

CSCI 1411: Fundamentals of Computing
Lab 12
Due Date: 8:30 AM November 10, 2020

Name: Kerry Gip

Goals:

- Learn about debugging your Python code using IDLE debugger

Development Environment: IDLE

Deliverables:

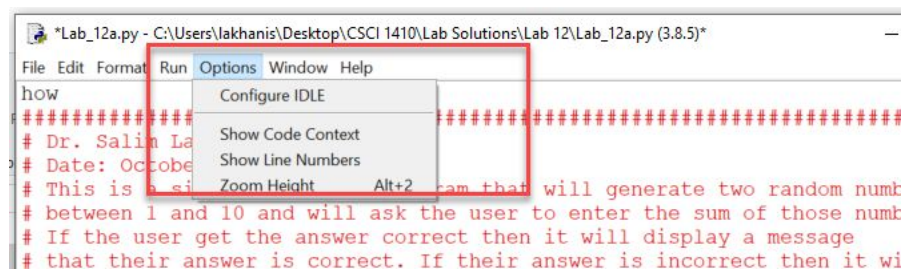
1. This lab handout with 9 screen shots (3 for part I and 6 for part II).
2. Your Python code for Part II of this lab. Name the file using the following format: YourlastnameFirstnameLab12b.py and YourlastnameFirstnameLab12c.py
Example: If your name is Jamal Jones then you will name the file as follows: JonesJamalLab12b.py and JonesJamalLab12c.py

How to take a **screen shot**:

- For a Windows 10: Use Snipping Tool to copy and CTRL + V to paste screen shot.
- For Mac: Shift + Command + 4 to copy and CTRL + V to paste screen shot.

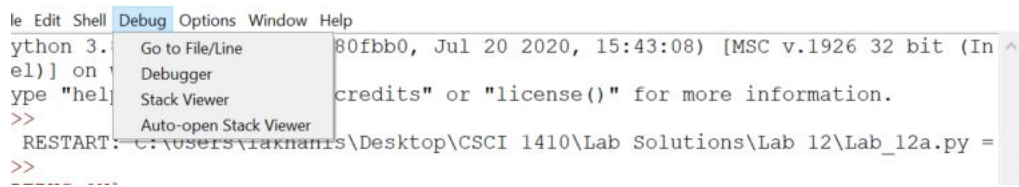
Part I – Skill Practice (10 Points)

- **Note: Take two screen shots at two different points as you follow the steps given below to run your debugger.**
- In this portion of the lab you will follow along with your instructor. Please do not get too far ahead of your instructor as they explain the use of debugger.
- Start IDLE
- Open the file Lab_12a.py and save it as YourlastnameFirstnameLab12a.py
- Select “Show Line Numbers” in the Options menu of your Editor window to display the line numbers. This will help you with following this tutorial.

