EE 381 Lab 4 Report

Normal (Gaussian) Distribution

Daniel Duong and Long Nguyen

Instructor: Dr. Duc Tran

Due 12/03/20

Table of Contents

Introduction:

The purpose of this lab was for us to get used to the normal distribution function as well as how the value of mean and sigma will affect the shape of the graph. We also have a chance to understand the central limit theorem through the graph using the provided code and changing the input value to observe the change on the histogram as well as comparing it to the normal distribution function. The last question on the lab helps visualize the distribution of the sum of exponential random variables.

Procedure:

1. **Question 1:**
    a. We created two functions from scratch which are the probability function and the distribution function by the formula provided on the instructions on lab 4.
    b. We generated a random list of x values inclusively from -6 to 6.
    c. We created 6 different lists for the cdf graphs and 6 different lists for the pdf graphs with a subtotal of 12 different lists for different values of variance and mean.
    d. We passed in to the PDF and CDF functions that list of random value of X to calculate PDF and CDF values and saved them into the lists on the step above.
    e. We created 2 different figures, one figure to contain 6 different PDF curves and the other figure containing 6 different graphs for the CDF curves.
    f. The expectation affects normal pdf and cdf graphs by shifting the origin to the left of the x-axis. sigma squared changes the slope/steepness of the graph. The smaller

the sigma squared, the more steep it is. The bigger the sigma squared, the less steep it is.

2. Question 2:

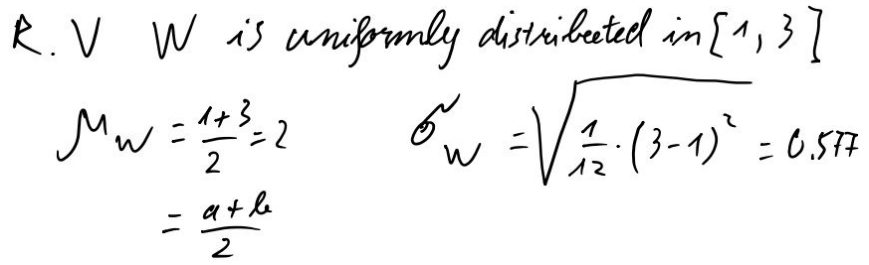
a. We calculated the mean thickness and standard deviation of a single book, and mean thickness and deviation of 5 and 15 books using the table provided in the lab prompt.
b. We copied and pasted the sample code provided in the lab prompt.
c. The only changes that we have made was nbooks to 1, 5, and 15 after each trial to get a better understanding of the standard deviation.

3. Question 3:

a. We have copy and pasted the code from question 2 and made good modifications in order to satisfy the requirements.
b. We call the random.exponetial function while passing the value of while passing in the value of beta = 45 and the number of batteries which is 24.
c. We ran the experiments for 10,000 times and used the values generated from the random.expoenential function to graph the PDF curve.
d. We used the cumulative sum function (cumsum) to graph the values of histogram multiplied with the barwidth to get the CDFvalues. And then we graph the CDF values that we obtained with the class mark of the histogram.

Results and Discussions:

Lab4Q2.py:



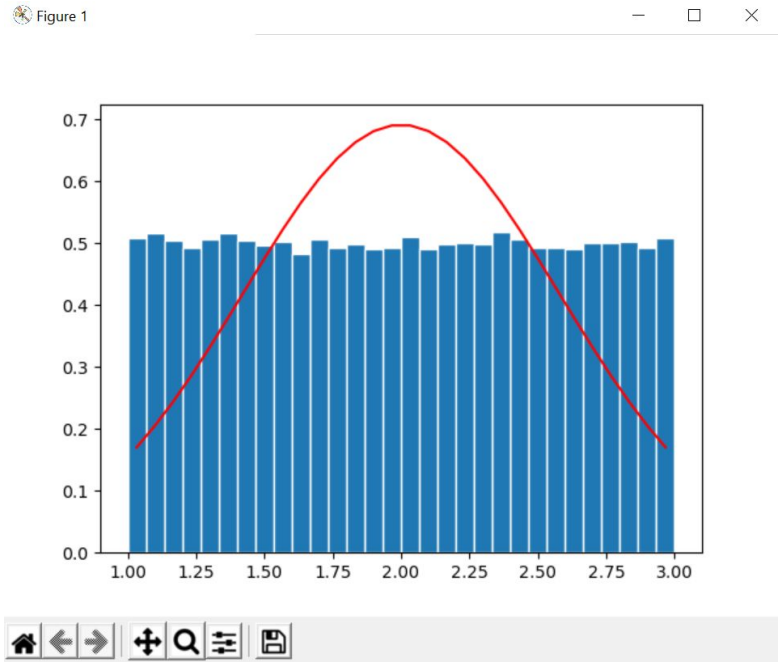
NOTE: This is for one book

Mean thickness of a single book (cm)	Standard deviation of 1 book
μW = 2	0.577

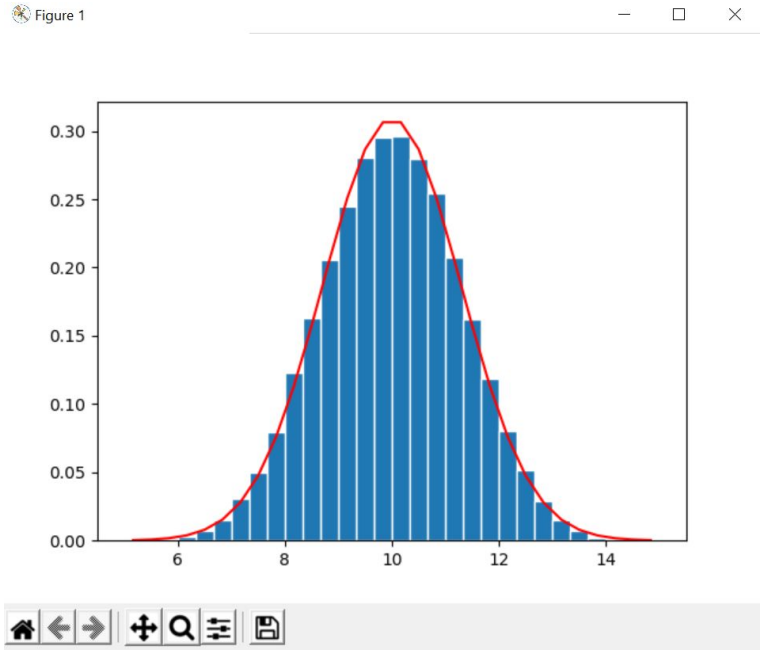
This random variable has a mean $\mu Sn = n\mu$ and a standard deviation of $\sigma Sn = \sigma\sqrt{n}$.

Number of books n	Mean thickness of a stack of n books (cm)	Standard deviation of the thickness for n books
n = 1	2	0.577
n= 5	10	1.290
n= 15	30	2.235

