Jerry Chen
A13310365
January 23, 2018

ECE 16 Lab 1: Python and Arduino Communication

1. Introduction

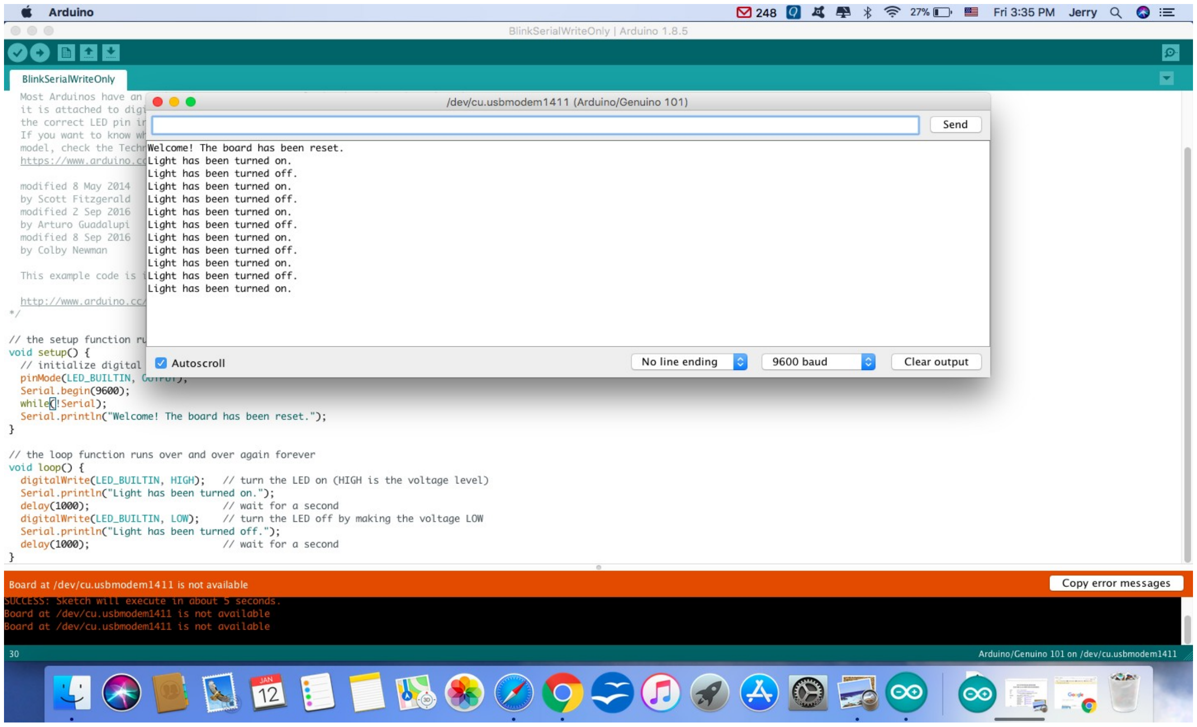
This lab introduced us to using the Python language, Arduino, and the Pyserial class that enables them to communicate. The first three objectives showed us the basics of how to use Arduino. Specifically, in order, how to run a sketch in Arduino and the basic structure of the code, how to manipulate the light source on the board and write to the output, and how to take input from the monitor and display output to the monitor. Objectives 4 and 5 showed how to use python code in terminal and how to structure it's code and perform complex tasks with it, such as rotating a matrix or perform something as simple as a greeting to a few specific people. Lastly, Objectives 6 and 7 showed how to facilitate communication between the Arduino board and a Python program with Input and Output.

2. Objective 1: Arduino Blink

The first objective is rather simple; it merely involves testing an already completed piece of code. First, we connect the Arduino board to the computer as we always do and make sure the indicator light is on. Next we open up Arduino and click on the tools tab and select the port connected to the board. We open the “File” tab in the top left corner, view the “Examples” folder and the nested “01: Basics” folder, and select “Blink” to open up the code to get the light on the board to Blink. We then select the verify button and upload it to the board and observe the LED light on the board blink (turn on and off periodcally).

3. Objective 2: Serial Write

With the Blink code we opened in objective 1, we modify both sections of the code (setup() and loop()) to write information from the board to the computer, specifically when the board resets and when the light is on or off. In the setup() function, we write Serial.begin(9600) at the end of the original code to setup output and bit transfer rate to serial, give it time to startup by writing a While(!Serial); loop, and write Serial.print() with a welcome message string argument to output a welcome message each time the board is reset. In the loop() function, we add Serial.println() with a string argument immediately after each digitalWrite() function to indicate the status of the light to the computer each time it changes. Once we save the modified code as BlinkSerialWriteOnly and verify and upload the code to the board and open the Serial monitor, we should see it print out a welcome message each time we open the Serial monitor and print out the status of the light on the board each time it changes.