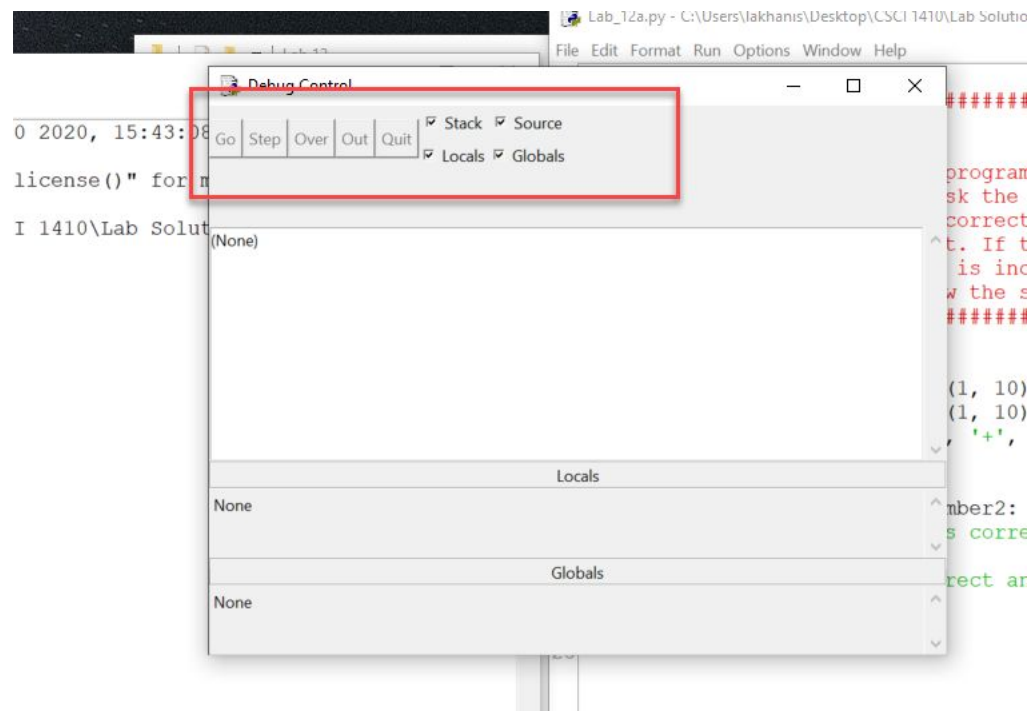


- This is a simple math quiz program that will generate two random numbers between 1 and 10, it will display those numbers and ask the user to enter the sum of those numbers. If the user answers the question correctly then it will display the message that the answer is correct. If the user answers the question incorrectly then it will display the message Nope with the correct answer.

- Run the program. This program has a bug and it will display Nope every time you run this program. It will display Nope even if you get the answer correct. We will use a debugger to find the error.
- Start the debugger by selecting Debugger from the Debug menu:

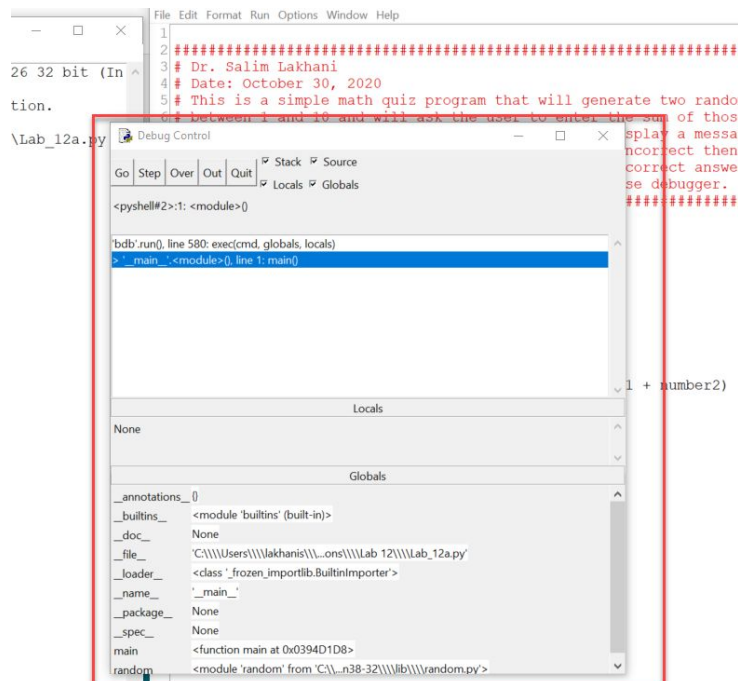


- This will open the Debug Control window.

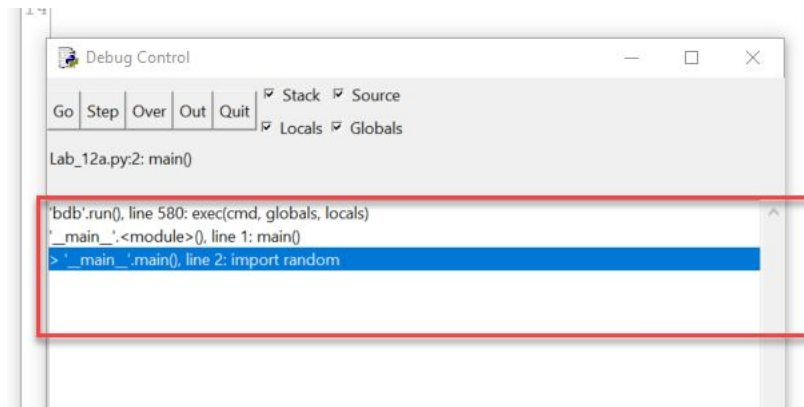


- Make sure that you select all the following check boxes: Stack, Locals, Source, and Globals.

