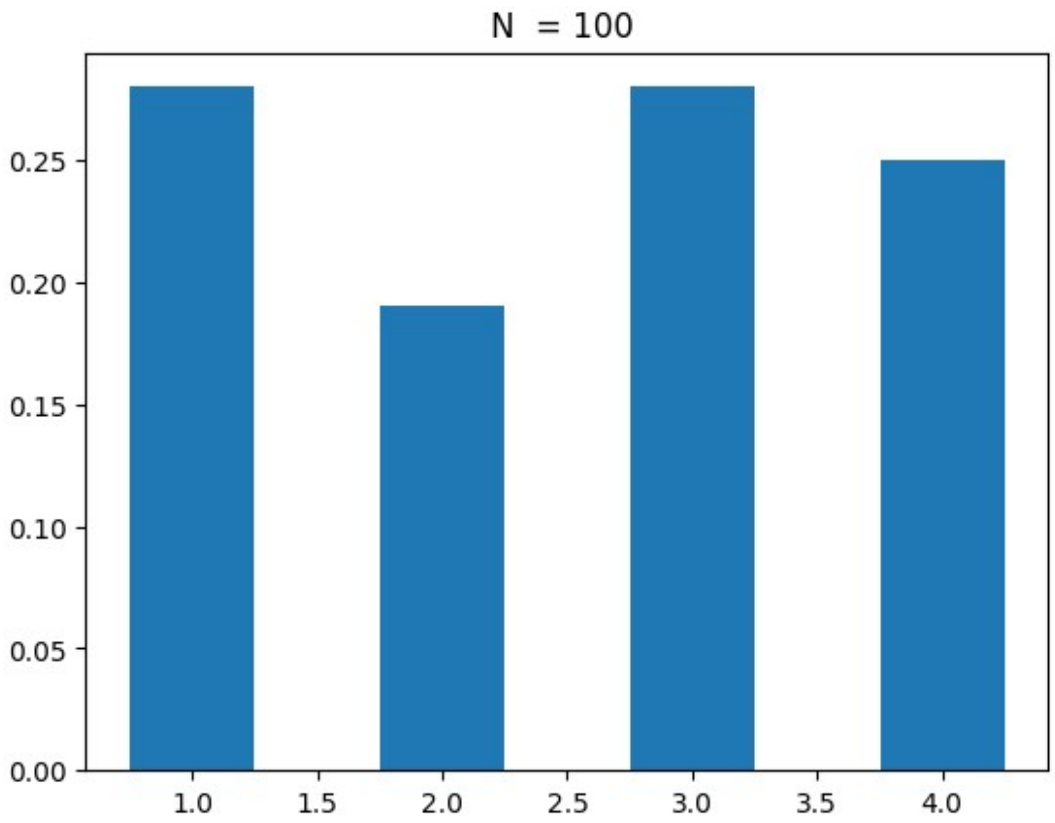
Setup

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import poisson
```

[10 Marks] Task 1 : Plot Histogram

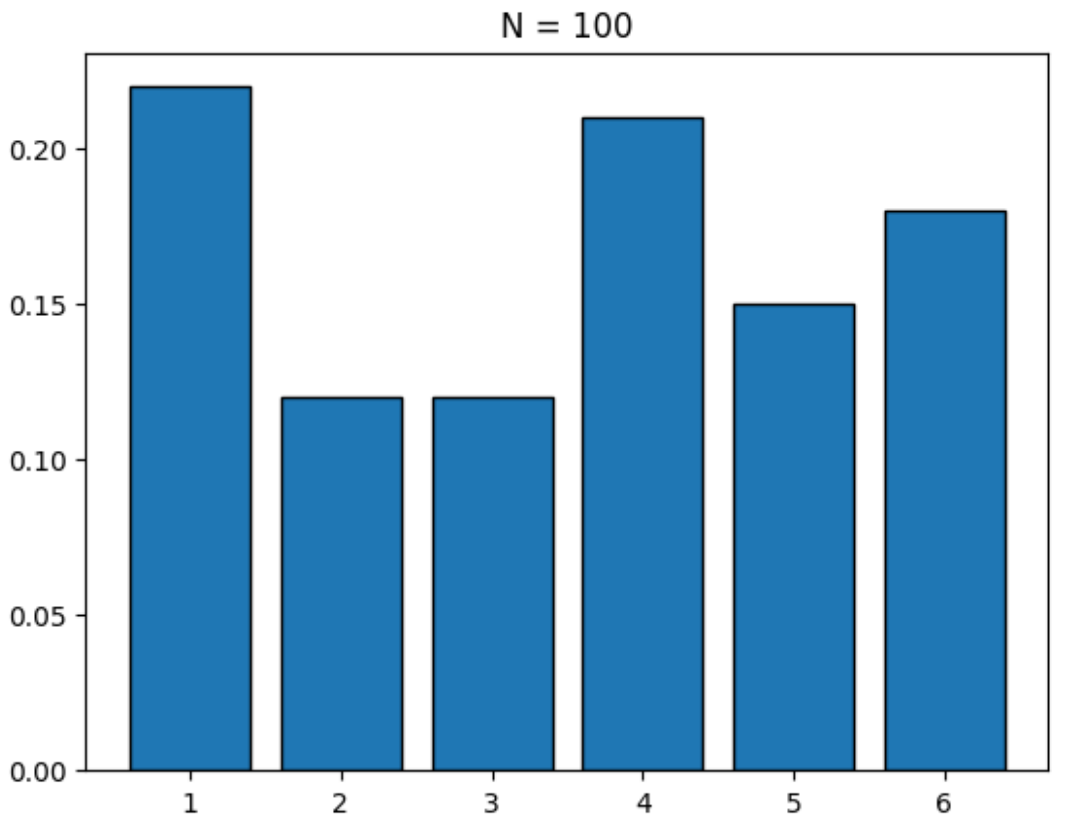
The following code simulates the rolling of a fair 4-sided die 100 times and plots the histogram of the outputs.

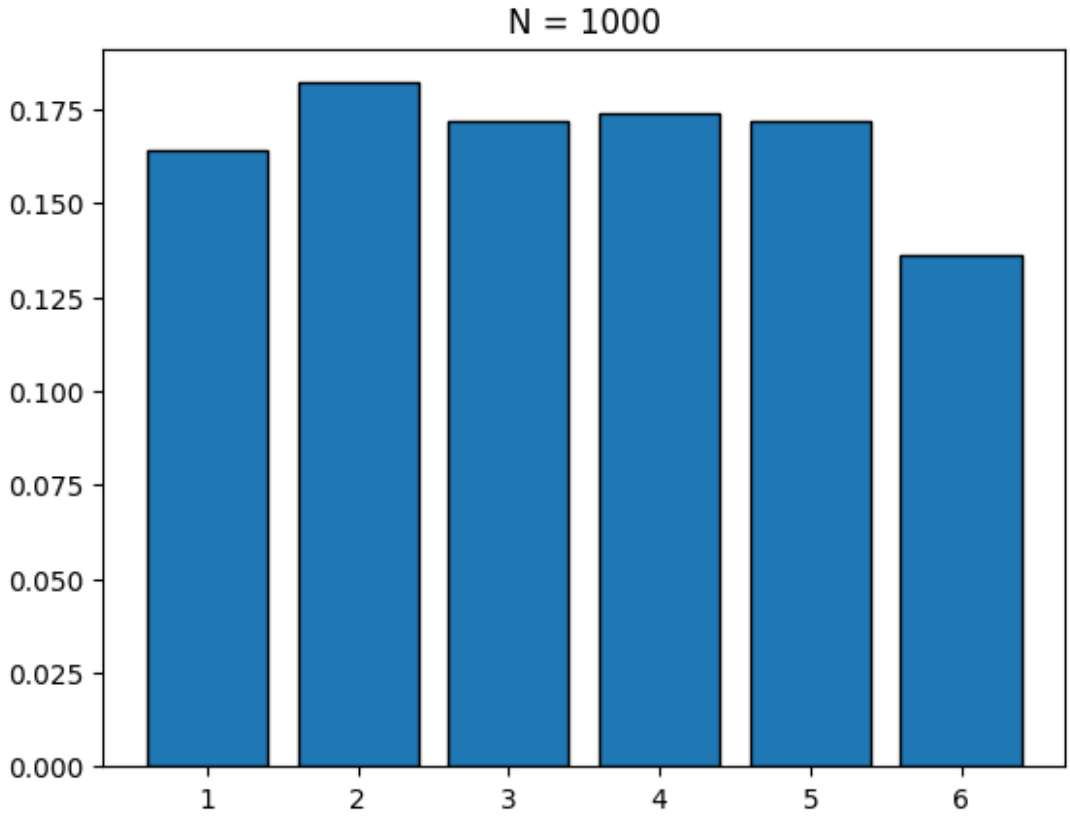
```
# Simulates the roll of a a fair 4-sided die. Size determines the
number of times the die is rolled.
fair  = np.random.randint(low=1,high=5, size=100)
# Plot Histogram.
plt.hist(fair,bins=[1,2,3,4,5],rwidth = 0.5, density=True,
align='left')
plt.title("N  = %i "%100)
plt.show()
```

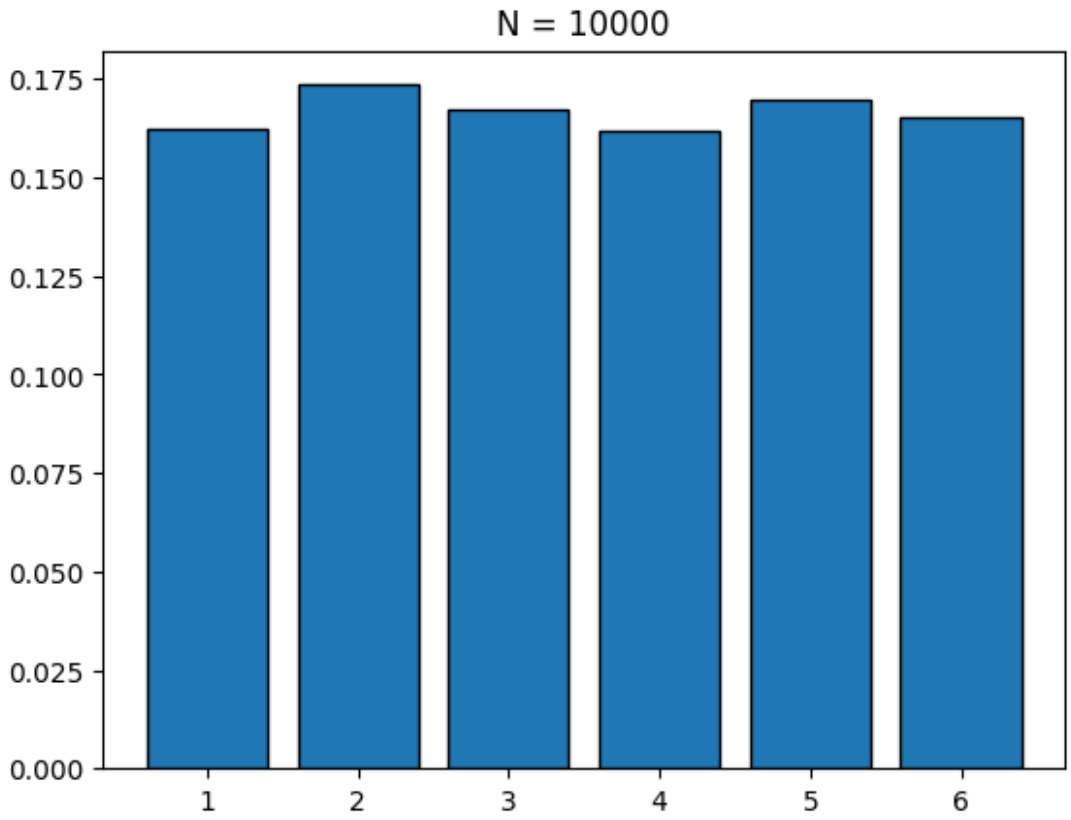


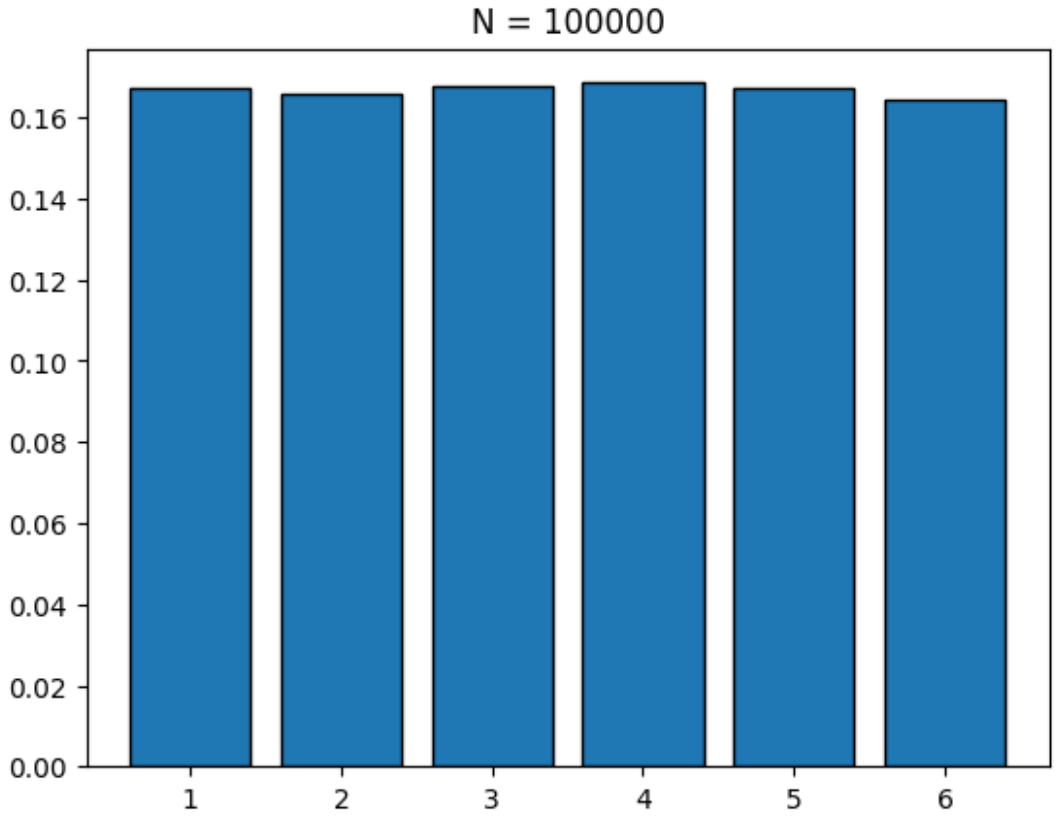
In this task, you need to simulate the throwing of a fair 6-sided die (Octahedron) N times and plot the corresponding histogram of outputs. You are required to experiment with different values of N and comment how the shape of the histogram changes with increasing values of N.

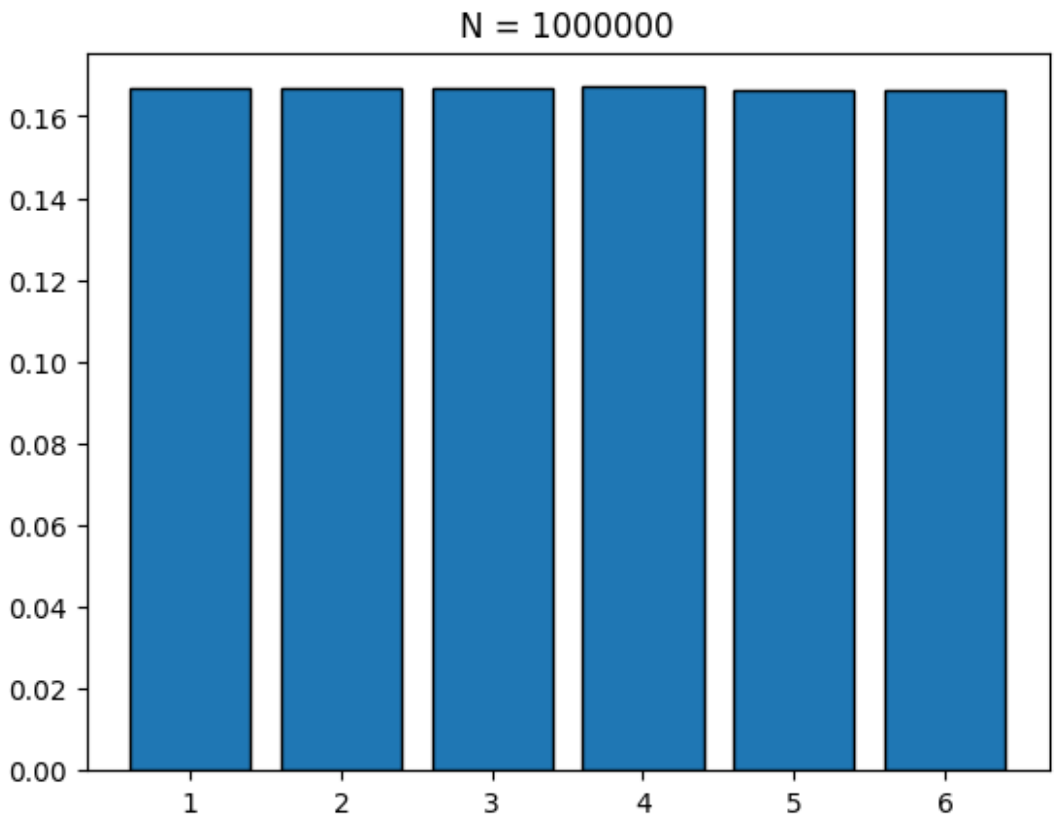
```
# YOUR CODE FOR TASK 1
'''Breeha Qasim bq08283 and Hammad Malik hm08298'''

