Justin May
RUID: 170003686
02/19/2018

Probability and Random Processes
Assignment 1

I programmed the program in Python 2.7 using Sublime 3 Text as my text editor and i term 2 as my terminal. I imported the matplotlib.pyplot API onto my machine and used the native random library to help with the implementation of this code.

Code:

probability.py

```
1  import matplotlib.pyplot as omega #importing plotting library
2  import random #importing random function
3
4  p = .6388 #last 4 digits of my RUID divided by 36, probability of heads
5
6  print "Probability and Random Proc. Assignment 1\nJustin May RUID:170003686 \n\np=.6388 \n" #printing user data to terminal
7
8  n = int(input("please enter an 'n':")) #taking in input for n
9
10 howManyHeads = 0 #initalizing inital heads count to 0
11 table = [] #initalizing a python list of n probabilities
12
13 for i in range(0,n): #will iterate the following code n times
14     if(random.randint(1,101)>(p*100)): #generates a random variable between 1 and 100 and compares it with 100*p
15         table.append(float(howManyHeads) / (i+1)) #if it's greater than, then the simulated fliop is tails and divide the #heads by current # tosses
16     else:
17         howManyHeads += 1 #updatig howManyHeads by 1
18         table.append(float(howManyHeads) / (i+1)) #if it's less than, then the simulated fliop is heads and divide the #heads by current # tosses
19
20 omega.plot(table,'r.',ms = 2.0) #creates a plot using data table
21 omega.ylabel('Probability of Heads') #sets y axis label
22 omega.xlabel('Coin Toss Number') #sets x axis label
23 omega.axis([0,n,0,1]) #determines axis domains and ranges
24 omega.show() #produces the plot
```

Most of the code is explained in the comments. Line 15 and 18 have a type cast to float because Python 2.7 does not automatically type cast and will round down to zero, as supposed to Python 3+ which will automatically type cast. Line 20 has modifiers in the argument ‘r.” makes the plot red dots, ‘ms’ is the markerSize argument and makes the dots smaller and easier to differentiate. The arguments for matplotlib.pyplot.plot are the same as the Matlab plotting implementation. This library automatically will assume any list passed as an argument will be the y coordinates and automatically assign x coordinates starting at 1, which is convenient for this assignment.

The user will determine n for the function, as seen in line 8. The red dot in line 8 is the sublime linter “Bandit” for python code. It throws an error because “input” in Python 2.7 automatically type casts as supposed to “raw_input”. For the purposes of this project this is okay, because the user will always input the correct answer. The implemented function exists in the for loop logic. A blank list is initialized to NULL and a #of heads variable howManyHeads is initialized to zero before the loop. In line 14, the linter throws an error that random.randint is not good practice when used for cryptography which is not the case, so it is ignored and okay. The for loop will loop through from 0 to n, assigning the current loop value to i. if the randomly generated number from 1 to 100 is greater than 100*p or 63.88 then the logical simulation is tails, therefore #heads is not incremented by one and the probability added is the current #heads divided by the number of trials (i+1). The # of trials is i+1 because i starts at zero, as do the procedure in most coding languages. Else the same code (simulated heads) is repeated except #heads is incremented.

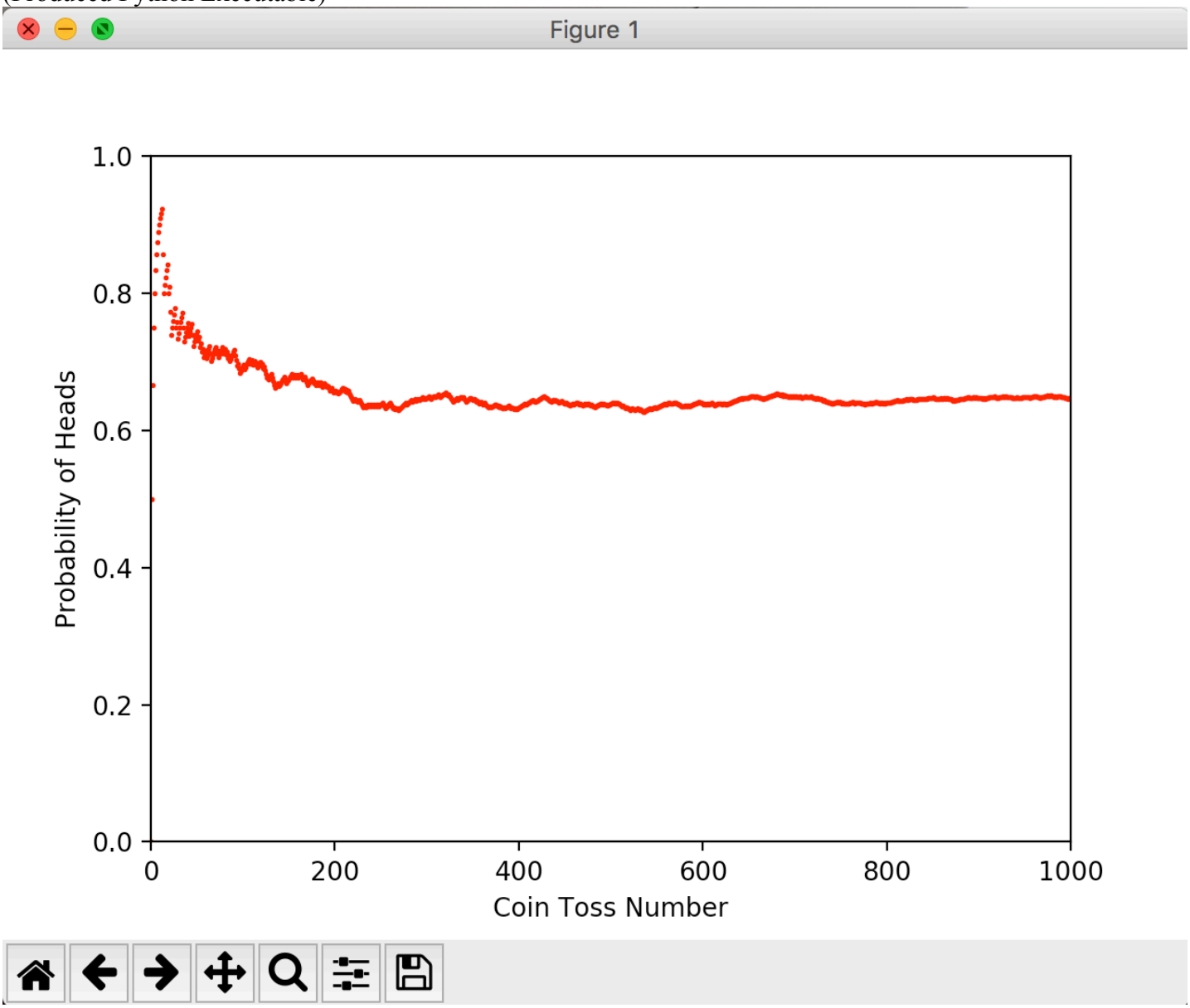
Implementation of Code:
(Terminal)

```
nbp-149-251:coding justinmay$ python probability.py
Probability and Random Proc. Assignment 1
Justin May RUID:170003686

p=.6388

please enter an 'n':1000
```

(Produced Python Executable)



Example 2
(terminal)

```
nbp-149-251:coding justinmay$ python probability.py
Probability and Random Proc. Assignment 1
Justin May RUID:170003686

p=.6388

please enter an 'n':100
```

(Produced Python Executable)