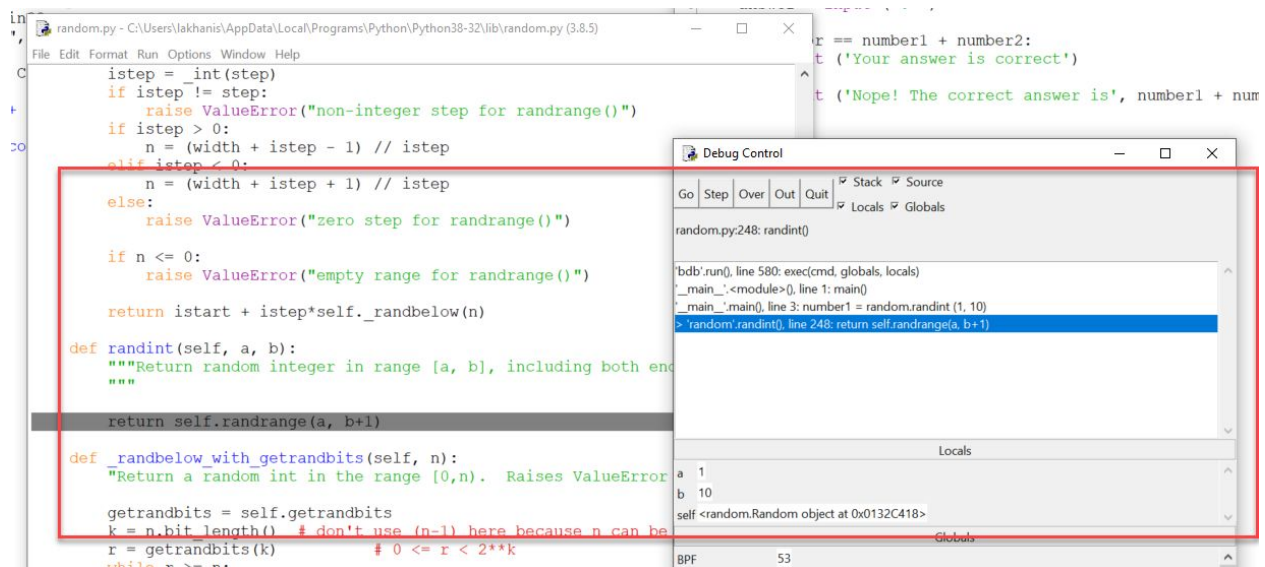
- The debugger lets you execute one instruction at a time. This is called stepping. There are two options: you can either click on Step or Over to execute the current Python instruction. If your statement is going to call a function then Step will step into the function and Over will execute the function but will not take you into the function. You can also click on the Out button to finish execution of current function and jump back to the calling function. We will look at some examples later in this lab. The Go button will execute the code from the current location either to the end of the program or a break point.
- You can insert a break point by right clicking on a line of code and selecting “Set Breakpoint” from short cut menu. You can clear the break point by right clicking on a line of code and selecting “Clear Breakpoint” from the short cut menu. IDLE will highlight the line with breakpoint. If you think that there is an error in certain part of the program (like loop) then you can set a breakpoint at start of that loop, click on Go button to execute the program. Your debugger will stop at the line of code with breakpoint.
- You can use the Quit button to quit the debugger.
- Globals area in the Debug control window will display all the global variables. We will not use it for this lab.
- Locals area in the Debug control window will display all the local variables and their values (as the variables are created during the execution of your program).
- Type `main()` in the shell to start the execution of your program. Your debugger Control will show that it is stopped at the first line (line 1) of your code (`def main()`)



- Now click on the Step button to complete the execution of the first line and move to the second line of your code. The Debug Control window will display line 2 of your code.

