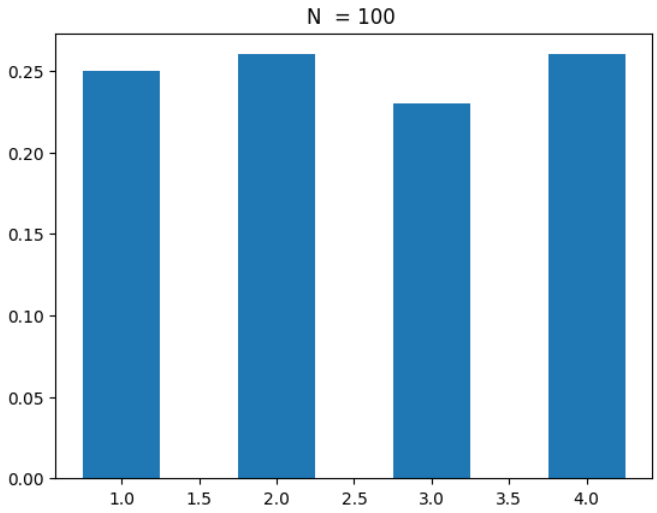
In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[15 Marks] Task 1 : Plot Histogram

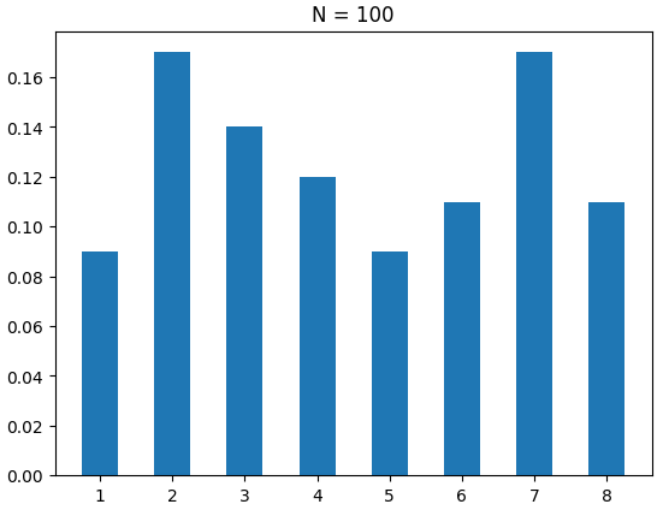
The following code simulates the rolling of a fair 4-sided die 100 times and plots the histogram of the outputs.

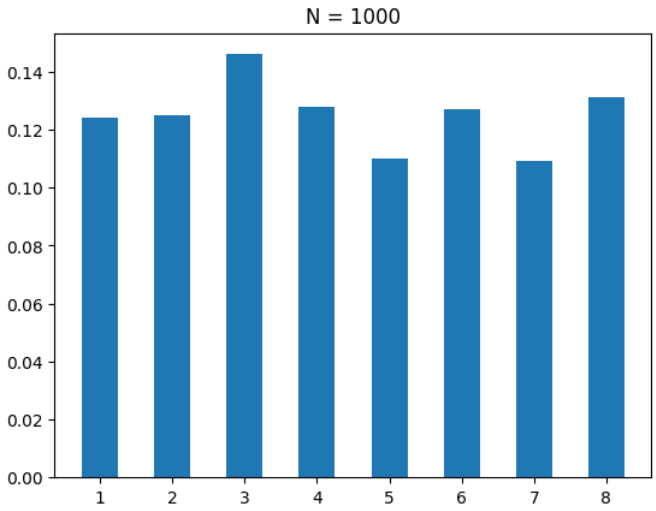
In [ ]:
```
# Simulates the roll of a a fair 4-sided die. Size determines the number of times the die is rolled.
fair  = np.random.randint(low=1,high=5, size=100)
# Plot Histogram.
plt.hist(fair,bins=[1,2,3,4,5],rwidth = 0.5, density=True, align='left')
plt.title("N  = %i "%100)
plt.show()
```

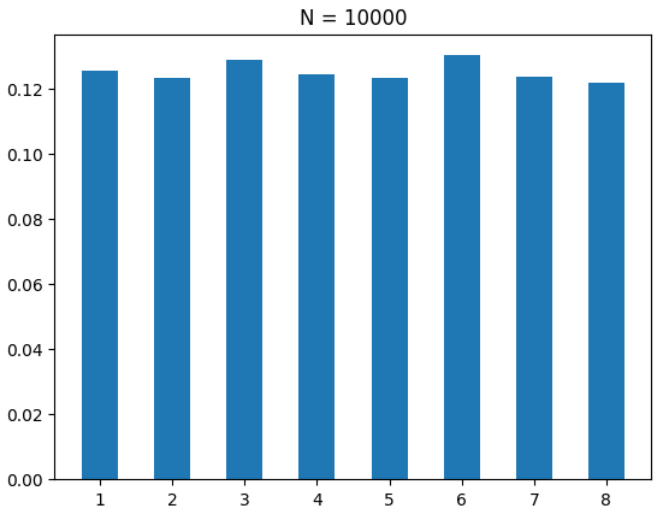


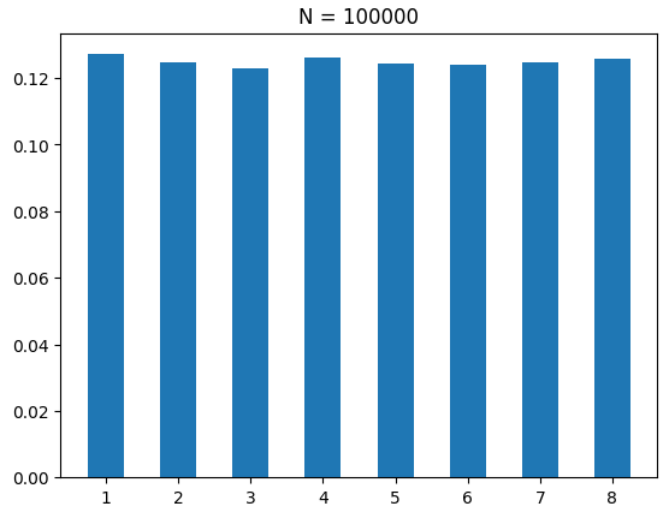
In this task, you need to simulate the throwing of a fair 8-sided die (Octahedron) N times and plot the corresponding histogram of outputs. You are required to experiment with different values of N and comment how the shape of the histogram changes with increasing values of N.