# Simulate rolling a fair 6-sided die (cube) various times
roll_100_times = np.random.randint(1, 7, 100)  # Simulate 100 rolls
roll_1000_times = np.random.randint(1, 7, 1000)  # Simulate 1,000
rolls
roll_10000_times = np.random.randint(1, 7, 10000)  # Simulate 10,000
rolls
roll_100000_times = np.random.randint(1, 7, 100000)  # Simulate
100,000 rolls
roll_1000000_times = np.random.randint(1, 7, 1000000)  # Simulate
1,000,000 rolls

# Defining bins to plot the histogram
bins = np.arange(1, 8)  #defining bin edges from 1 to 7 (6 faces + 1
for the edge)

# Plotting histograms for different sample sizes to observe
distribution changes
plt.hist(roll_100_times, bins=bins, rwidth=0.8, density=True,
edgecolor='black', align= 'left')
plt.title("N = %i"%100)
plt.show()

plt.hist(roll_1000_times, bins=bins, rwidth=0.8, density=True,
edgecolor='black', align= 'left')
plt.title("N = %i"%1000)
plt.show()

plt.hist(roll_10000_times, bins=bins, rwidth=0.8, density=True,
edgecolor='black',align= 'left')
plt.title("N = %i"%10000)
plt.show()

plt.hist(roll_100000_times, bins=bins, rwidth=0.8, density=True,
edgecolor='black', align= 'left')
plt.title("N = %i"%100000)
plt.show()

plt.hist(roll_1000000_times, bins=bins, rwidth=0.8, density=True,
edgecolor='black', align= 'left')
plt.title("N = %i"%1000000)
plt.show()
```


N = 100
0.20
0.15
0.10
0.05
0.00
1
2
3
4
5
6


N = 1000
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
1
2
3
4
5
6


N = 10000
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
1
2
3
4
5
6


N = 100000
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
1
2
3
4
5
6



Comments
'''Breeha Qasim bq08283 and Hammad Malik hm08298'''

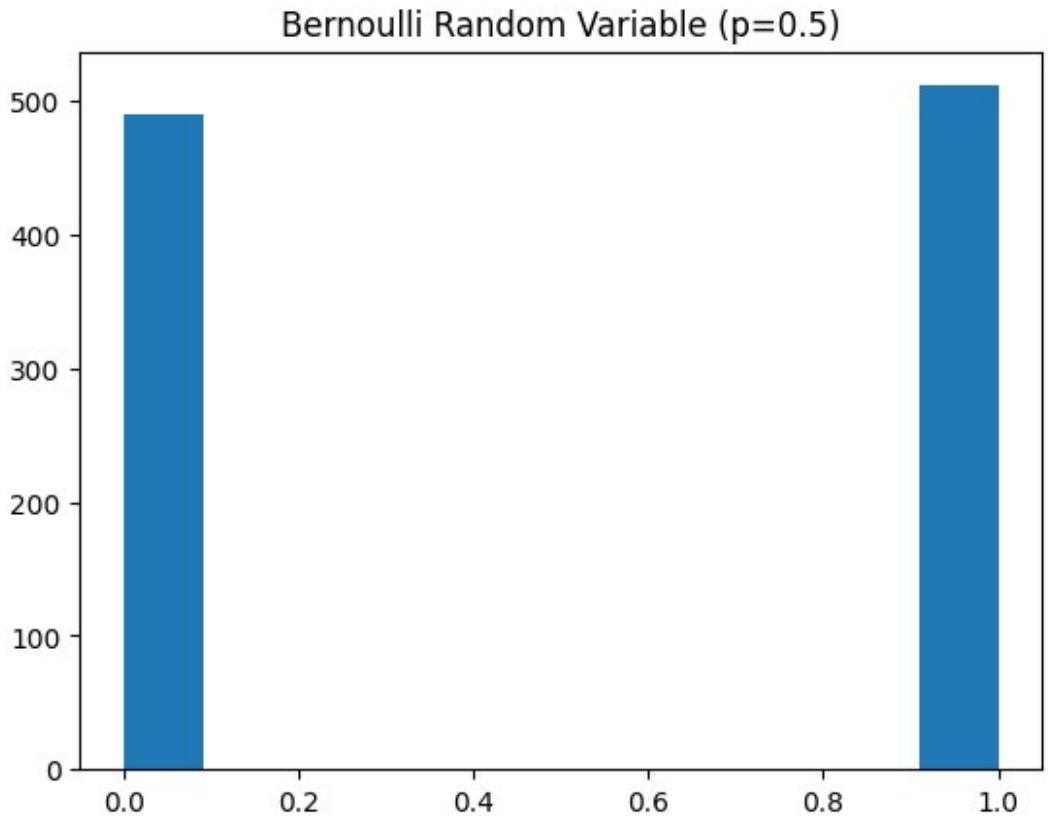
As we roll a fair 6-sided die more times, we see each number on the die come up more often. The more we roll the die, the more each side gets a chance to appear, and eventually, all the sides show up about the same number of times. This is because each side of the die has the same chance of landing face up. When we roll the die lots of times, the number of times each side comes up evens out, showing that every side has an equal chance. This idea matches the law of large numbers, which says that if you do something many times, you'll start to see the expected result more clearly.

[20 Marks] Task 2: Common Discrete Random Variables

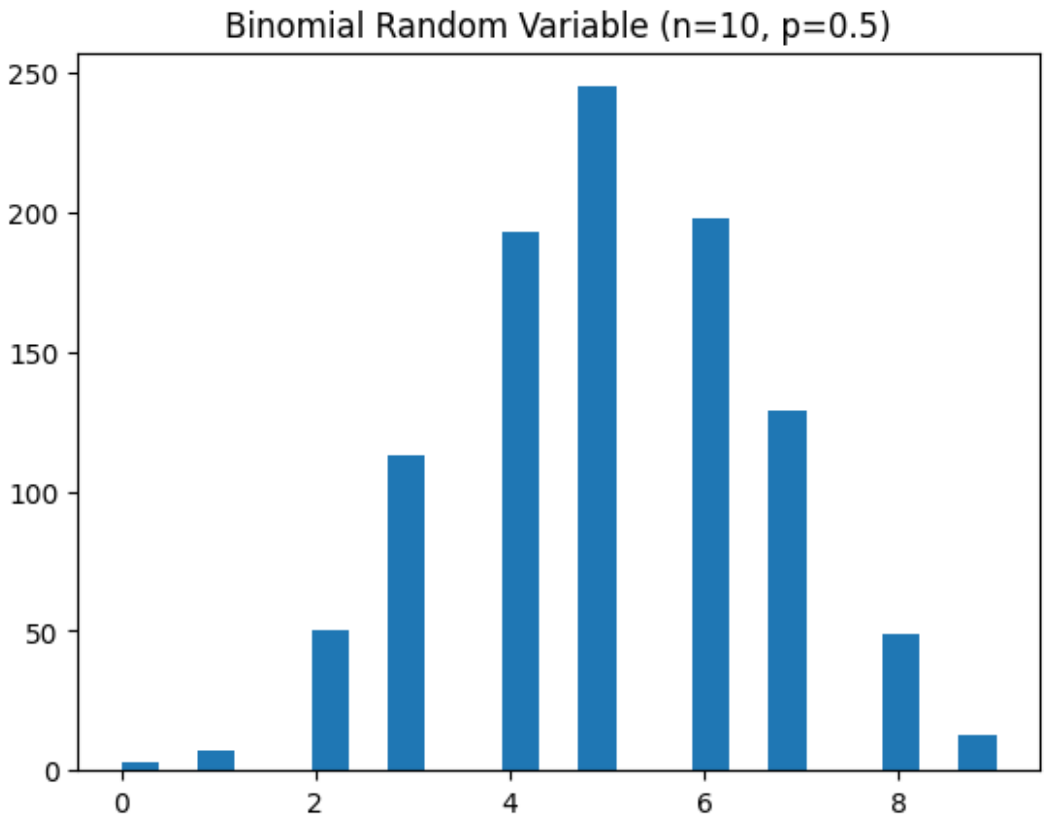
In Task 1, you were introducted to the relationship between histogram and PMF of Random Variables. In Task 2, you will be introduced to common Discrete Random Variables (DRVs) that you have already discussed in class. The aim of Task 2 is to introduce you to the simulation of these DRVs in Python. In this task, you are required to vary parameters for each DRV while keeping others constant and comment on how does this affect the shape of the their histogram.

The following code snippets show you how to simulate the generation of 1000 samples of the following types of random variables: Bernoulli, Binomial, Geometric, Poisson. The code also plots the histogram for the 1000 samples of each type of random variable.

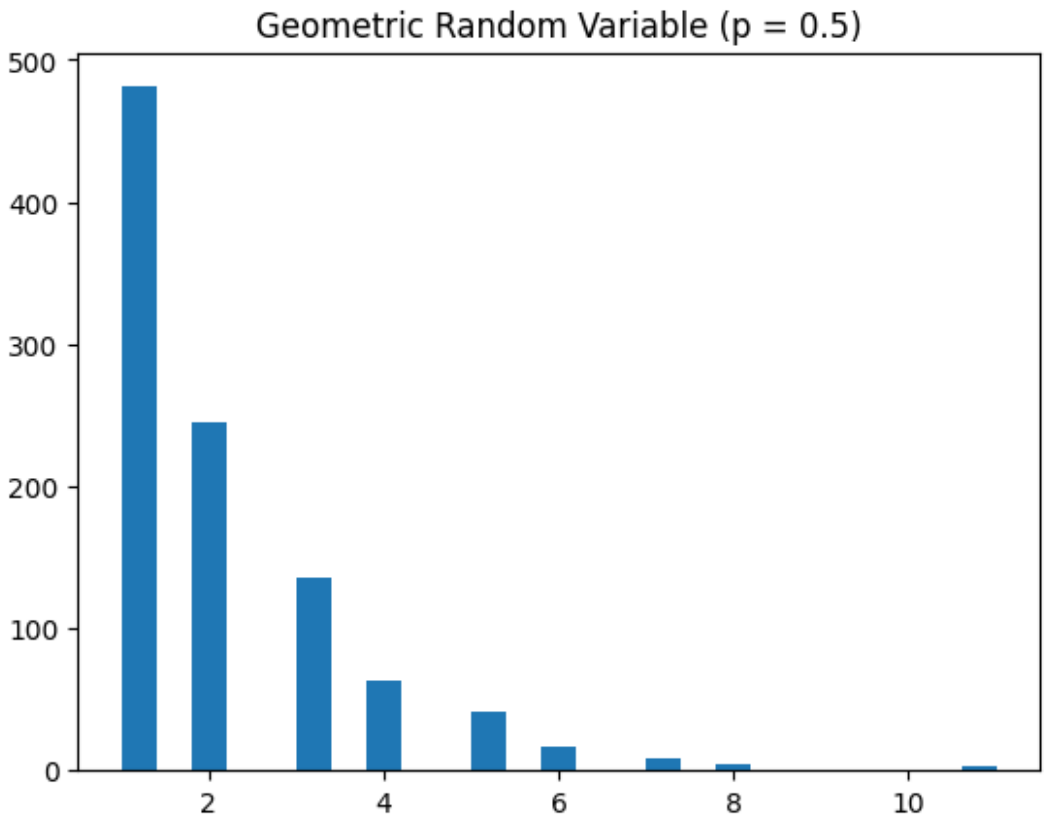
```
# Generate 1000 samples of a Bernoulli Random Variable
p = 0.5 # Probability of Success
n = 1   # Number of states (When n = 1, Binomial Random Variable
becomes a Bernoulli Random Variable)
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Bernoulli Random Variable (p=0.5)")
plt.hist(X,bins='auto')
plt.show()
```



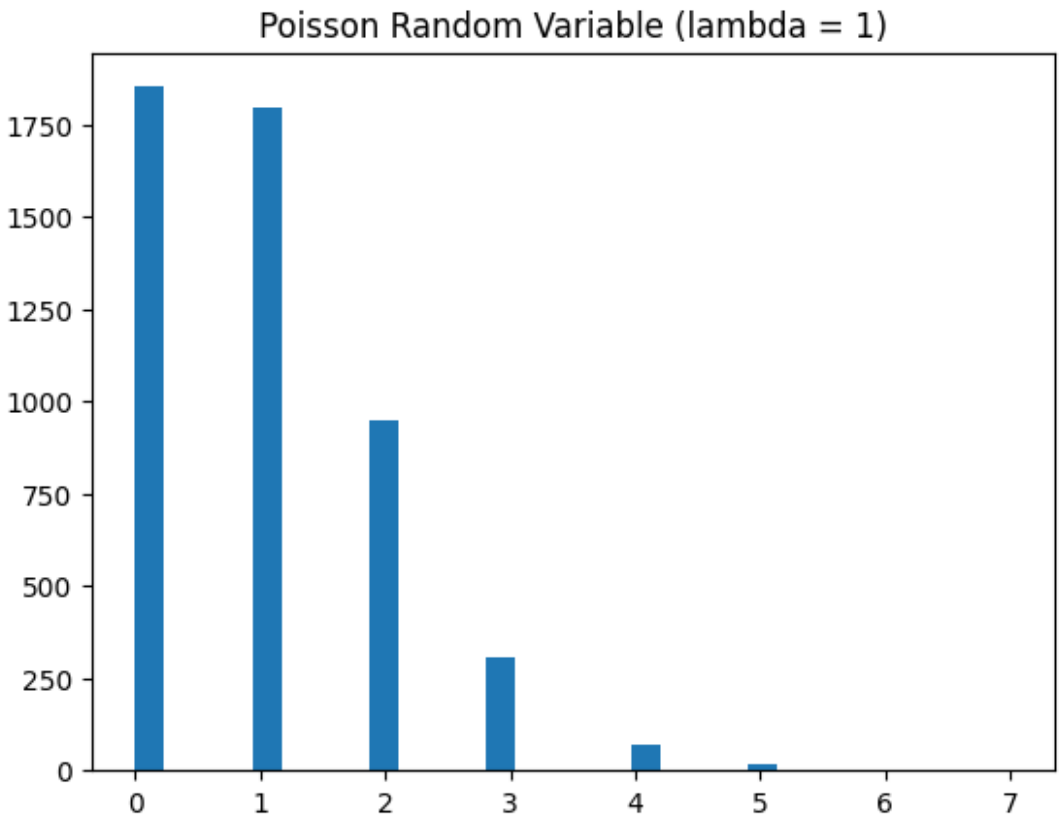

```
# Generate 1000 samples of a Binomial Random Variable
p = 0.5 # Probability of a single success
n = 10  # Number of states
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=10, p=0.5)")
plt.hist(X,bins="auto");
plt.show()
```