Arduino
BlinkSerialWriteOnly | Arduino 1.8.5
BlinkSerialWriteOnly
/dev/cu.usbmodem1411 (Arduino/Genuino 101)
Send
Autoscroll
No line ending
9600 baud
Clear output
Copy error messages
Arduino/Genuino 101 on /dev/cu.usbmodem1411

```
Welcome! The board has been reset.
Light has been turned on.
Light has been turned off.
Light has been turned on.
Light has been turned off.
Light has been turned on.
Light has been turned off.
Light has been turned on.
Light has been turned off.
Light has been turned on.
Light has been turned off.
Light has been turned on.
```

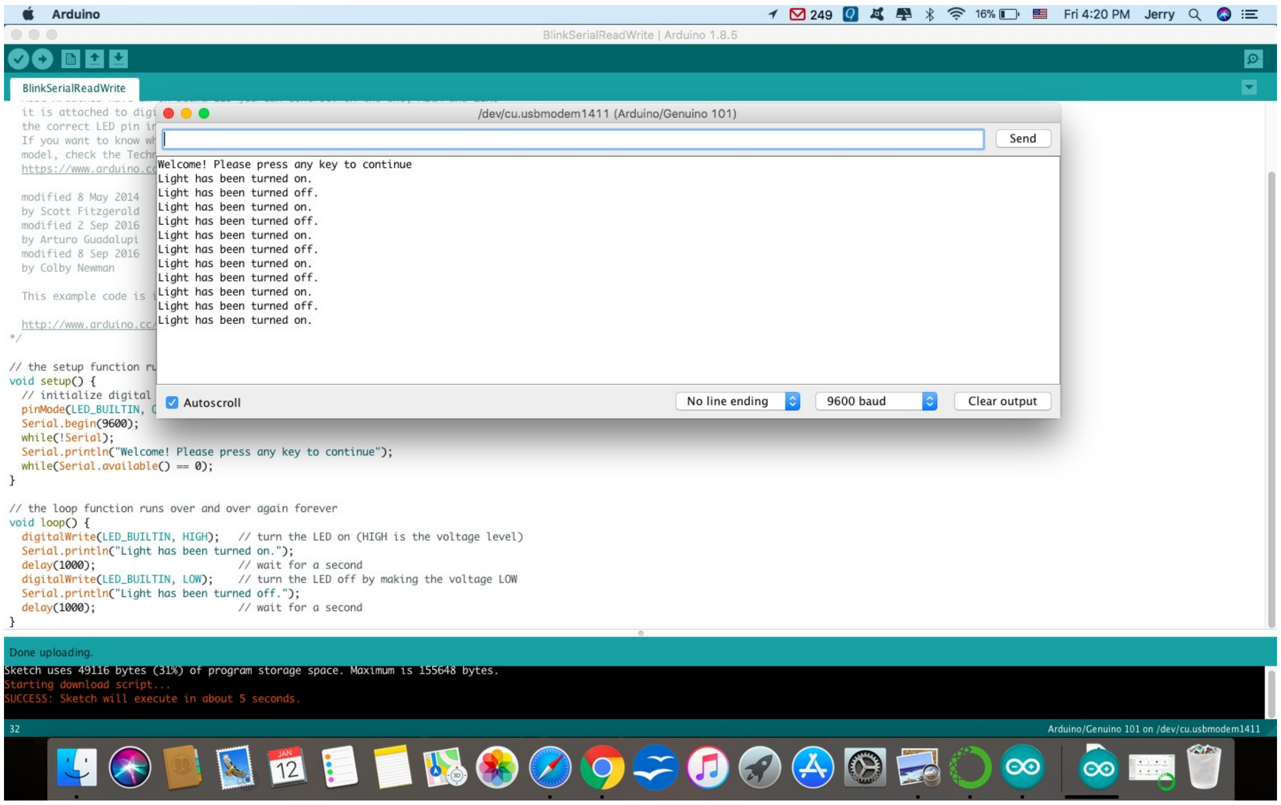
```
// the setup function ru
void setup() {
  // initialize digital
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
  while(!Serial);
  Serial.println("Welcome! The board has been reset.");
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  Serial.println("Light has been turned on.");
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  Serial.println("Light has been turned off.");
  delay(1000);                       // wait for a second
}
```

```
Board at /dev/cu.usbmodem1411 is not available
SUCCESS: Sketch will execute in about 5 seconds.
Board at /dev/cu.usbmodem1411 is not available
Board at /dev/cu.usbmodem1411 is not available
```

Figure 1: The code for BlinkSerialWrite is shown in the background while the Serial output is shown in the serial monitor in the foreground.

