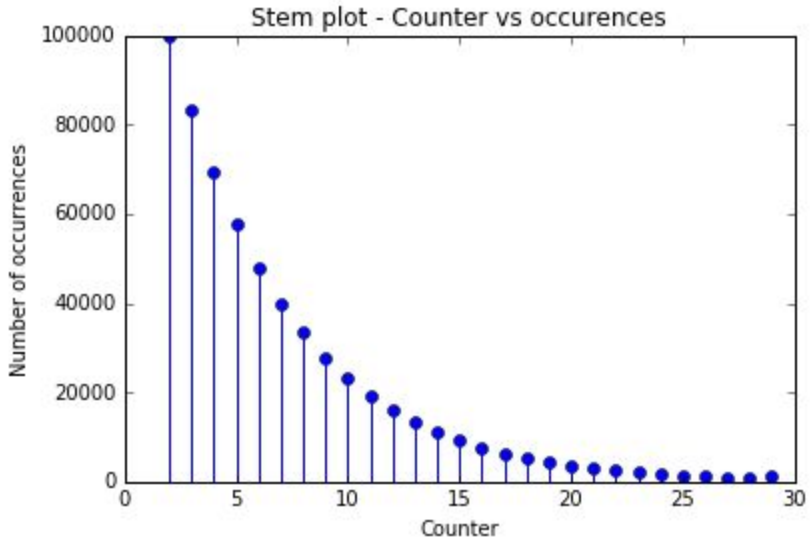
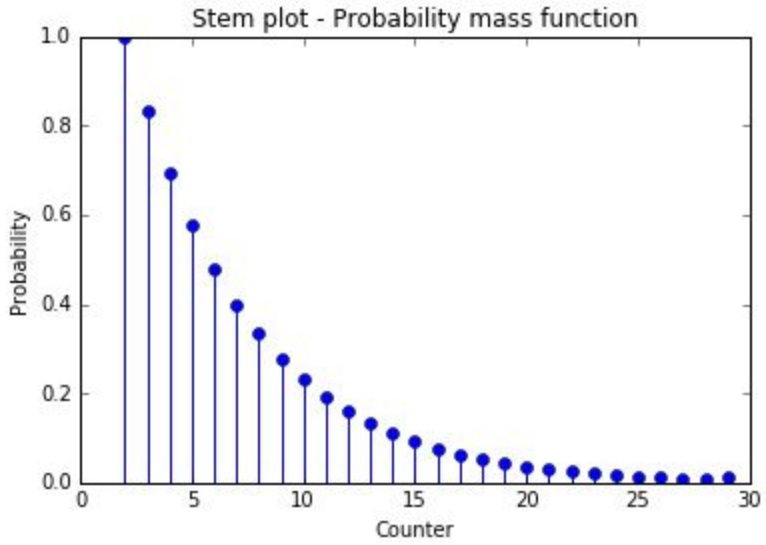
Roberto Sanchez (014587792)
September 16, 2017
EE 381 – Probability and Statistics with Applications to Computing
Lab 1 – Random Numbers and Stochastic Experiment

1) Number of rolls needed to get a “7” with two dice
    a) **INTRODUCTION:**
        i) The problem we are trying to solve is what is the necessary number of rolls we need to get a 7 with two dice. In this problem we are plotting the occurrences vs the number of rounds into a graph along with another graph plotting the probability vs number of rounds
    b) METHODOLOGY:
        i) We solved this problem by first running the base experiment of rolling a pair of dice and adding their value. We used At that point a comparison is done to see if a “7” has been achieved. The program then records that a “7” has been found. The program then runs that base experiment 100,000 times and keeps track of how many occurrences happened. With that number we used python’s histogram libraries to generate the graph.
    c) RESULTS AND CONCLUSIONS:
        i) From running the program, the plots that were generated are the following:

```
In [23]: runfile('/home/beryl/test.py', wdir='/home/beryl')
```





Getting the sum of 7 is highest the first time than later

2) Getting 50 heads when tossing 100 coins

a) INTRODUCTION:

i) This experiment involves tossing 100 coins and tabulating how many occurrences of exactly 50 heads per toss. Then running that experiment 100,000 times to calculate the probability of getting 50 heads when 100 coins are tossed.

b) METHODOLOGY:

i) For this experiment we had to import the random library in order to create random 1's and 0's to represent heads and tails. A list was created to store all the values that are generated when tossing 100 coins. Once the base experiment was run 100,000 times then use the sum function of python to add up the 1's since 1 represent 50 heads out of 100 coins. Then divide the sum by the number of trials the base experiment ran. In this case it's 100,000.

c) RESULTS AND CONCLUSIONS:

Probability of 50 heads in tossing 100 fair coins	
Ans.	**P =0.0796**

There is a 7.96% chance that 50 heads will appear if you toss 100 fair coins

3) Getting 4-of-a-kind
    a) **INTRODUCTION:**
        i) This experiment involves having a deck of 52 cards and shuffling the deck and drawing 4 cards. At that point a comparison is made to see if you got 4 of a kind. If there is then it is considered a success, if not, its a fail. The main experiment runs this base experiment 100,000 times to find the probability of getting 4 of a kind in 100,000 rounds.
    b) **METHODOLOGY:**
        i) To solve this problem, a card class was created to store both the suite and rank value. In the base experiment, the program generated 52 cards and drew the first 5 cards in the list. Then those cards got stored on a hand list to be counted based on rank. Once the program tabulated the results if there are any 4 of a kind then the base experiment returns a '1' or if not returns a '0'. The main experiment runs the base experiment 100,000 times to calculate the probability of getting 4 of a kind.
    c) **RESULTS AND CONCLUSIONS:**

Probability of 4 of a kind	
Ans.	P =0.000238

There is a 0.238% chance of getting a 4 of a kind.

4) The Password Hacking Problem
    a) **INTRODUCTION:**
        i) This problem involves a hacker creating a list of four character passwords 10^5 long. We then compare our password with that of the hacker's list and see if our password is on there. If it is then it will record it in the main experiment and calculate the probability of getting your password generated.
    b) **METHODOLOGY:**
        i) This problem was solved by creating the base experiment where the list of 10^5 passwords were generated and stored on a list object. The main experiment passes the password to the base experiment to see if exists on the list. The main experiment runs the base experiment 1,000 times to calculate the probability of your password appearing on that list.

c) RESULTS AND CONCLUSIONS:

Prob that at least one of the words matches the password 10^5	P = 0.189
Prob that at least one of the words matches the password 10^6	P = 0.875
Approximate number of words in the list (for p=0.5)	M = 320,000 (p=0.489)

5) Appendix
   a) Code for #1

```
'''
Name: Roberto Sanchez (014587792)
Date: Sept 16, 2017
Assignment 1 - Part 1
'''
from random import randint
from numpy import histogram
import matplotlib.pyplot as plt
def sum2dice(N):
    record = []
    # Generates value for each die
    for x in range (0,N):
        sum1 = 0
        counter = 0
        while (sum1 != 7):
            counter += 1
            d1=randint(1,6)
            d2=randint(1,6)
            sum1=d1+d2
            record.append(counter)
    b=range(1,30)
    h1, bin_edges = histogram(record,bins=b)
    b1=bin_edges[1:max(record)]
    # Warning undefine name 'close'??? Couldn't run
    #close('all')
    # Begin generating Plot
    fig1=plt.figure(1)
    plt.stem(b1,h1)
    plt.title('Stem plot - Counter vs occurences')
    plt.xlabel('Counter')
    plt.ylabel('Number of occurrences')
    fig1.savefig('1 EE381 Proj Stoch Exper-1.jpg')
    # Second Figure
    fig2=plt.figure(2)
    p1=h1/N
    plt.stem(b1,p1)
    plt.title('Stem plot - Probability mass function')
    plt.xlabel('Counter')
    plt.ylabel('Probability')
    fig2.savefig('1 EE381 Proj Stoch Exper-1.jpg')

sum2dice(100000)
```

b) Code for #2

```
# -*- coding: utf-8 -*-
"""
Created on Sat Sep 16 22:19:17 2017

@author: Administrator
"""
from random import randint
def coin2(N):
    coins = []
    for x in range(0,N):
        #Let 0 be tails and 1 be heads
        coin = randint(0,1)
        coins.append(coin)
    #print ("List of Coin Values = ",coins)
    heads = sum(coins)

    if heads == 50:
        #print ("This experiment was a success!")
        return 1
    else:
        #print ("This experiment was a failure...")
        return 0
def experiment(N):
    result = []
    for x in range(0,N):
        res = coin2(100)
        result.append(res)
   # print("list", result)
    prob = sum(result)/N
    print("Probability of getting 50 heads when tossing 100 coins 100,000 times are ",prob)

experiment(100000)
```

