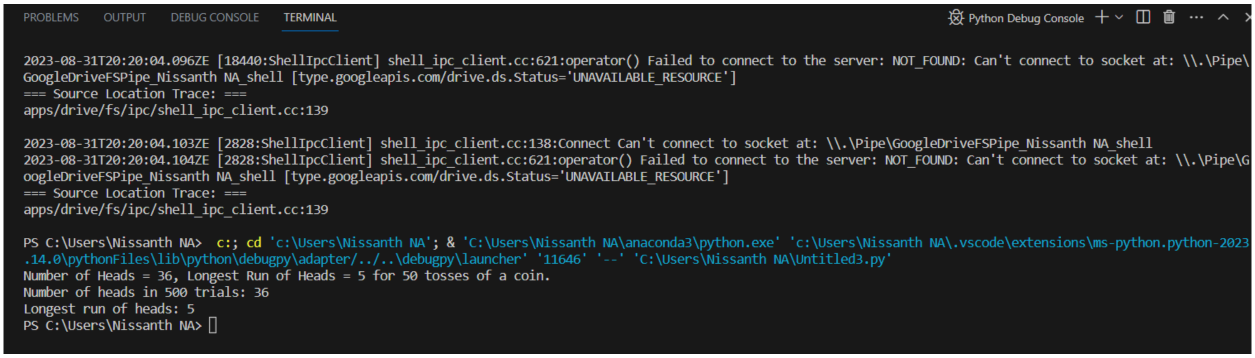
Homework #1
EE 541: Fall 2023

Name: Nissanth Neelakandan Abirami
USC ID: 2249203582
Instructor: Dr. Franzke

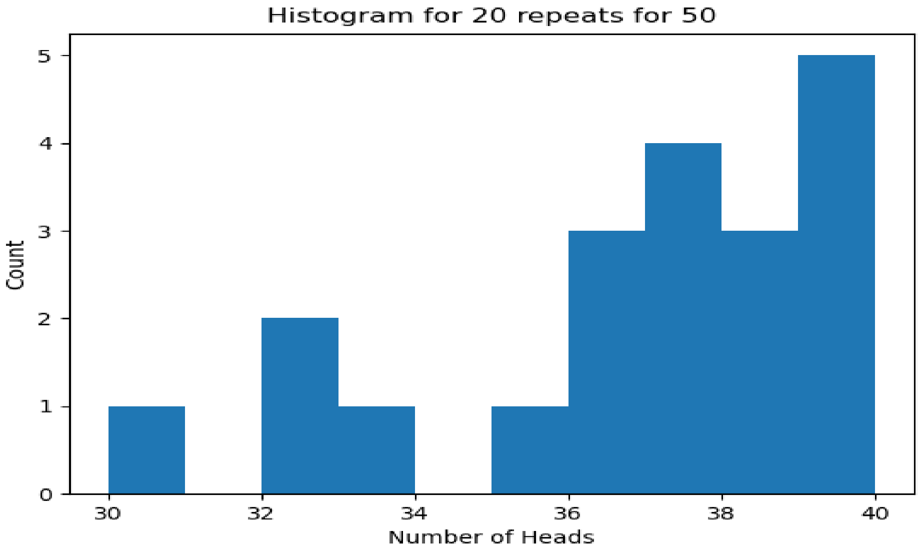
1. Simulate tossing a biased coin (a Bernoulli trial) where P[HEAD] = 0.70.
(a) Count the number of heads in 50 trials. Record the longest run of heads.
(b) Repeat the 50-flip experiment 20, 100, 200, and 1000 times. Use matplotlib to generate a histogram showing the observed number of heads for each case. Comment on the limit of the histogram.
(c) Simulate tossing the coin 500 times. Generate a histogram showing the heads run lengths.

Solutions:



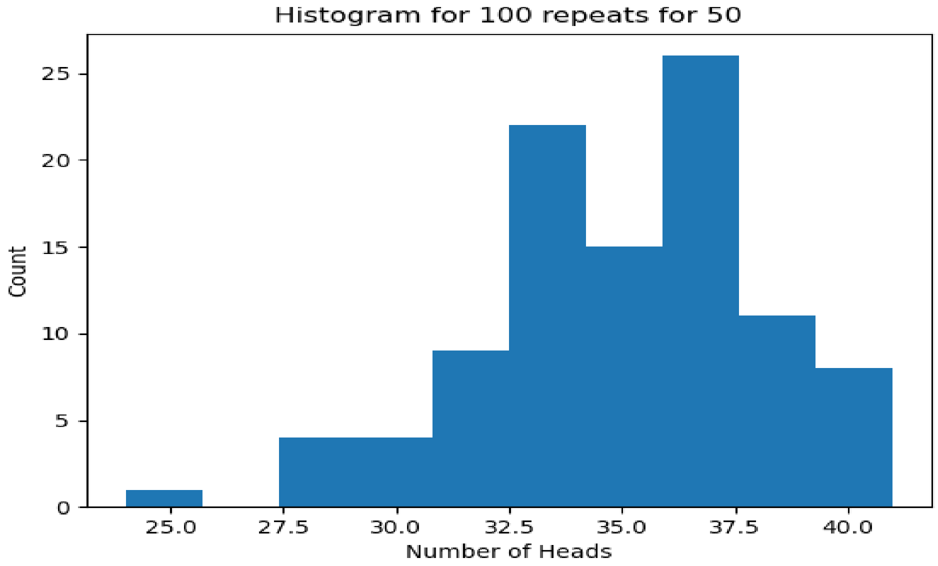
```
2023-08-31T20:20:04.096ZE [18440:ShellIpcClient] shell_ipc_client.cc:621:operator() Failed to connect to the server: NOT_FOUND: Can't connect to socket at: \\.\Pipe\
GoogleDriveFSPipe_Nissanth NA_shell [type.googleapis.com/drive.ds.Status='UNAVAILABLE_RESOURCE']
=== Source Location Trace: ===
apps/drive/fs/ipc/shell_ipc_client.cc:139

2023-08-31T20:20:04.103ZE [2828:ShellIpcClient] shell_ipc_client.cc:138:Connect Can't connect to socket at: \\.\Pipe\GoogleDriveFSPipe_Nissanth NA_shell
2023-08-31T20:20:04.104ZE [2828:ShellIpcClient] shell_ipc_client.cc:621:operator() Failed to connect to the server: NOT_FOUND: Can't connect to socket at: \\.\Pipe\G
oogleDriveFSPipe_Nissanth NA_shell [type.googleapis.com/drive.ds.Status='UNAVAILABLE_RESOURCE']
=== Source Location Trace: ===
apps/drive/fs/ipc/shell_ipc_client.cc:139

PS C:\Users\Nissanth NA>  c:; cd 'c:\Users\Nissanth NA'; & 'C:\Users\Nissanth NA\anaconda3\python.exe' 'c:\Users\Nissanth NA\.vscode\extensions\ms-python.python-2023
.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '11646' '--' 'C:\Users\Nissanth NA\Untitled3.py'
Number of Heads = 36, Longest Run of Heads = 5 for 50 tosses of a coin.
Number of heads in 500 trials: 36
Longest run of heads: 5
PS C:\Users\Nissanth NA> 
```

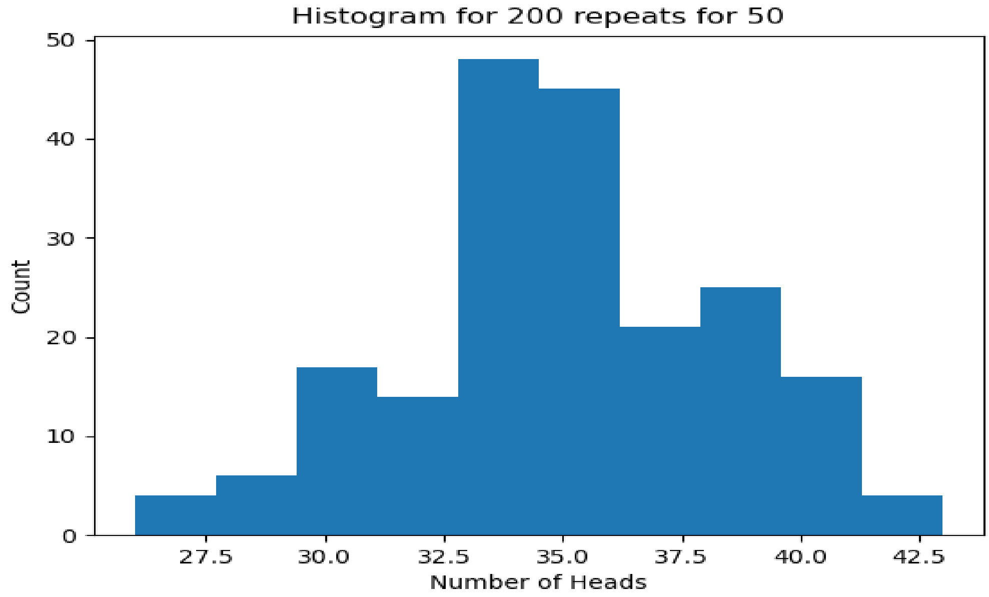
a) No of Heads = 36
   Longest No of Heads(Streaks)= 5 for 50 flips