4. Objective 3: Serial Read

Taking the BlinkSerialWriteOnly code we already wrote, we modify it to take an input from the computer to start the blinking. We start by altering the string in the Serial.println() function in the setup() function from stating the board's status as reset to request for input from the computer to start blinking. To ensure that the loop() function does not start running and cause the light on the board to blink until the computer has provided input, we either add a line of code at the end of the setup function that checks for input continuously until it is provided by the computer. In this case, we provide a while loop that checks for available input from Serial until it is provided (while(Serial.available() == 0);)



```
/dev/cu.usbmodem1411 (Arduino/Genuino 101)

Welcome! Please press any key to continue
Light has been turned on.
Light has been turned off.
Light has been turned on.
Light has been turned off.
Light has been turned on.
Light has been turned off.
Light has been turned on.
Light has been turned off.
Light has been turned on.
Light has been turned off.
Light has been turned on.
```

```
  it is attached to digi
  the correct LED pin in
  If you want to know wh
  model, check the Techn
  https://www.arduino.cc

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is i

  http://www.arduino.cc/
*/

// the setup function ru
void setup() {
  // initialize digital
  pinMode(LED_BUILTIN, O
  Serial.begin(9600);
  while(!Serial);
  Serial.println("Welcome! Please press any key to continue");
  while(Serial.available() == 0);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  Serial.println("Light has been turned on.");
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  Serial.println("Light has been turned off.");
  delay(1000);                       // wait for a second
}
```

```
Done uploading.
Sketch uses 49116 bytes (31%) of program storage space. Maximum is 155648 bytes.
Starting download script...
SUCCESS: Sketch will execute in about 5 seconds.
```

Arduino
BlinkSerialReadWrite | Arduino 1.8.5
BlinkSerialReadWrite
Send
Autoscroll
No line ending
9600 baud
Clear output
Arduino/Genuino 101 on /dev/cu.usbmodem1411

Figure 2: The code for BlinkSerialReadWrite is shown in the background while output is shown in the serial monitor in the foreground. Note that aside from the modified string in the Serial.println() in the setup() function and the second while loop awaiting input, the code is nearly identical to BlinkSerialWriteOnly.

5. **Objective 4: Python "Hello World!"**

   Now to write a simple Python code in terminal and with a Python IDE (Integrated Development Environment) that provides a basic "Hello World!" greeting for many initial programs. In terminal, we simply enter the command "python" and type "print("Hello <names>!")", where "<names>" is a list of names of people before hitting the return key and it should print out the greeting. In the Python IDE, we write and save a code "hello_friends" with the same line of code and run it, reproducing the effect.

6. **Objective 5: Python matrix rotation**

   This objective is a complicated one, as it involves using logic outside of Python documentation to find a solution. Using a sample array of m rows and n columns, we can be certain that when we rotate a matrix 90 degrees, the new matrix will have n columns and m rows. So with the original sample and altered sample matrix, we can determine that the entries will resemble these:

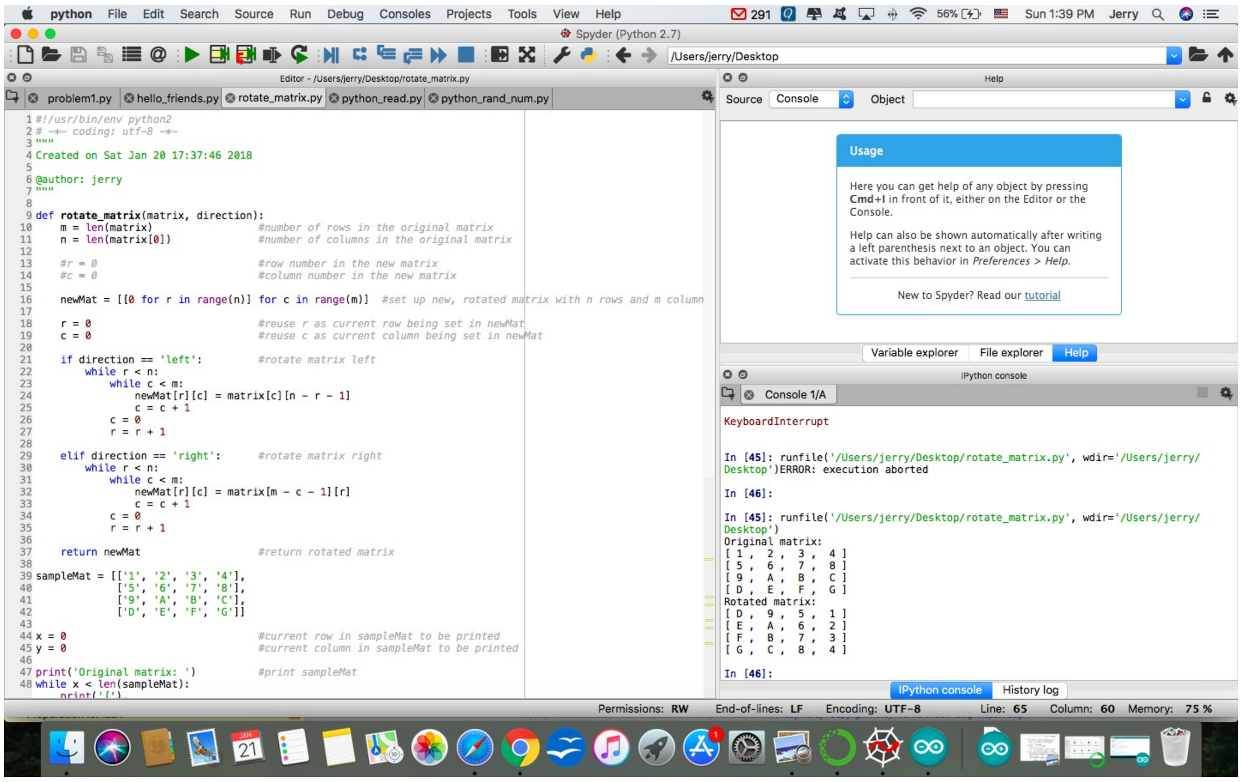
Original:

entry(0, 0)	entry(0, 1)	….	entry(0, n - 2)	entry(0, n - 1)
entry(1, 0)	entry(1, 1)	….	entry(1, n - 2)	entry(1, n - 1)
….	….	….	….	….
entry(m - 2, 0)	entry(m - 2, 1)	….	entry(m - 2, n - 2)	entry(m - 2, n - 1)
entry(m - 1, 0)	entry(m - 1, 1)	….	entry(m - 1, n - 1)	entry(m - 1, n - 1)

Rotated Right:

entry(m - 1, 0)	entry(m - 2, 0)	….	entry(0, n - 2)	entry(0, n - 1)
entry(m - 1, 1)	entry(m - 2, 1)	….	entry(1, n - 2)	entry(1, n - 1)
….	….	….	….	….
entry(m - 2, 0)	entry(m - 2, 1)	….	entry(m - 2, n - 2)	entry(m - 2, n - 1)
entry(m - 1, 0)	entry(m - 1, 1)	….	entry(m - 1, n - 1)	entry(m - 1, n - 1)

From the above we can notice that there is a pattern in which whenever the matrix (origMat) is rotated right each entry in the new matrix newMat[r][c] = origMat[m-1-c][r]. Using the same method, we can also determine that when rotating left, newMat[r][c] = origMat[c][n-1-r]. Thus, in rotate_matrix.py, in "def rotate_matrix(matrix, direction):" we write m = len(matrix) and n = len(matrix[0]) to gather the number of rows and columns in the matrix respectively, and setup a new matrix array newMat with n rows and m columns and set each entry to 0 at first and then set each entry according to "direction" and the principles discovered above and use two while loops (one column based loop in the new matrix nested in another row based loop) for either direction, making sure the column number resets to 0 at the end of each row loop and finally return the new matrix, ending the function definition. We then use a sample matrix, which we print and then run it in the function and print out the result and verify that it works.



```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sat Jan 20 17:37:46 2018

@author: jerry
"""

def rotate_matrix(matrix, direction):
    m = len(matrix)                 #number of rows in the original matrix
    n = len(matrix[0])              #number of columns in the original matrix

    #r = 0                          #row number in the new matrix
    #c = 0                          #column number in the new matrix

    newMat = [[0 for r in range(n)] for c in range(m)]  #set up new, rotated matrix with n rows and m column

    r = 0                           #reuse r as current row being set in newMat
    c = 0                           #reuse c as current column being set in newMat

    if direction == 'left':         #rotate matrix left
        while r < n:
            while c < m:
                newMat[r][c] = matrix[c][n - r - 1]
                c = c + 1
            c = 0
            r = r + 1

    elif direction == 'right':      #rotate matrix right
        while r < n:
            while c < m:
                newMat[r][c] = matrix[m - c - 1][r]
                c = c + 1
            c = 0
            r = r + 1

    return newMat                   #return rotated matrix

sampleMat = [['1', '2', '3', '4'],
             ['5', '6', '7', '8'],
             ['9', 'A', 'B', 'C'],
             ['D', 'E', 'F', 'G']]

x = 0                               #current row in sampleMat to be printed
y = 0                               #current column in sampleMat to be printed

print('Original matrix: ')          #print sampleMat
while x < len(sampleMat):
    print('[')
```

