EE 381 Lab 1 Report

Random Number Experiments

Daniel Duong and Long Nguyen

Instructor: Dr. Duc Tran

Due 9/24/20

Table Of Contents

Introduction

The purpose of this lab is getting used to histogram and new modules and Python. We looked at the sample code provided in the lab 1instructions to run the code and to get an idea of how we are going to do the lab questions. We apply the concept of probability and classical approach to run simulations for drawing “4 of a kind” out of a deck of 52 cards and flipping coins multiple times to find the probability of getting 35 heads. The purpose of the question in the lab is to apply the distribution function and visualize how it works. The purpose of question 1 is to find the probability of a 7 and to get used to the histogram function by passing to it an array.

Procedure:

1. Question 1:
    a. We created a while loop to keep generate two random integers between 1 and 6 inclusive and take the sum of those 2 two numbers as long as the sum of those numbers are not 7
    b. If the sum for the two numbers is die is 7, then it breaks out of the loop
    c. The counter increases every time the loop generates 2 new numbers.
    d. The number of rolls to achieve a 7 is recorded into an array
    e. After the while loop the sum is reset to 0, and the numbers of rolls to achieve a sum of is also reset
    f. The experiment is then run 100000 times using a for loop.
    g. Then we used the array data in order to get our points on the graph.
2. Question 2:
    a. We stored the value of the distribution into a list.
    b. For each experiment, we generate a random number between 0 and 1 and round it to 2 decimal places.
    c. We use multiple if and else if statements to compare the random number to ranges that represent each face of the dice
    d. The face of the dice is then store into a numpy array
    e. We then run the experiments 100000 times using a for loop
    f. Then we used the result array to graph using the histogram function.

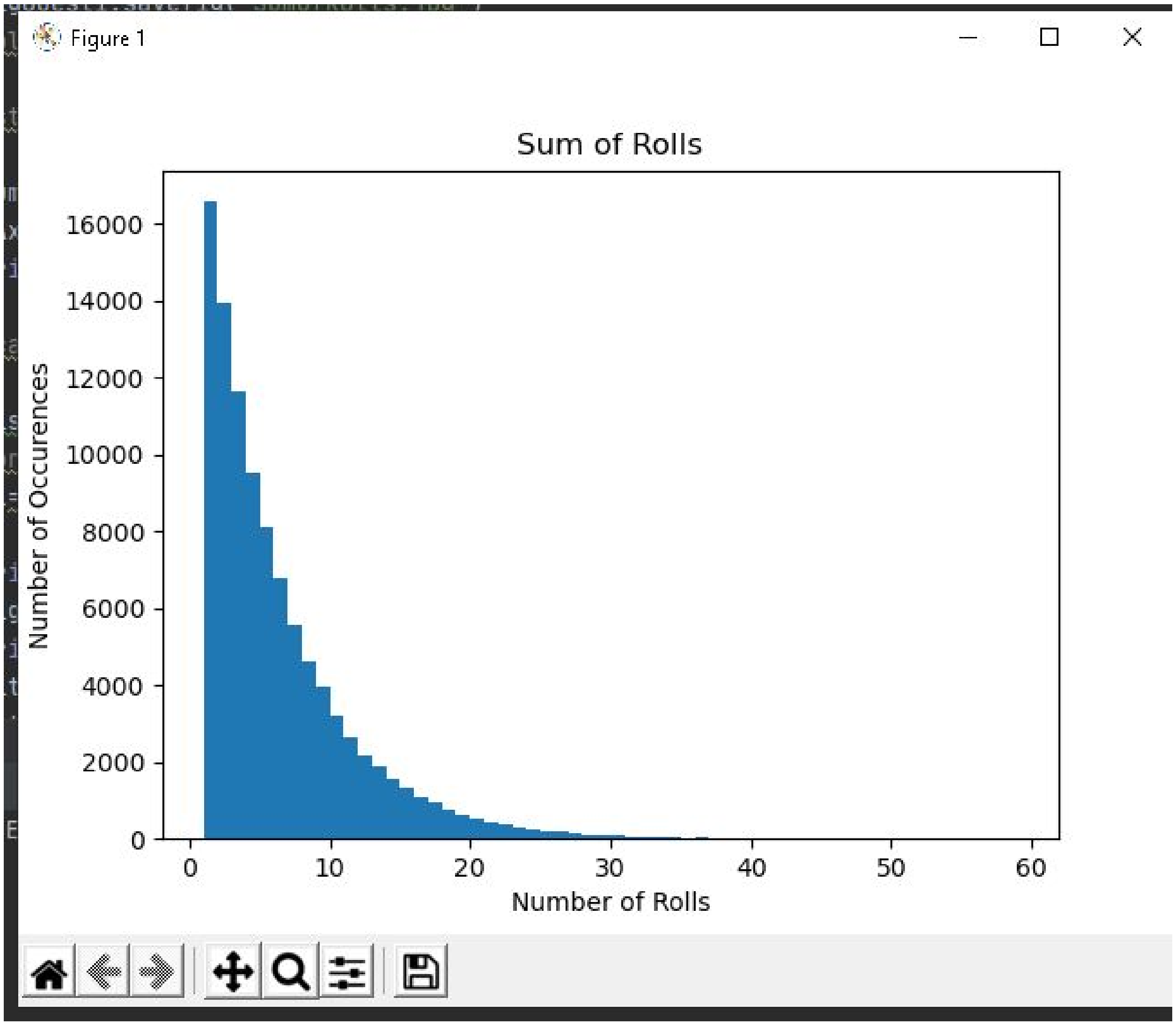
3. Question 3:
    a. We created a for loop to toss a coin 100 times per experiment
    b. We use the random functions to generate the a number between 0 and 1 then round it.
    c. If result is 0 we increment the head counter by 1
    d. After tossing the coin 100 times, and we get 35 heads we increment the number of experiments that has 35 heads out 100 toss by one.
    e. And then we reset the counter of heads to 0
    f. We repeat the experiment 100000 times by using a nested for loop
    g. We divided the number of experiments that have 35 heads by the total number of experiments.