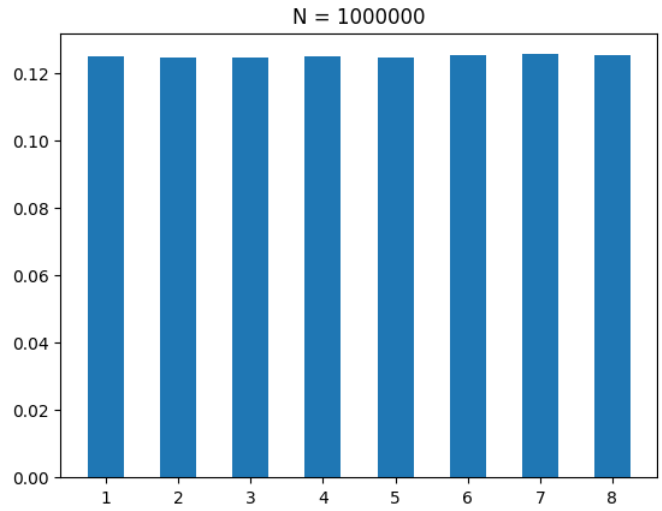
In [ ]:
```
''' Ali Muhammad Asad aa07190'''
# YOUR CODE FOR TASK 1
#Simulate the roll of a fair 8-sided die (Octahedron) n times
fair_hundred = np.random.randint(low = 1, high = 9, size = 100) #Randomly generating numbers 100 times
fair_thousand = np.random.randint(low = 1, high = 9, size = 1000) #Randomly generating numbers 1000 times
fair_tenthou = np.random.randint(low = 1, high = 9, size = 10000)  #Randomly generating numbers 10000 times
fair_hundthou = np.random.randint(low = 1, high = 9, size = 100000) #Randomly generating numbers 100000 times
fair_million = np.random.randint(low = 1, high = 9, size = 1000000)  #Randomly generating numbers 1000000 times
#The above creates an array of size "size" with randomly selected elements from 1 - 8 [9 not inclusive] hence simulating the rolls of an 8 sided

## Plotting Histogram ##
bin = [1, 2, 3, 4, 5, 6, 7, 8, 9] #the values to be displayed on the histogram from 1 - 9 not inclusive of 9

# Showing for different increasing values where n can take any of these values - to show contrast between increasing values of n #
plt.hist(fair_hundred, bins = bin, rwidth= 0.5, density= True, align= 'left') #Setting up the histogram
plt.title("N = %i"%100); plt.show() #Setting title and then displaying the histogram

plt.hist(fair_thousand, bins = bin, rwidth= 0.5, density= True, align= 'left') #Setting up the histogram
plt.title("N = %i"%1000); plt.show() #Setting title and then displaying the histogram

plt.hist(fair_tenthou, bins = bin, rwidth= 0.5, density= True, align= 'left') #Setting up the histogram
plt.title("N = %i"%10000); plt.show() #Setting title and then displaying the histogram

plt.hist(fair_hundthou, bins = bin, rwidth= 0.5, density= True, align= 'left') #Setting up the histogram
plt.title("N = %i"%100000); plt.show() #Setting title and then displaying the histogram

plt.hist(fair_million, bins = bin, rwidth= 0.5, density= True, align= 'left') #Setting up the histogram
plt.title("N = %i"%1000000); plt.show() #Setting title and then displaying the histogram
```


N = 100
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
1
2
3
4
5
6
7
8


N = 1000
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
1
2
3
4
5
6
7
8


N = 10000
0.12
0.10
0.08
0.06
0.04
0.02
0.00
1
2
3
4
5
6
7
8





Ali Muhammad Asad (aa07190)

Comments

As we increase the value of n, we get more and more number of occurrences for each face of the 8-sided die. Hence the sample space / number of trials keeps on increasing and all the bars approach towards a single value. Since this is simulating a fair 8-sided die, each number should have equal probability, and at very large sample spaces - where there are very large number of trials, that is seen - all bars have the same height. This happens as each number has equal probability, so the occurances of all numbers become roughly the same on very large values of the sample space. This also follows the law of large numbers in probability which states that as a sample size grows or the same experiment is performed a very large number of times, the average of the results obtained tends to converge towards the expected value (mean) of the experiment.

[10 Marks] Task 2 : Expectation and Variance

In Task 1, you plotted the histogram for simulated throwing of a fair 8-sided die N times. In Task 2, you have been provided functions that compute the expectation and variance for each value of N. You are required to comment on the comparison of these calculated values with the theoretical values of expaction and variance (obtained by considering the outcome of rolling a fair 8-sided die as a Discrete Uniform Random Variable).

In [ ]:
```
# Compute Expectation
def compute_expectation (x: np.array, p: np.array ):
   return np.sum( p * x)

# Compute Variance
def compute_variance (x: np.array, p: np.array ):
   EX = compute_expectation(x,p)
   x = np.square(x - EX)
   return compute_expectation(x,p)
```

In [ ]:
```
def compute_expectation_variance (N: list):
    for n in N:
      print ("N = {val}".format(val = n ))
      # Simulate the rolling of a fair 8-sided die
      q = np.random.randint(low=1,high=9,size=n)
      x,p = np.unique(q,return_counts=True)
      p = p/n
      # Compute Expectation
      EX = compute_expectation(x,p)
      # Compute Variance
      Var = compute_variance(x,p)

      print ("Expectation : {ex:.2f} ,  Variance : {var:.2f}".format(ex=EX,var=Var))
```

```
compute_expectation_variance([100,1000,10000,100000])
```

```
N = 100
Expectation : 4.31 ,  Variance : 5.13
N = 1000
Expectation : 4.49 ,  Variance : 5.24
N = 10000
Expectation : 4.50 ,  Variance : 5.26
N = 100000
Expectation : 4.50 ,  Variance : 5.25
```

Ali Muhammad Asad (aa07190)

Comments

The Mean or the Expectation of a Discrete Random Variable 'X' can be considered to be the weighted average, or the center of gravity of the PMF of X. So in theory, since each face in an 8-sided die has equal probability to occur, and the PMF would give us bars of equal height on each discrete value of x, then the mean will be the center point of the values of x. Or in theory, we can calculate it using the formula for the Mean, which comes out to be 4.5 [Expectation = (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8) / 8 = 4.5]. Now in calculation, we can observe that for smaller values of n, where we have a smaller sample space or fewer trials, the mean is very close but not the exact value as we calculated. However, as we increase the value of 'n', we get closer and closer to the expected value, and on very large values of 'n', where each number has roughly occurred the same number of times (we have conducted a huge number of trials), the Expectation is the same as the one we got in theory. So the calculated expectation converges to the true expectation of the distribution as we increase the number of trials.

In theory, the variance is the measure of spread/distribution of a PMF around its mean. Based on the Expected value of a Discrete Random Variable X, the variance can be calculated using its formula var(X) = ∑(x - E[X])^2 px(x). In theory, the variance calculated is 5.25. Now in calculation, we can observe that on lower values of n, the variance deviates from the true variance as there are less number of trials, hence the Expected value is not the true Expected value, hence a difference in variance. This also checks out by theory as the spread of the PMF also deviates from the mean. However, as we increase the value of n, and hence the number of trials, we get closer to the true mean of the PMF, hence the variance also decreases as the spread around the mean decreases for larger numbers and we get closer to the true mean of the PMF. Hence the variance decreases and gets closer to the true variance.

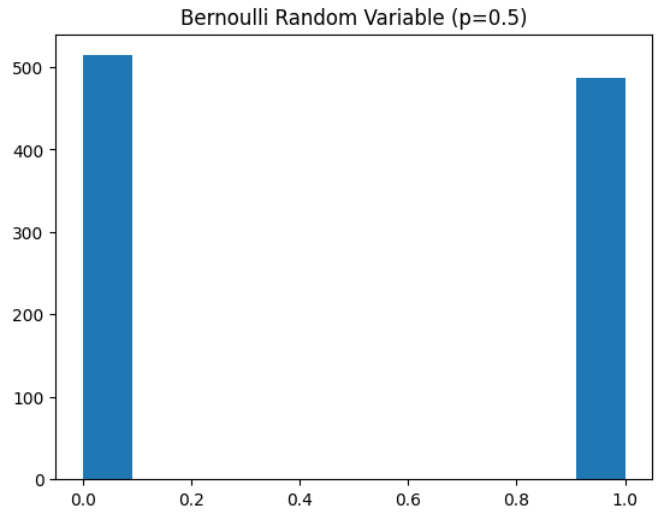
This can be interpreted in another way, that as we increase the number of trials, or increase the value of n, the outcome varies less from the expected outcome, so it becomes more predictable and approaches the values observed in theory - again which follows the law of large numbers as explained in Task 1.