```
# Generate 1000 samples of a Geometric Random variables
p = 0.5
size = 1000
plt.title("Geometric Random Variable (p = 0.5)")
X = np.random.geometric(p,size=size)
plt.hist(X,bins="auto")
plt.show()
```



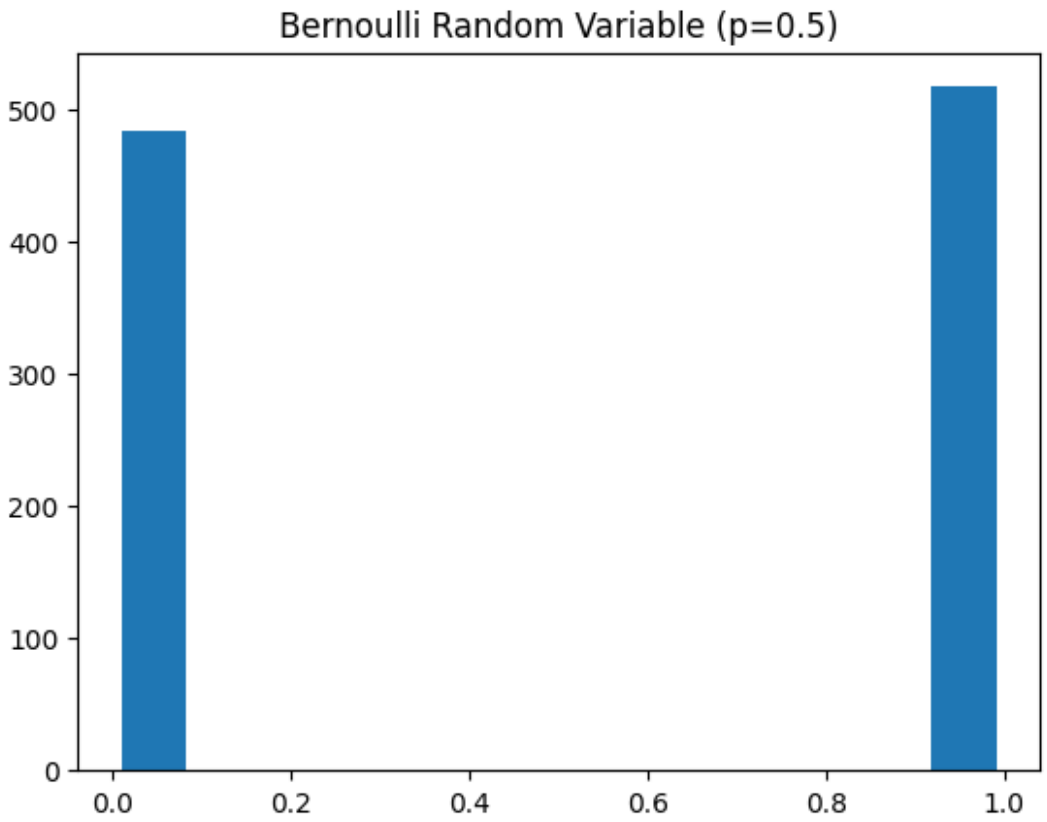

```
# Generate 5000 samples of a Poisson Random Variable
lambd = 1
X = np.random.poisson(lambd,size=5000)
plt.title("Poisson Random Variable (lambda = 1)")
plt.hist(X,bins="auto")
plt.show()
```

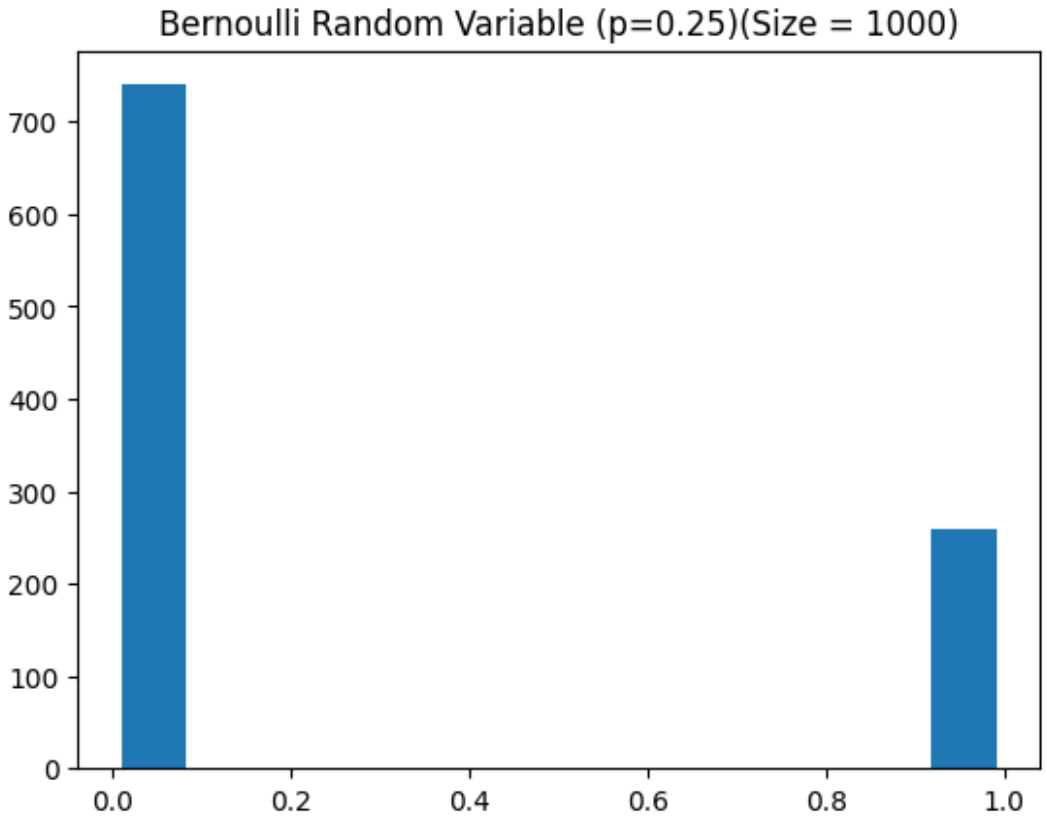


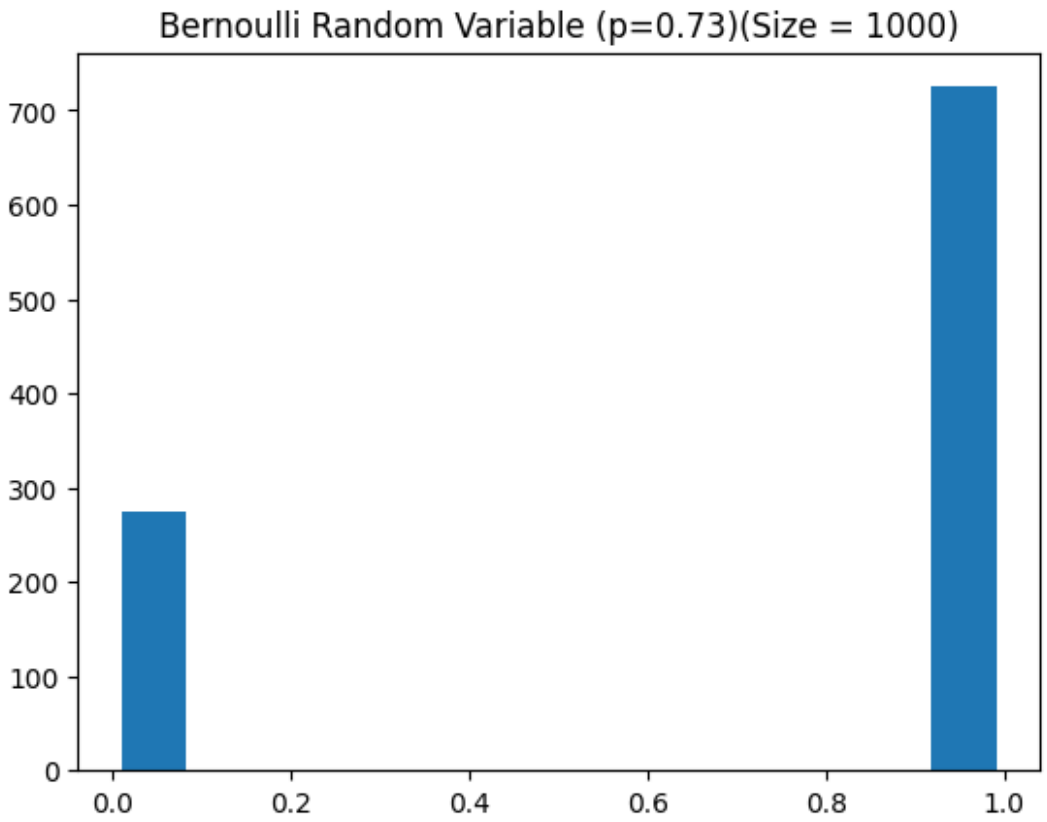
In this task, you are required to vary parameters for each Discrete RV (Bernoulli, Binormail, Geometric, Poission) while keeping others constant and display how does this affect the shape of the their histogram.

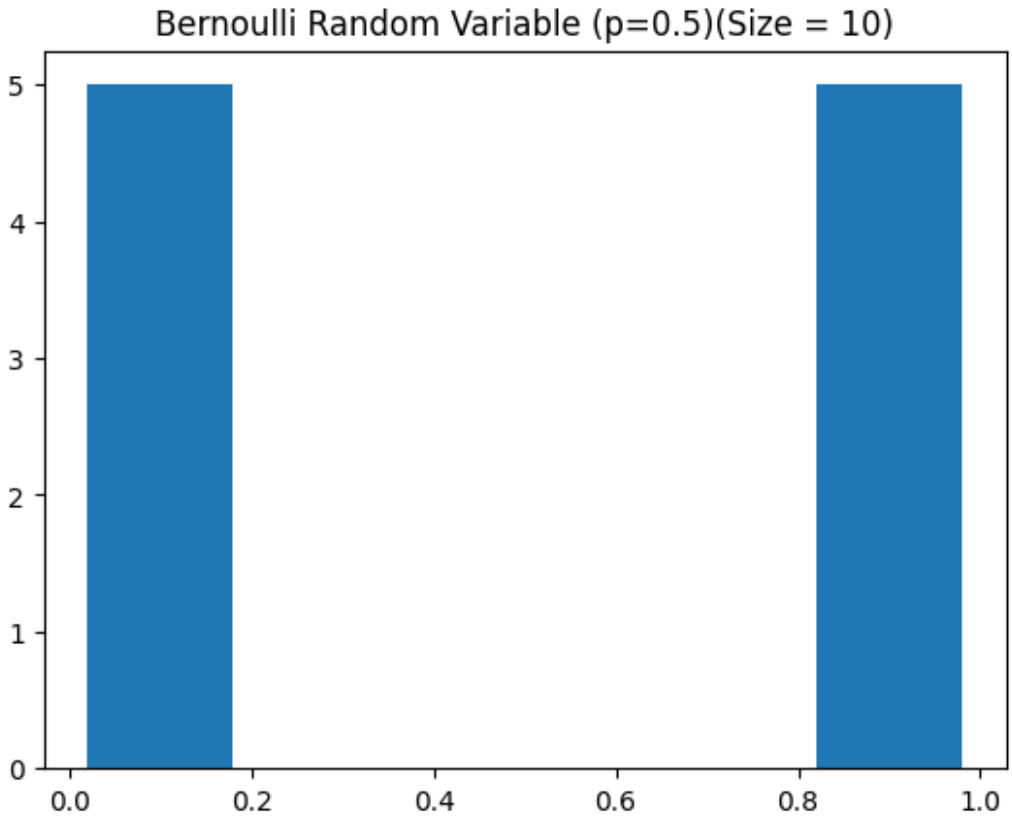
```
# BERNOULLI RANDOM VARIABLE
# Your code goes here
'''Breeha Qasim bq08283 and Hammad Malik hm08298'''
# Generate 1000 samples of a Bernoulli Random Variable
p = 0.5 # Probability of Success
n = 1   # Number of states (When n = 1, Binomial Random Variable
becomes a Bernoulli Random Variable)
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Bernoulli Random Variable (p=0.5)")
plt.hist(X,bins='auto',rwidth=0.8)
plt.show()

# VARYING PROBABILITY OF SUCCESS

p = 0.25 # Probability of Success
n = 1   # Number of states (When n = 1, Binomial Random Variable
becomes a Bernoulli Random Variable)
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
```

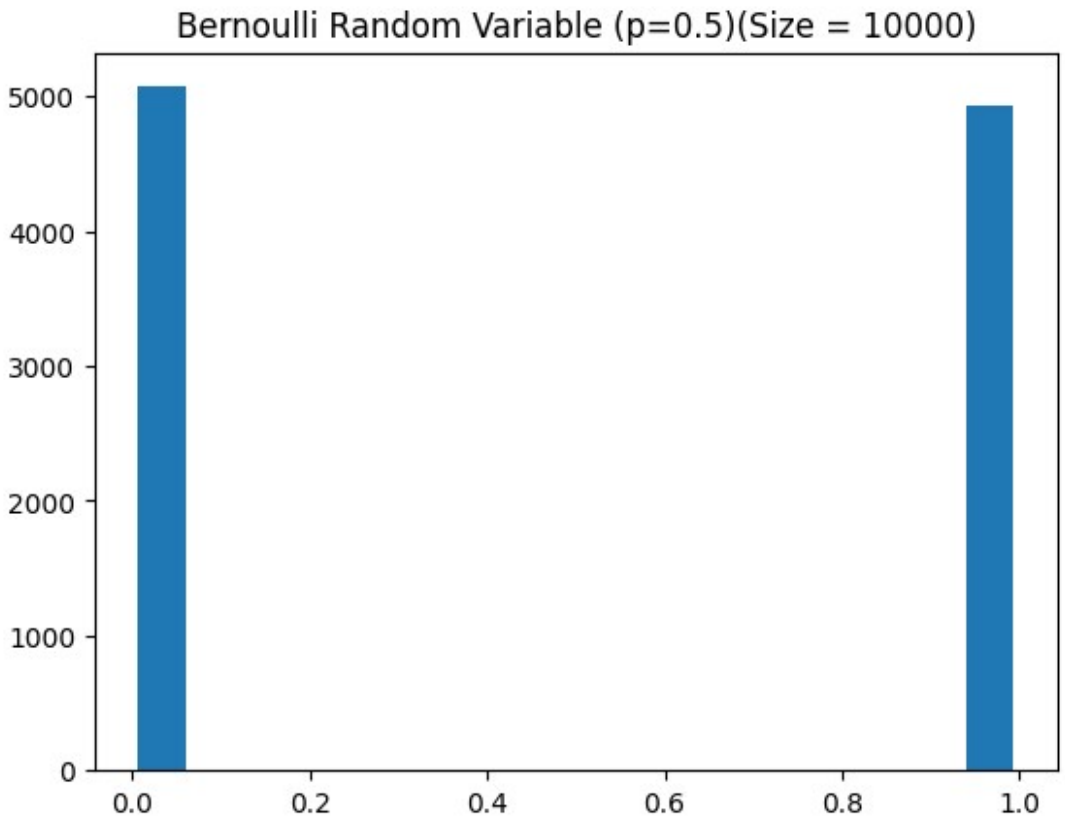
```
plt.title("Bernoulli Random Variable (p=0.25)(Size = 1000)")
plt.hist(X,bins='auto',rwidth=0.8)
plt.show()

p = 0.73 # Probability of Success
n = 1   # Number of states (When n = 1, Binomial Random Variable
becomes a Bernoulli Random Variable)
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Bernoulli Random Variable (p=0.73)(Size = 1000)")
plt.hist(X,bins='auto',rwidth=0.8)
plt.show()