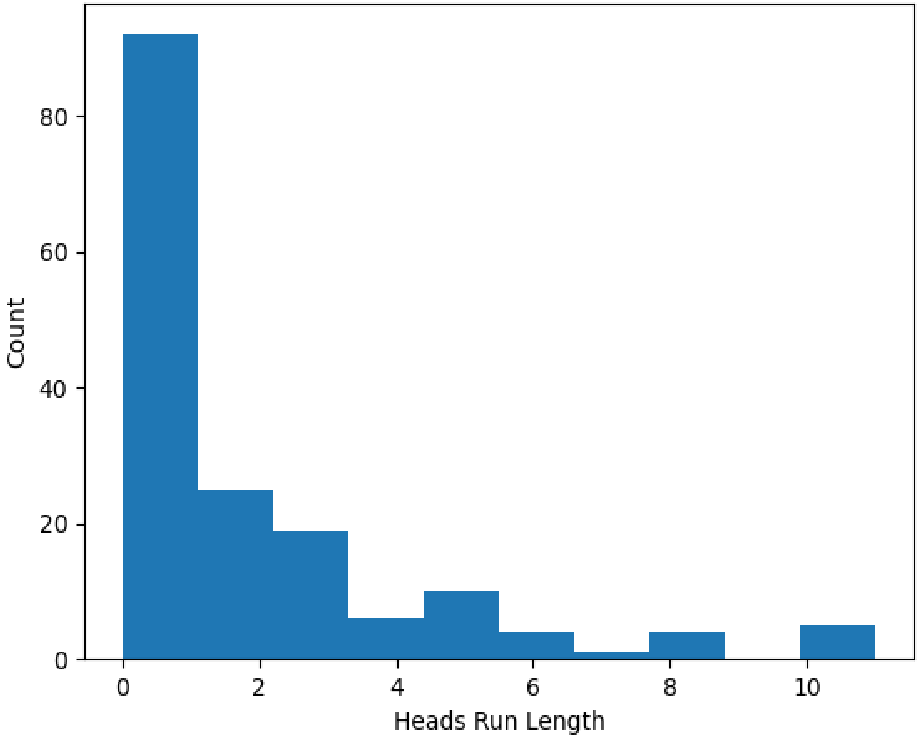
b) Histogram is used to plot the data's for better visualization. Since the Biased coin with probability of head is 0.7 is used. The average values of the Histogram revolves around 35.The Limit for the Histogram for visualization is set close to 30 and 45.

P[H] = 0.7*50
= 35


Histogram for 100 repeats for 50
Count
0
5
10
15
20
25
25.0
27.5
30.0
32.5
35.0
37.5
40.0
Number of Heads

