NE 334 Assignment 1
Due October 8, 2024

The purpose of this assignment is to gain experience with numerical simulations and data analysis. The required Python scripts are available on the UW Learn page, in the same content section this Assignment PDF file was found. You can run the calculations on your own computer. You will need to have Python installed in your computer. Upload a *single* PDF file to the learn dropbox for Assignment 1.

1. **Normal distribution**. The python script normal.py samples the normal distribution for a random variable x. Open the script in a text editor to modify calculation parameters. After running the script, a file named xnormal.dat is created and contains the sampled data. The input parameters are xmean, standarddeviation, and sample_size.

   (a) Modify the input parameters in normal.py and calculate the average value of x: $\langle x\rangle$. This can be done by either modifying the python script to compute and print the average value, or by importing the xnormal.dat datafile using tools such as MS Excel. Increase the value of samplesize and recompute $\langle x\rangle$. What happens to $|\langle x\rangle - \text{xmean}|^2$ as you increase samplesize? Report all your input parameters. Be careful as not to increase samplesize too much as each real number stored in computer RAM occupies 8 bytes and each character saved in the x_normal.dat files requires 1 byte of disk storage.

```
f=open('x_normal.dat','w')
for x in x_norm:
  f.write(str(x)+'\n')
  x_sum += x
f.close()
x_av = x_sum/sample_size
```

Sample Result when samplesize = 10000:

x_av=0.9527696078914627

The highlighted lines are the added code to calculate the average.

When the sample size is increased, the <x> (average) gets closer to the x mean (1). And $|\langle x\rangle - \text{xmean}|^2$ gets closer to zero.

Sample Result (after changing to 50000 results)

x_av=1.0098024509335186

x_diff =9.608804430403995e-05

   (b) Choose input parameters and compute the variance of x, σ_x^2, based on the sampled data. This can be done by modifying the python script or by analyzing the data in a program such as MS Excel. Also calculate the standard deviation and compare your result to the input parameter standarddeviation. Report all your input parameters.

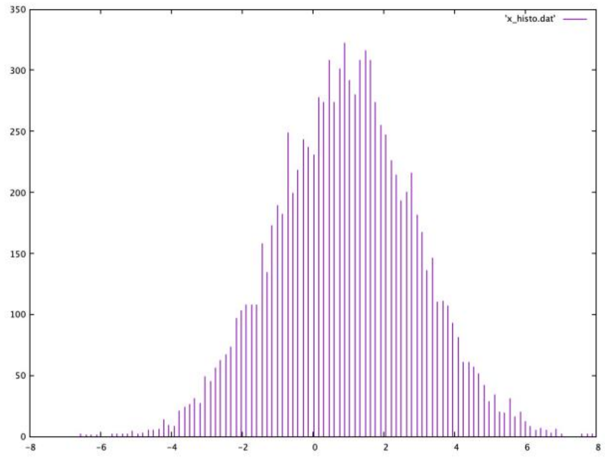
```
import statistics
x_std = statistics.stdev(x_norm)
```

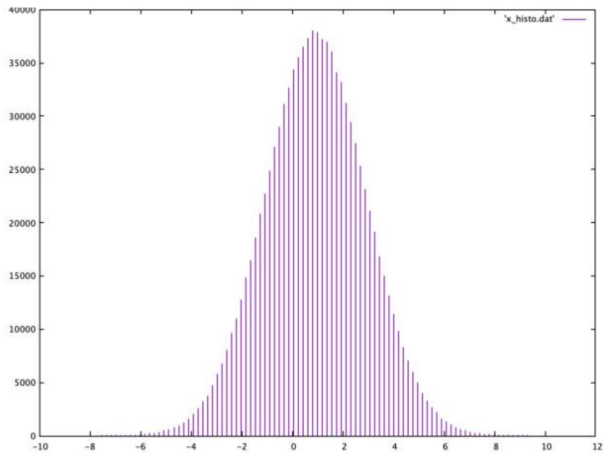
Sample Result samplesize = 10000:

x_std = 1.9771595672207578

The calculated variance is very close to the input parameter (2)