# VARYING NUMBER OF TRIALS

p = 0.5 # Probability of Success
n = 1   # Number of states (When n = 1, Binomial Random Variable
becomes a Bernoulli Random Variable)
size  = 10 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Bernoulli Random Variable (p=0.5)(Size = 10)")
plt.hist(X,bins='auto',rwidth=0.8)
plt.show()

p = 0.5 # Probability of Success
n = 1   # Number of states (When n = 1, Binomial Random Variable
becomes a Bernoulli Random Variable)
size  = 10000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Bernoulli Random Variable (p=0.5)(Size = 10000)")
plt.hist(X,bins='auto',rwidth=0.8)
plt.show()
```


Bernoulli Random Variable (p=0.5)
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0


Bernoulli Random Variable (p=0.25)(Size = 1000)
700
600
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0


Bernoulli Random Variable (p=0.73)(Size = 1000)
700
600
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0


Bernoulli Random Variable (p=0.5)(Size = 10)
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0



Comments
'''Breeha Qasim bq08283 and Hammad Malik hm08298'''

A Bernoulli random variable is a discrete random variable that takes the value 1 with probability p and the value 0 with probability 1-p.

Varying the value of 'p':

For instance, when p=0.2, the random variable takes the value 1 with probability 0.2 and the value 0 with probability 0.8. If you adjust p to, for instance, 0.6, the random variable then has a 0.6 probability of being 1 and a 0.4 probability of being 0.

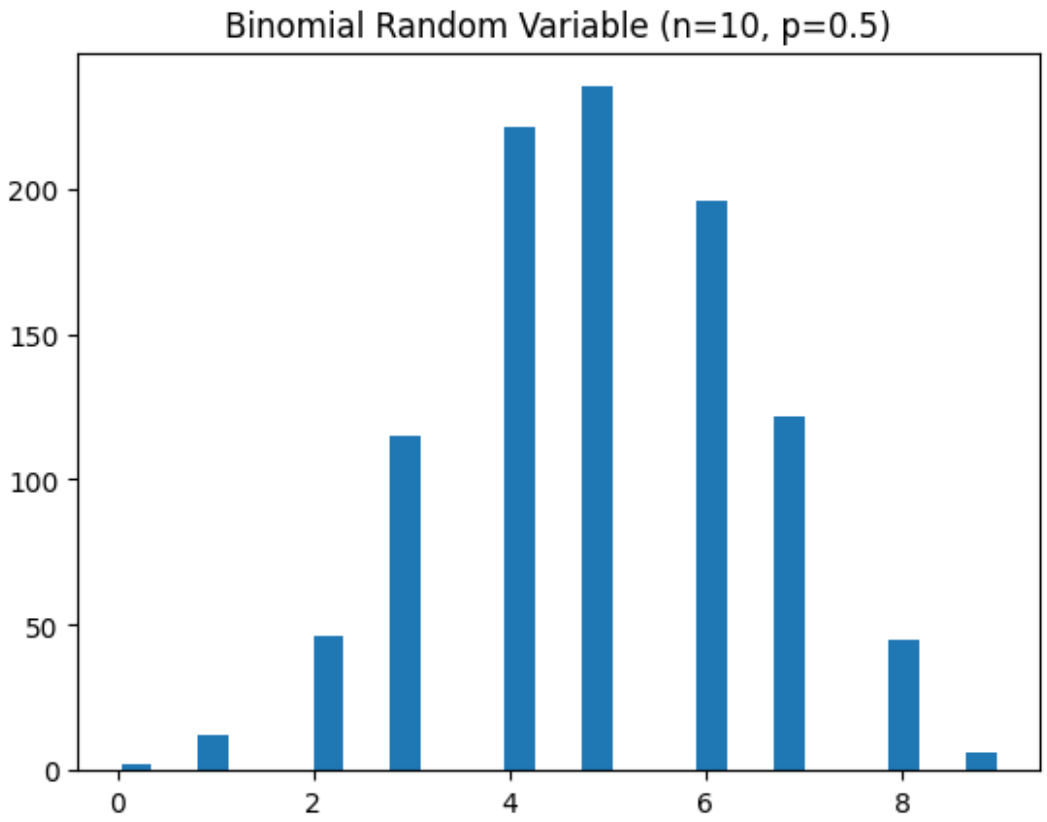
In simpler terms, when p increases, success becomes more likely while failure becomes less likely. This is because in a Bernoulli random variable, there's only one event with two outcomes: success or failure. So, if the chance of success rises, the event is more likely to succeed.

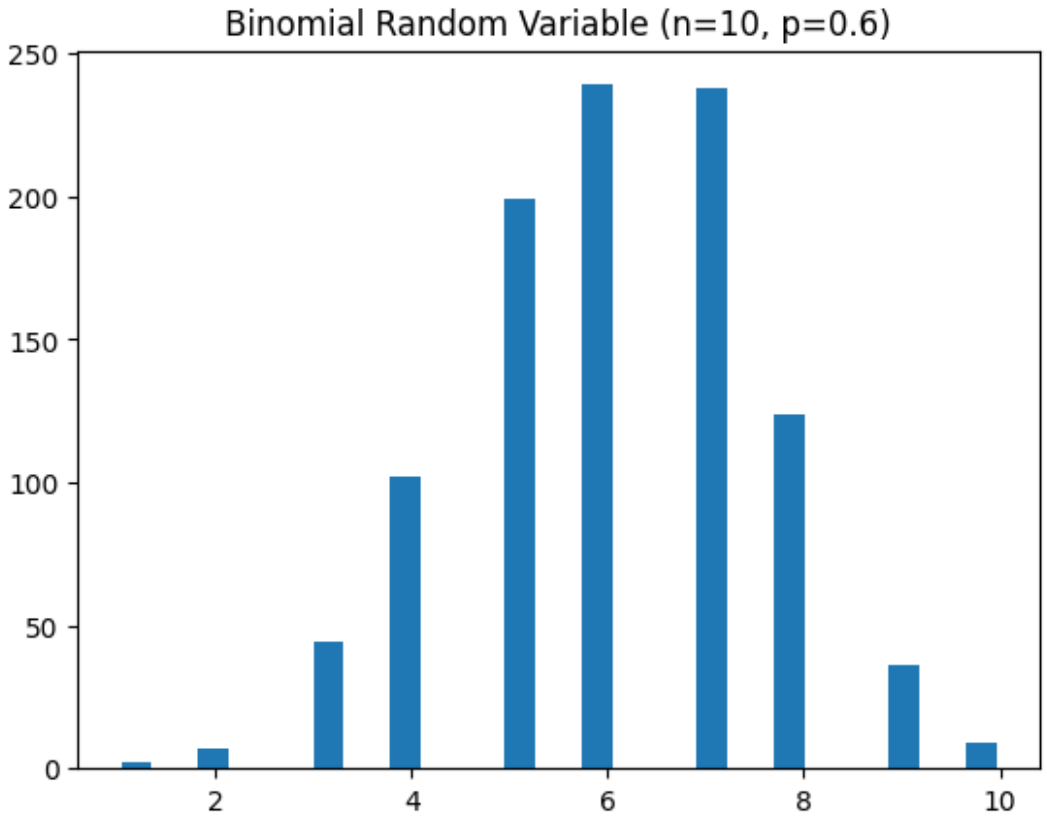
Varying the value of 'size':

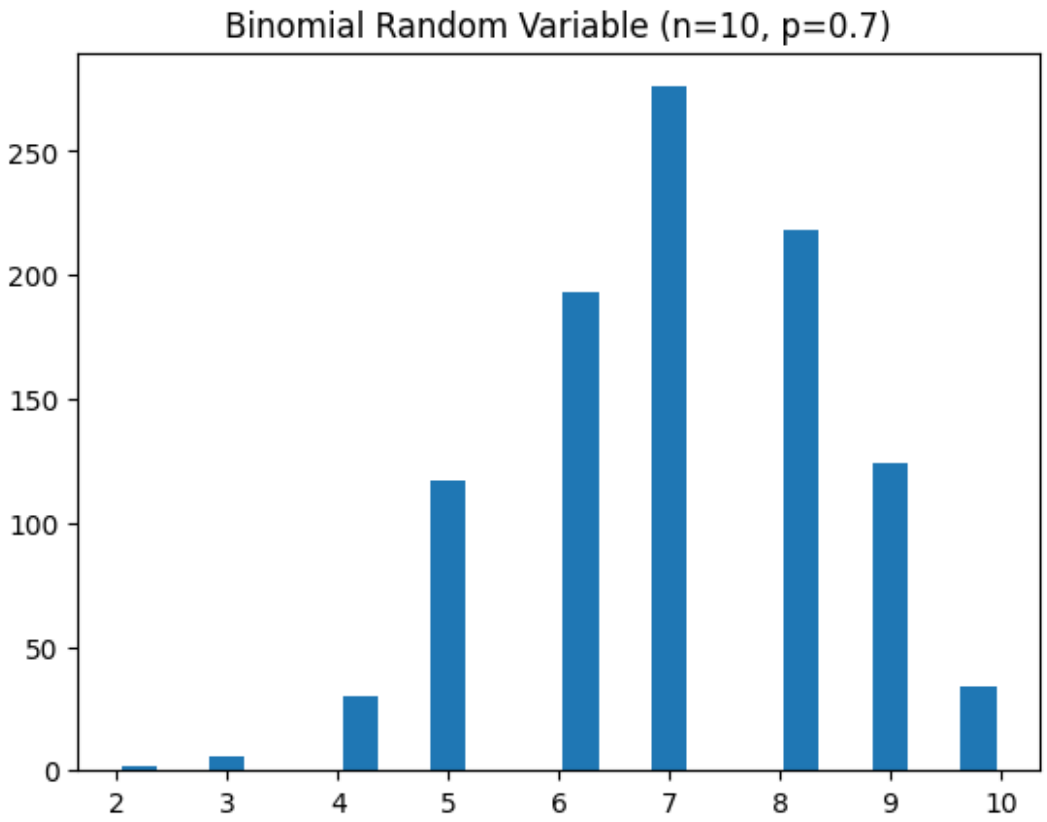
With small sample sizes, histograms may not accurately represent the true shape of a Bernoulli random variable's distribution, as they can be highly variable and misleading due to increased sampling error. Conversely, larger sample sizes lead to more stable histograms that closely resemble the actual distribution, as they provide more information and reduce the impact of random sampling variability.

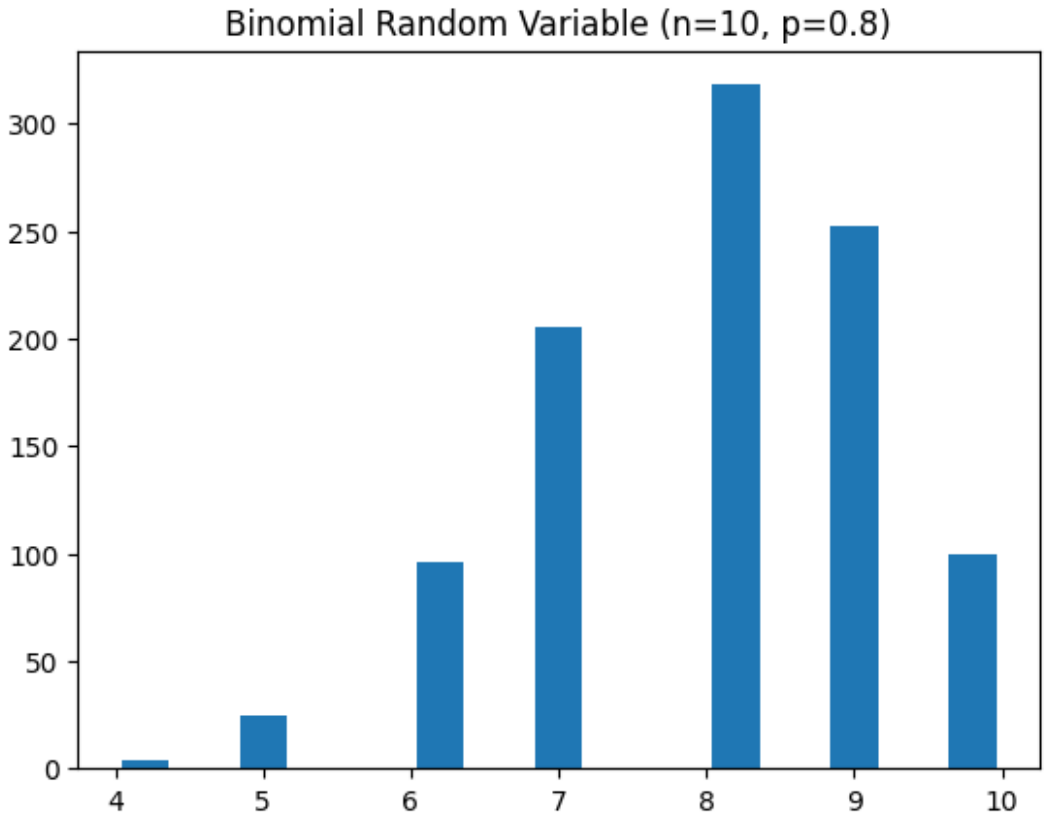
```
# BINOMIAL RANDOM VARIABLE
# Your code goes here
'''Breeha Qasim bq08283 and Hammad Malik hm08298'''
# Generate 1000 samples of a Binomial Random Variable
p = 0.5 # Probability of a single success
n = 10  # Number of states
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=10, p=0.5)")
plt.hist(X,bins="auto",rwidth=0.8);
plt.show()

# VARYING PROBABILITY OF SUCCESS

p = 0.6 # Probability of a single success
n = 10  # Number of states
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=10, p=0.6)")
plt.hist(X,bins="auto",rwidth=0.8);
plt.show()

p = 0.7 # Probability of a single success
n = 10  # Number of states
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=10, p=0.7)")
plt.hist(X,bins="auto",rwidth=0.8);
plt.show()

p = 0.8 # Probability of a single success
n = 10  # Number of states
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=10, p=0.8)")
plt.hist(X,bins="auto",rwidth=0.8);
plt.show()

# VARYING NUMBER OF STATES

p = 0.5 # Probability of a single success
n = 8  # Number of states
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=8, p=0.5)")
plt.hist(X,bins="auto",rwidth=0.8);
plt.show()

p = 0.5 # Probability of a single success
n = 9  # Number of states
```

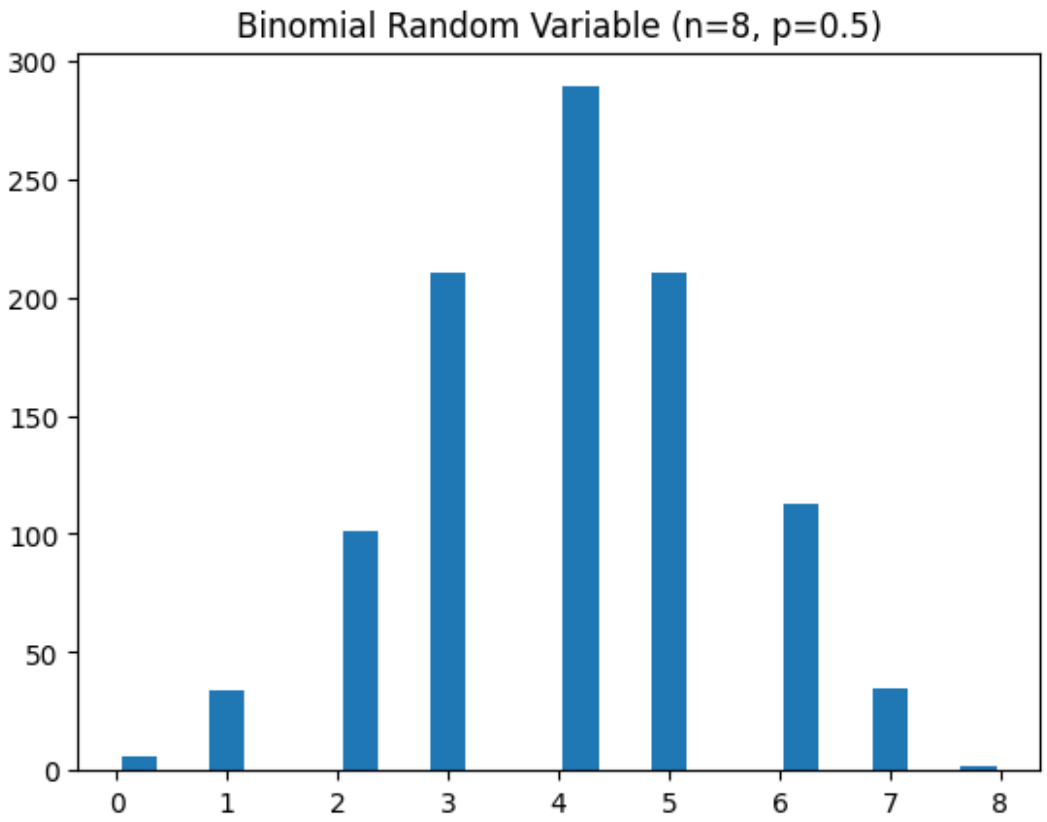
```
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=9, p=0.5)")
plt.hist(X,bins="auto",rwidth=0.8);
plt.show()