Task 3 [25 Marks] : Common Discrete Random Variables

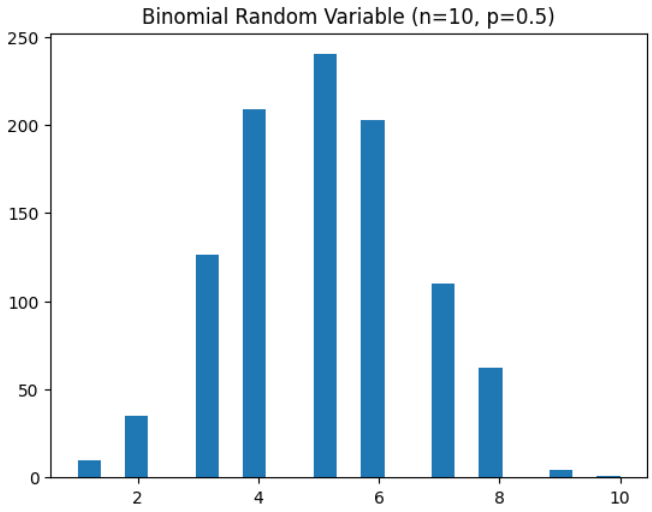
In Task 1, you were introducted to the relationship between histogram and PMF of Random Variables. In Task 2, you were introduced to computing Expectation and Variance. In Task 3, you will be introduced to Common DRVs that you have already discussed in class. The aim of Task 3 is to introduce you to the simulation of these DRVs in Python.

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.

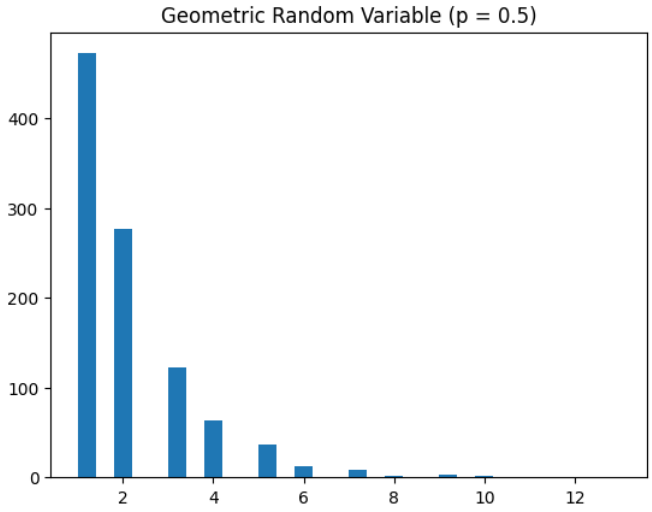
In [ ]:
```
# Generate 1000 samples of a Bernoulli Random Variable
p = 0.5 # Probability of Success
n = 1   # Number of states (When n = 1, Binomial Random Variable becomes a Bernoulli Random Variable)
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Bernoulli Random Variable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



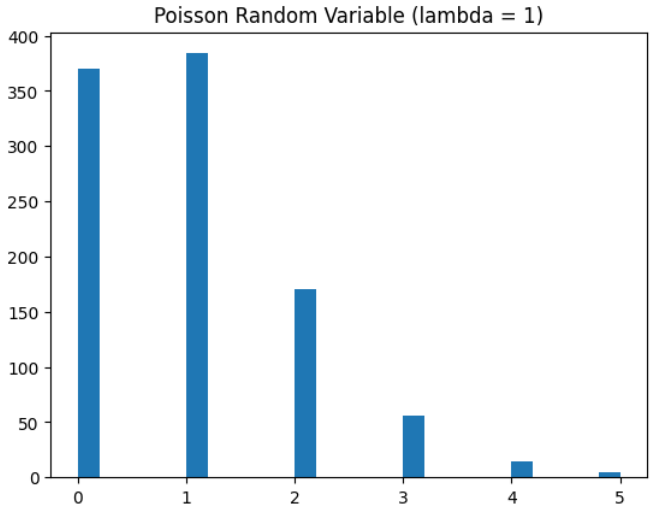
In [ ]:
```
# Generate 1000 samples of a Binomial Random Variable
p = 0.5 # Probability of a single success
n = 10  # Number of states
size  = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=10, p=0.5)")
plt.hist(X,bins="auto")
plt.show()
```



In [ ]:
```
# 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()
```



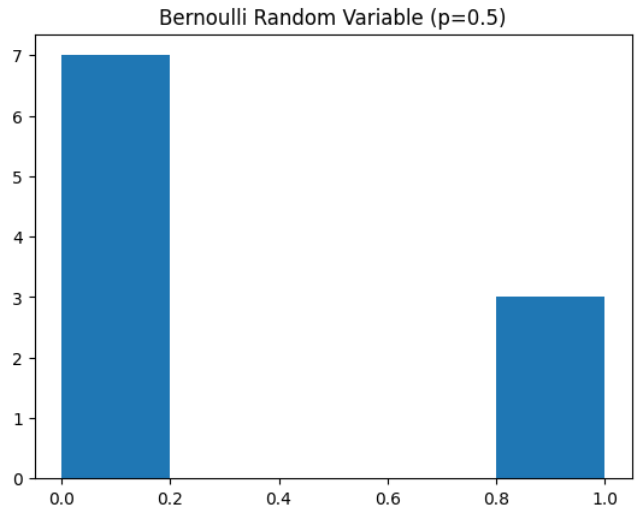
In [ ]:
```
# Generate 1000 samples of a Poisson Random Variable
lambd = 1
X = np.random.poisson(lambd,size=1000)
plt.title("Poisson Random Variable (lambda = 1)")
plt.hist(X,bins="auto")
plt.show()
```

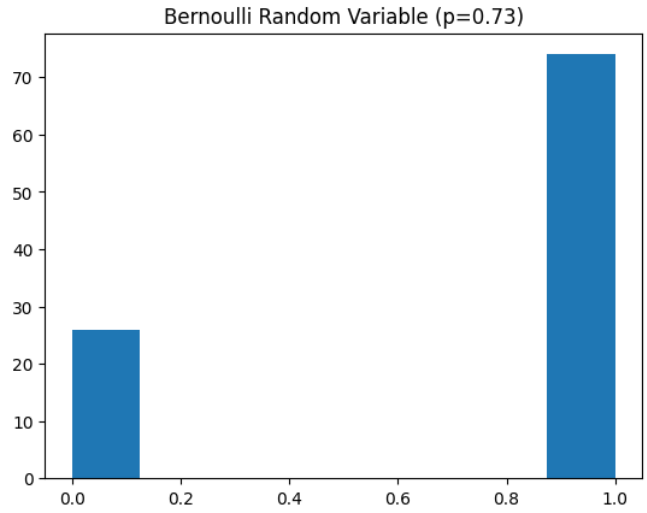


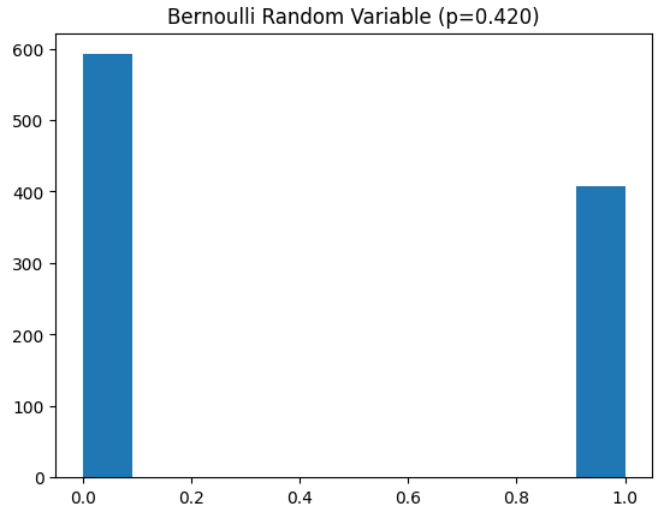
In this task, you are required to vary parameters for each Discrete RV (Bernoulli, Binormail, Geometric, Poission) while keeping others constant and comment on how does this affect the shape of the their histogram.

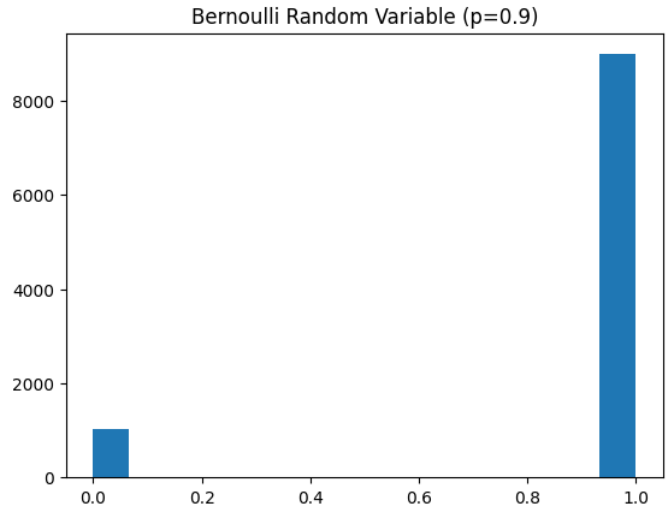
In [ ]:
```
''' Ali Muhammad Asad aa07190'''
# BERNOULLI RANDOM VARIABLE
```