Usage

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

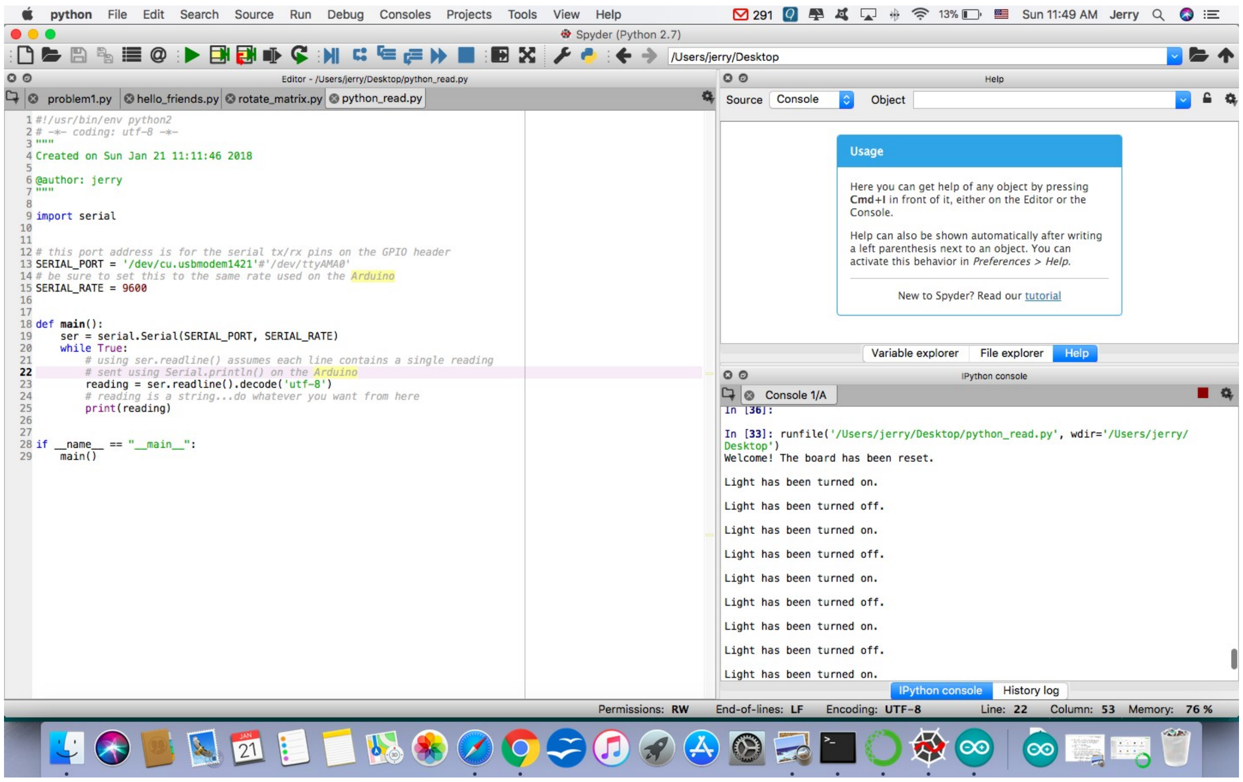
```
KeyboardInterrupt


In [45]: runfile('/Users/jerry/Desktop/rotate_matrix.py', wdir='/Users/jerry/
Desktop')ERROR: execution aborted

In [46]:

In [45]: runfile('/Users/jerry/Desktop/rotate_matrix.py', wdir='/Users/jerry/
Desktop')
Original matrix:
[ 1 ,  2 ,  3 ,  4 ]
[ 5 ,  6 ,  7 ,  8 ]
[ 9 ,  A ,  B ,  C ]
[ D ,  E ,  F ,  G ]
Rotated matrix:
[ D ,  9 ,  5 ,  1 ]
[ E ,  A ,  6 ,  2 ]
[ F ,  B ,  7 ,  3 ]
[ G ,  C ,  8 ,  4 ]

In [46]:
```

python File Edit Search Source Run Debug Consoles Projects Tools View Help
Spyder (Python 2.7)
/Users/jerry/Desktop
Editor - /Users/jerry/Desktop/rotate_matrix.py
problem1.py
hello_friends.py
rotate_matrix.py
python_read.py
python_rand_num.py
Help
Source Console Object
Variable explorer File explorer Help
IPython console
Console 1/A
IPython console History log
Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 65 Column: 60 Memory: 75 %

Figure 3: The definition of rotate_matrix in rotate_matrix.py, and the results of the test are shown.