4. Question 4:
    a. Because the experiment is based on a deck of 52 playing cards we use a nested for loop in other to generate and store all of our cards
    b. We shuffle the full deck of cards.
    c. We use a for loop to draw 6 cards from the deck and save it to an array(hand)
    d. The function collections.Counter() is used to count the number of duplicates for a specific type of card.
    e. We travers through the number of values to check if there is a four of a kind then we increment the counter for group of four.
    f. We repeated the experiment one million times.

Results and Discussions

Lab 1 Question 1 Results

Professor's Number: 8.63 * 10^-4 = 0.000863 (Found in Lab Notes #1)



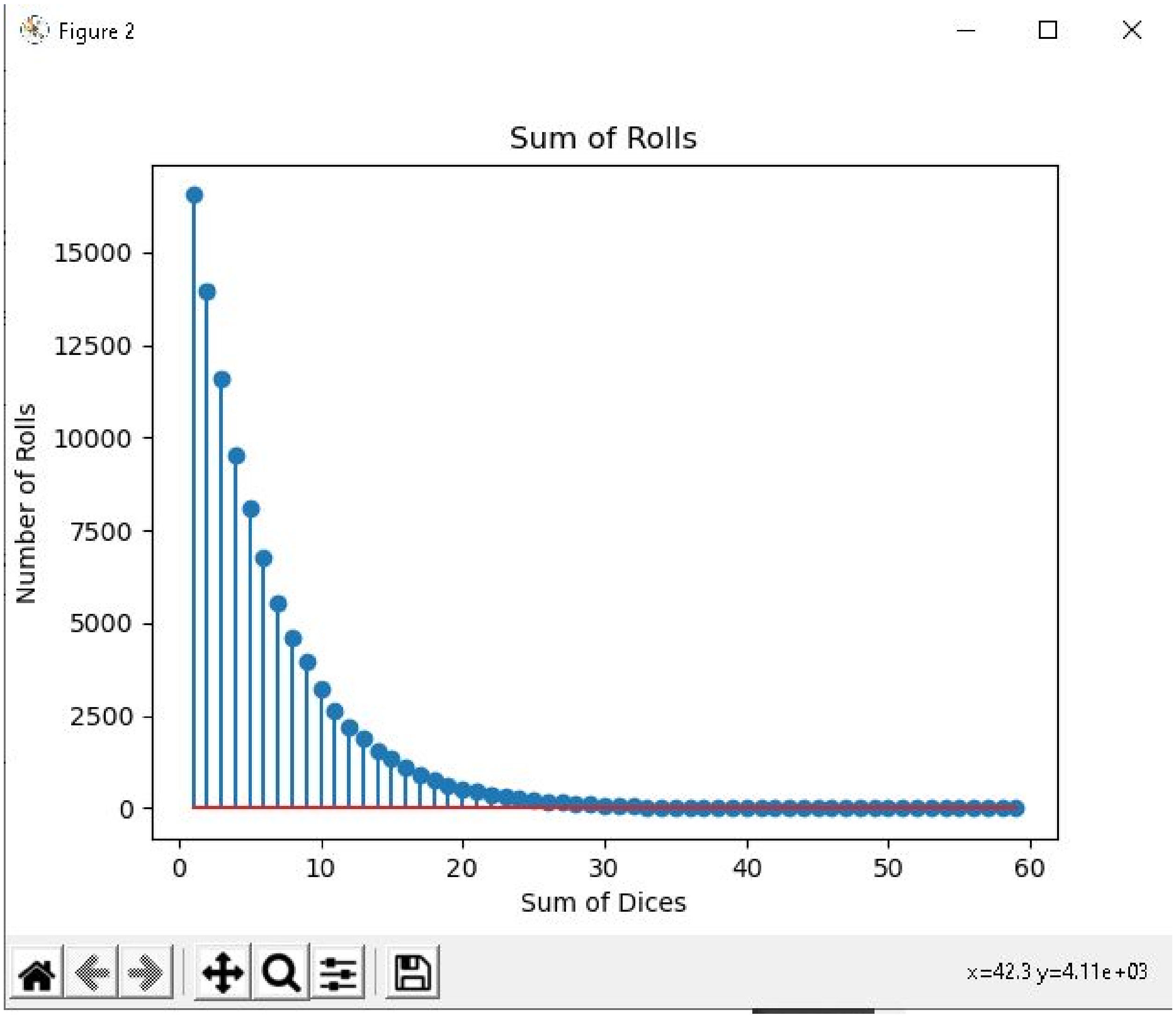
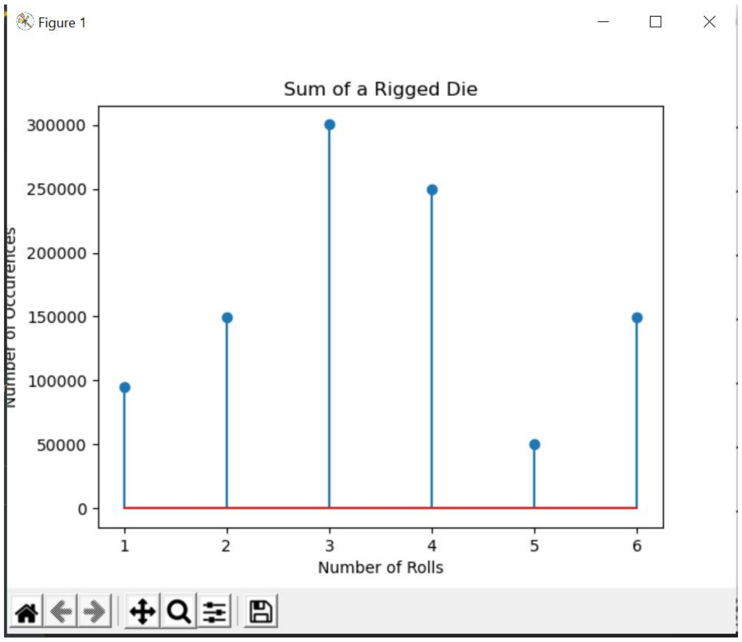
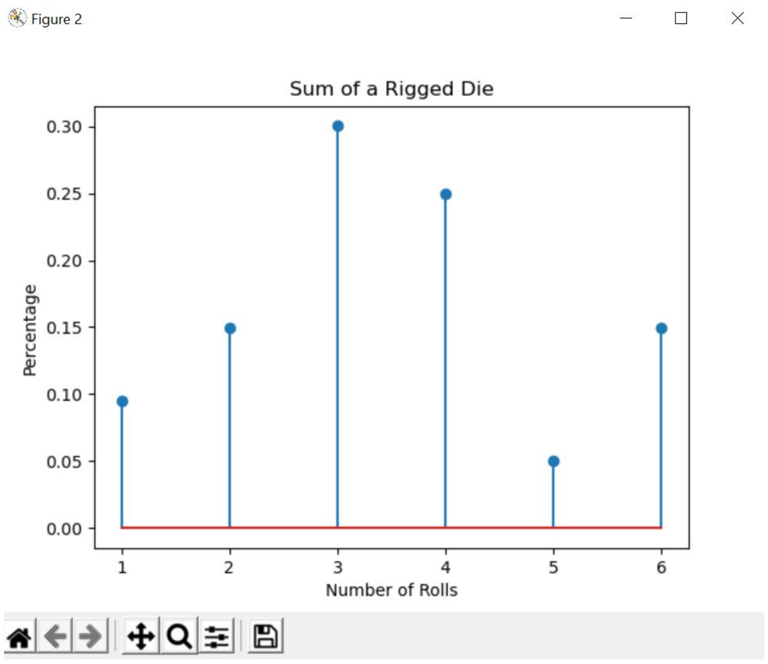