```
# Your code goes here

# 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  = 10 # Number of trials
X = np.random.binomial(n,p,size=size) #Setting up the Bernoulli Random Variable
plt.title("Bernoulli Random Variable (p=0.5)") #Title
plt.hist(X,bins='auto') #Histogram
plt.show() #Showing the distribution

# Varying 'p' value

p = 0.73 # Probability of Success
n = 1   # Number of states (When n = 1, Binomial Random Variable becomes a Bernoulli Random Variable)
size  = 100 # Number of trials
X = np.random.binomial(n,p,size=size) #Setting up the Bernoulli Random Variable
plt.title("Bernoulli Random Variable (p=0.73)") #title
plt.hist(X,bins='auto') #histogram
plt.show() #Showing the distribution

p = 0.420 # 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) #Setting up the Bernoulli Random Variable
plt.title("Bernoulli Random Variable (p=0.420)") #title
plt.hist(X,bins='auto') #histogram
plt.show() ##Showing the distribution

p = 0.9 # 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) #Setting up the Bernoulli Random Variable
plt.title("Bernoulli Random Variable (p=0.9)") #title
plt.hist(X,bins='auto') #histogram
plt.show() #Showing the distribution
```









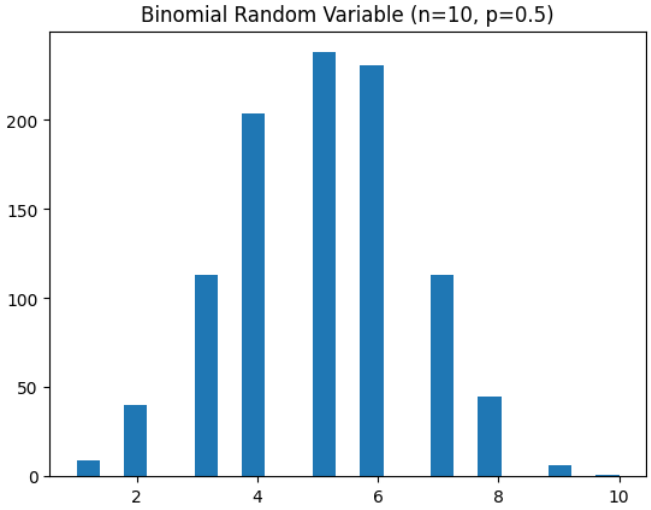
Ali Muhammad Asad (aa07190)

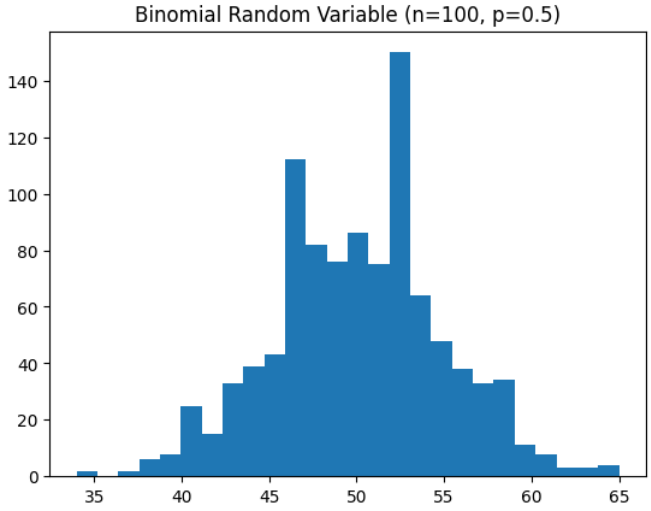
Comments

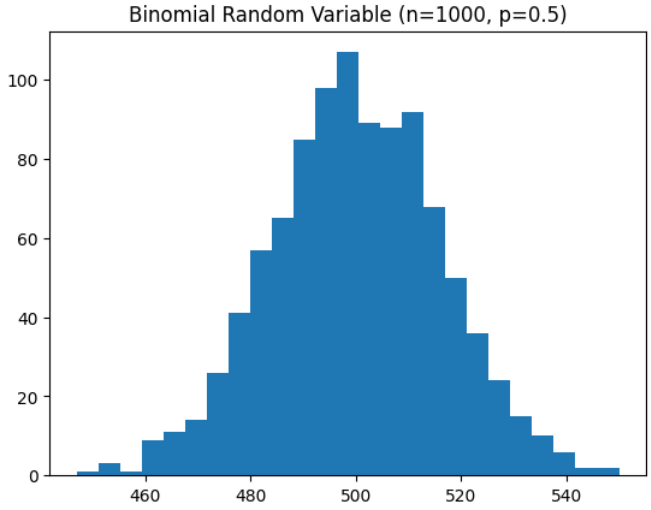
We only vary the 'p' values since this is a Bernoulli Random Variable so it can have only two states, so n can't be changed. The variables that can be changed are the number of trials, and the probability of success/the event happening denoted by 'p'. As 'p' is increased, then the value at 1 increases as can be seen since that is the likelihood of the event happening, and the corresponding value at 0 decreases as that is the likelihood of the event not happening and vice versa.

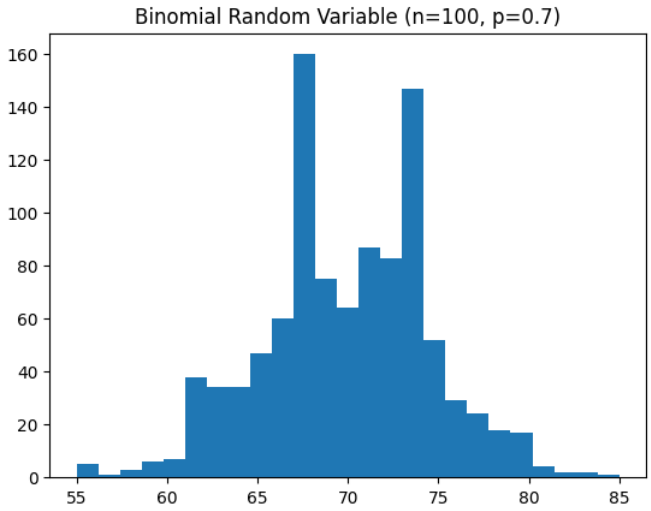
In [ ]:
```
''' Ali Muhammad Asad aa07190'''
# BINOMIAL RANDOM VARIABLE
# Your code goes here