# VARYING NUMBER OF TRIALS

p = 0.5 # Probability of a single success
n = 10  # Number of states
size  = 10 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=10, p=0.5,size=10)")
plt.hist(X,bins="auto",rwidth=0.8);
plt.show()

p = 0.5 # Probability of a single success
n = 10  # Number of states
size  = 100 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=10, p=0.5,size=100)")
plt.hist(X,bins="auto",rwidth=0.8);
plt.show()

p = 0.5 # Probability of a single success
n = 10  # Number of states
size  = 10000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=10, p=0.5,size=10000)")
plt.hist(X,bins="auto",rwidth=0.8);
plt.show()
```

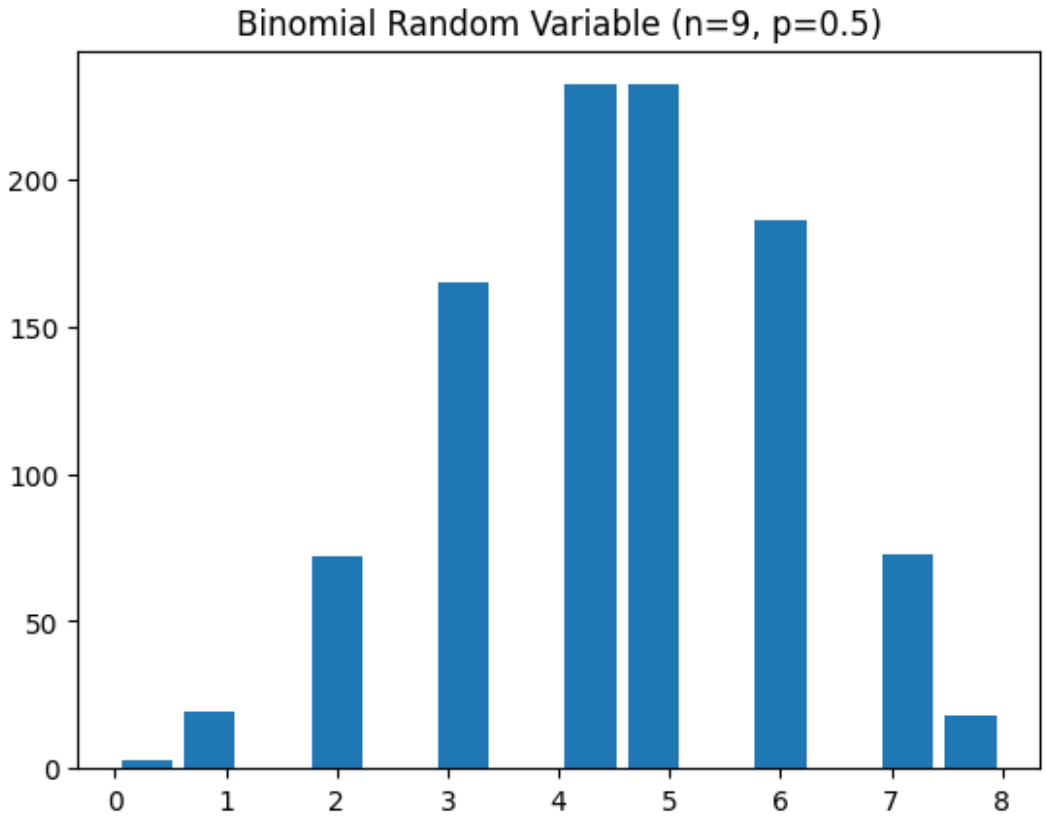

Binomial Random Variable (n=10, p=0.5)
0
50
100
150
200
0
2
4
6
8

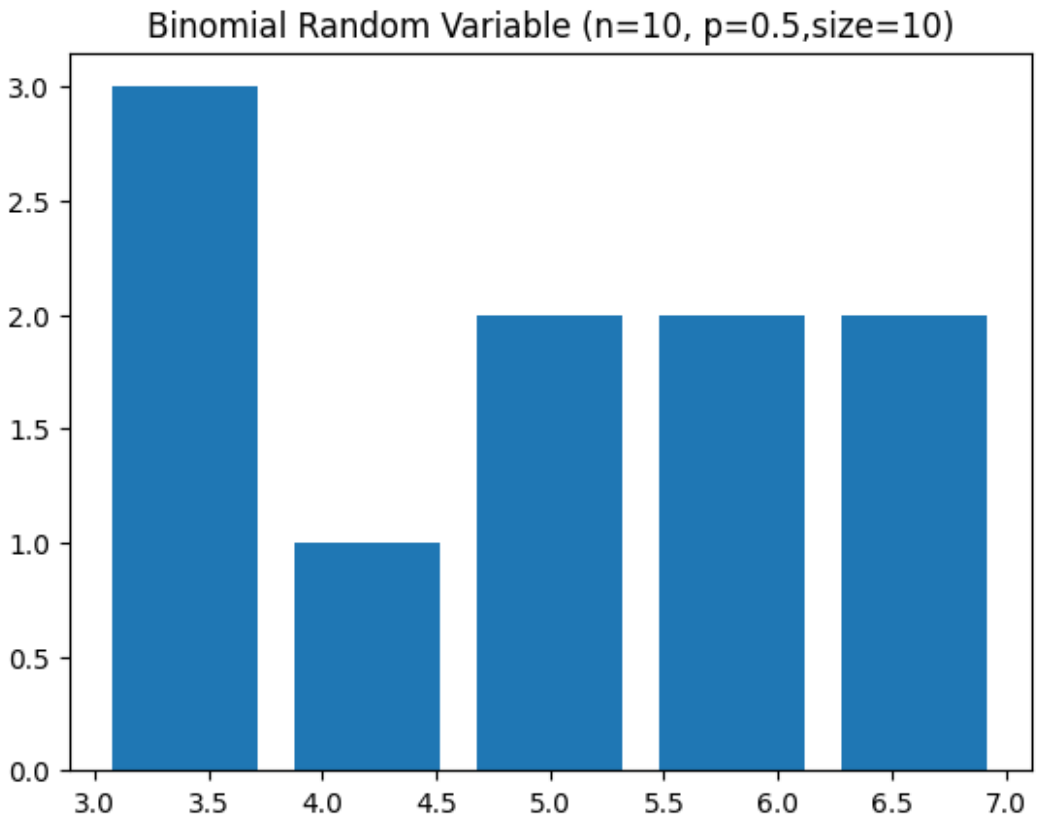

Binomial Random Variable (n=10, p=0.6)
0
50
100
150
200
250
2
4
6
8
10

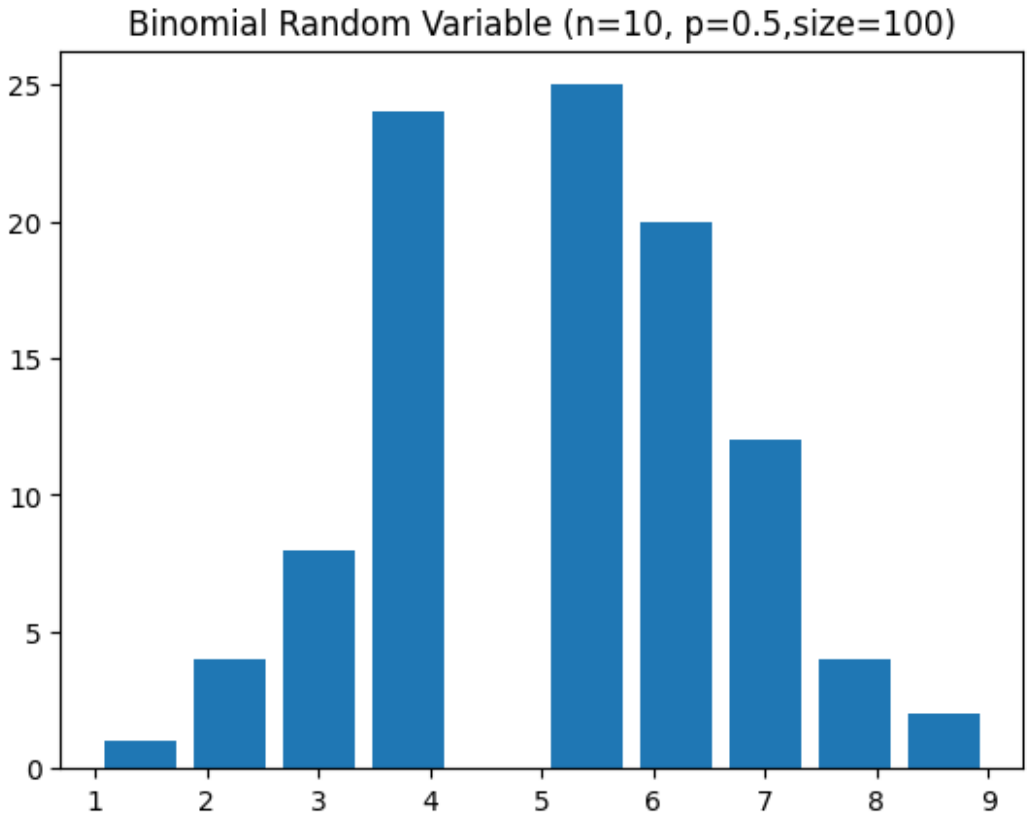

Binomial Random Variable (n=10, p=0.7)
250
200
150
100
50
0
2
3
4
5
6
7
8
9
10

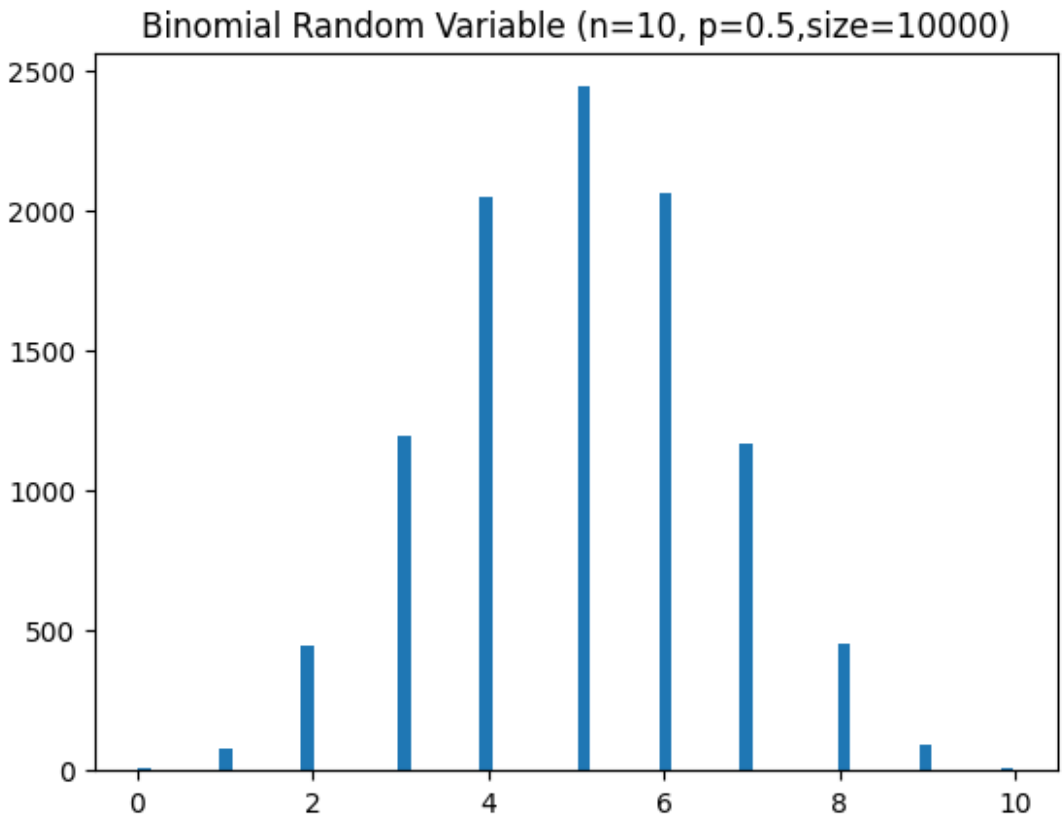

Binomial Random Variable (n=10, p=0.8)
300
250
200
150
100
50
0
4
5
6
7
8
9
10


Binomial Random Variable (n=8, p=0.5)
300
250
200
150
100
50
0
0
1
2
3
4
5
6
7
8


Binomial Random Variable (n=9, p=0.5)
200
150
100
50
0
0
1
2
3
4
5
6
7
8


Binomial Random Variable (n=10, p=0.5,size=10)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0


Binomial Random Variable (n=10, p=0.5,size=100)
25
20
15
10
5
0
1
2
3
4
5
6
7
8
9



Comments
'''Breeha Qasim bq08283 and Hammad Malik hm08298'''

Varying the value of 'p':

When P approaches 0.5, the distribution tends to be symmetrical with a bell-shaped histogram. This symmetry arises from nearly equal probabilities of success and failure, centering the distribution around the mean.