Figure 2
Sum of Rolls
Number of Rolls
Sum of Dices
0
2500
5000
7500
10000
12500
15000
0
10
20
30
40
50
60
x=42.3 y=4.11e+03

Lab 1 Question 2 Results

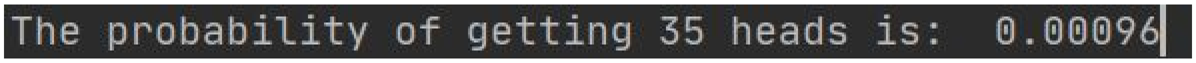




Lab 1 Question 3 Results

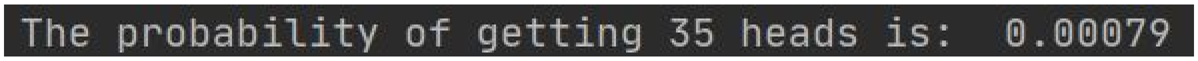
Theoretical Result: 8.63 * 10^-4 = 0.000863

Trial 1:



```
The probability of getting 35 heads is:  0.00096
```

Trial 2:



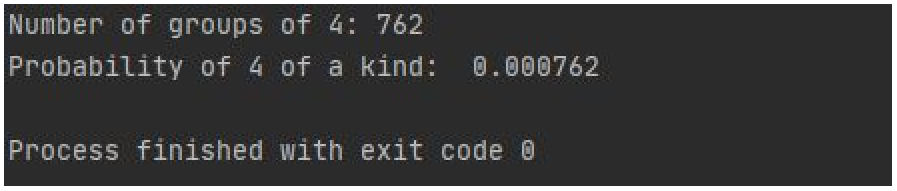
```
The probability of getting 35 heads is:  0.00079
```

Lab 1 Question 4 Results

Theoretical Result: 0.0007

Lab Result:

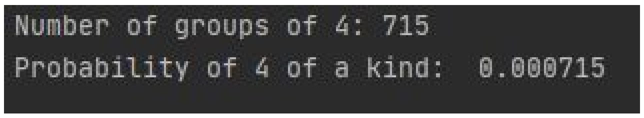
Trial 1:



```
Number of groups of 4: 762
Probability of 4 of a kind:  0.000762

Process finished with exit code 0
```

Trial 2:



```
Number of groups of 4: 715
Probability of 4 of a kind:  0.000715
```

Conclusion

What we learned in this lab was to use python and the new modules that were able to call a histogram graph, to create a numpy array, and the collections import Counter in order to create a dictionary in order to traverse through them to determine the number of duplicates.
-We applied the classical approach by running a program multiple times. We also applied the concept of the distribution function in order to conduct the experiment for question 2 by forming ranges for random numbers to represent the probability of a rigged dice. We used the basic probability function to calculate the probability for questions 1, 3, and 4 by dividing the number of occurrences by the number of experiments.