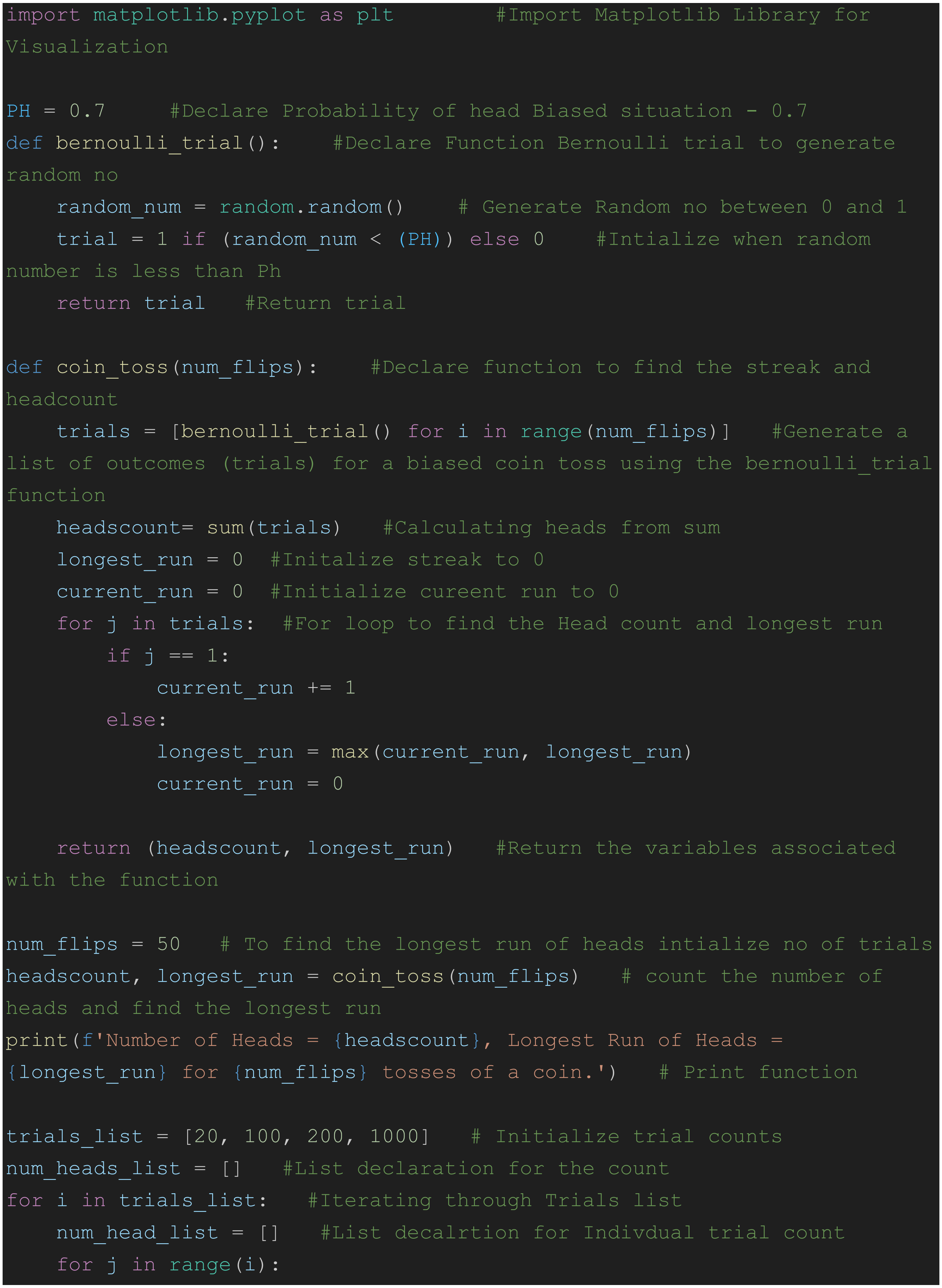

Histogram for 200 repeats for 50
Count
0
10
20
30
40
50
27.5
30.0
32.5
35.0
37.5
40.0
42.5
Number of Heads

c) Longest No of Heads: 5

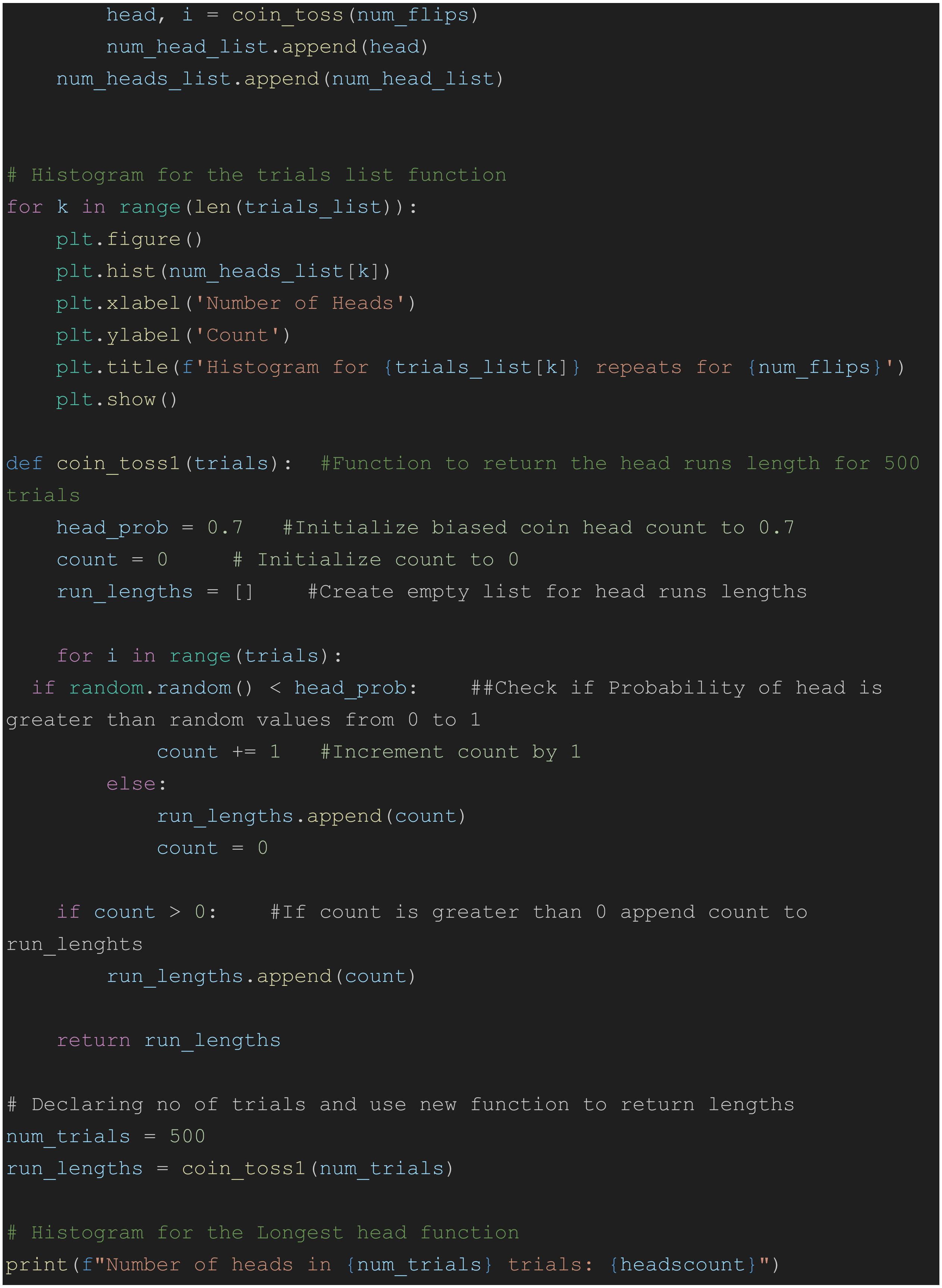


APPENDIX:

```
"""". Simulate tossing a biased coin (a Bernoulli trial) where P[HEAD] =
0.70.
(a) Count the number of heads in 50 trials. Record the longest run of
heads.
(b) Repeat the 50-flip experiment 20, 100, 200, and 1000 times. Use
matplotlib to generate a histogram showing the
observed number of heads for each case. Comment on the limit of the
histogram.
(c) Simulate tossing the coin 500 times. Generate a histogram showing the
heads run lengths."""

import random        # Import Random to generate random numericals in the
experiment
```