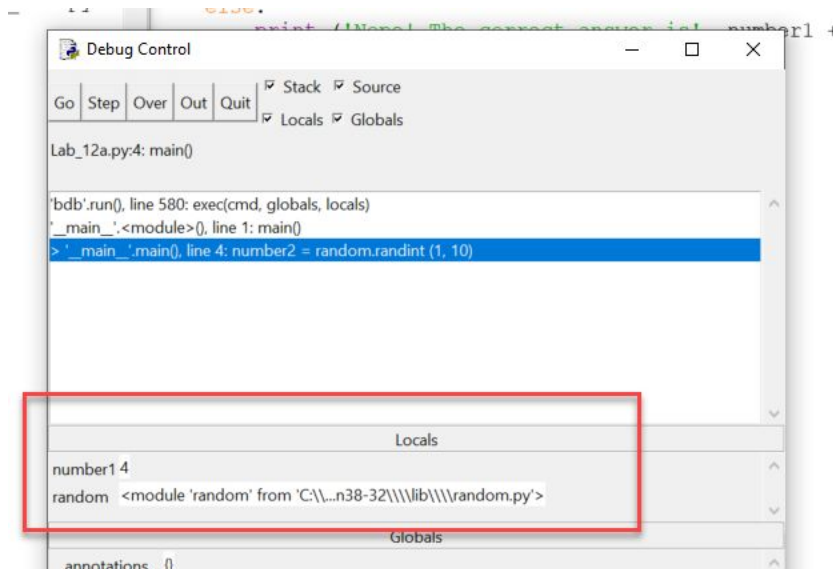


- Line 2 of your code will import the random module. Click on Step Button to complete the execution of line 2 and move to the next statement (line 3) in your Python code. Now you can either click on Step button which will take you into randint function or you can click on the Over button to complete the execution of randint function line of your code. Go ahead and click on the Step button. This will open a new code window displaying the random.py file with line highlighted in the randint function. The Locals area will display the value of variables a and b (which are local variables for randint function.)

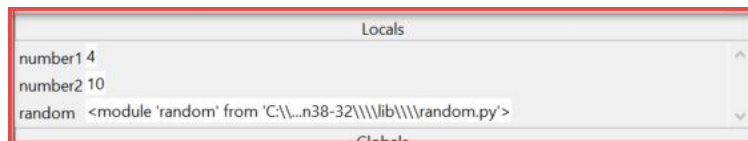


- Now click on the Out button to get out of the randint function and go back to your Python code. This will finish execution of the line 3 of your Python code. You can close the random.py window if you want. Now the Locals area will display the value of

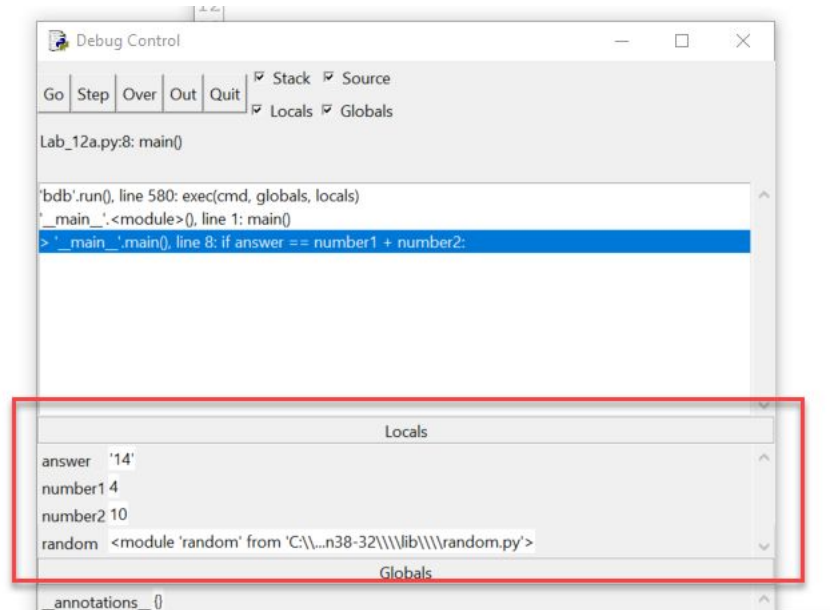
`number1` (variable in your Python code). Note that the Locals area also show that you have imported `Random` module into your Python code.



- Click on **Over** button to execute the next line of code (line 4) in your Python code. This will create another local variable `number2` and its value will be displayed in the Locals area.



- Click on the **Over** button to execute the `print` statement which will display the message in the shell.
- Click on **Over** again. It will execute the `input` statement and wait for you to type your input in the shell. Go ahead and type the correct answer. Your Python code will read your input complete the execution of line 6 of your code, which is `input` statement. Now the Locals area will display three variables `answer`, `number1`, and `number2` (variables are displayed in alphabetical order).