7. Objective 6: Serial Read from Arduino to Python

Now to facilitate communication from Arduino to Python, using the BlinkSerialWriteOnly Arduino code as an example. First, we connect the board to the computer and connect to the proper port. We open up BlinkSerialWriteOnly and upload it to the board, but we refrain from opening serial monitor. In the Python IDE (in my case it is spyder) we import serial to enable the connection and locate the port where the board is connected and save the address as a string SERIAL_PORT. In my case it is '/dev/cu.usbmodem1411'. We also match the SERIAL_RATE to that of the Arduino (9600). In the main function, we open up the connection to the by applying the ser = serial.Serial(SERIAL_PORT, SERIAL_RATE) to set ser as the serial connection. Using a while loop we read lines from serial and print them to enable the same effect BlinkSerialWriteOnly had in the Serial Monitor, only in Python.



```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 21 11:11:46 2018

@author: jerry
"""

import serial


# this port address is for the serial tx/rx pins on the GPIO header
SERIAL_PORT = '/dev/cu.usbmodem1421'#'/dev/ttyAMA0'
# be sure to set this to the same rate used on the Arduino
SERIAL_RATE = 9600


def main():
    ser = serial.Serial(SERIAL_PORT, SERIAL_RATE)
    while True:
        # using ser.readline() assumes each line contains a single reading
        # sent using Serial.println() on the Arduino
        reading = ser.readline().decode('utf-8')
        # reading is a string...do whatever you want from here
        print(reading)


if __name__ == "__main__":
    main()
```

Here you can get help of any object by pressing Cmd+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help.*

New to Spyder? Read our tutorial

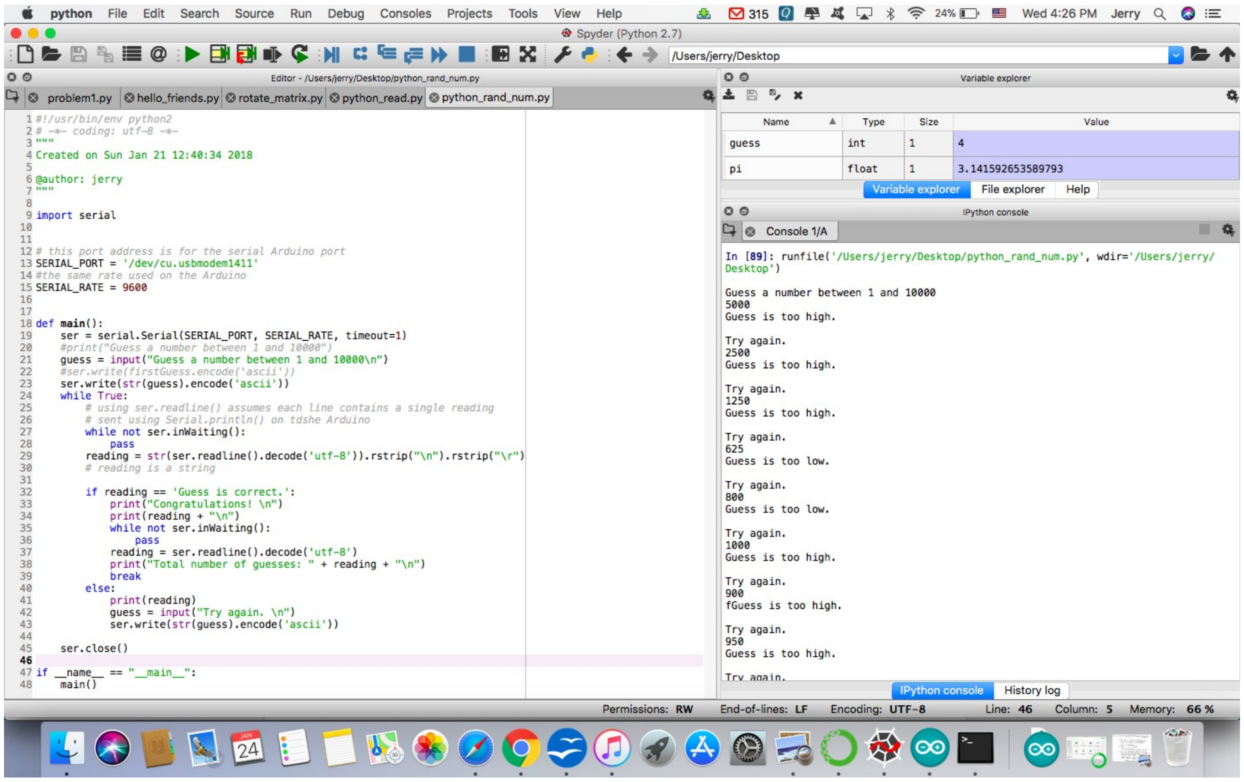
```
In [36]:

In [33]: runfile('/Users/jerry/Desktop/python_read.py', wdir='/Users/jerry/
Desktop')
Welcome! The board has been reset.

Light has been turned on.

Light has been turned off.

Light has been turned on.

Light has been turned off.

Light has been turned on.

Light has been turned off.

Light has been turned on.

Light has been turned off.

Light has been turned on.
```