For question 1, we were new to histograms, so we did not know initially how to plot the functions on top of our heads and figure out the template. In order to find the template for the histograms, we ran sample code that was provided to us during the lab session and looked at lines of code where the graphing part begins. Once we were able to figure out the template, we copied and pasted it over into our code and made minor changes such as the xlabel and ylabel and changing to the appropriate context of the problem such as x label labeled as "Number of Rolls" and the y label labeled as "Number of Occurrences". We had to find a way to count the number of duplicates that a number or a rank has within the 6 cards we drew. We used the from collections import Counter to store the values in the dictionary to list out the cards and the values that we drew and the multiplicity of them. Sometimes we had to run the code more than 100000 times in order to verify that our code works correctly because the results come close to theoretical results but not exactly. We printed the data types of each object by having print(type(variableName)) in order to determine the data type of each variable.

We were stuck on question 1 when it comes to graphing the data that we acquired because we were not sure what value we should put on the X-Axis. We were stuck on how to configure our graph and context that goes with it.

Appendix

Lab1Q1.py

```
#Authors: Daniel Duong and Long Nguyen

#Lab 1 Question 1
import numpy as np
import matplotlib.pyplot as plt
import random

N = 100000
sum = 0

sumsArray = np.zeros((N, 1))
numRolls = 0


for i in range(N):
    while sum != 7:
        numRolls += 1
        sum = random.randint(1, 6) + random.randint(1, 6)
        #print("Sum: ", sum)

    sumsArray[i, :] = numRolls
    #print("Number of rolls: ", numRolls)

    sum = 0 #reset the SUM!!!
    numRolls = 0 #reset the number of rolls

print(sumsArray)

#start the 1st graph here

bins = np.arange(1, 60, 1) #beginning, max number, incrementer
#the first parameter is the starting value, 2nd is the max value, 3rd is incrementer
#or bins = np.arange(-0.95, 0.95, 0.1)
plt.hist(sumsArray, bins)
figQuest1 = plt.figure(1)
plt.title("Sum of Rolls")
plt.xlabel("Number of Rolls")
plt.ylabel("Number of Occurences")
figQuest1.savefig("SumofRolls.jpg")
plt.show()
```

```
#start on the 2nd graph here

numsOnxAxis = range(1, 61)
xAxis = np.size(numsOnxAxis)
print("Size of xAxis is: ", xAxis)

#calling histograms

histo1, bin_edges = np.histogram(sumsArray,numsOnxAxis) #calls the histrogram function,
passes in sumsArray, and b
#print("Bin Edges are : ", bin_edges)
b1=bin_edges[0:xAxis-1] #set the above the x axis which is 13 bc you subtract 1

print("Histol 0 is: ", histo1[0])
figQuest2 = plt.figure(2)
print("The probability of the first element is: ", histo1[0] / N)
plt.stem(b1, histo1)
plt.title("Sum of Rolls")
plt.xlabel("Sum of Dices")
plt.ylabel("Number of Rolls")
figQuest2.savefig("SumofRolls.jpg")
plt.show()
```

