**Name: Session:**

**Programming I**

**Using Python to do your Statistics Homework**

**Lab Exercise 12.8.2020**

Two tools we use to explore the field of probability and statistics are a pair of dice and a coin. We can flip the coin and have an outcome of heads or tails. We can roll a pair of dice and have outcomes of the integer values of 2 through 12. The flip of the coin will have the probability of coming up heads of ½. The probability of coming up 2 through 12 is as follows:

|  |  |
| --- | --- |
| **X** | **Probability of X** |
| 2 | 1/36 |
| 3 | 2/36 |
| 4 | 3/36 |
| 5 | 4/36 |
| 6 | 5/36 |
| 7 | 6/36 |
| 8 | 5/36 |
| 9 | 4/36 |
| 10 | 3/36 |
| 11 | 2/36 |
| 12 | 1/36 |

Why is the probability of rolling a 7 higher than any other? Why does rolling a 2 or a 12 have the same probability?

In this lab we are going to build a package of statistical functions and mathematical models that will allow us to conduct probability and statistical experiments. I have created a skeleton file on the server to get you started. (\\Ada\Data Files\Programming I\Lab Exercise 11.30.2016)

We will start by building a model of a coin flip. Be sure to import the random module.

def flipCoin():

'''

This models a flip of a fair coin

'''

rNumber = random.random()

if rNumber> 0.5:

return 'H'

else:

return 'T'

Now we will build a model of a dice roll.

def rollDice():

'''

This models the roll of two fair die

'''

die1 = random.randint(1, 6)

die2 = random.randint(1, 6)

return die1 + die2

Now we will create a function to simulate flipping a coin n times.

def flipSim(n):

'''

This simulates the flip of a coin n times. Stores the outcome of each roll in a list that contains all n outcomes.

''''

result = []

for i in range(n):

result.append(flipCoin())

return result

Now we will create a function to simulate rolling a pair of dice n times.

def rollSim(n):

'''

This simulates the roll of two die n times. Stores the outcome of each roll in a list that contains all n outcomes.

''''

result = []

for i in range(n):

result.append(rollDice())

return result

Since we are building a statistical package, we should create functions that will allow us to calculate two of the most common statistics namely the mean and standard deviation.**Note that these two functions will only work with a numeric list.**

def calcMean(X):

'''

This returns the arithmetic mean of any numeric list

'''

return sum(X) / len(X)

def stdDev(X):

'''

This returns the standard deviation of any numeric list

'''

mean = calcMean(X)

total = 0.0

for x in X:

tot += (x - mean)\*\*2

return (total/len(X))\*\*0.5

Finally, we would like a method of displaying our statistics in a graphical manner. One of the most useful statistical graphs is the Histogram which gives us a visual representation of the probability distribution of a simulation.

def printHistogram(X):

'''

This will print a histogram of a dictionary containing numeric values

'''

test = ('H', 'T')

print()

print ("Histogram")

keyList = list(X.keys()) #get a list of keys in the dictionary

maxValue = 0

for e in keyList: #find the maximum value of data in the dictionary

if X[e] > maxValue:

maxValue = X[e]

scaleFactor = 40.0 / maxValue #determine the scale factor based on data to ensure

#Histogram fits on screen

for e in keyList: #scale the data

X[e] \*= scaleFactor

for e in keyList: #typecast the data to an integer value

X[e] = int(X[e])

vals = list(X.values()) #make a list of the values in the dictionary

for i in range(len(vals)): #print the histogram

if keyList[i] not in test:

if keyList[i] < 10:

print (keyList[i], " ",end = '')

else:

print (keyList[i], "",end = '')

else:

print (keyList[i], "",end = '')

for i in range(vals[i]):

print ("X",end = '')

print()

print()

print()

Now to make this project more accurate we need to have the ability to run our experiment multiple times, referred to as trials. Here is a simulation function.

def runSimulation(function, n, numTrials, simDict , totalDict):

for trial in range(1, numTrials+1):

dict = simDict.copy() #reset the dictionary counts to 0

print ("Trial", trial)

answer = function(n) #roll or flip n times

for e in answer:

dict[e] += 1 #add up the values for each outcome

print (dict)

keyList = list(dict.keys())

for e in keyList:

totalDict[e] += dict[e] #add values to totalDict

print()