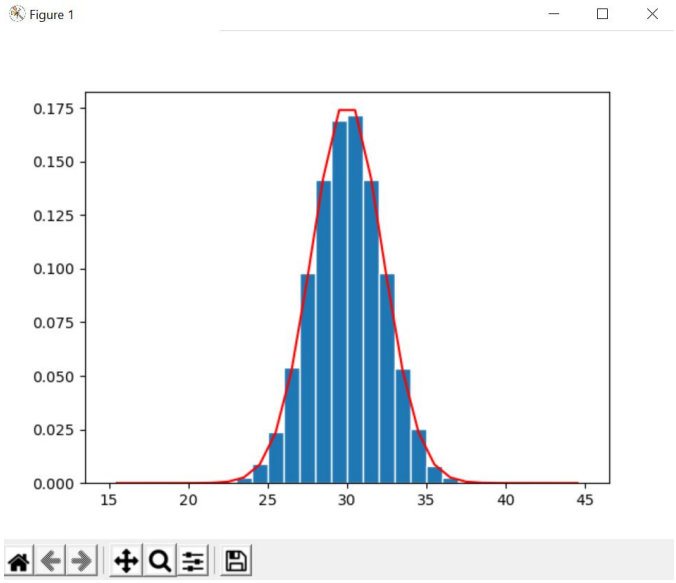
n = 1 book



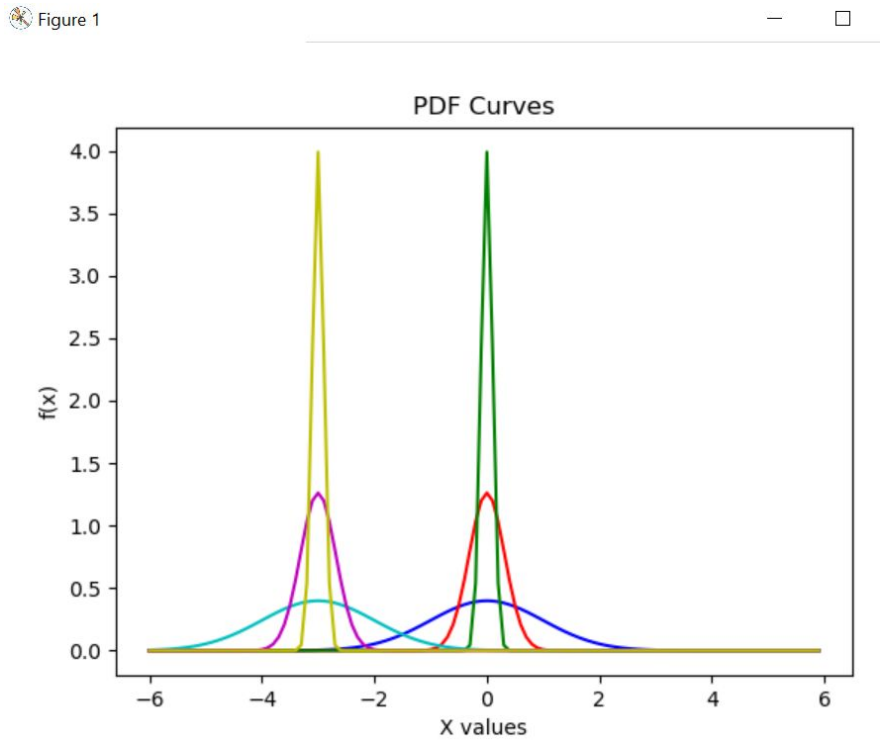
n = 5 books



n = 15 books



Lab4Question1:



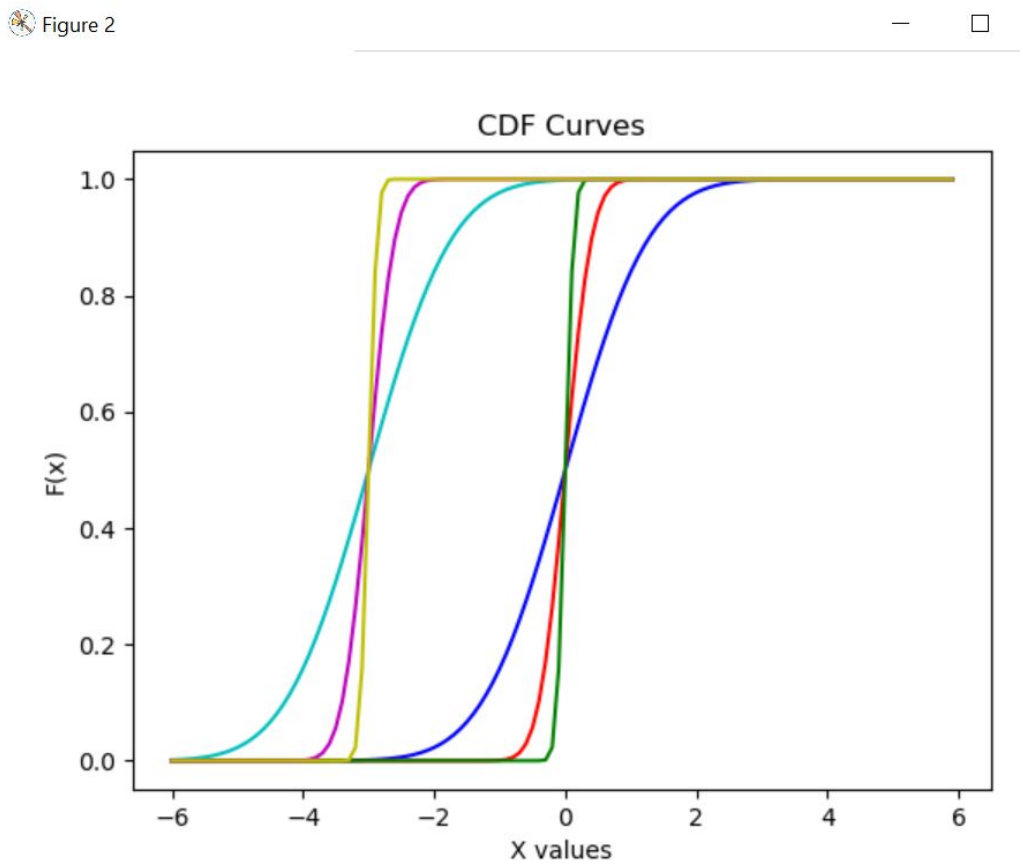
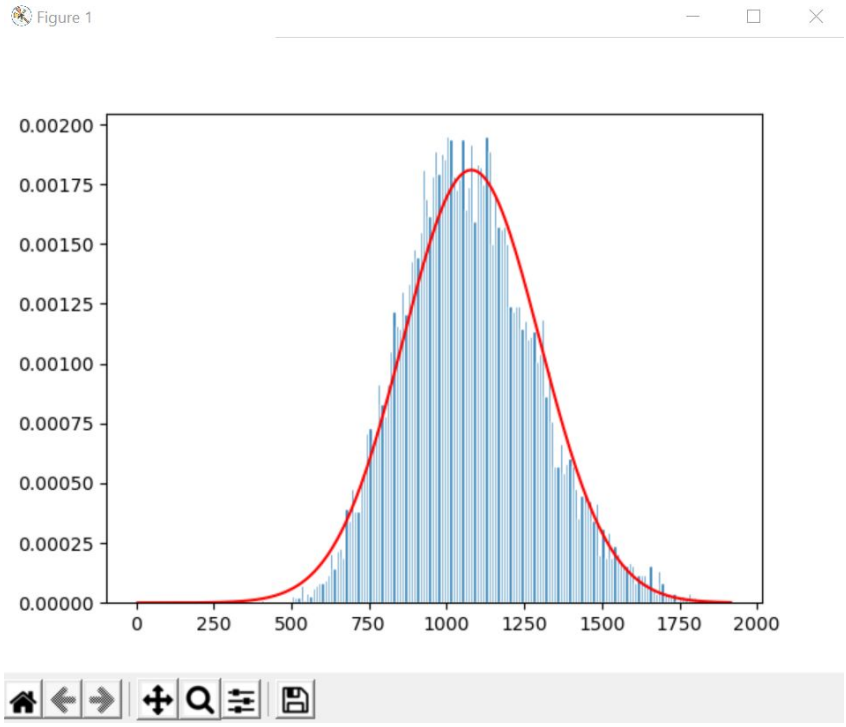