- Click on the Step button. Debugger will execute the `if` statement and will jump to the `else` part (even if you inputted the correct answer). Look at the values of `answer`, `number1`, and `number2` in your Debug Control window. If you think that you know what is wrong with the code then click on the Quit button to quit the debugger, close the Debug Control window, fix the code and run it to make sure it does what it is supposed to do.
- **Once you are satisfied with your results then run your program and take a screen shot of your output.**

Paste all 3 of your screenshots below this line

```

*Lab_12a.py - C:\Users\kerry\Desktop\Lab_12a.py (3.8.5)*
File Edit Format Run Options Window Help
1 def main():
2     import random
3     number1 = random.randint(1, 10)
4     number2 = random.randint(1, 10)
5     print('What is', number1, '+', number2, '?')
6
7     answer = eval(input('What is your answer? '))
8
9     if answer == number1 + number2:
10        print('Your answer is correct')
11    else:
12        print('Nope! The correct answer is', number1 + number2)

```

```
=====  
===== RESTART: C:\Users\kerry\Desktop\Lab_12a.py =====  
>>> main()  
What is 5 + 7 ?  
What is your answer? 12  
Your answer is correct  
>>>
```

```
=====  
===== RESTART: C:\Users\kerry\Desktop\Lab_12a.py =====  
>>>  
[DEBUG ON]  
>>> main()  
What is 8 + 8 ?  
What is your answer? 16
```

Debug Control

Go Step Over Out Quit

☒ Stack ☒ Source
☒ Locals ☒ Globals

Lab_12a.py:10: main()

'bdb'.run(), line 580: exec(cmd, globals, locals)
'__main__'.<module>(), line 1: main()
> '__main__.main(), line 10: if answer == number1 + number2:

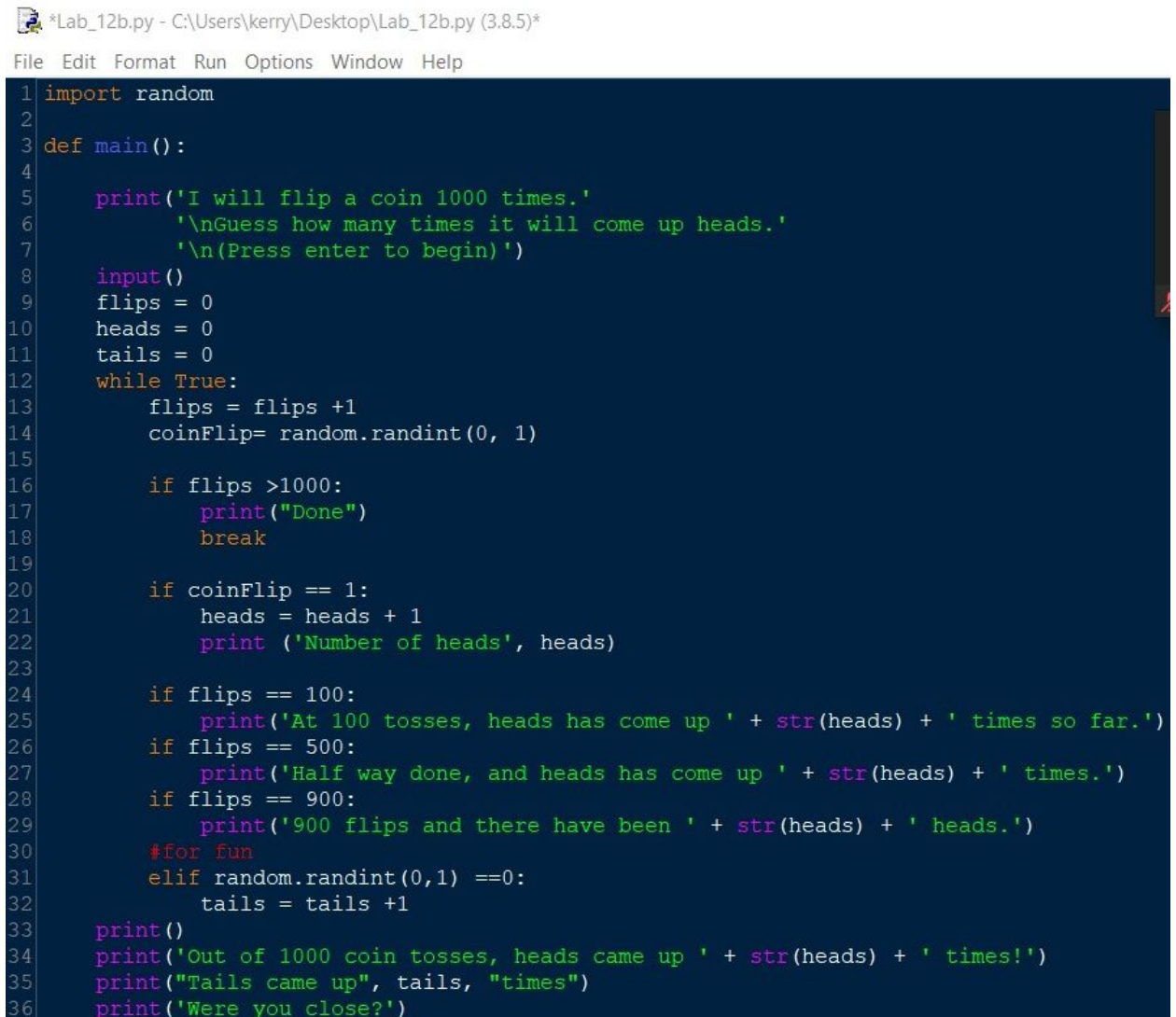
Locals

answer 16
number1 8
number2 8

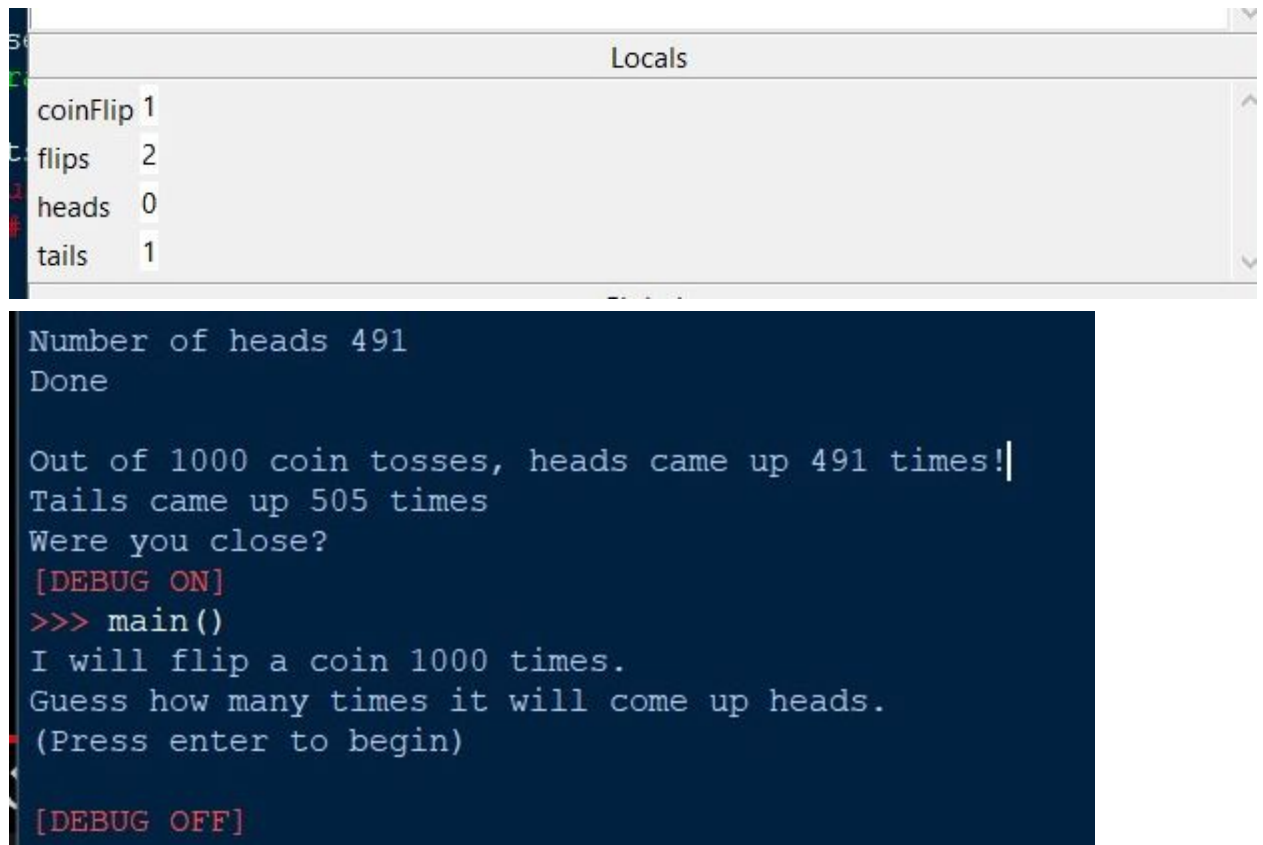
Part II – Debug a Python Program (15 Points)