Permissions: RW   End-of-lines: LF   Encoding: UTF-8   Line: 22   Column: 53   Memory: 76 %

Figure 4: Running python_read.py after uploading BlinkSerialWriteOnly to the Arduino board and printing each time enables a connection through pyserial that provides a similar response.

8. Objective 7

Now to create full communication between Python and Arduino with a Serial program in which Python requests an inputted number and gives it to the Arduino to process it. This Arduino code has global variables such that they can be setup in the setup function and modified in the loop function. The random number is setup in the setup function, while the other global numerical variables start out as 0. In loop, the program waits for an input in while(!Serial.available()); and progresses to read the input from serial and convert it into a number, incrementing the number of guesses made in the process. It then compares the guess to the random number and if it is low, blinks the board twice. If it is high it blinks once. And if it is correct, it blinks continuously and prints out such to serial. In Python, we import serial, locate the port address and match the serial and open up the serial port like in the previous objective, only this time, to prevent stalling from the input, we include a timeout. The Python then requests an input from the computer and user, takes it and then converts it to ASCII to pass it to the Arduino, which processes the answer. Each time before the Python code reads from Arduino, it waits for the serial to finish whatever it is doing before reading the line. Each time a number is inputted, it waits before reading and determines from reading, whether or not the last inputted number is correct. If not, it then prints out whether it was too high or too low from Arduino and prompts another input. Once the correct guess is provided, the user is congratulated and informed of the number of guesses made before the Python loop breaks and the serial closes, and the program ends.



Editor - /Users/jerry/Desktop/python_rand_num.py

```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 21 12:40:34 2018

@author: jerry
"""

import serial


# this port address is for the serial Arduino port
SERIAL_PORT = '/dev/cu.usbmodem1411'
#the same rate used on the Arduino
SERIAL_RATE = 9600


def main():
    ser = serial.Serial(SERIAL_PORT, SERIAL_RATE, timeout=1)
    #print("Guess a number between 1 and 10000")
    guess = input("Guess a number between 1 and 10000\n")
    #ser.write(firstGuess.encode('ascii'))
    ser.write(str(guess).encode('ascii'))
    while True:
        # using ser.readline() assumes each line contains a single reading
        # sent using Serial.println() on tdshe Arduino
        while not ser.inWaiting():
            pass
        reading = str(ser.readline().decode('utf-8')).rstrip("\n").rstrip("\r")
        # reading is a string

        if reading == 'Guess is correct.':
            print("Congratulations! \n")
            print(reading + "\n")
            while not ser.inWaiting():
                pass
            reading = ser.readline().decode('utf-8')
            print("Total number of guesses: " + reading + "\n")
            break
        else:
            print(reading)
            guess = input("Try again. \n")
            ser.write(str(guess).encode('ascii'))

    ser.close()

if __name__ == "__main__":
    main()
```