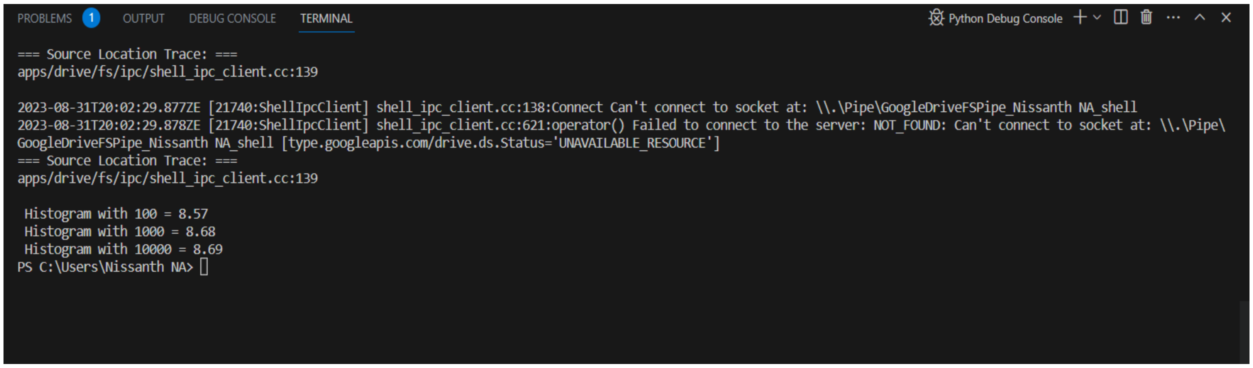
```
import matplotlib.pyplot as plt        #Import Matplotlib Library for
Visualization

PH = 0.7     #Declare Probability of head Biased situation - 0.7
def bernoulli_trial():    #Declare Function Bernoulli trial to generate
random no
    random_num = random.random()    # Generate Random no between 0 and 1
    trial = 1 if (random_num < (PH)) else 0    #Intialize when random
number is less than Ph
    return trial   #Return trial

def coin_toss(num_flips):    #Declare function to find the streak and
headcount
    trials = [bernoulli_trial() for i in range(num_flips)]   #Generate a
list of outcomes (trials) for a biased coin toss using the bernoulli_trial
function
    headscount= sum(trials)   #Calculating heads from sum
    longest_run = 0  #Initalize streak to 0
    current_run = 0  #Initialize cureent run to 0
    for j in trials:  #For loop to find the Head count and longest run
        if j == 1:
            current_run += 1
        else:
            longest_run = max(current_run, longest_run)
            current_run = 0

    return (headscount, longest_run)   #Return the variables associated
with the function

num_flips = 50   # To find the longest run of heads intialize no of trials
headscount, longest_run = coin_toss(num_flips)   # count the number of
heads and find the longest run
print(f'Number of Heads = {headscount}, Longest Run of Heads =
{longest_run} for {num_flips} tosses of a coin.')   # Print function

trials_list = [20, 100, 200, 1000]   # Initialize trial counts
num_heads_list = []   #List declaration for the count
for i in trials_list:   #Iterating through Trials list
    num_head_list = []   #List decalrtion for Indivdual trial count
    for j in range(i):
```



```
        head, i = coin_toss(num_flips)
        num_head_list.append(head)
    num_heads_list.append(num_head_list)


# Histogram for the trials list function
for k in range(len(trials_list)):
    plt.figure()
    plt.hist(num_heads_list[k])
    plt.xlabel('Number of Heads')
    plt.ylabel('Count')
    plt.title(f'Histogram for {trials_list[k]} repeats for {num_flips}')
    plt.show()

def coin_toss1(trials):  #Function to return the head runs length for 500
trials
    head_prob = 0.7   #Initialize biased coin head count to 0.7
    count = 0     # Initialize count to 0
    run_lengths = []    #Create empty list for head runs lengths

    for i in range(trials):
  if random.random() < head_prob:    ##Check if Probability of head is
greater than random values from 0 to 1
            count += 1   #Increment count by 1
        else:
            run_lengths.append(count)
            count = 0

    if count > 0:    #If count is greater than 0 append count to
run_lenghts
        run_lengths.append(count)

    return run_lengths

# Declaring no of trials and use new function to return lengths
num_trials = 500
run_lengths = coin_toss1(num_trials)

# Histogram for the Longest head function
print(f"Number of heads in {num_trials} trials: {headscount}")
print(f"Longest run of heads: {longest_run}")
plt.hist(run_lengths)
plt.xlabel('Heads Run Length')
plt.ylabel('Count')
plt.show()
```

2. Define the random variable $N = \min\{n : \sum_{i=1}^{n} X_i > 4\}$ as the smallest number of standard uniform random samples whose sum exceeds four. Generate a histogram using 100, 1000, and 10000 realizations of N. Comment on the expected value E[N].