# Generate 1000 samples of a Binomial Random Variable

# Varying the number of states 'n' first while keeping probability 'p' constant#

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()

p = 0.5 # Probability of a single success
n = 100  # Number of states
size = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=100, p=0.5)")
plt.hist(X,bins="auto")
plt.show()

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

# Varying the probability 'p' while keeping number of states 'n' constant

p = 0.7 # Probability of a single success
n = 100  # Number of states
size = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
```

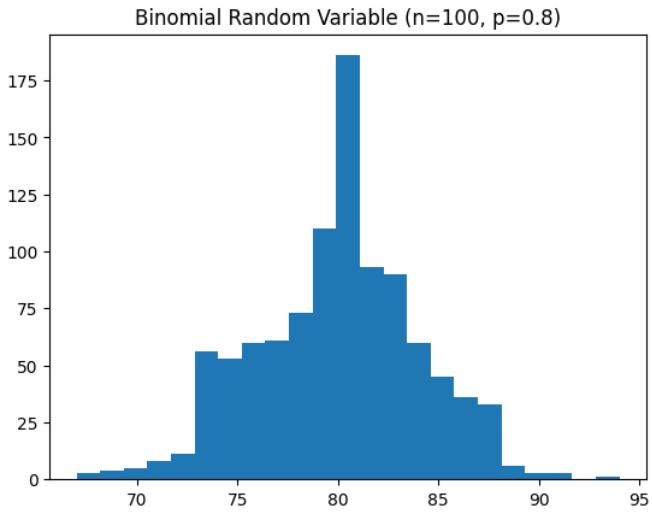
```
plt.title("Binomial Random Variable (n=100, p=0.7)")
plt.hist(X,bins="auto")
plt.show()

p = 0.8 # Probability of a single success
n = 100  # Number of states
size = 1000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=100, p=0.8)")
plt.hist(X,bins="auto")
plt.show()

p = 0.9 # Probability of a single success
n = 100  # Number of states
size = 10000 # Number of trials
X = np.random.binomial(n,p,size=size)
plt.title("Binomial Random Variable (n=100, p=0.9)")
plt.hist(X,bins="auto")
plt.show()

p = 0.2 # 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=0.2)")
plt.hist(X,bins="auto")
plt.show()
```

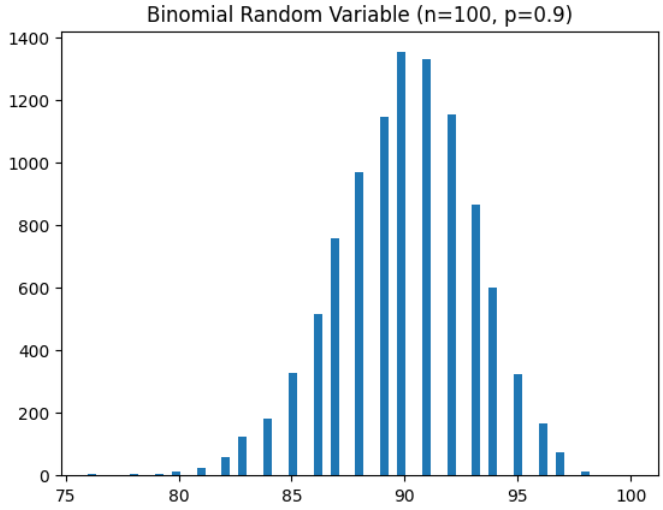


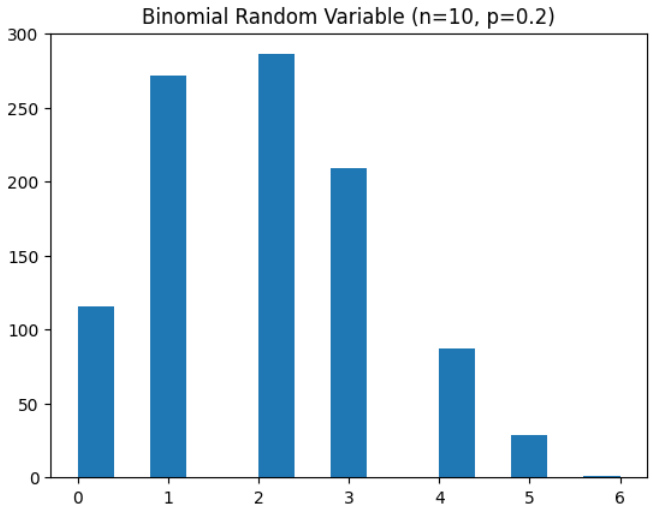



Binomial Random Variable (n=1000, p=0.5)
0
20
40
60
80
100
460
480
500
520
540


Binomial Random Variable (n=100, p=0.7)
0
20
40
60
80
100
120
140
160
55
60
65
70
75
80
85


Binomial Random Variable (n=100, p=0.8)
0
25
50
75
100
125
150
175
70
75
80
85
90
95





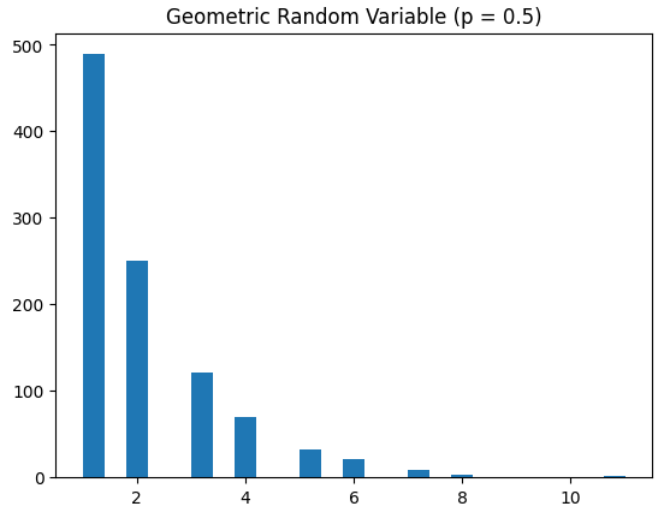
Ali Muhammad Asad (aa07190)

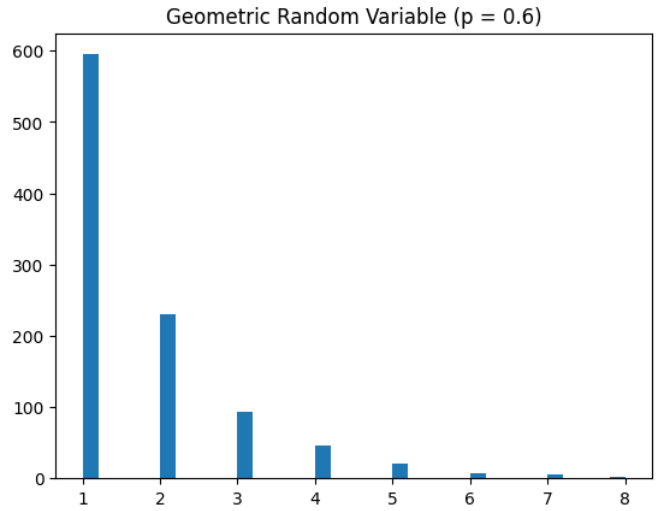
Comments

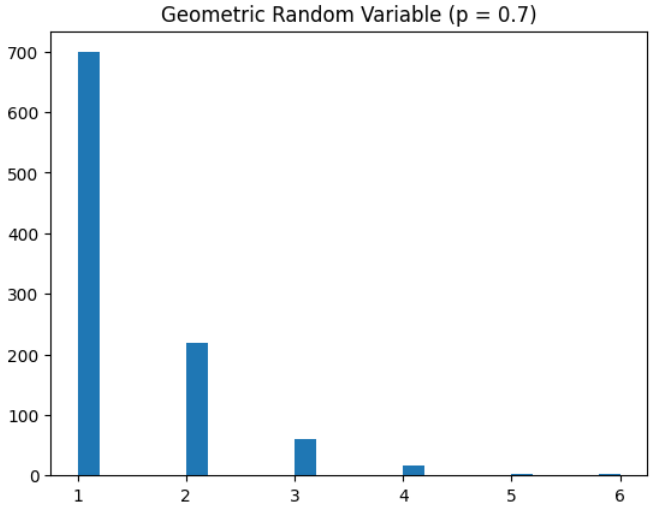
It can be seen from the various distributioins that if probability is kept constant but number of states are changed, then accordingly our variance has also changed. If number of states are increased, then variance increases, and if number of states are decreased then variance also decreases. If number of states are kept constant and we increase the probability 'p', then peak of the graph shifts towards the right, and if 'p' is decreased, then the peak of the graph shifts towards the left. This indicates that the expected value(mean value) increases as we increase 'p' and decreases as we decrease 'p'. This is also called as skew of the graph. For p > 0.5, the mean shifts to the right, hence is called skewed right. For p < 0.5, the mean shifts towards the left, hence is called skewed left.

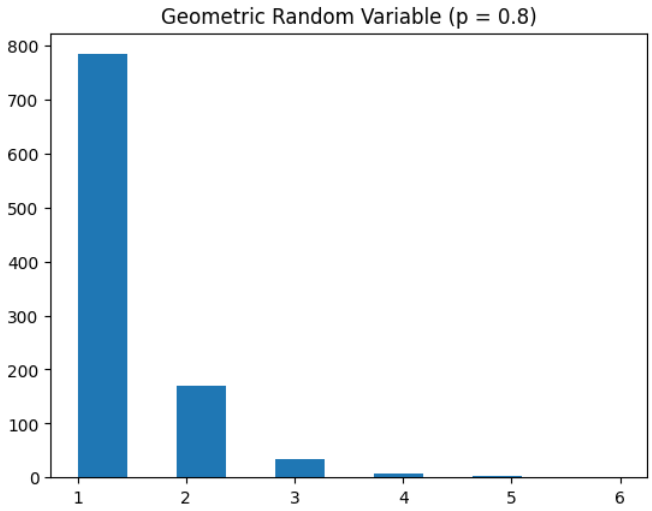
In [ ]:
```
''' Ali Muhammad Asad aa07190'''
# GEOMETRIC RANDOM VARIABLE
# Your code goes here