- **Note: Take two screen shots at two different points as you run the debugger to debug the given Python code.**
- Open the file Lab_12b.py and save it as YourlastnameFirstnameLab12b.py
- Run the program and you will see that it is stuck in an infinite loop. Press CTRL-C in shell to kill the execution of your code. Use the debugger to find the bug, fix the code and run it to make sure it does what it is supposed to do.
- **Once you are satisfied with your results then run your program and take a screen shot of your output.**

Paste all 3 of your screen shot below this line

A screenshot of a Python IDE window titled '*Lab_12b.py - C:\Users\kerry\Desktop\Lab_12b.py (3.8.5)*'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code is as follows:

```
1 import random
2
3 def main():
4
5     print('I will flip a coin 1000 times.'
6           '\nGuess how many times it will come up heads.'
7           '\n(Press enter to begin)')
8     input()
9     flips = 0
10    heads = 0
11    tails = 0
12    while True:
13        flips = flips + 1
14        coinFlip= random.randint(0, 1)
15
16        if flips >1000:
17            print("Done")
18            break
19
20        if coinFlip == 1:
21            heads = heads + 1
22            print ('Number of heads', heads)
23
24        if flips == 100:
25            print('At 100 tosses, heads has come up ' + str(heads) + ' times so far.')
26        if flips == 500:
27            print('Half way done, and heads has come up ' + str(heads) + ' times.')
28        if flips == 900:
29            print('900 flips and there have been ' + str(heads) + ' heads.')
30        #for fun
31        elif random.randint(0,1) ==0:
32            tails = tails +1
33    print()
34    print('Out of 1000 coin tosses, heads came up ' + str(heads) + ' times!')
35    print("Tails came up", tails, "times")
36    print('Were you close?')
```

- **Note: Take two screen shots at two different points as you run the debugger to debug the given Python code.**
- Open the file Lab_12c.py and save it as YourlastnameFirstnameLab12c.py
- Run the program and you will see that it is stuck in an infinite loop. Press CTRL-C in shell to kill the execution of your code. Use the debugger to find the bug, fix the code and run it to make sure it does what it is supposed to do.
- **Once you are satisfied with your results then run your program and take a screen shot of your output.**

Paste all 3 of your screen shot below this line

```
'bdb'.run(), line 580: exec(cmd, globals, locals)
> '.__main__'.<module>(), line 1: main()
'__main__'.main(), line 13: number = float(num)
```

Locals

count	9
infile	<_io.TextIOWrapper name='da... mode='r' encoding='cp1252'>
line	'1, 2, 3, 4, 5, 6, 7, 8, 9,...8, 9\\n5, 6, 7, 8, 9, 10\\n'
my_list	['1', '2', '3', '4', '5', '...', '6', '7', '8', '9', '10\\n']
num	'10\\n2'
number	9.0
sum1	45.0

```
===== RESTART: C:\Users\kerry\Desktop\Lab_12c.py =====
>>> main()
Number of data: 33
Sum of numbers is: 614.0
Average is: 18.61
>>> |
```

```
'bdb'.run(), line 580: exec(cmd, globals, locals)
'__main__'.<module>(), line 1: main()
> '__main__'.main(), line 15: count = count + 1
```

Locals

average	5.5
count	10
infile	<_io.TextIOWrapper name='da... mode='r' encoding='cp1252'>
line	'2, 13, 16, 19, 75\\n'
my_list	['2', '13', '16', '19', '75\\n']
num	'2'
sum1	57.0

Globals

```
8
9     while line != "":
10         my_list = line.split(',')
11
12         for num in my_list:
13
14             sum1 = sum1 + float(num)
15             count = count + 1
16         line = infile.readline()
17         average = sum1 / count
18
19     print ('Number of data:', count)
20     print ('Sum of numbers is:', sum1)
21     print ('Average is: {0:.2f}'.format(average))
22
23     infile.close()
24
```

- Upload this lab handout with required screen shots and your code files to Canvas to submit the lab.