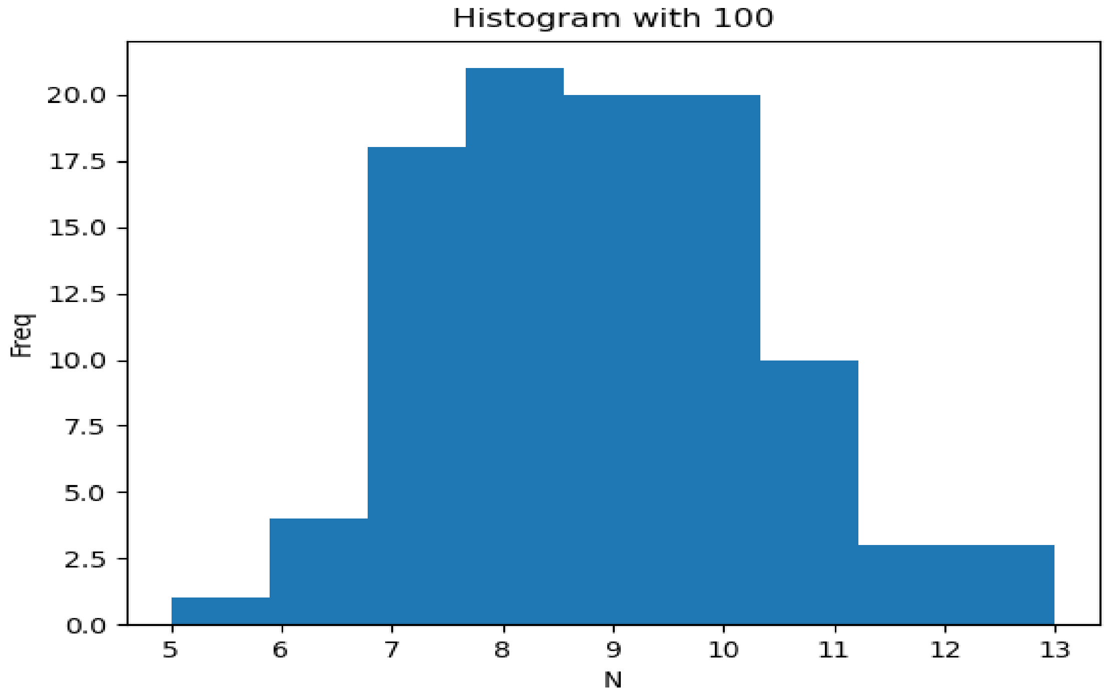
Solutions:



```
=== Source Location Trace: ===
apps/drive/fs/ipc/shell_ipc_client.cc:139

2023-08-31T20:02:29.877ZE [21740:ShellIpcClient] shell_ipc_client.cc:138:Connect Can't connect to socket at: \\.\Pipe\GoogleDriveFSPipe_Nissanth NA_shell
2023-08-31T20:02:29.878ZE [21740:ShellIpcClient] shell_ipc_client.cc:621:operator() Failed to connect to the server: NOT_FOUND: Can't connect to socket at: \\.\Pipe\
GoogleDriveFSPipe_Nissanth NA_shell [type.googleapis.com/drive.ds.Status='UNAVAILABLE_RESOURCE']
=== Source Location Trace: ===
apps/drive/fs/ipc/shell_ipc_client.cc:139

 Histogram with 100 = 8.57
 Histogram with 1000 = 8.68
 Histogram with 10000 = 8.69
PS C:\Users\Nissanth NA> 
```

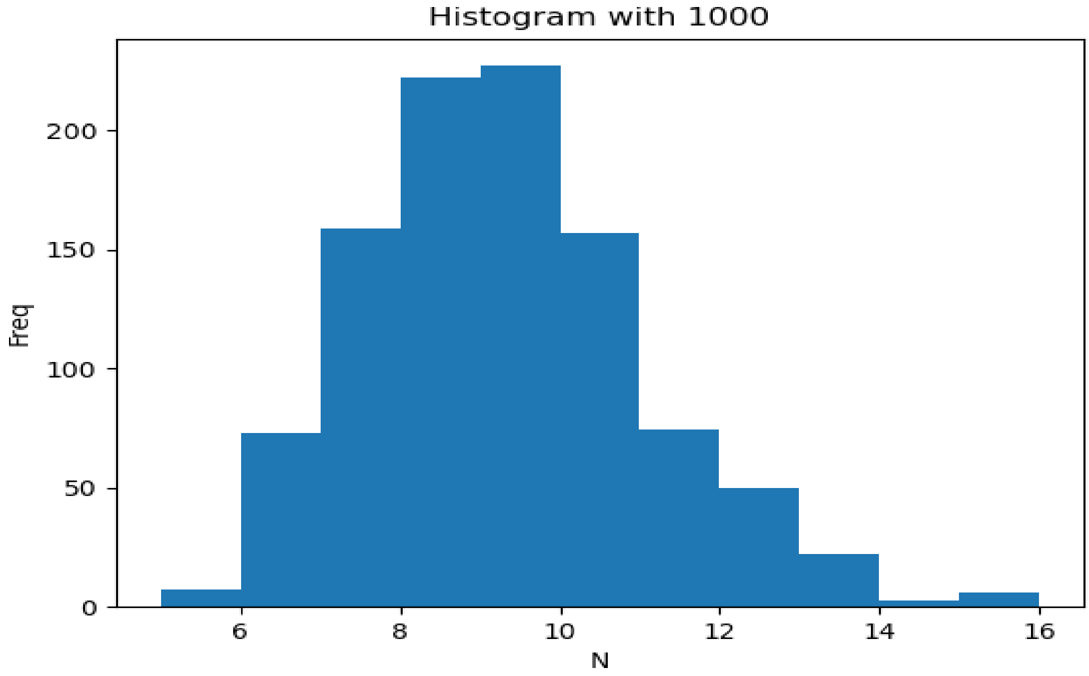
Histogram for 100 Realization:

The Expected value for the Realization is 8.57



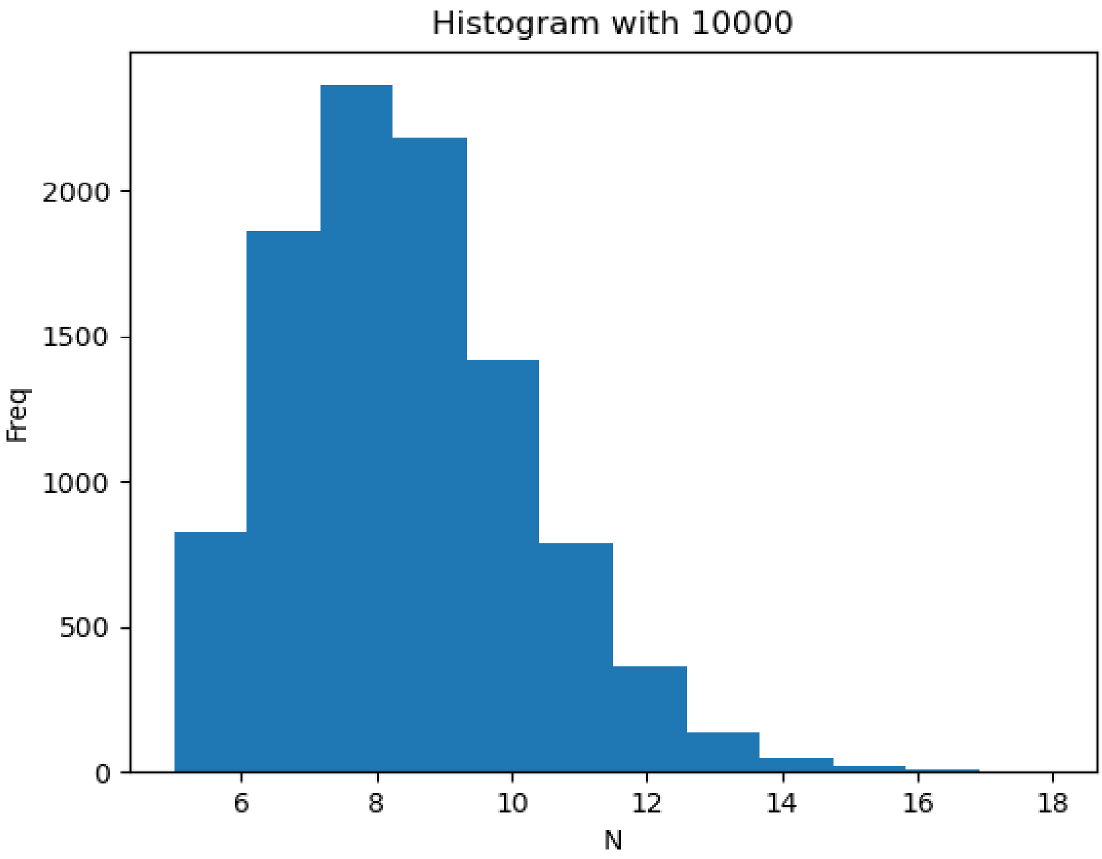
Histogram for 1000 Realization:

The Expected value for the Realization is 8.68



Histogram for 100000 Realization:

The Expected value for the Realization is 8.69

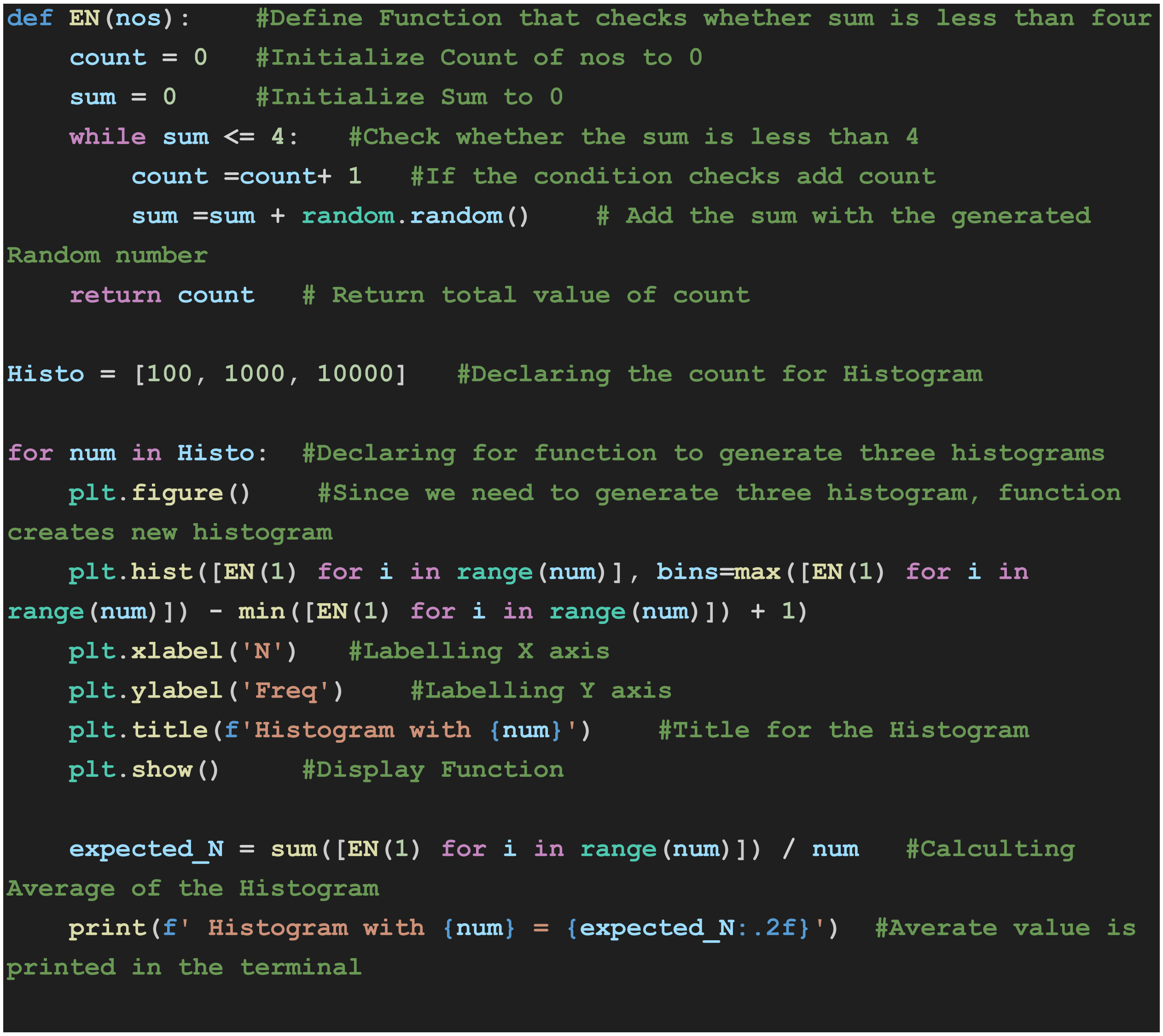


Expected value:

The Expected value of the variables is close to 8.5 to 8.7 for the Histograms 100,1000,10000 variables. The results can be seen graphically.

APPENDIX:

```
"""Define the random variable N = min {n :Pn i=1 Xi > 4} as the smallest
number of standard
uniform random samples whose sum exceeds four. Generate a histogram using
100, 1000, and 10000 realizations of N.
Comment on the expected value E[N]."""

import matplotlib.pyplot as plt   #Import Matplotlib Library for
Visualization
import random  # Import Random to genrate random numericals in the
experiment
```



```
def EN(nos):     #Define Function that checks whether sum is less than four
    count = 0    #Initialize Count of nos to 0
    sum = 0      #Initialize Sum to 0
    while sum <= 4:    #Check whether the sum is less than 4
        count =count+ 1    #If the condition checks add count
        sum =sum + random.random()     # Add the sum with the generated
Random number
    return count    # Return total value of count

Histo = [100, 1000, 10000]    #Declaring the count for Histogram

for num in Histo:  #Declaring for function to generate three histograms
    plt.figure()     #Since we need to generate three histogram, function
creates new histogram
    plt.hist([EN(1) for i in range(num)], bins=max([EN(1) for i in
range(num)]) - min([EN(1) for i in range(num)]) + 1)
    plt.xlabel('N')    #Labelling X axis
    plt.ylabel('Freq')     #Labelling Y axis
    plt.title(f'Histogram with {num}')     #Title for the Histogram
    plt.show()      #Display Function

    expected_N = sum([EN(1) for i in range(num)]) / num    #Calculting
Average of the Histogram
    print(f' Histogram with {num} = {expected_N:.2f}')  #Averate value is
printed in the terminal
```