However, if P is significantly smaller than 0.5, the distribution skews rightward, evident in a longer tail on the right side of the histogram. Here, the low probability of success concentrates the distribution toward the left, extending the tail to the right.

Conversely, a P much larger than 0.5 leads to a left-skewed distribution, characterized by a longer left tail on the histogram. With high success probability, the distribution centers toward the right, extending the tail leftward.

Varying the value of 'n':

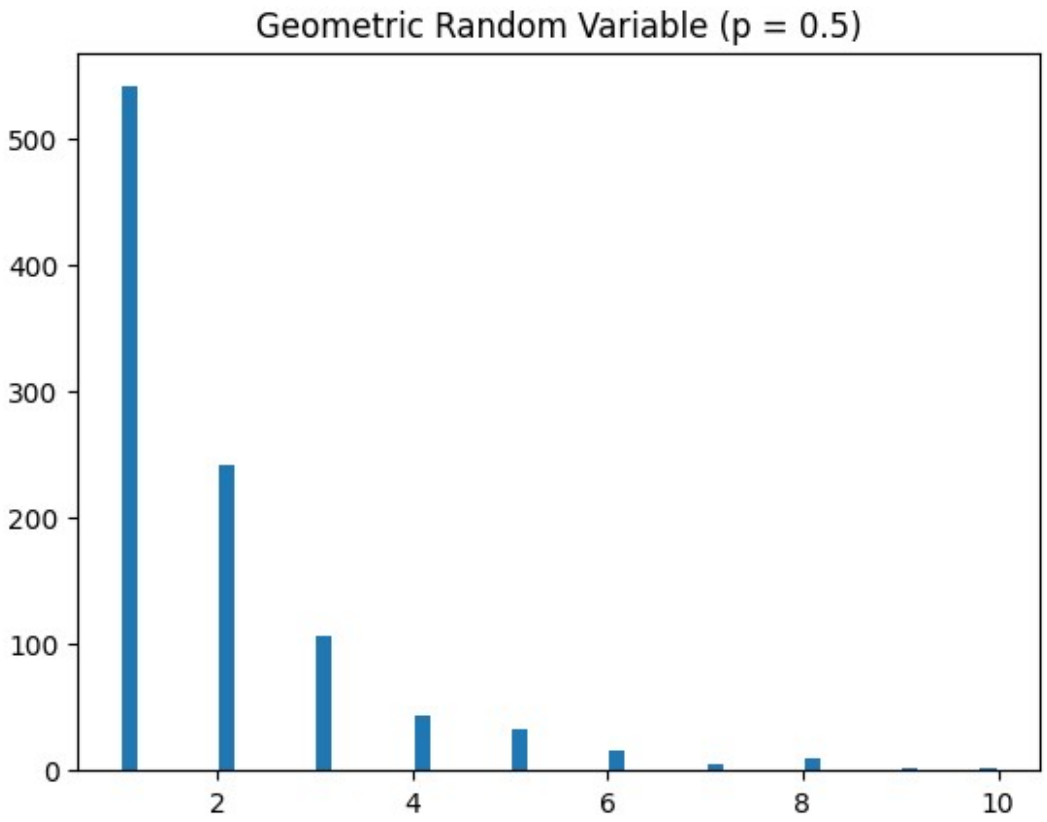
Increasing 'n' yields more histogram bins, offering a detailed view of the data with finer variations and features. Conversely, reducing 'n' results in fewer bins, presenting the data with less detail and a smoother histogram that downplays individual data points.

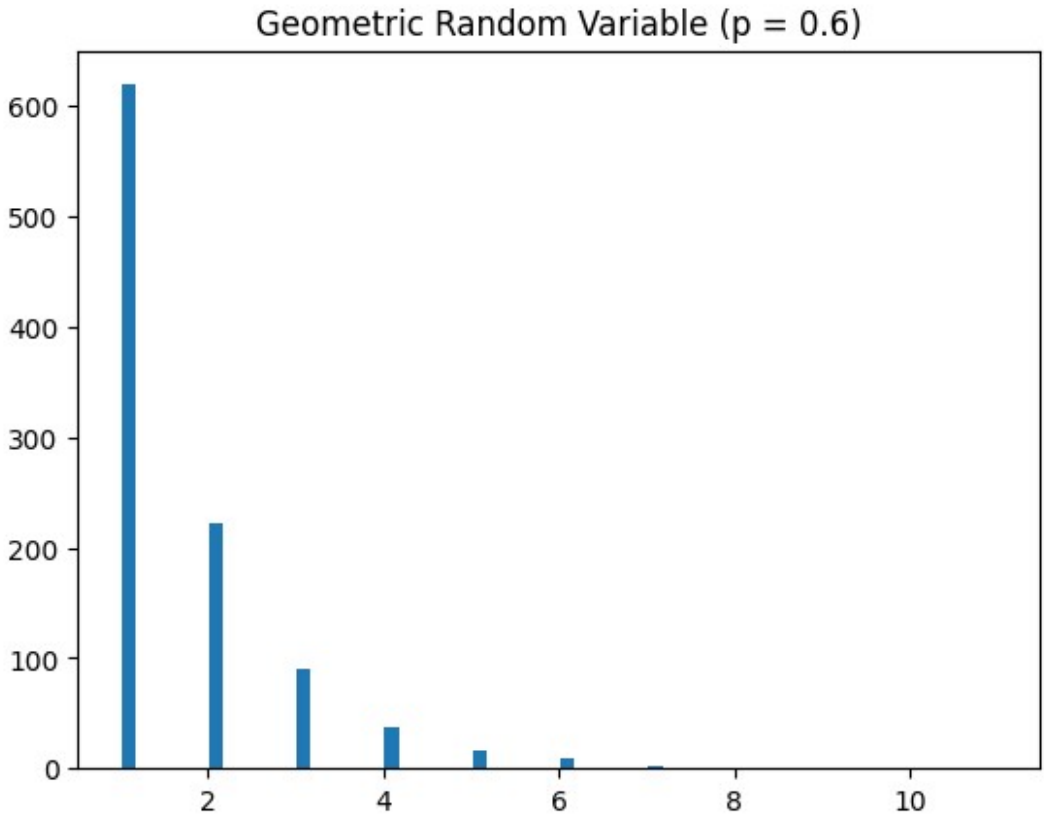
Varying the value of 'size':

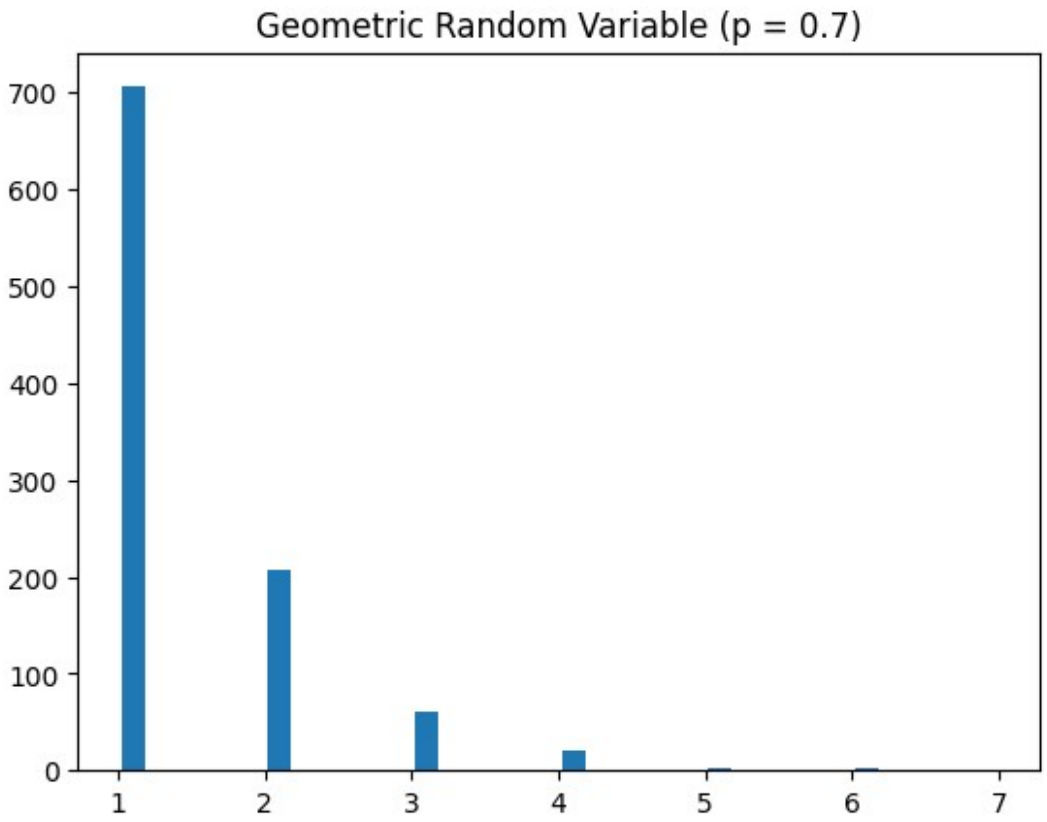
With increasing sample size, the histogram of the binomial random variable becomes smoother, symmetrical, and reflective of its central tendency. Larger samples provide more accurate estimates of the true distribution.

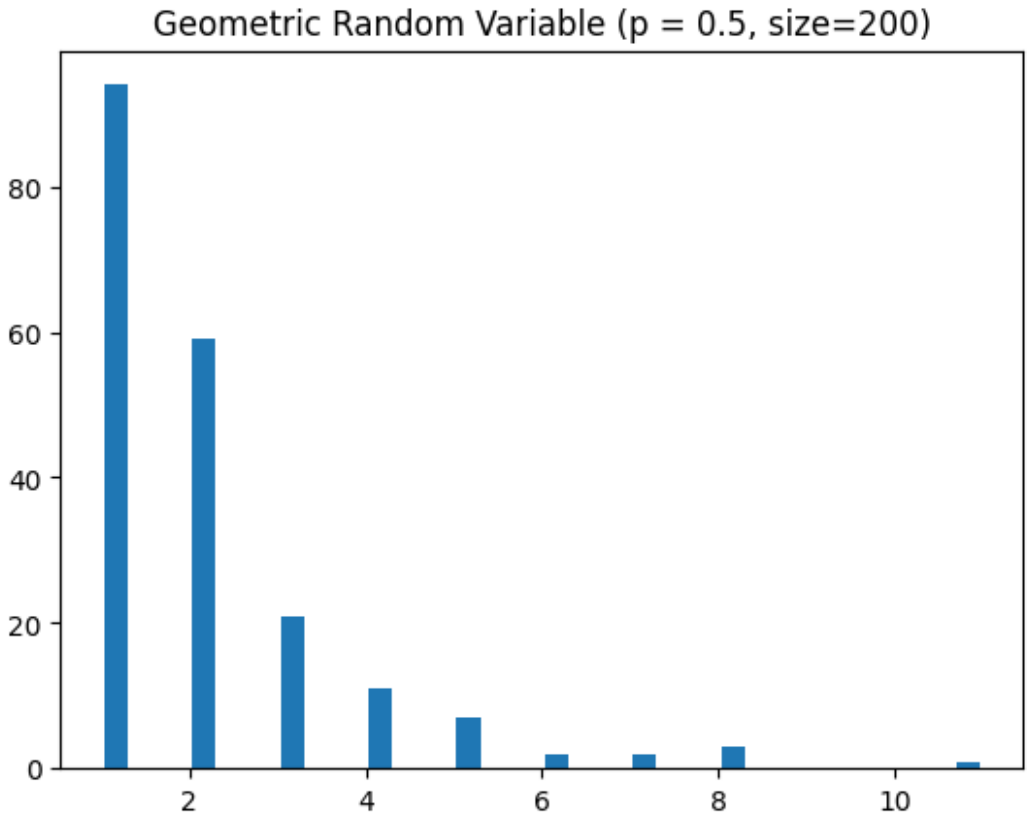
Conversely, smaller sample sizes may produce more variable and skewed histograms, influenced by the specific data sample obtained. Smaller samples are more likely to deviate from the true distribution of the binomial random variable.

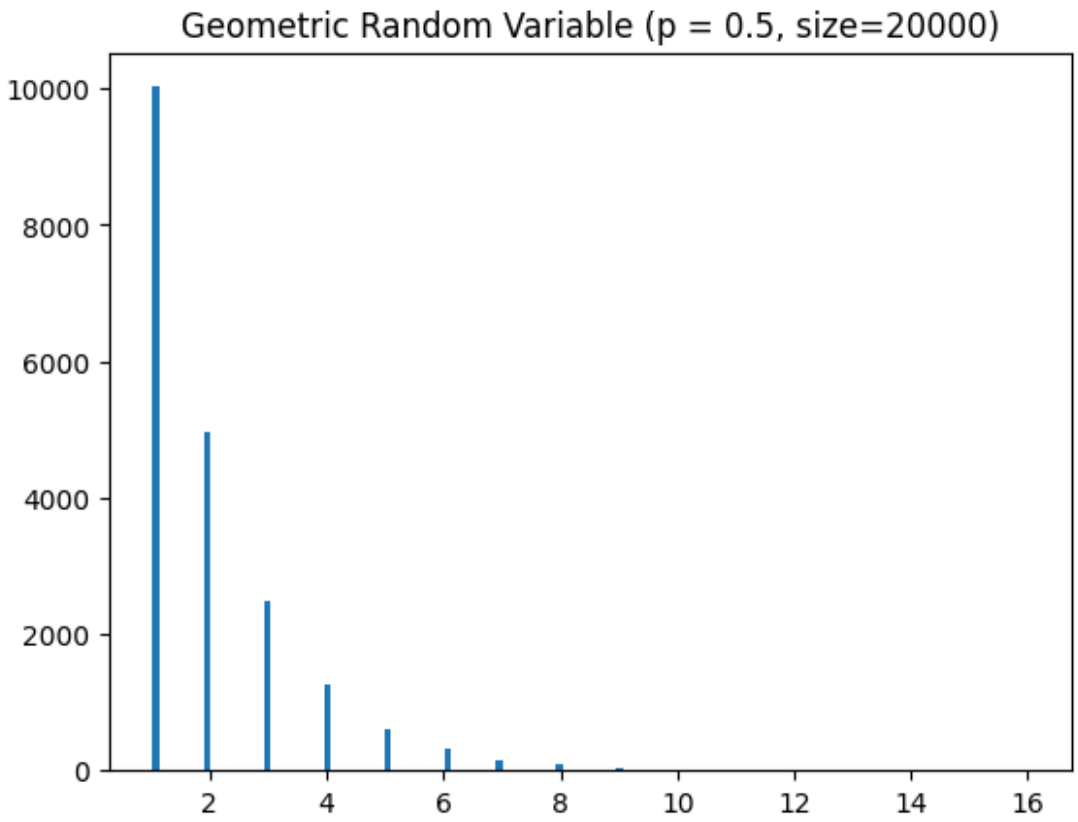
```
# GEOMETRIC RANDOM VARIABLE
# Your code goes here
'''Breeha Qasim bq08283 and Hammad Malik hm08298'''
# Generate 1000 samples of a Geometric Random variables
p = 0.5
size = 1000
plt.title("Geometric Random Variable (p = 0.5)")
X = np.random.geometric(p,size=size)
plt.hist(X,bins="auto",rwidth=0.8)
plt.show()

