.*

DriveDirect(0,0)

By telling the wheels to travel 0 cm/s , the robot will stop. If that’s all you want the robot to do, then all that’s left is to close communication by running this code:

closeRobotCommunication()

**Let’s run these lines of code together and demonstrate the robot.**

It is recommended that you make a copy of the MyPythonRobotController.py file to your computer before editing in case you ever encounter issues with the edited file on the supplied USB Drive.

The end of your MyPythonRobotController.py code on the USB drive should look like this:

#-------------Start My robot program------------

DriveDirect(25,25)

time.sleep(10)

DriveDirect (0,0)

closeRobotCommunication()

This code can be edited in Python or in a simple plain text editor such as Notepad.

**Do not connect the cable to the robot until instructed.**

Place the supplied modified cable in the top-right USB port in the Raspberry Pi as indicated in green in the diagram.



Connect the red and black wires with the connector to the three pins in the top-right indicated in green. Make sure the empty slot is on the top, red in the second spot down and black is in the third spot down.



Now place the USB drive with your code in any of the available USB ports on the Raspberry Pi.

Take the robot off it’s charging dock and place it on the floor, ensuring that there is an area clear from obstacles 5 feet wide and 8 feet long for the robot to drive.

Now place the Raspberry Pi on top of the robot. You can secure it with masking tape or hook and loop material if you wish. Now you can plug the modified cable into the port on the robot.

It will take approximately 60 seconds for the robot to start moving.

Run this basic program 5 times from the same starting point. Have students chart the robot’s path on a piece of paper. Note that the robot’s path will vary slightly each time. Robotics can be inexact and it’s an important lesson for the students to learn. As they run through their own code, they should make multiple runs and chart the results.

Now, introduce the concept of turns. Since the code includes a speed for each wheel, the robot will rotate with a differential in wheel speeds.

Modify your Python program to include a turn and another straight run. Do not show this code to students as they will need to determine how best to turn the robot on their own in the next lesson:

#-------------Start My robot program------------

DriveDirect(25,25)

time.sleep(10)

DriveDirect(25,15)

time.sleep(10)

DriveDirect(25,25)

time.sleep(10)

DriveDirect (0,0)

closeRobotCommunication()

Through testing, a speed of (25,15) for 3 seconds makes an approximate 90 degree turn. Since the left wheel is accelerating at a faster speed, the robot will turn left.

Now that students are familiar with the code basics, they are ready to test it for themselves.

***Lesson 4: Writing Robot Code***

**Time:** Three45-50 minute classes

**Materials:** Computers for each student with internet access, iRobot Create Kit including Robot, Cable, Raspberry Pi, USB Drive, painter’s tape, meter sticks or tape measure.

**Prior Knowledge:** Basic Python

|  |
| --- |
| **Objective:** *Students will learn about Python code specifically made for the Create and see a demonstrator of the robot.* |

**Writing Robot Code**

Prior to class, set up a course for the students using painter’s tape on the floor. Leave plenty of room for drift as it is inevitable during testing. Use a few turns to make sure students cannot simply have the robot go straight, but it shouldn’t be overly complicated.

This is an example of the type of course you could design:



Pair students up for this project so they can share ideas. Have each group measure the course for their calculations.

Return students to a computer with a USB containing the text file of the code to edit. Go around the room and advise students if they are struggling with the concepts or if their code looks significantly incorrect. It is ok for them to be unsuccessful. Continue to have students go through trials until they fine tune their code.

Once students can successfully navigate their course, begin timing their trials. Keep a leaderboard displayed in your room for motivation. If there are still students struggling to navigate the course at this stage, help them fix their code so they can be successful.

The challenges do not need to end here. Once the foundations are in place, there are endless possibilities and challenges you can design for your students. Adding a second iRobot Create allows for direct competition and simultaneous testing to keep students engaged. Have fun!