c) Code for #3

```
# -*- coding: utf-8 -*-
"""
Created on Sat Sep 16 22:19:17 2017

@author: Administrator
"""
from random import randint
def coin2(N):
    coins = []
    for x in range(0,N):
        #Let 0 be tails and 1 be heads
        coin = randint(0,1)
        coins.append(coin)
    #print ("List of Coin Values = ",coins)
    heads = sum(coins)

    if heads == 50:
        #print ("This experiment was a success!")
        return 1
    else:
        #print ("This experiment was a failure...")
        return 0
def experiment(N):
    result = []
    for x in range(0,N):
        res = coin2(100)
        result.append(res)
   # print("list", result)
    prob = sum(result)/N
    print("Probability of getting 50 heads when tossing 100 coins 100,000 times are ",prob)

experiment(100000)# -*- coding: utf-8 -*-
"""
Name: Roberto Sanchez (014587792)
Date: Sept 16, 2017
Assignment 1 - Part 3
"""
from random import randint
from random import shuffle
class card:
    def __init__(self,rank,suite):
        self.rank = rank
        self.suite = suite
    def getRank(self):
        return self.rank
    def getSuite(self):
        return self.suite
    def __str__(self):
        return ('{0} of {1}'.format(self.rank,self.suite))
    def __repr__(self):
        return ('{0} of {1}'.format(self.rank,self.suite))
```

```
52
53 def drawingCards():
54     # Could be easier just to make a 2d array...too late
55     cardRank = ['Ace',1,2,3,4,5,6,7,8,9,'Jack','Queen','King']
56     cardSuit = ["Clubs","Spades","Hearts","Diamonds"]
57     shuffle(cardRank)
58     shuffle(cardSuit)
59
60     deck = []
61     for x in range(0,5):
62         myCard = card(cardRank[randint(0,12)],cardSuit[randint(0,3)])
63         #print(myCard)
64         deck.append(myCard)
65     #print("Deck",deck)
66
67     # Calculating 4 of a kind
68     #Tracks each suite
69     numClubs = 0
70     numSpades = 0
71     numHearts = 0
72     numDiamonds = 0
73
74     # Iterates through deck tabulating suite occurances
75     for x in range(0,5):
76         if deck[x].getSuite() == "Clubs":
77             numClubs= numClubs + 1
78         elif deck[x].getSuite() == "Spades":
79             numSpades= numSpades + 1
80         elif deck[x].getSuite() == "Hearts":
81             numHearts= numHearts + 1
82         elif deck[x].getSuite() == "Diamonds":
83             numDiamonds= numDiamonds + 1
84     #Calculating Rank
85     nA = 0
86     n1 = 0
87     n2 = 0
88     n3 = 0
89     n4 = 0
90     n5 = 0
91     n6 = 0
92     n7 = 0
93     n8 = 0
94     n9 = 0
95     nj = 0
96     nq = 0
97     nk = 0
98     for x in range (0,5):
99         if deck[x].getRank() == 'Ace':
100            nA = nA + 1
101        elif deck[x].getRank() == 1:
102            n1 = n1 + 1
103        elif deck[x].getRank() == 2:
104            n2 = n2 + 1
105        elif deck[x].getRank() == 3:
106            n3 = n3 + 1
```

```
    for x in range (0,5):
        if deck[x].getRank() == 'Ace':
            nA = nA + 1
        elif deck[x].getRank() == 1:
            n1 = n1 + 1
        elif deck[x].getRank() == 2:
            n2 = n2 + 1
        elif deck[x].getRank() == 3:
            n3 = n3 + 1
        elif deck[x].getRank() == 4:
            n4 = n4 + 1
        elif deck[x].getRank() == 5:
            n5 = n5 + 1
        elif deck[x].getRank() == 6:
            n6 = n6 + 1
        elif deck[x].getRank() == 7:
            n7 = n7 + 1
        elif deck[x].getRank() == 8:
            n8 = n8 + 1
        elif deck[x].getRank() == 9:
            n9 = n9 + 1
        elif deck[x].getRank() == "Jack":
            nj = nj + 1
        elif deck[x].getRank() == "Queen":
            nq = nq + 1
        elif deck[x].getRank() == "King":
            nk = nk + 1
    #Printing Result
    #print("Results of this draw",numClubs," ",numSpades," ",numHearts," ",numDiamonds)
    #Checking for 4 of a kind, if true return 1 for success
    """
    if numClubs == 4 or numSpades == 4 or numHearts == 4 or numDiamonds == 4:
        return 1
    """
    if nA == 4 or n1 == 4 or n2 == 4 or n3 == 4 or n4 == 4  or n5 == 4 or n6 == 4 or n7 == 4 or n8 == 4 or n9 == 4:
        return 1
    elif nj == 4 or nq == 4 or nk == 4:
        return 1
    else:
        return 0
def experiment(N):
    result = []
    for x in range(0,N):
        res = drawingCards()
        result.append(res)
   # print("list", result)
    prob = sum(result)/N
    print("Probability of getting 4 of a kind in  100,000 trails are ",prob)
experiment(100000)
```

d) Code for #4

```
"""
Name: Roberto Sanchez (014587792)
Date: Sept 16, 2017
Assignment 1 - Part 4
"""
from random import randint
def comparePasswords(ps):
    characters = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']
    for x in range (0,100000):
        if ps == characters[randint(0,25)] + characters[randint(0,25)] + characters[randint(0,25)] + characters[randint(0,25)]:
            return 1
    return 0
def experiment(N):
    #Creates List of passwords
    ps = 'aaaa'
    # Checking Experiment
    result = 0
    for x in range(0,N):
        print(x)
        result += comparePasswords(ps)
    print("Result = ",result)
    print("The probablity of that at least one of the words in the hacker's list will match your password is  ",result/N)

experiment(1000)
```