#Calculate average of numTrials trials

for e in keyList:

totalDict[e] /= float(numTrials)

#Round the values in the dictionary

for e in keyList:

totalDict[e] = round(totalDict[e], 2)

#Print the rounded average

print ("Average = ", totalDict)

printHistogram(totalDict)

The runSimulation function has several features that you should understand. The first parameter sent to runSimulation is function which can hold any function sent to it. In this case it can either be fliptSim or rollSim. The parameter n is actually the parameter for the function passed to runSimulation. The parameters simDict and totalDict are dictionaries that can hold the result of either rollSim or flipSim. simDict is copied into another dictionary dict that is “re-zeroed” for each trial whereas totalDict keeps a running total for all trials. Note: a trial is one run of either flipSim or rollSim.

At the top of your file, you should add this code which imports random and defines the dictionaries we will be using.

import random

rollTally = {2:0, 3:0, 4:0, 5:0, 6:0, 7:0, 8:0, 9:0, 10:0, 11:0, 12:0}

flipTally = {'H':0, 'T':0}

rollTotal = {2:0, 3:0, 4:0, 5:0, 6:0, 7:0, 8:0, 9:0, 10:0, 11:0, 12:0}

flipTotal = {'H':0, 'T':0}

Notice that if we are running flipSim, we will be using flipTally and flipTotal whereas if we are running rollSim, we will be using rollTally and rollTotal.

Finally, we need to call`runSimulation by entering the following code at the bottom of your file.

#simulate a million flip coin toss done 3 times

runSimulation(flipSim, 1000000, 3, flipTally, flipTotal)

#simulate a million roll dice roll done 3 times

runSimulation(rollSim, 1000000, 3, rollTally, rollTotal)

If you run this program, you should get an output of something like this.

>>> ================================ RESTART ================================

Trial 1

{'H': 498523, 'T': 501477}

Trial 2

{'H': 500290, 'T': 499710}

Trial 3

{'H': 499668, 'T': 500332}

Average = {'H': 499493.67, 'T': 500506.33}

Histogram

H X XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

T X XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Trial 1

{2: 27611, 3: 55914, 4: 82873, 5: 111158, 6: 138514, 7: 166248, 8: 139233, 9: 111372, 10: 83556, 11: 55703, 12: 27818}

Trial 2

{2: 27819, 3: 55629, 4: 83510, 5: 110673, 6: 139072, 7: 166429, 8: 139306, 9: 111243, 10: 83023, 11: 55535, 12: 27761}

Trial 3

{2: 27934, 3: 55348, 4: 83514, 5: 110758, 6: 139026, 7: 166282, 8: 139465, 9: 111081, 10: 83592, 11: 55190, 12: 27810}

Average = {2: 27788.0, 3: 55630.33, 4: 83299.0, 5: 110863.0, 6: 138870.67, 7: 166319.67, 8: 139334.67, 9: 111232.0, 10: 83390.33, 11: 55476.0, 12: 27796.33}

Histogram

2 X XXXXX

3 X XXXXXXXXXXXX

4 X XXXXXXXXXXXXXXXXXXX

5 X XXXXXXXXXXXXXXXXXXXXXXXXX

6 X XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

7 X XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

8 X XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

9 X XXXXXXXXXXXXXXXXXXXXXXXXX

10 X XXXXXXXXXXXXXXXXXXX

11 X XXXXXXXXXXXX

12 X XXXXX

Now that your program is working well, you are to add the capability of simulating the spin of a roulette wheel. We will use the European model. The wheel contains the numbers 1 to 36 with the odds being red and the evens being black. There is also a green 0.

In order to do this you need to do the following three things:

1. Create rouletteTally and rouletteTotal dictionaries.
2. Create a spinWheel function which returns a random integer from 0 to 36.
3. Create a spinSim function which calls spinWheel n times.

To run this new simulation, enter the following code:

#simulate a million spins of a roulette done 3 times

runSimulation(spinSim, 1000000, 3, rouletteTally, rouletteTotal)

**When your new simulation is running, submit the IDLE shell output of your program and attach it to this handout.**