Lab1Q2.py

```
#Authors: Daniel Duong and Long Nguyen

#Lab 1 Question 2
import numpy as np
import matplotlib.pyplot as plt
import random

r = 0.0
N = 1000000 #number of experiments
#r is a random generated number

F = [0, 0.1, 0.25, 0.55, 0.8, 0.85, 1] #F is the culmative probability function
resultsArray = np.zeros((N, 1))

for i in range(N):
    r = round(random.random(), 2) #round function rounds this stuff to 2 decmial places according
to the professor
```

```
    if r >= F[0] and r < F[1]:
        resultsArray[i, :] = 1
    elif r >= F[1] and r < F[2]:
        resultsArray[i, :] = 2
    elif r >= F[2] and r < F[3]:
        resultsArray[i, :] = 3
    elif r >= F[3] and r < F[4]: #problem here
        resultsArray[i, :] = 4
    elif r >= F[4] and r < F[5]:
        resultsArray[i, :] = 5
    elif r >= F[5] and r < F[6]:#problem here
        resultsArray[i, :] = 6

    #print("r is: ", r)

#print("List has: ", resultsArray)

#Start the histrogram function
'''bins = np.arange(1, 6, 1)
plt.hist(resultsArray, bins)
figQuestOne = plt.figure(1)
plt.title("Sum of a Rigged Die")
plt.xlabel("Die Faces")
plt.ylabel("Number of Occurences")
figQuestOne.savefig("UnfairDie.jpg")
plt.show() #show the graph'''

numsOnxAxis = range(1, 8)
xAxis = np.size(numsOnxAxis)

histo1, bin_edges = np.histogram(resultsArray,numsOnxAxis) #calls the histrogram function,
passes in sumsArray, and b
print("The hiso1 is a: ", type(histo1))
#print("Bin Edges are : ", bin_edges)
b1=bin_edges[0:xAxis-1] #set the above the x axis which is 8 because the die has 1 - 6 sides
inclusively

#first graph
figQuest1 = plt.figure(1)
plt.stem(b1, histo1)
plt.title("Sum of a Rigged Die")
plt.xlabel("Number of Rolls")
plt.ylabel("Number of Occurences")
figQuest1.savefig("UnfairDie.jpg")
```

```
# a new second graph
figQuest2 = plt.figure(2)

percentage = histo1 /N
print("The type percentage is a: ", type(percentage))

plt.stem(b1, percentage)
plt.title("Sum of a Rigged Die")
plt.xlabel("Number of Rolls")
plt.ylabel("Percentage")
figQuest2.savefig("UnfairDie.jpg")
plt.show()
```

Lab1Q3.py

```
#Authors: Daniel Duong and Long Nguyen

#Lab 1 Question 3

import numpy as np
import random

total_flips = 100000
#total_flips = 1000000 #(debug)
numCoinTosses = 100
totalNumHeads = 0
#heads=np.zeros((total_flips,1))
headCounter = 0 #let heads be the head counter
for i in range(total_flips):
    for j in range(numCoinTosses): #tosses 100 times
        result = round(random.random()) #either going to be in heads or tails

        if result == 0:
            headCounter += 1 #increment your heads by 1 every time that the coin lands on a head

    if headCounter == 35: #once you toss 100 per experiment, if you get the total number of heads
to be 35
        totalNumHeads += 1 #increment the total number of heads and log it

    #print("The number of heads is: ", headCounter)
    headCounter = 0 #reset the heads to 0

print("The probability of getting 35 heads is: ", (totalNumHeads/total_flips))
```

Lab1Q4.py

```
#Authors: Daniel Duong and Long Nguyen

import random
from collections import Counter

#NOTE: List index starts at 0, NOT 1

numExperiments = 100000
ranks = ["A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"]
#print(len(ranks))
groupOf4 = 0

for i in range(numExperiments):
    deck = []
    hand = []
    for i in range(4):
        for j in range(len(ranks)):
            deck.append(ranks[j]) #adding cards to the deck

#print(deck)
    random.shuffle(deck) #shuffle the deck
#print(deck)
#print("The size of the deck is: ", len(deck))

    for j in range(0, 6):
        hand.append(deck.pop()) #pop removes the last card from the deck

    #print("The size of the deck right now is: ", len(deck))

    #print(hand)
    #print(deck)
    #print("The hand has: ", len(hand), " cards")
    #print("The deck has: ", len(deck), " cards")

    handDups = Counter(hand)
    #print("The data type of handdups is: ", type(handDups))
    #print("Hand Dupes:  ", handDups)

    #print(Counter(handDups).values())

    numberOfDuplicates = Counter(handDups).values()
    for i in numberOfDuplicates:
        if i == 4:
            groupOf4 += 1
            print("Hand: ", hand)

print("Number of groups of 4:", groupOf4)
print("Probability of 4 of a kind: ", groupOf4 / numExperiments)
```