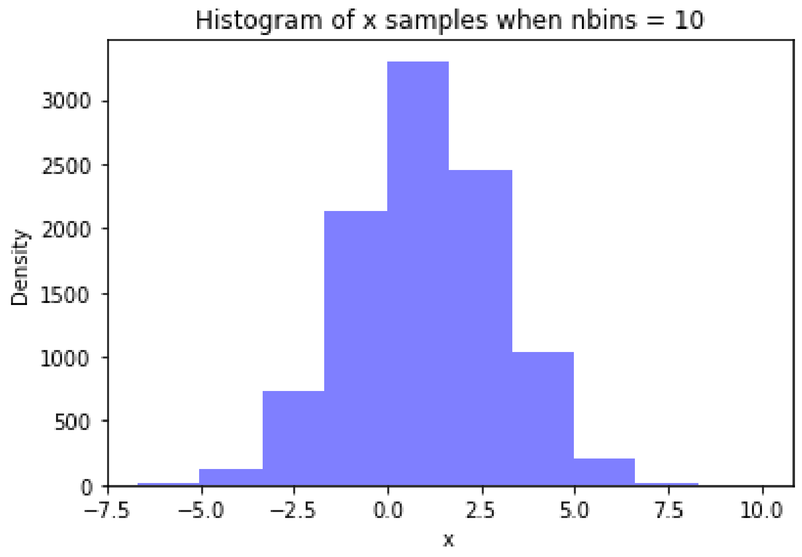
(c) The file xhisto.dat contains the histogram of *x* sampled from the normal distribution. Plot the histogram of *x* for different values of samplesize and your choice of input parameters. The graphs should look like this:

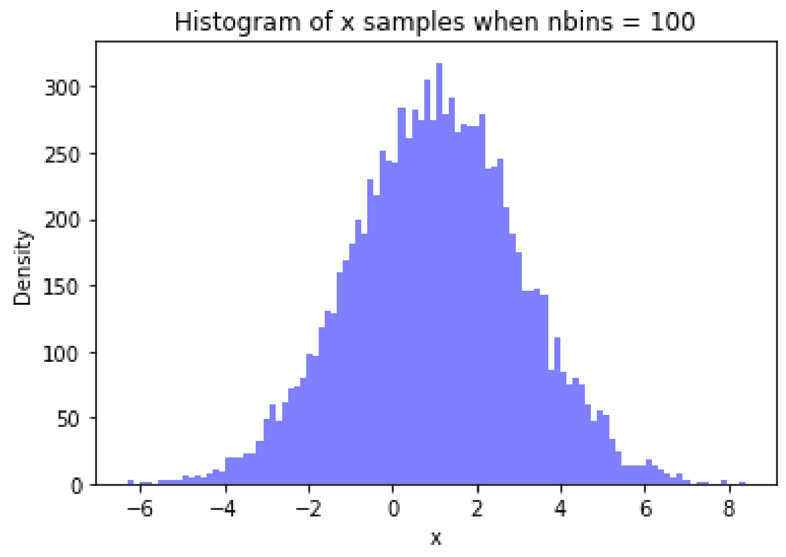


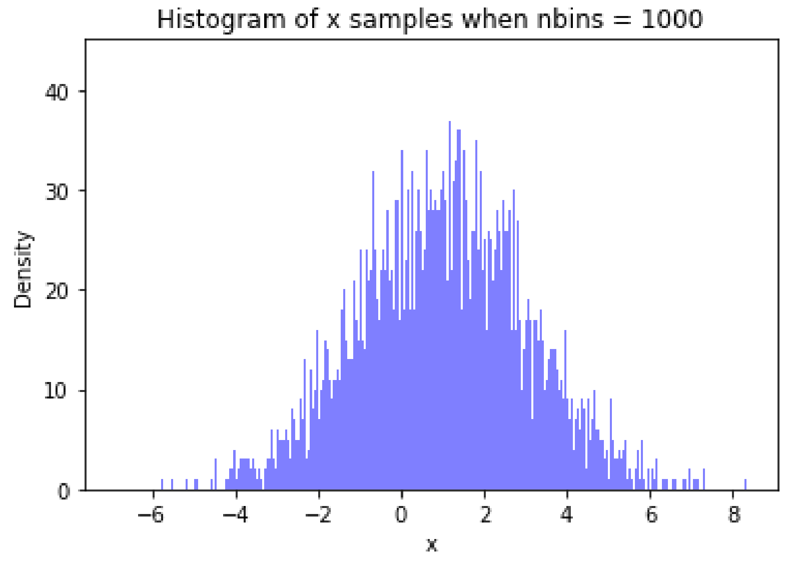


The script parameters nbins can be adjusted to generate as smoother graph. When samplesize is smaller, nbins should also be smaller. Report all your input parameters.