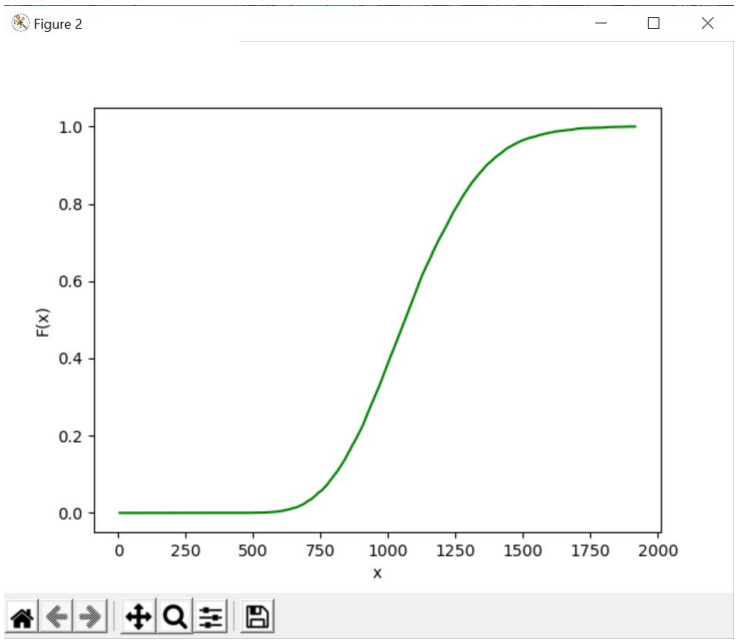
Figure 2
CDF Curves
1.0
0.8
0.6
0.4
0.2
0.0
F(x)
−6
−4
−2
0
2
4
6
X values

Lab4Q3.py:

PDF graph with Normal Distribution



CDF Curve Graph



1. Find the probability that the carton will last longer than three years, i.e. $P(S > 3 * 365) = 1 - P(S \leq 3 * 365) = 1 - F(1095)$. Use the graph of the CDF F(t) to estimate this probability.

Because we tried to get the exact point of 1095, but we could not, we used 1096 instead because it is closest to it. F(1096) = 0.573. 1 - F(1096) = 1 - 0.540 = 0.46.

