Project #1

Random Number Experiments

The objective of this project is to first *understand* the Python "histogram" function and *duplicate* the results of the example of generating a histogram ("stem" plot as shown) of the probability of the sum of two dice for a user definable number of rolls. The second objective is to submit the four problems at the end of this handout.

First consider the following two programs, which determining the number of heads and tails after flipping a coin 10000 times. Both programs provide the same results, but they differ in the way the models are coded.

- The first model is programmed in Python using "for loops".
- The second model makes use of the arrays, and it is computationally more efficient.

MODEL 1

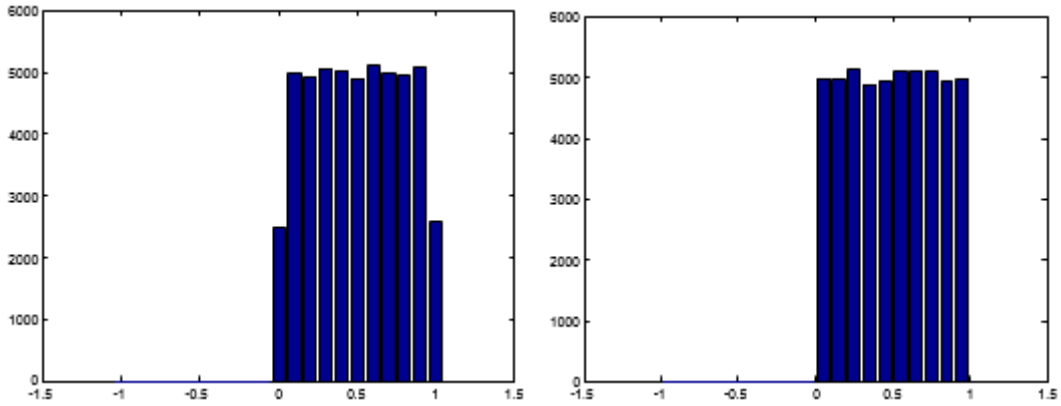
```
import random
total_flips=10000
heads=0
tails=0
for i in range(total_flips):
  coin=round(random.random())
  if coin==0:
     heads=heads+1
  else:
     tails=tails+1
print("number of heads: " ,heads, ".")
```

MODEL 2

```
import numpy as np
import random
total_flips=10000
heads=np.zeros((total_flips,1))
for i in range(total_flips):
   heads[i,:]=round(random.random())
print("number of heads: ",sum(sum(heads)))
print("number of tails: ",sum(total_flips-sum(heads)))
```

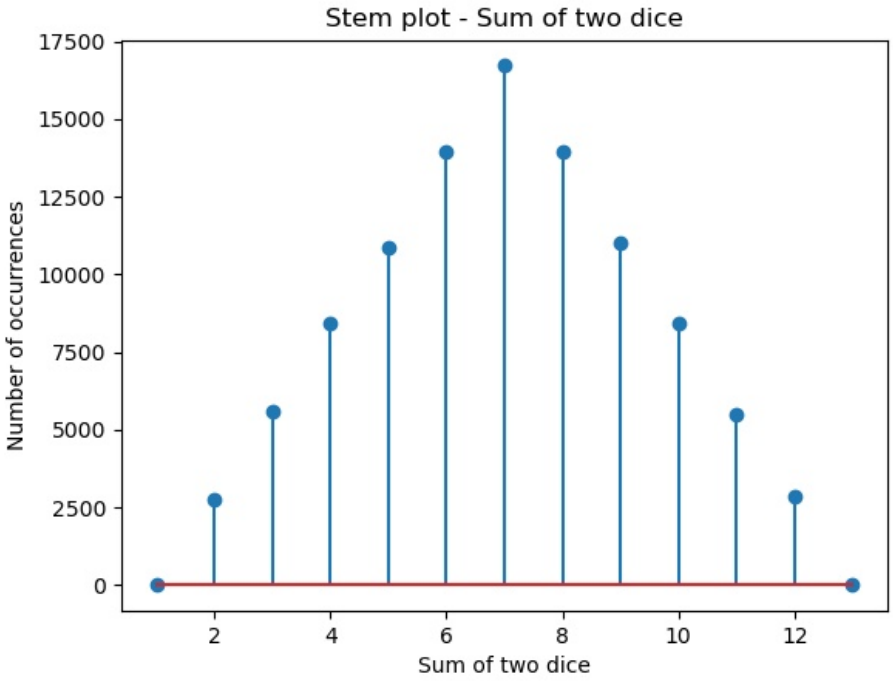
Next, experiment with the "histogram" function by making modifications to the simple program listing below; noting the different outcomes when defining the bin limits differently.