# VARYING PROBABILITY OF SUCCESS

p = 0.6
size = 1000
plt.title("Geometric Random Variable (p = 0.6)")
X = np.random.geometric(p,size=size)
plt.hist(X,bins="auto",rwidth=0.8)
plt.show()

p = 0.7
size = 1000
plt.title("Geometric Random Variable (p = 0.7)")
X = np.random.geometric(p,size=size)
plt.hist(X,bins="auto",rwidth=0.8)
plt.show()

# VARYING NUMBER OF TRIALS

p = 0.5
size = 200
plt.title("Geometric Random Variable (p = 0.5, size=200)")
X = np.random.geometric(p,size=size)
plt.hist(X,bins="auto",rwidth=0.8)
plt.show()

p = 0.5
size = 20000
plt.title("Geometric Random Variable (p = 0.5, size=20000)")
X = np.random.geometric(p,size=size)
plt.hist(X,bins="auto",rwidth=0.8)
plt.show()
```


Geometric Random Variable (p = 0.5)
500
400
300
200
100
0
2
4
6
8
10


Geometric Random Variable (p = 0.6)
600
500
400
300
200
100
0
2
4
6
8
10


Geometric Random Variable (p = 0.7)
700
600
500
400
300
200
100
0
1
2
3
4
5
6
7


Geometric Random Variable (p = 0.5, size=200)
80
60
40
20
0
2
4
6
8
10



Comments
'''Breeha Qasim bq08283 and Hammad Malik hm08298'''

Varying the value of 'p':

When p is small, indicating a low likelihood of success in a single trial, it requires more trials on average to achieve the first success, resulting in a higher expected value for the geometric random variable. Conversely, if p is large, signifying a higher chance of success in a single trial, fewer trials are typically needed to attain the first success.

Decreasing p shifts the histogram of the geometric random variable to the right, indicating an increase in the average number of trials required for the first success. This adjustment may result in a narrower and taller peak, along with a wider spread of the distribution. Conversely, increasing p shifts the histogram to the left, potentially leading to a wider and shorter peak, and a narrower spread of the distribution.

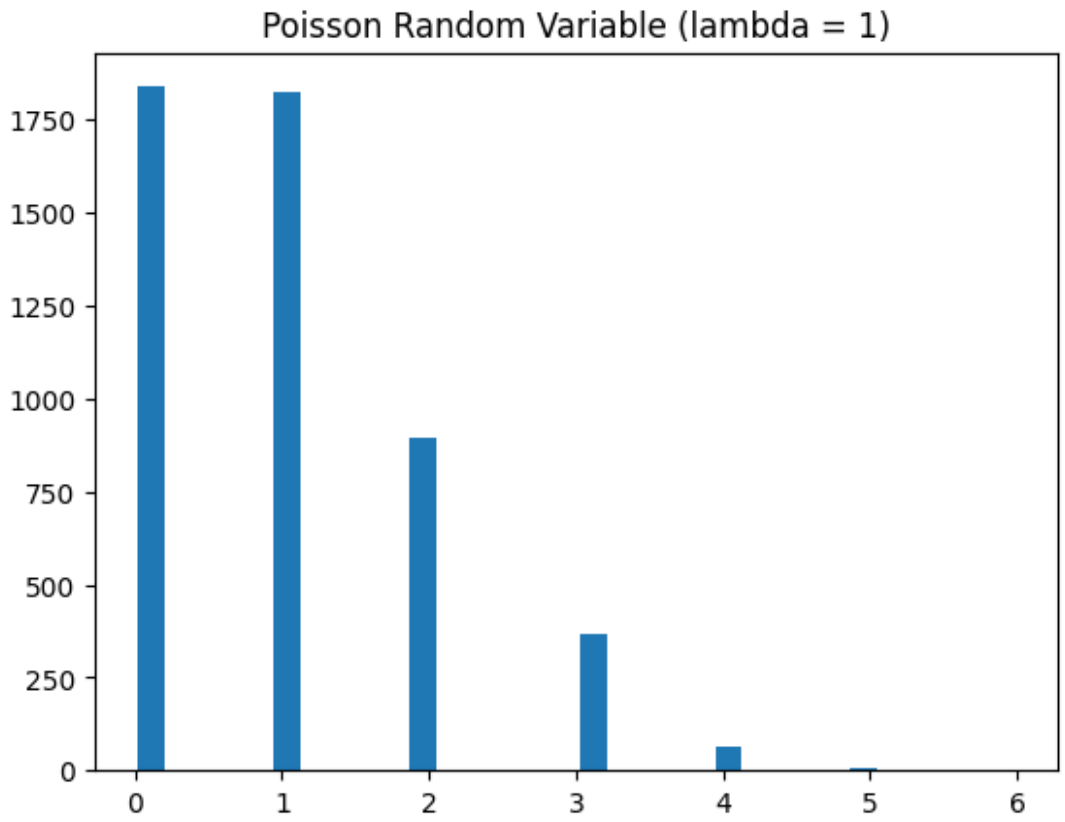
Varying the value of 'size':

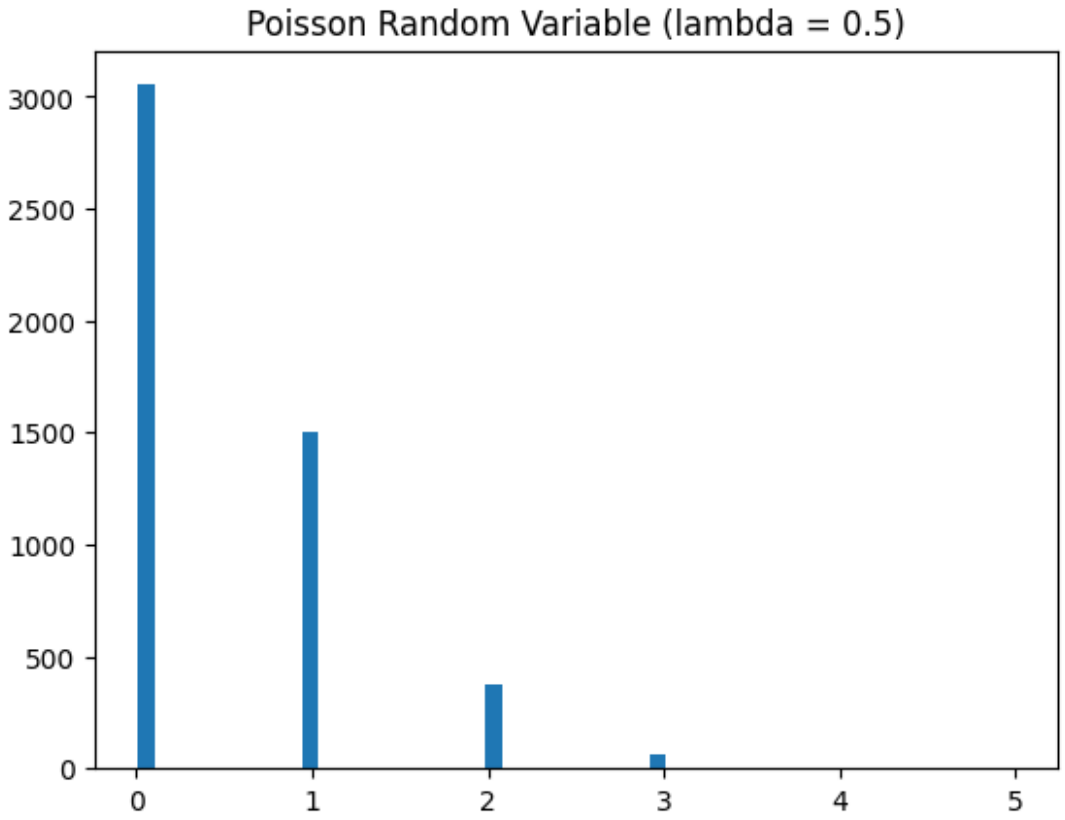
As the sample size expands, the histogram becomes smoother and more symmetric, revealing a clearer central tendency of the distribution. This enhancement is due to the increased accuracy in estimating the true distribution of the geometric random variable with larger sample sizes.

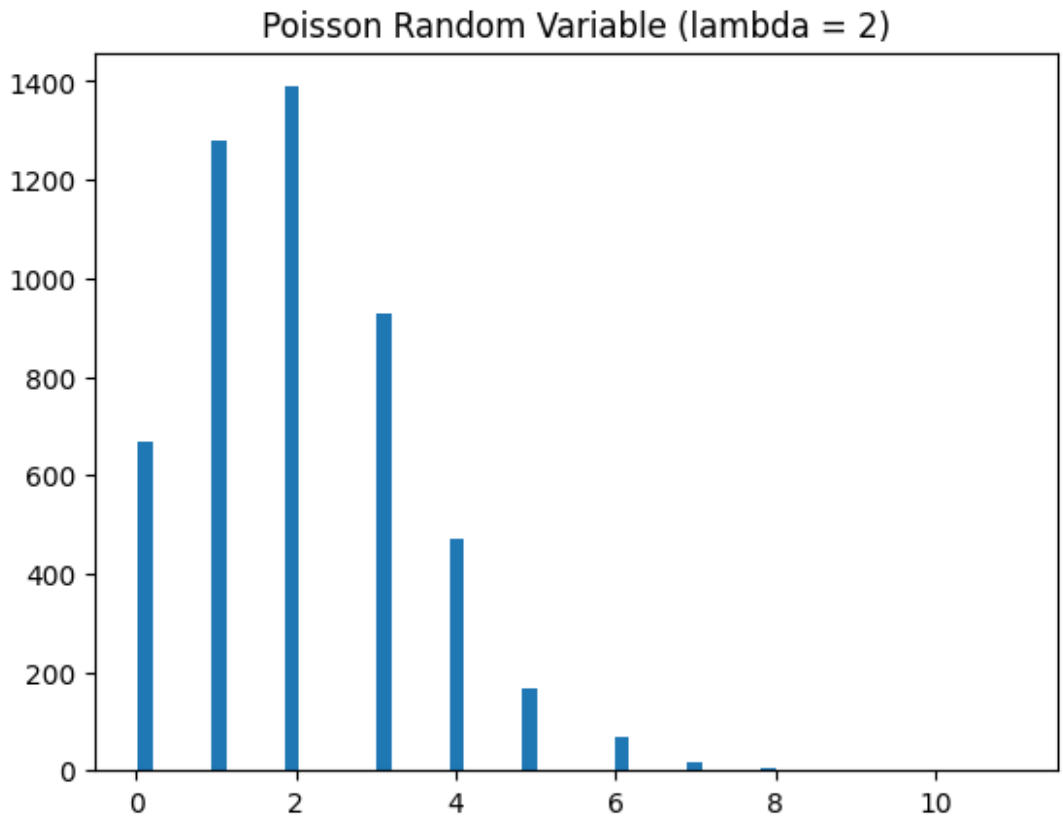
In contrast, smaller sample sizes yield histograms that are more variable and skewed, influenced by the specific sample data obtained. With smaller sample sizes, there is a greater likelihood of the histogram deviating from the true distribution of the geometric random variable.

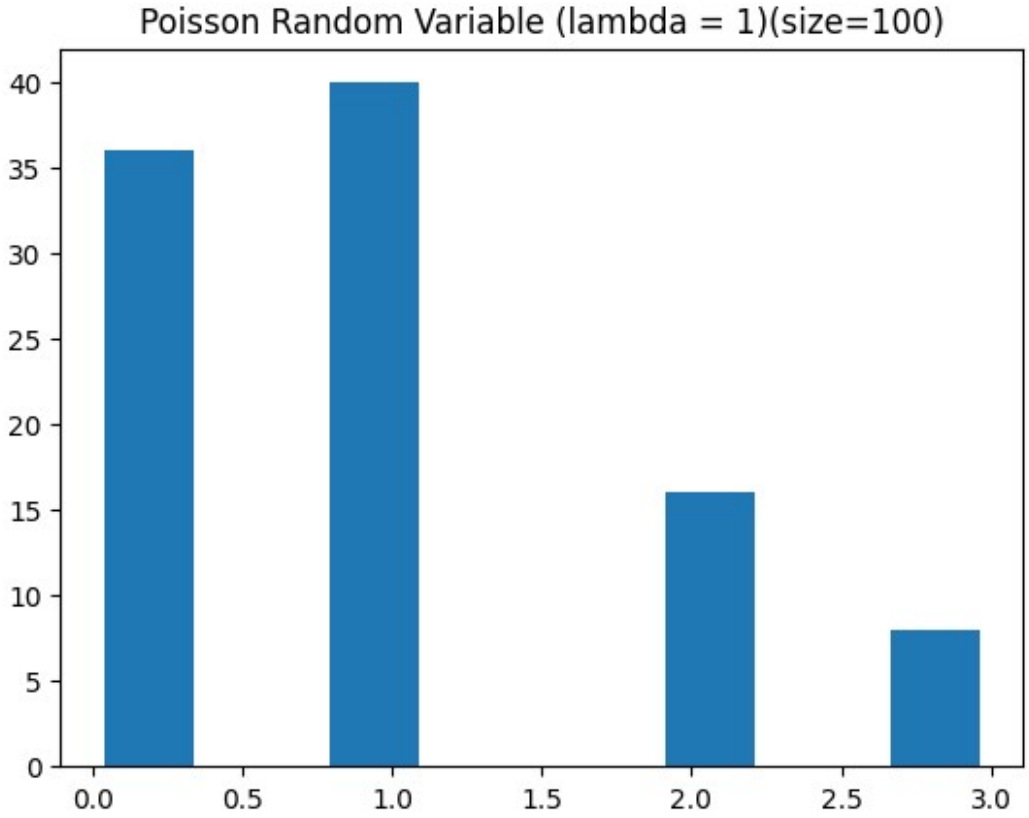
Consequently, sample size significantly impacts the accuracy and precision of the histogram, with larger samples generally producing more representative and precise histograms.

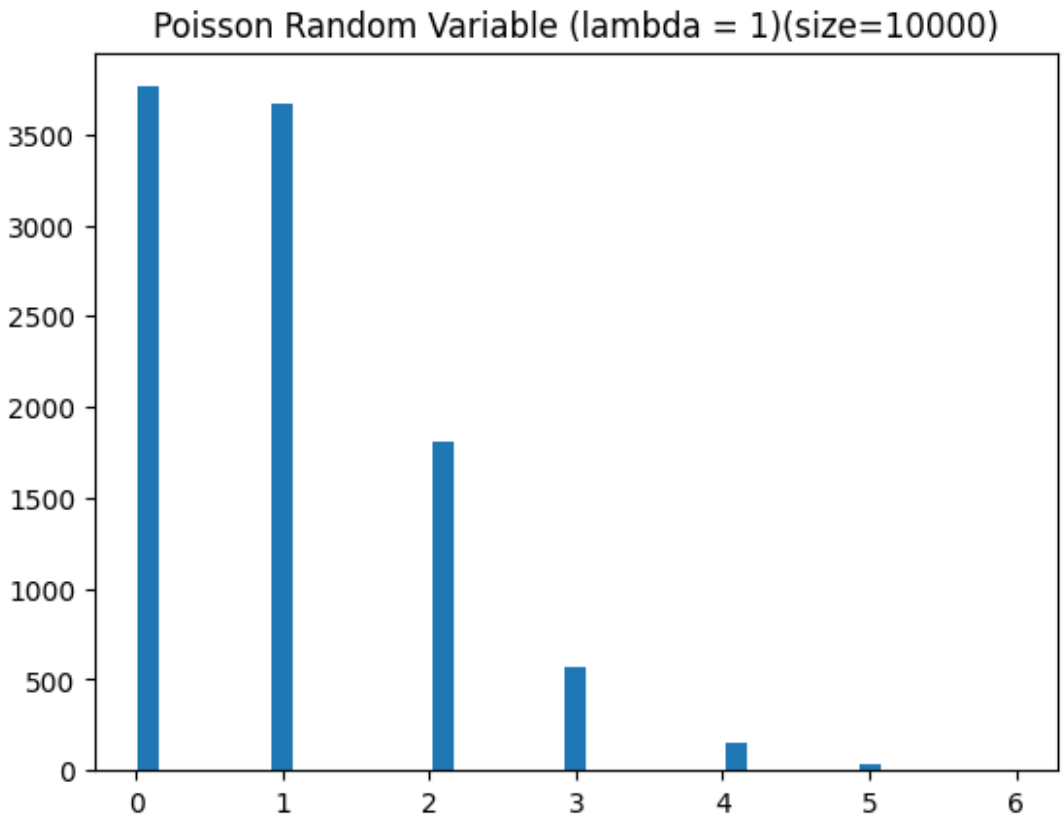
```
# POISSON RANDOM VARIABLE
# Your code goes here
'''Breeha Qasim bq08283 and Hammad Malik hm08298'''

# Generate 5000 samples of a Poisson Random Variable
lambd = 1
X = np.random.poisson(lambd,size=5000)
plt.title("Poisson Random Variable (lambda = 1)")
plt.hist(X,bins="auto",rwidth=0.8)
plt.show()

# VARYING VALUE OF LAMBD

lambd = 0.5
X = np.random.poisson(lambd,size=5000)
plt.title("Poisson Random Variable (lambda = 0.5)")
plt.hist(X,bins="auto",rwidth=0.8)
plt.show()

lambd = 2
X = np.random.poisson(lambd,size=5000)
plt.title("Poisson Random Variable (lambda = 2)")
plt.hist(X,bins="auto",rwidth=0.8)
plt.show()