P(S > 3 * 365) = 1 - P(S <= 3 * 365) = 1 - F(1096) = 1 - 0.540 = 0.46

2. Find the probability that the carton will last between 2.0 and 2.5 years (i.e between 730 and 912 days): $P(730 < S < 912) = F(912) - F(730)$ .Use the graph of the CDF F(t) to estimate this probability.

Because we tried to get the exact point of 730 and 912, we used 731 and 913.

F(731) = 0.049, F(913) = 0.260

P(731 < x < 913) = F(913) - F(731) = 0.260 - 0.049 = 0.211

Conclusion:

For the first question we observe how the value of mean and variance and sigma can affect the normal distribution function. The variance changes the steepness, the mean shifts the position of the graph. Sigma squared changes the slope/steepness of the graph. The smaller the sigma squared, the more steep it is. The bigger the sigma squared, the less steep it is.

We learned how to use two new modules, which are numpy.random.exponential, which generates an amount of certain values exponential probability distribution. The second new module we learned was the numpy.cumsum to calculate the CDF function by multiplying the histogram values by the bar width. It was meant to be the interval of the pdf. For the second question, we only changed the value of the number of books and observed that the number of books increases the steep the graph gets. For question 3, we needed to determine the minimum and maximum battery life prior to conducting our experiment. The way we determined the minimum battery life was by setting it to 0 and the maximum battery life PDF function and tried to take the limit as t approaches infinity; however, on the limit calculator, the limit was 0 which did not tell us anything useful. Thus, we ended up eyeballing the maximum battery life to 80 because it gave us a full graph. At first, when we attempted to graph the CDF function with the number of bins, the graph figure was correct; however, we had to change the x axis from the number of bins to the class mark; thus, leading to the expected results. When we calculated the probability of the cartons, we could not get the exact number on the CDF graph, which led to us having to get the number that was closest to the expected number which ended up being exactly 1 over the original number and took the same procedure as instructed. Our results for the questions were close to the exact numbers which are 0.46 and 0.211.

Appendix

Lab4Q1.py

```
#Authors: Daniel Duong and Long Nguyen
import math
import numpy as np
import scipy.stats
import matplotlib.pyplot as plt

'''pi will be 3.14, e will be 2.718'''
e = 2.71828182845904523536028747135 27
pi = 3.14159265359

def probabilityFunction(x, expectation, sigma):
    exponentEquation = -(pow((x - expectation), 2) / (2 * sigma))
    probX = (1 / math.sqrt(2 * pi * sigma)) * (pow(e, exponentEquation))
    #print(probX)
    return probX

def distributionFunction(x, expectation, sigma):
    distribX = 0.5 * (math.erf((x - expectation) / math.sqrt(2 * sigma))) + 0.5

    return distribX

def main():
    print("Part A")
    probFunc = probabilityFunction(0.5, 0.5, 0.5)

    print("The probabilty function is: ", probFunc)

    disFunc = distributionFunction(0.5, 0.5, 0.5)

    print("The distribution function is: ", disFunc)
    print()

    print("Part B")
    probFuncList = []
    probFuncList2 = []
    probFuncList3 = []
    probFuncList4 = []
    probFuncList5 = []
    probFuncList6 = []
```

```
    cdfFuncList = []
    cdfFuncList2 = []
    cdfFuncList3 = []
    cdfFuncList4 = []
    cdfFuncList5 = []
    cdfFuncList6 = []

    xValues = np.arange(-6, 6, 0.1)
    print(np.size(xValues))
    for i in range(0, np.size(xValues)): #second paramter is excluded from the range, so by having
7 as the range, we include 6
        #print(i)

        probFuncList.append(probabilityFunction(xValues[i], 0, 1))
        cdfFuncList.append(distributionFunction(xValues[i], 0, 1))


        probFuncList2.append(probabilityFunction(xValues[i], 0, 0.1))
        cdfFuncList2.append(distributionFunction(xValues[i], 0, 0.1))


        probFuncList3.append(probabilityFunction(xValues[i], 0, 0.01))
        cdfFuncList3.append(distributionFunction(xValues[i], 0, 0.01))


        probFuncList4.append(probabilityFunction(xValues[i], -3, 1))
        cdfFuncList4.append(distributionFunction(xValues[i], -3, 1))


        probFuncList5.append(probabilityFunction(xValues[i], -3, 0.1))
        cdfFuncList5.append(distributionFunction(xValues[i], -3, 0.1))


        probFuncList6.append(probabilityFunction(xValues[i], -3, 0.01))
        cdfFuncList6.append(distributionFunction(xValues[i], -3, 0.01))



    print("probFunctionList: ", probFuncList)
    print("size of probFunctionList: ", len(probFuncList))
    print("cdfFuncList: ", cdfFuncList)
    print("size of cdfFuncList: ", len(cdfFuncList))
    print("probFunctionList2: ", probFuncList2)
    print("size of probFunctionList2: ", len(probFuncList2))
    print("cdfFuncList2: ", cdfFuncList2)
```

```
    print("size of cdfFuncList2: ", len(cdfFuncList2))
    print("probFunctionList3: ", probFuncList3)
    print("size of probFunctionList3: ", len(probFuncList3))
    print("cdfFuncList3: ", cdfFuncList3)
    print("size of cdfFuncList3: ", len(cdfFuncList3))

    #plt.subplot(221)
    figure1 = plt.figure(1)
    plt.plot(xValues,probFuncList, 'b') #pass in the xValues and list of probabilities given by the x
values
    plt.plot(xValues, probFuncList2, 'r')
    plt.plot(xValues, probFuncList3, 'g')
    plt.plot(xValues, probFuncList4, 'c')
    plt.plot(xValues, probFuncList5, 'm')
    plt.plot(xValues, probFuncList6, 'y')
    plt.title("PDF Curves")
    plt.xlabel("X values")
    plt.ylabel("f(x)")

    figure2 = plt.figure(2)
    plt.plot(xValues, cdfFuncList, 'b')
    plt.plot(xValues, cdfFuncList2, 'r')
    plt.plot(xValues, cdfFuncList3, 'g')
    plt.plot(xValues, cdfFuncList4, 'c')
    plt.plot(xValues, cdfFuncList5, 'm')
    plt.plot(xValues, cdfFuncList6, 'y')
    plt.title("CDF Curves")
    plt.xlabel("X values")
    plt.ylabel("F(x)")

    print("The expectation affects normal pdf and cdf graphs by "
        "shifting the origin to the left of the x-axis"
        "sigma squared changes the slope/steepness of the graph"
        "The smaller the sigma squared, the more steep it is"
        "The bigger the sigma squared, the less steep it is ")

    plt.show()

main()
```