# 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()

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")
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")
plt.show()

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

p = 0.4
size = 1000
plt.title("Geometric Random Variable (p = 0.4)")
```

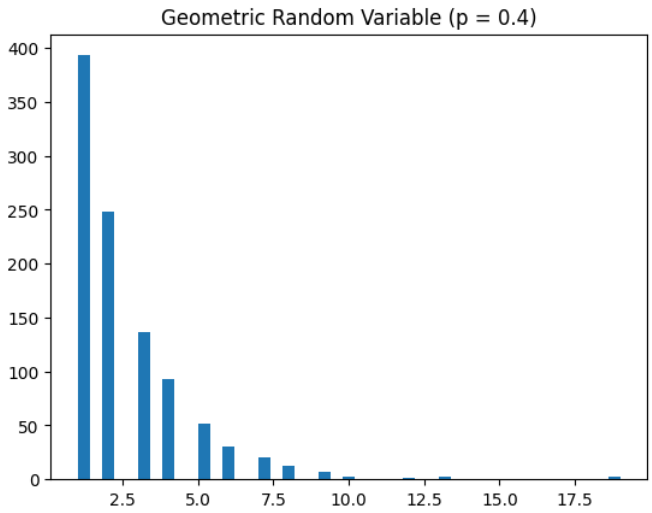
```
X = np.random.geometric(p,size=size)
plt.hist(X,bins="auto")
plt.show()

p = 0.3
size = 1000
plt.title("Geometric Random Variable (p = 0.3)")
X = np.random.geometric(p,size=size)
plt.hist(X,bins="auto")
plt.show()

p = 0.2
size = 1000
plt.title("Geometric Random Variable (p = 0.2)")
X = np.random.geometric(p,size=size)
plt.hist(X,bins="auto")
plt.show()