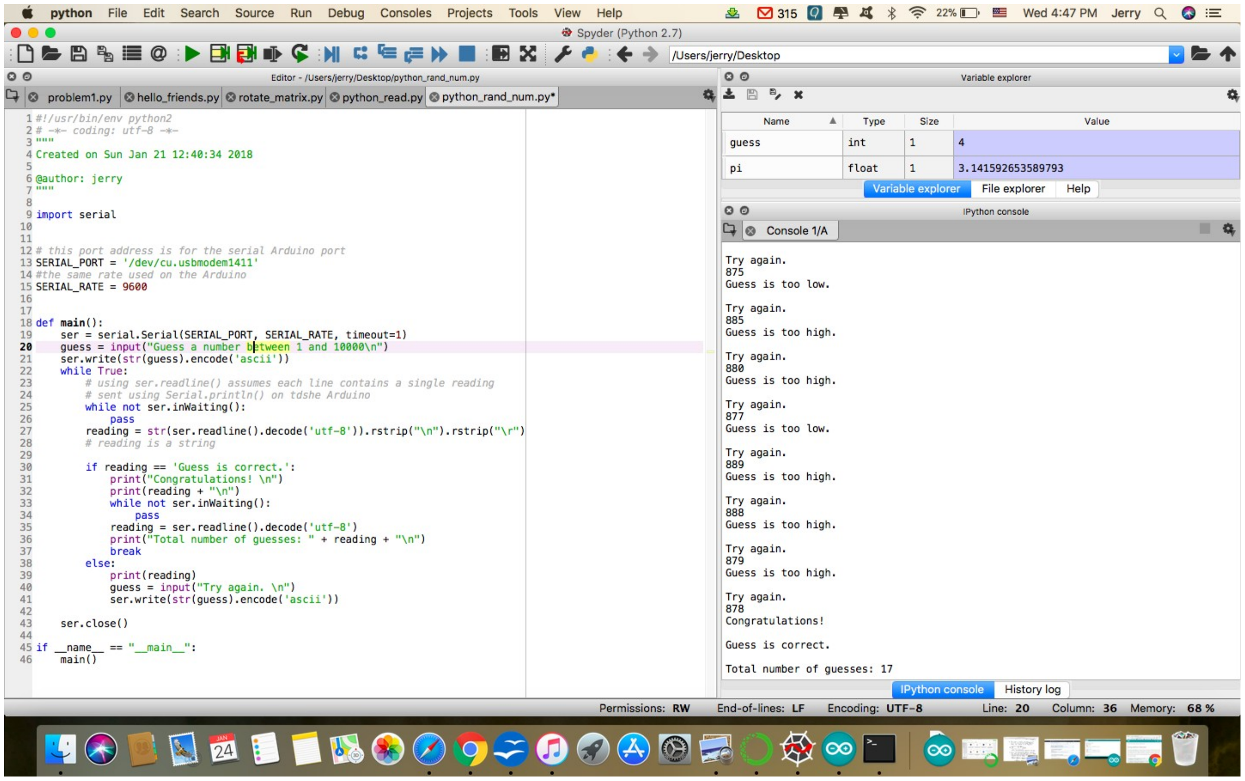
Variable explorer

Name	Type	Size	Value
guess	int	1	4
pi	float	1	3.141592653589793

IPython console

```
In [89]: runfile('/Users/jerry/Desktop/python_rand_num.py', wdir='/Users/jerry/
Desktop')

Guess a number between 1 and 10000
5000
Guess is too high.

Try again.
2500
Guess is too high.

Try again.
1250
Guess is too high.

Try again.
625
Guess is too low.

Try again.
800
Guess is too low.

Try again.
1000
Guess is too high.

Try again.
900
fGuess is too high.

Try again.
950
Guess is too high.

Try again.
```

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 46 Column: 5 Memory: 66 %



```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 21 12:40:34 2018

@author: jerry
"""

import serial


# this port address is for the serial Arduino port
SERIAL_PORT = '/dev/cu.usbmodem1411'
#the same rate used on the Arduino
SERIAL_RATE = 9600


def main():
    ser = serial.Serial(SERIAL_PORT, SERIAL_RATE, timeout=1)
    guess = input("Guess a number between 1 and 10000\n")
    ser.write(str(guess).encode('ascii'))
    while True:
        # using ser.readline() assumes each line contains a single reading
        # sent using Serial.println() on tdshe Arduino
        while not ser.inWaiting():
            pass
        reading = str(ser.readline().decode('utf-8')).rstrip("\n").rstrip("\r")
        # reading is a string

        if reading == 'Guess is correct.':
            print("Congratulations! \n")
            print(reading + "\n")
            while not ser.inWaiting():
                pass
            reading = ser.readline().decode('utf-8')
            print("Total number of guesses: " + reading + "\n")
            break
        else:
            print(reading)
            guess = input("Try again. \n")
            ser.write(str(guess).encode('ascii'))

    ser.close()

if __name__ == "__main__":
    main()
```

Name	Type	Size	Value
guess	int	1	4
pi	float	1	3.141592653589793

```
Try again.
875
Guess is too low.

Try again.
885
Guess is too high.

Try again.
880
Guess is too high.

Try again.
877
Guess is too low.

Try again.
889
Guess is too high.

Try again.
888
Guess is too high.

Try again.
879
Guess is too high.

Try again.
878
Congratulations!

Guess is correct.

Total number of guesses: 17
```

Permissions: RW   End-of-lines: LF   Encoding: UTF-8   Line: 20   Column: 36   Memory: 68 %

Figure 5&6: Shown is the Python code and the beginning and end of the guesses.

5. Problems and Issues

Describe any problems or issues you may have had for this lab. If you did not have any problems or issues, you may skip this section.

6. Conclusion

This lab ultimately enabled us to use Arduino and Python, and taught us how to provide input and output to the two and between each other and how to process information between them.