# VARYING NUMBER OF TRIALS

lambd = 1
X = np.random.poisson(lambd,size=100)
plt.title("Poisson Random Variable (lambda = 1)(size=100)")
plt.hist(X,bins="auto",rwidth=0.8)
plt.show()

lambd = 1
X = np.random.poisson(lambd,size=10000)
plt.title("Poisson Random Variable (lambda = 1)(size=10000)")
plt.hist(X,bins="auto",rwidth=0.8)
plt.show()
```


Poisson Random Variable (lambda = 1)
1750
1500
1250
1000
750
500
250
0
0
1
2
3
4
5
6


Poisson Random Variable (lambda = 0.5)
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5


Poisson Random Variable (lambda = 2)
1400
1200
1000
800
600
400
200
0
0
2
4
6
8
10


Poisson Random Variable (lambda = 1)(size=100)
40
35
30
25
20
15
10
5
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0



Comments
'''Breeha Qasim bq08283 and Hammad Malik hm08298'''

Varying the value of 'lambd':

When λ decreases, the histogram of the Poisson random variable shifts to the left, indicating a reduction in the average number of events observed within a fixed interval. This shift may result in a narrower and taller peak in the distribution, accompanied by a decrease in its spread. Conversely, increasing λ shifts the histogram to the right, potentially widening and shortening the peak while enlarging the spread of the distribution.

Varying the value of 'size':

Altering the size parameter does not directly impact the Poisson random variable. However, larger values of size lead to a closer approximation of a normal or theoretical distribution compared to smaller values.

[20 Marks] Task 3 : Mathematical Model for the Number of Maiden Overs in an ODI Cricket Match

For Task 3, you have been provided a CSV file (ODIData.csv) that contains some information about all the One Day International (ODI) cricket matches played in the 21st century. An ODI cricket match consists of 100 overs (divided into two innings of 50 overs each). A maiden over is an over in which no run is scored. In this task, you need to develop a mathematical model for

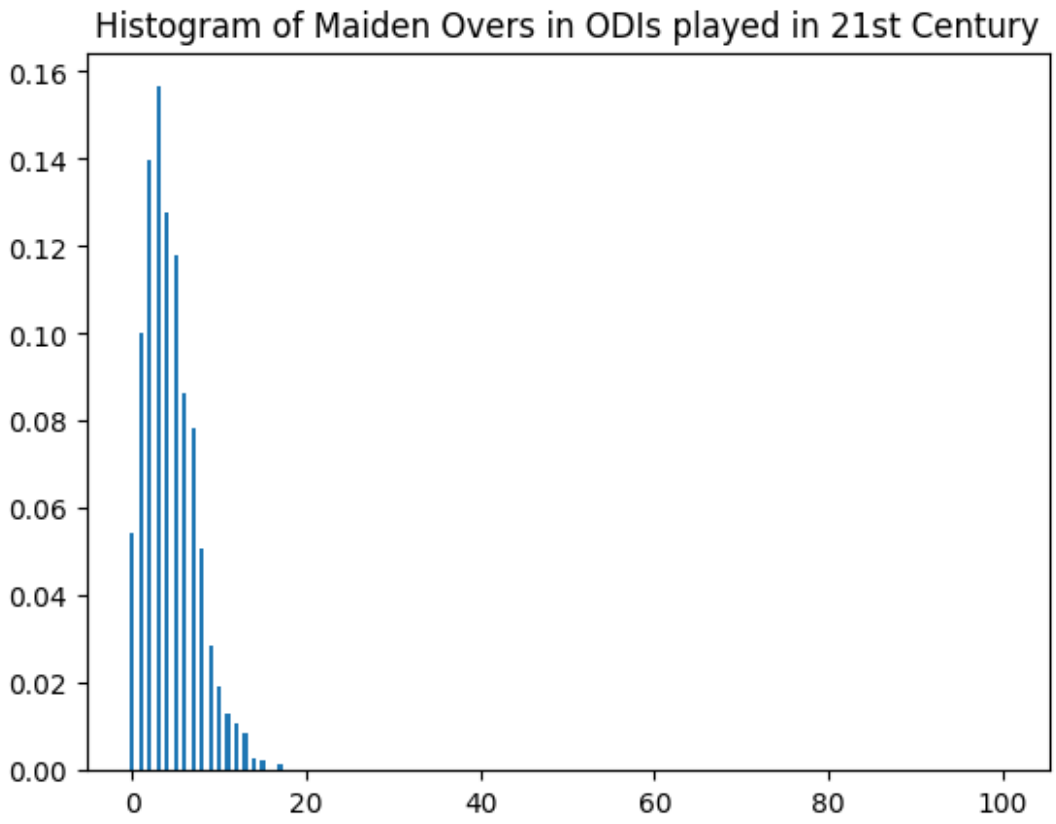
number of maiden overs in an ODI cricket match using the probability terminology discussed in class.

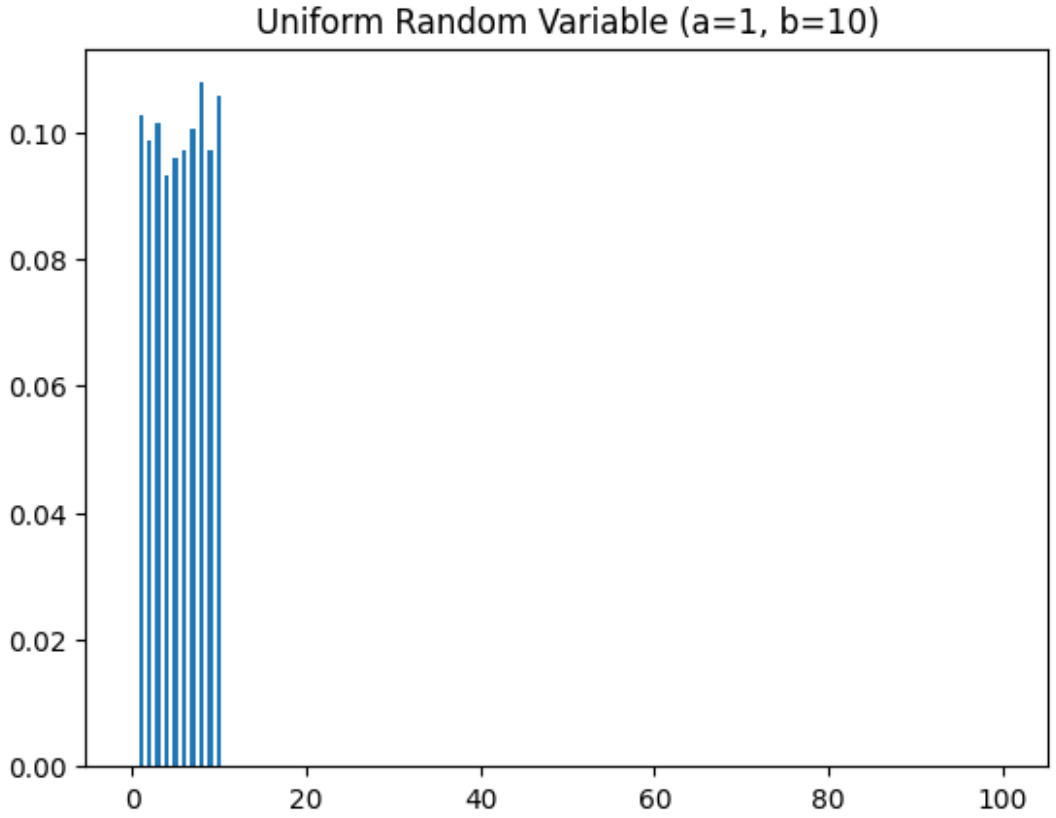
Let's model the number of maiden overs in an ODI cricket match next week as a random variable X. As part of this modeling exercise, what will you choose as the PMF of X? You can choose from various possible PMFs of commmon DRVs in the last section.

However, you have to justify your answer based on the historical data available in ODIData.csv file. In particular, justify your answer by calculating the "error" between the histogram of maiden overs in the past matches with the histogram of the data generated by your proposed PMF of X. The "error" is defined as the sum of the absolute difference between the corresponding values in the histogram.

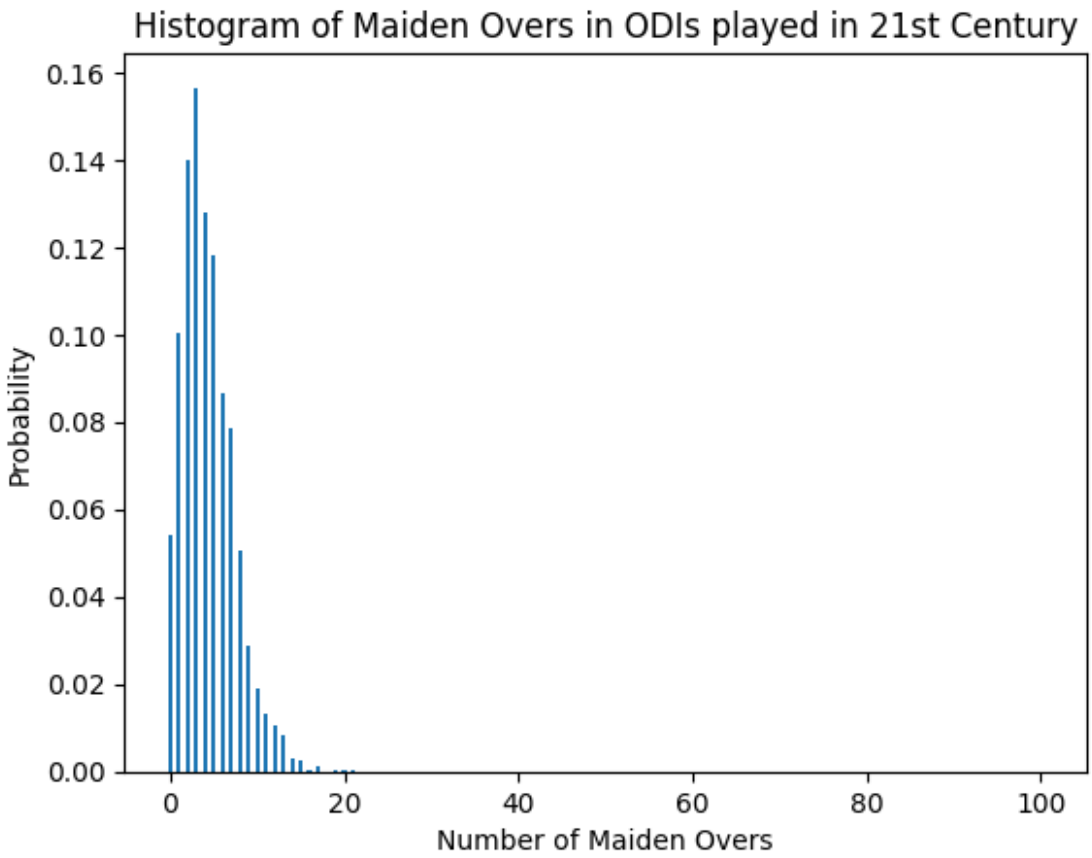
For example, the code snippet given below does the following: 1) plots the histogram of maiden overs in ODIs based on the given data 2) plots the histogram of the data generated by a uniform RV with parameter a=1 and b=10. 3) calculates the "error" between the historical data and the proposed PMF

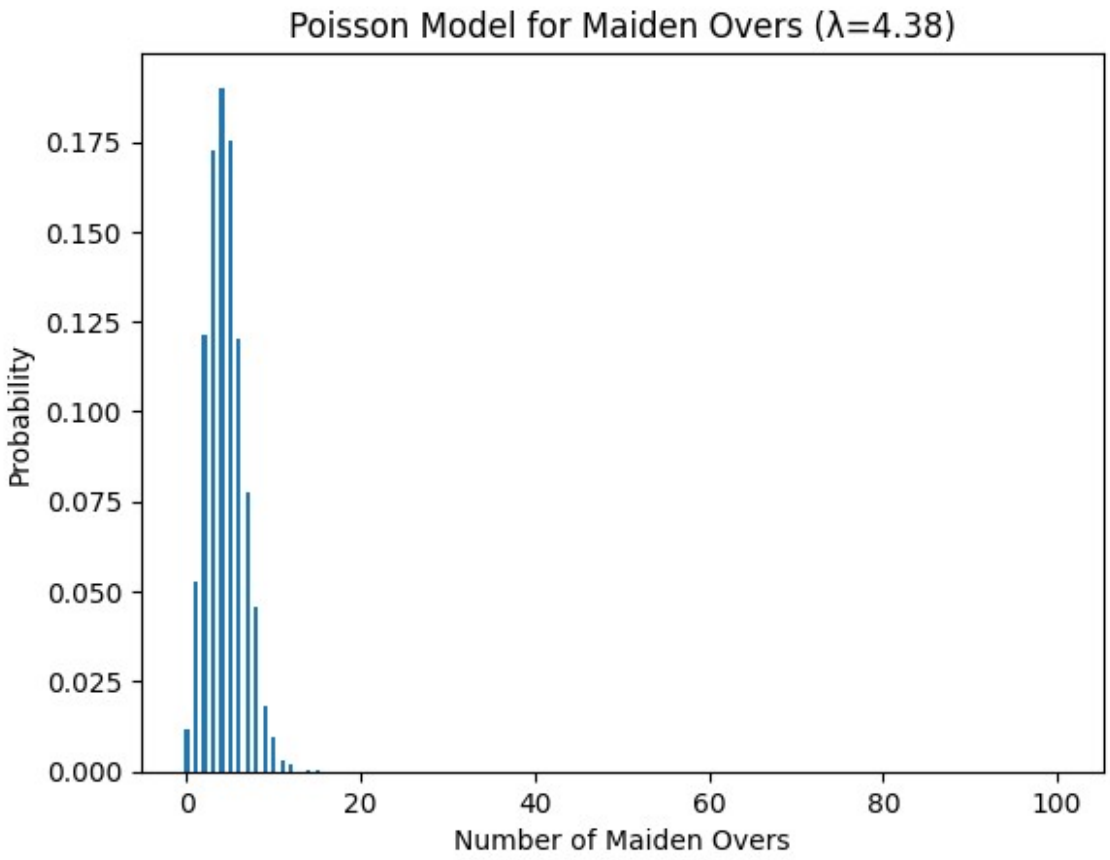
```
odi_data_df = pd.read_csv('ODIData.csv')
number_of_maiden_overs = np.array((odi_data_df['Mdns']))
over_bins = np.arange(102)

(bin_values_historical_data, bins, patches) =
plt.hist(number_of_maiden_overs, bins=over_bins, rwidth = 0.5,
density=True, align='left')
plt.title('Histogram of Maiden Overs in ODIs played in 21st Century')
plt.show()

# Generate 4000 samples of a Uniform Random Variable with a = 1, b =
10.
a = 1 # parameter a of uniform RV
b = 10  # parameter b of uniform RV
size  = 4000 # Number of trials
X = np.random.randint(low=a,high=b+1, size=size)
plt.title("Uniform Random Variable (a=1, b=10)")
(bin_values_pmf_data, bins, patches) = plt.hist(X,bins=over_bins,
rwidth = 0.5, density=True, align='left' );
plt.show()

difference_in_bin_values = bin_values_historical_data -
bin_values_pmf_data
error = np.sum(abs(difference_in_bin_values))
print("Error when the Proposed PMF is Unifrom with parameter a=1 and
b=10: " + str(error))
```


Histogram of Maiden Overs in ODIs played in 21st Century
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80
100


Uniform Random Variable (a=1, b=10)
0.10
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80
100

```
Error when the Proposed PMF is Unifrom with parameter a=1 and b=10:
0.49411345852895144

# Mathematical Model of Number of Maiden Overs in an ODI Cricket Match
# Your code goes here
'''Breeha Qasim bq08283 and Hammad Malik hm08298'''
# Load the ODI match data
odi_data_df = pd.read_csv('ODIData.csv')

# Extract the number of maiden overs data
number_of_maiden_overs = np.array(odi_data_df['Mdns'])
over_bins = np.arange(102)  # Assuming a reasonable maximum number of
maiden overs

# Plot the histogram of historical maiden overs data
plt.hist(number_of_maiden_overs, bins=over_bins, rwidth=0.5,
density=True, align='left')
plt.title('Histogram of Maiden Overs in ODIs played in 21st Century')
plt.xlabel('Number of Maiden Overs')
plt.ylabel('Probability')
plt.show()

# Estimate the parameter lambda (λ) of the Poisson distribution as the
mean of historical maiden overs
# Generate the Poisson distributed data based on the estimated λ
lambd = np.mean(number_of_maiden_overs)
X = np.random.poisson(lambd,size=4000)
# poisson_data = np.random.poisson(lambda_estimate, size=size)

# Plot the histogram of the Poisson model
plt.hist(X, bins=over_bins, rwidth=0.5, density=True, align='left')
plt.title("Poisson Model for Maiden Overs (λ={:.2f})".format(lambd))
plt.xlabel('Number of Maiden Overs')
plt.ylabel('Probability')
plt.show()

# Calculate the error between the historical and Poisson model
histograms
(hist_values_historical, _) = np.histogram(number_of_maiden_overs,
bins=over_bins, density=True)
(hist_values_poisson, _) = np.histogram(X, bins=over_bins,
density=True)

difference_in_bin_values = hist_values_historical -
hist_values_poisson
error = np.sum(np.abs(difference_in_bin_values))
print("Error when the Proposed PMF is Poisson with λ={:.2f}:
{}".format(lambd, error))
```


Histogram of Maiden Overs in ODIs played in 21st Century
Probability
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80
100
Number of Maiden Overs




```
Error when the Proposed PMF is Poisson with λ=4.38: 0.338848200312989
```

Comments '''Breeha Qasim bq08283 and Hammad Malik hm08298'''

The Poisson distribution is the most appropriate discrete random variable for modeling the number of maiden overs in a One Day International (ODI) match. Analysis of historical data reveals a mean close to 4, with the distribution of values resembling the bell-shaped curve characteristic of a Poisson distribution when the value of n (number of trials) is large. The binomial distribution was deemed unsuitable due to the variable number of overs in each match, preventing a consistent n. Furthermore, the number of matches analyzed was significantly large. The geometric distribution also proved to be an inadequate model since it focuses on the count of overs bowled until the occurrence of the first maiden over, which does not align with our modeling goals.