p = 0.1
size = 1000
plt.title("Geometric Random Variable (p = 0.1)")
X = np.random.geometric(p,size=size)
plt.hist(X,bins="auto")
plt.show()
```

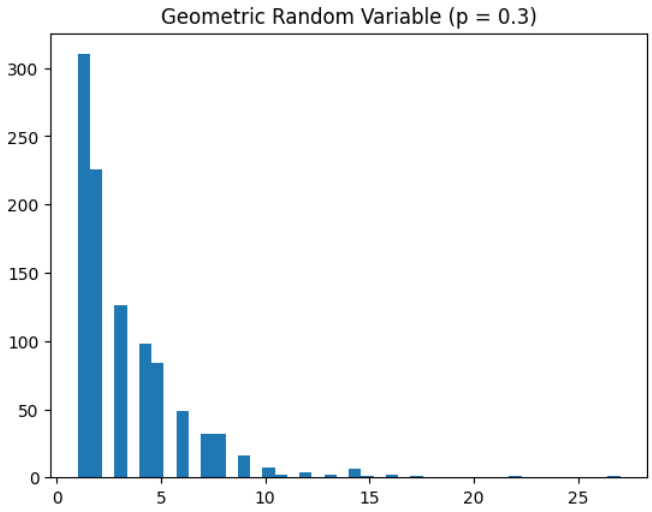


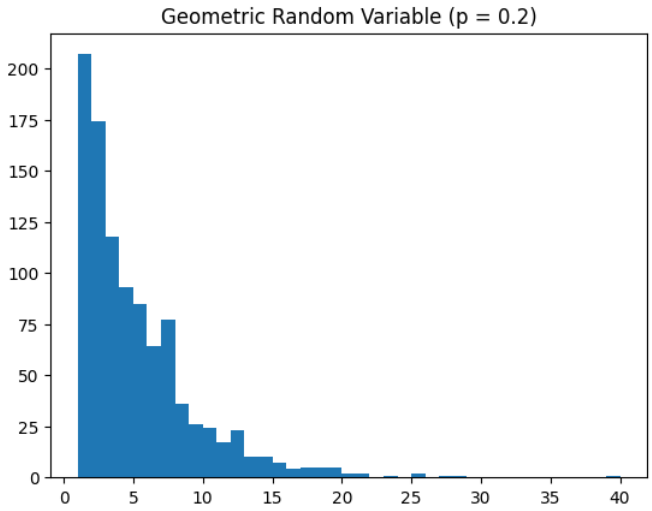


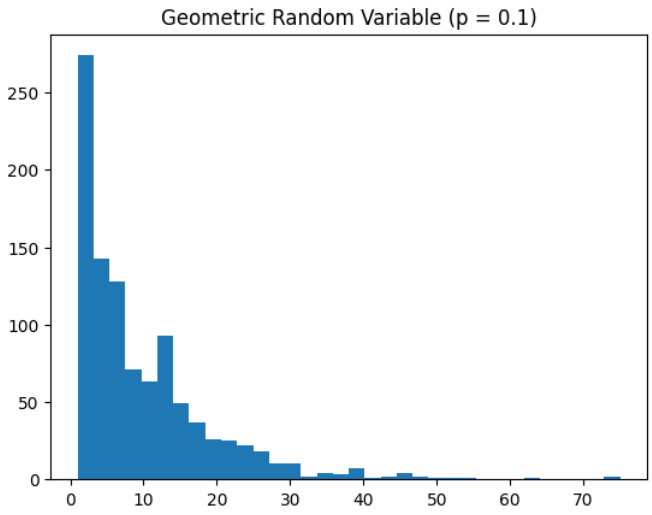

Geometric Random Variable (p = 0.7)
700
600
500
400
300
200
100
0
1
2
3
4
5
6


Geometric Random Variable (p = 0.8)
800
700
600
500
400
300
200
100
0
1
2
3
4
5
6


Geometric Random Variable (p = 0.4)
400
350
300
250
200
150
100
50
0
2.5
5.0
7.5
10.0
12.5
15.0
17.5







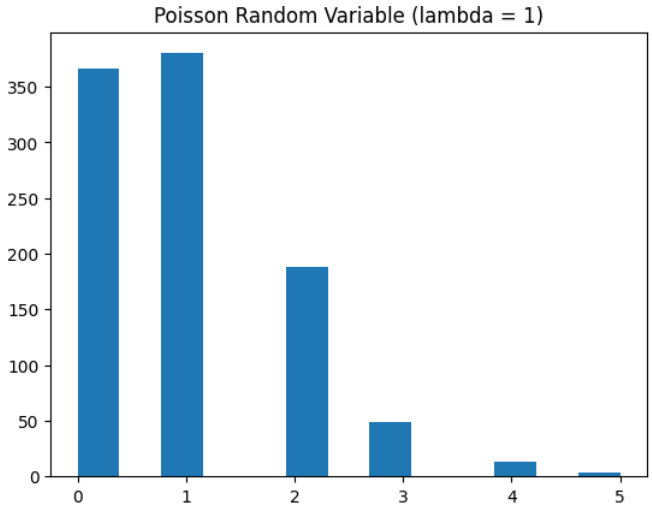
Ali Muhammad Asad (aa07190)

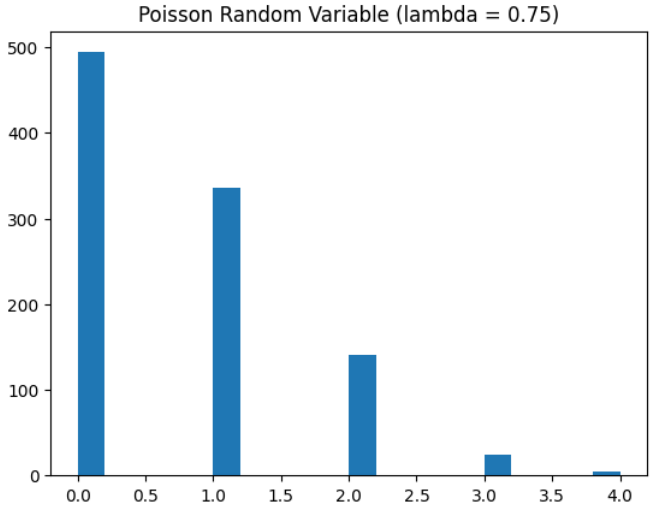
Comments

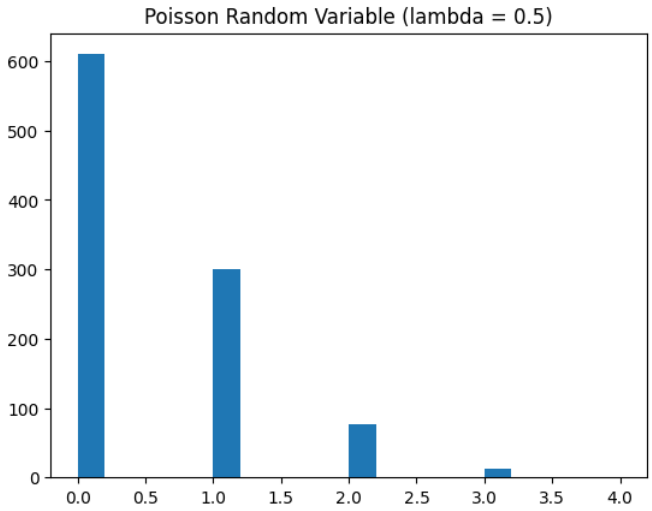
For a Geometric Random Variable, the Expectation = 1/p where 'p' is the probability of success. So as 'p' increases, the Expectation decreases, and as 'p' decreases, Expectation increases. As a direct conseqeunce of this, the variance also has an inverse proportion to the value of 'p' and it can be seen by the various plost as well that on increasing the probability of success 'p', the variance decresaes, however, on decreasing 'p', the variance increases.

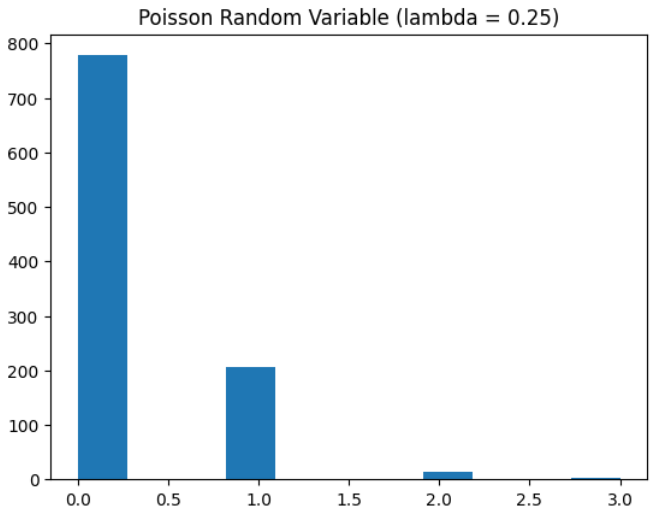
In [ ]:
```
''' Ali Muhammad Asad aa07190'''
# POISSON RANDOM VARIABLE
# Your code goes here

# Generate 1000 samples of a Poisson Random Variable
lambd = 1
X = np.random.poisson(lambd,size=1000)
plt.title("Poisson Random Variable (lambda = 1)")
plt.hist(X,bins="auto")
plt.show()
```

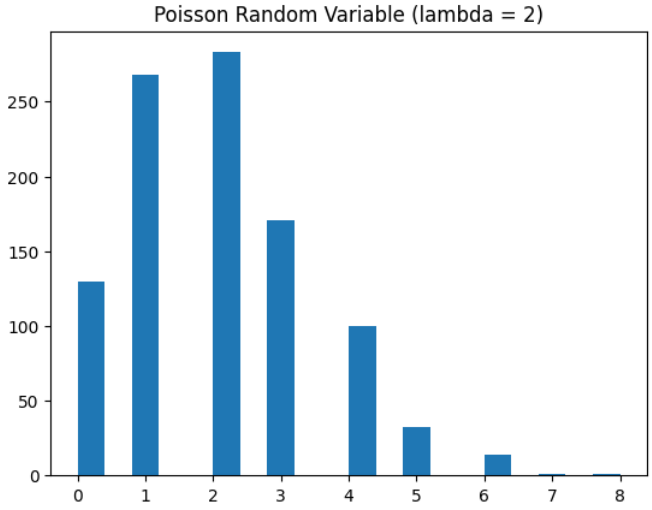
```
lambd = 0.75
X = np.random.poisson(lambd,size=1000)
plt.title("Poisson Random Variable (lambda = 0.75)")
plt.hist(X,bins="auto")
plt.show()

lambd = 0.5
X = np.random.poisson(lambd,size=1000)
plt.title("Poisson Random Variable (lambda = 0.5)")
plt.hist(X,bins="auto")
plt.show()

lambd = 0.25
X = np.random.poisson(lambd,size=1000)
plt.title("Poisson Random Variable (lambda = 0.25)")
plt.hist(X,bins="auto")
plt.show()

lambd = 2
X = np.random.poisson(lambd,size=1000)
plt.title("Poisson Random Variable (lambda = 2)")
plt.hist(X,bins="auto")
plt.show()