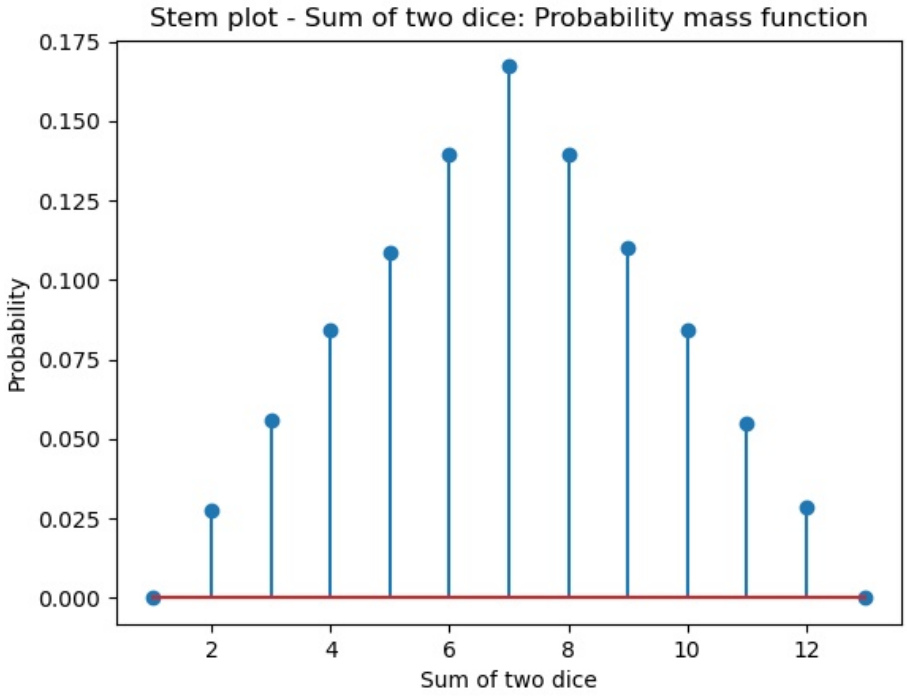
```
import matplotlib.pyplot as plt
import numpy as np
import random
N = 50000
x=np.zeros((N,1))
for i in range(N):
  x[i,:] = random.random()
bins = np.arange(-1, 1, 0.1)
#or bins = np.arange(-0.95, 0.95, 0.1)
plt.hist(x, bins)
plt.show()
```



Now, let experiment the roll of a pair of fair dice. This experiment models the roll of a pair of dice for N times. The sum of each roll is recorded, and stored in vector "s". The probability of each possible outcome is calculated and plotted in a "probability Mass Function" (PMF) plot. To create the plots, the simulation has been run for N=100000 times.

```
import numpy as np
import matplotlib.pyplot as plt
import random
#
N=100000
d1=np.zeros((N, 1))
d2=np.zeros((N, 1))
for i in range(N):
   d1[i,:]=random.randint(1, 6)
for i in range(N):
   d2[i,:]=random.randint(1, 6)
s=d1+d2
b=range(1,15) ; sb=np.size(b)
h1, bin_edges = np.histogram(s,bins=b)
b1=bin_edges[0:sb-1]
#
fig1=plt.figure(1)
plt.stem(b1,h1)
plt.title('Stem plot - Sum of two dice')
plt.xlabel('Sum of two dice')
plt.ylabel('Number of occurrences')
fig1.savefig('Sum of two dice.jpg')
#
fig2=plt.figure(2)
p1=h1/N
plt.stem(b1,p1)
plt.title('Stem plot - Sum of two dice: Probability mass function')
plt.xlabel('Sum of two dice')
plt.ylabel('Probability')
fig2.savefig('PMF of sum of two dice.jpg')
```


Stem plot - Sum of two dice
Number of occurrences
17500
15000
12500
10000
7500
5000
2500
0
2
4
6
8
10
12
Sum of two dice


Stem plot - Sum of two dice: Probability mass function
Probability
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
2
4
6
8
10
12
Sum of two dice

Finally, turn in the required graphical or numerical results for *simulation* problems 1) through 4).

1) Generate a probability histogram of the number of rolls required of two dice before a sum of "7" appears (graphical answer followed by Python code). Note: Don't plot experiment that took more than 60 rolls.

2) Generate an unfair six-sided die. The die has sixe sides [1, 2, 3, 4, 5, 6] with probabilities: $[p_1, p_2, p_3, p_4, p_5, p_6] = [0.1, 0.15. 0.3, 0.25, 0.05, 0.15]$. Simulating the roll of the die for N = 10,000 times, and plot the PMF of your unfair die as stem plot.
The stem plot should verify that the six sides of your unfair die follow the required probabilities.

3) When 100 coins are tossed find the probability that exactly 35 will be heads (numerical answer followed by Python code), assuming the number of experiments is 100000.

4) Determine the probability of "4 of a kind" in a 6-card poker draw (numerical answer followed by Python code).