Lab4Q2.py

```
#Authors: Daniel Duong and Long Nguyen

'''pi will be 3.14, e will be 2.718'''
e = 2.71828182845904523536028747135 27
pi = 3.14159265359

import math
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
# Generate the values of the RV X

'''def normalDistrib(x, mean, sigma):
 exponentEquation = -(pow((x - mean), 2) / (2 * sigma * sigma))
 probNormal = (1 / (sigma * math.sqrt(2 * pi))) * (pow(e, exponentEquation))
 #print(probNormal)
 #print(type(probNormal))
 return probNormal'''


N=100000; nbooks=15; a=1; b=3;
mu_x=(a+b)/2 ; sig_x=np.sqrt((b-a)**2/12) #calculated the thickness of one book, mean and std
dev of one book
X=np.zeros((N,1)) #numpy array returning an array of zeros
print("X is: ", X)
print("X has ",np.size(X), " elements")
#sumofBigX = [] #we did a list to copy the big X's np array's value to work around the np array

for k in range(0,N): #100000 experiments
 x=np.random.uniform(a,b,nbooks) #generates books with random thicknesses within the range
from 1 to 3 inclusively
 #print("size of little x is: ", np.size(x))
 #print("little x is: ", x)
 w=np.sum(x) #w is the sum of all thicknesses of the books
 #print("w is: ", w)
 X[k]=w #populates the values of thicknesses of one experiment into an numpy array
 #sumofBigX.append(w)
 #X is the summation of the little x
 #print("Type of Big X is: ", type(X))
 #print("Big X is: ", X)
```

```
 #print("Big X size is: ", np.size(X))
# Create bins and histogram
nbins = 30; # Number of bins, the number of bars you will graph between the first and second parameters of bins
edgecolor='w'; # Color separating bars in the bargraph
#
bins=[float(x) for x in np.linspace(nbooks*a, nbooks*b,nbins+1)] #generating graph bars
#print("bins: ", bins)
h1, bin_edges = np.histogram(X,bins,density=True) #h1 is the array of probability of each thickess, bin_edges is the x label of the bar graph
#print("h1 is : ", h1)
#print("bin edges is: ", bin_edges)
# Define points on the horizontal axis
be1=bin_edges[0:np.size(bin_edges)-1] #the left side of each graph bar
print("be1 is: ", be1)
print("size of be1 is: ", np.size(be1))
be2=bin_edges[1:np.size(bin_edges)] #the right side of each graph bar
print("be2 is: ", be2)
print("size of be2 is: ", np.size(be2))
b1=(be1+be2)/2 #class mark of bin edges 1 and 2
print("b1 is: ", b1)
barwidth=b1[1]-b1[0] # Width of bars in the bar graph, subtract the 1st index of b1 by the 0th index of b1
#print("b1[1] is", b1[1])
#print("b1[0] is", b1[0])
#print("barwidth is: ", barwidth)
plt.close('all')
# PLOT THE BAR GRAPH
fig1=plt.figure(1)
plt.bar(b1,h1, width=barwidth, edgecolor=edgecolor)
#PLOT THE GAUSSIAN FUNCTION

'"print("sumOfBigX[0] is: ", sumofBigX[0])
print("X[0] is: ", X[0])

if(sumofBigX[0] == X[0]):
  print("copied successfully")

else:
 print("some error")"'


'"yValues = []
meanList = [2, 10, 30]
sigmaList = [0.577, 1.290, 2.235]
```

```
#test = normalDistrib(X[0], meanList[0], sigmaList[0])
#print("test is: ", test)

for l in range(0, N):
  yValues.append(normalDistrib(sumofBigX[l], meanList[0], sigmaList[0]))

print("yValues is: ", yValues)'"

def gaussian(mu,sig,z):
 f=np.exp(-(z-mu)**2/(2*sig**2))/(sig*np.sqrt(2*np.pi))
 return f

f=gaussian(mu_x*nbooks,sig_x*np.sqrt(nbooks),b1)
plt.plot(b1,f,'r')
#plt.plot(sumofBigX, yValues, 'm')

'"fig2 = plt.figure(2)
plt.plot(sumofBigX, yValues, 'm')"'

plt.show()
```