lambd = 3
X = np.random.poisson(lambd,size=1000)
plt.title("Poisson Random Variable (lambda = 3)")
plt.hist(X,bins="auto")
plt.show()

lambd = 5
X = np.random.poisson(lambd,size=1000)
plt.title("Poisson Random Variable (lambda = 5)")
plt.hist(X,bins="auto")
plt.show()

lambd = 10
X = np.random.poisson(lambd,size=1000)
plt.title("Poisson Random Variable (lambda = 10)")
plt.hist(X,bins="auto")
plt.show()

lambd = 20
X = np.random.poisson(lambd,size=1000)
plt.title("Poisson Random Variable (lambda = 20)")
plt.hist(X,bins="auto")
plt.show()

lambd = 100
X = np.random.poisson(lambd,size=1000)
plt.title("Poisson Random Variable (lambda = 100)")
plt.hist(X,bins="auto")
plt.show()
```

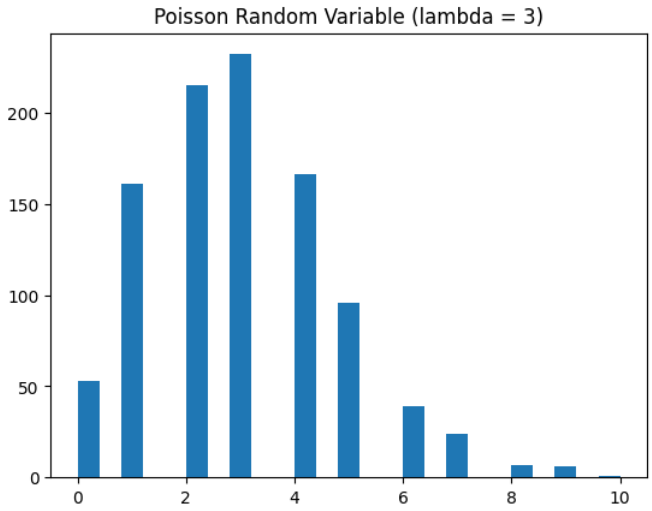


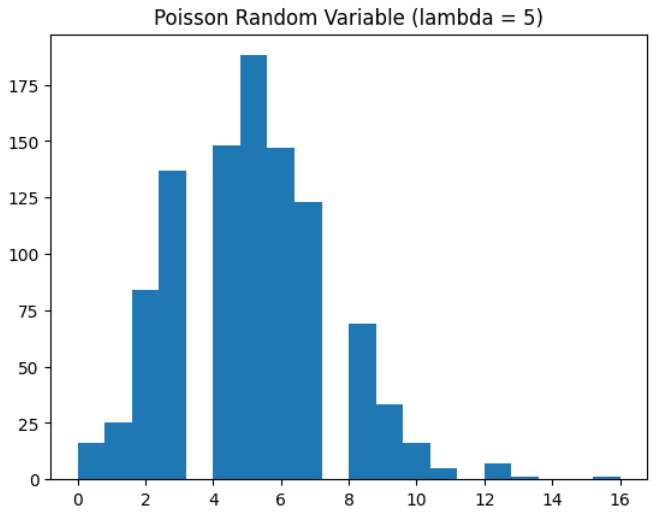

Poisson Random Variable (lambda = 0.75)
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

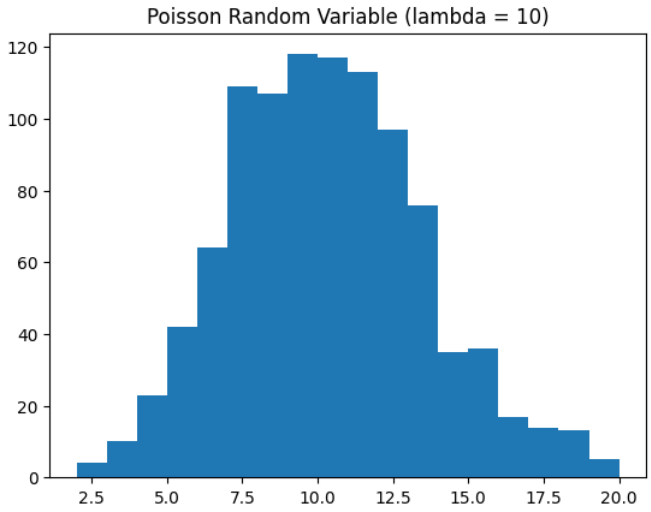

Poisson Random Variable (lambda = 0.5)
600
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

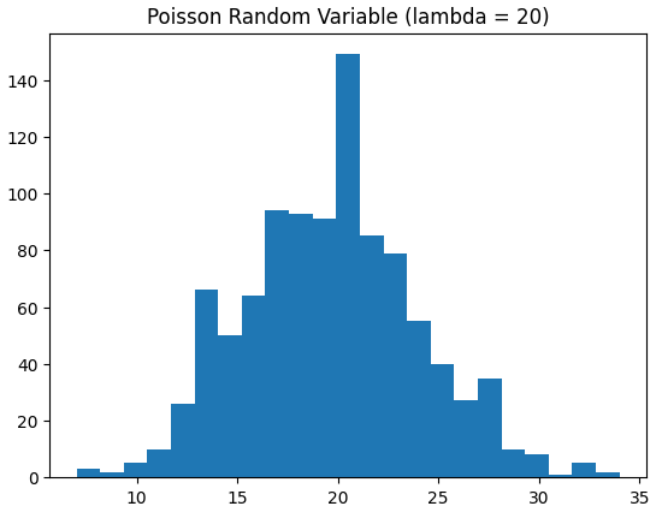

Poisson Random Variable (lambda = 0.25)
800
700
600
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

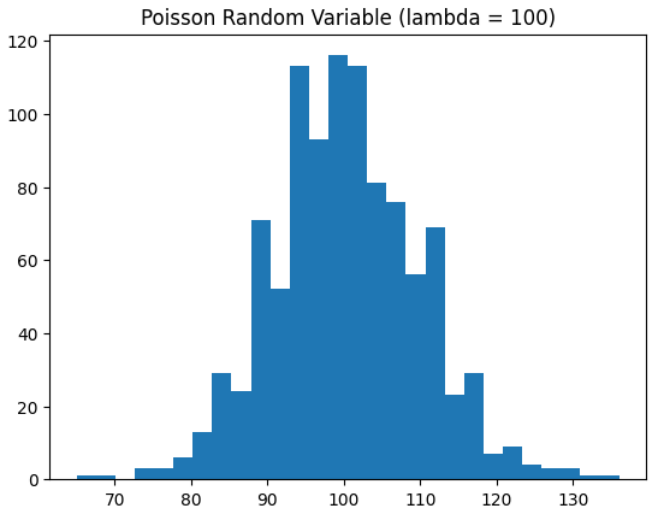

Poisson Random Variable (lambda = 2)
0
50
100
150
200
250
0
1
2
3
4
5
6
7
8


Poisson Random Variable (lambda = 3)
0
50
100
150
200
0
2
4
6
8
10


Poisson Random Variable (lambda = 5)
0
25
50
75
100
125
150
175
0
2
4
6
8
10
12
14
16







Ali Muhammad Asad (aa07190)

Comments

For a Poisson Random Variable, the lambda value is the Expected Value (mean) of the random variable, and also the variance. And that can be seen from the various distributions that on increasing the value of lambda, the expectation and variance increase, and on decreasing the value of lambda, the expectation and variance decrease. Note that as we increase the value of lamda, we start to get a bell shaped curve, and on large values of lambda, the shape begins to look similar to the normal distribution. Hence, at larger values of lamda, Normal Distribution can be used to approximate the Poisson Distribution to the Normal Distribution [values of lambda greater than 10].