Lab4Q3.py

```
#Authors: Daniel Duong and Long Nguyen

import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import math
# Generate the values of the RV X
N=10000; numBatteries=24;
#a=1; b=3,
batteryMinLife = 0; batteryMaxLife = 80
#we tried to take the limit of the function to infinity; however, the max is still 0 so it doesn't
make sense
beta = 45
mu_x= beta
sig_x= beta

mu_c = numBatteries * beta
#print("mu_c: ", mu_c)
sig_c = beta * math.sqrt(numBatteries)
#print("sig_c: ", sig_c)

X=np.zeros((N,1))
print("X size: ", np.size(X))
```

```
for k in range(0,N):
 x=np.random.exponential(beta, numBatteries)
 #print("x: ", x)
 w=np.sum(x) #takes the sum of the battery life for each battery
 #print("w: ", w)
 X[k]=w
# Create bins and histogram
nbins= 200; # Number of bins
edgecolor='w'; # Color separating bars in the bargraph
#
bins=[float(x) for x in np.linspace(numBatteries*batteryMinLife,
numBatteries*batteryMaxLife,nbins+1)]
h1, bin_edges = np.histogram(X,bins,density=True)
# Define points on the horizontal axis
be1=bin_edges[0:np.size(bin_edges)-1]
be2=bin_edges[1:np.size(bin_edges)]
b1=(be1+be2)/2 #class mark
barwidth=b1[1]-b1[0] # Width of bars in the bargraph
plt.close('all')
# PLOT THE BAR GRAPH
fig1=plt.figure(1)
plt.bar(b1,h1, width=barwidth, edgecolor=edgecolor)
#PLOT THE GAUSSIAN FUNCTION
def gaussian(mu,sig,z):
 f=np.exp(-(z-mu)**2/(2*sig**2))/(sig*np.sqrt(2*np.pi))
 return f

f=gaussian(mu_c,sig_c,b1) #f is the line of the graph

cdfFunc = np.cumsum(h1 * barwidth)
#print(np.size(cdfFunc)) #cumsum has 30 elements

#print("cdf: ", cdfFunc)

plt.plot(b1,f,'r')

fig2 = plt.figure(2)
plt.xlabel("x")
plt.ylabel("F(x)")
plt.plot(b1, cdfFunc, 'g')

plt.show()
```