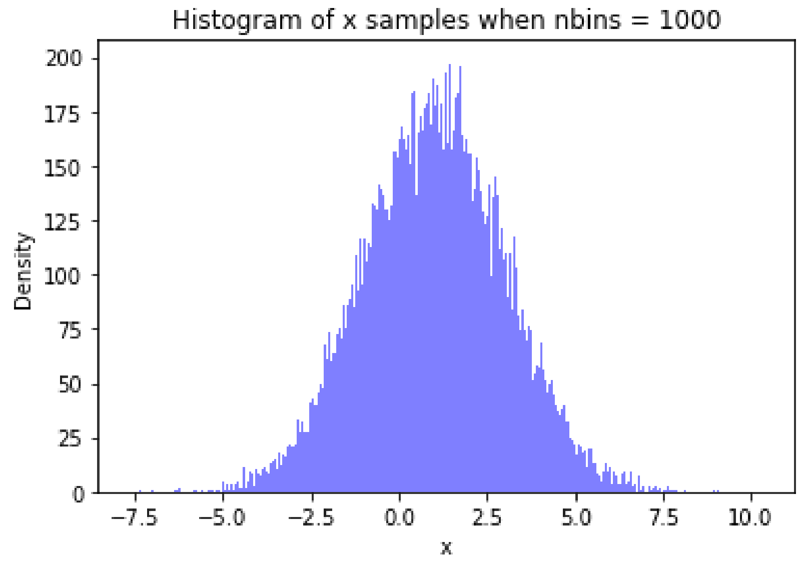
For samplesize = 10000







For samplesize = 50000



2. **Uniform distribution**. The python script uniform.py samples a uniform distribution of random numbers in the range $0 \le x \le 1$. A file named xuniform.dat is created and contains the sample data.

   (a) What do you expect $\langle x \rangle$ to be?

   I expect it to be close to 0.5

   (b) Use a text editor to modify the input parameters in uniform.py and calculate $\langle x \rangle$. How does $\langle x \rangle$ change when sample_size increases?

   Import statistics

   x_mean = statistics.mean(x_uniform)

   Sample Result

   x_mean = 0.5010676152809138

   When the samplesize increases, the x_mean approaches 0.5

   Sample Result:

   sample_size = 100000

   x_mean = 0.4999470723962127

   (c) Modify the script to generate random numbers between -1 and 3. Include your script with your submission.

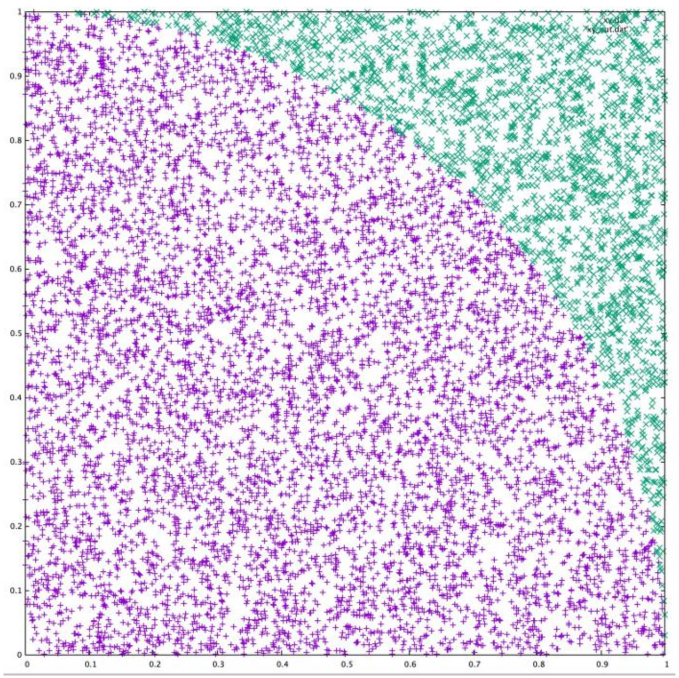
   import numpy as np

   x_uniform=np.random.uniform(-1,3,sample_size)

   Sample Result

   x_mean = 1.0057748669228945

3. **Can you calculate** π ? The python script pi.py samples uniform distributions for two sets of random numbers, $0 \le x \le 1$ and $0 \le y \le 1$. The total number of samples is samplesize. The code also reports the number of data points, countcircle, for which $x^2 + y^2 \le 1$ and those data points are stored in xy.dat. Below is a graph of the sample points in the (x,y) plane. The purple points are those for which $x^2 + y^2 \le 1$ holds true. The points in green are those for which the $x^2 + y^2 \le 1$ condition is NOT met. Those points are stored in the file xyout.dat.



(a) Devise an approach to estimate the value of π based on the data generated by running pi.py. Describe your procedure.

approx_pi=(count_circle/float(sample_size))*4

To approximate pi, the number of points that satisfy the condition is divided by the total number of points, then multiplied by 4. The points that satisfy the condition are inside the quarter circle, and the quotient has to be multiplied by 4 since it's only a quarter of the circle.

(b) Report your estimate of π for a series of samplesize values.

For samplesize = 10000

3.1232

For samplesize = 50000

3.1404

For samplesize = 100000

3.14104

With increasing the sample size, the pi approximation becomes closer to the correct value

(c) Estimate the standard error of your computed π value.
Hint: for N independent samples, the standard error of the mean of property A, $\langle A \rangle$, is defined as $\sigma_A/\sqrt{N}$.

err = (approx_pi-math.pi)/np.sqrt(sample_size)

Sample Result:

```
0.0003440734641020704
```

NOTE: You can use matplotlib to make your plots if you wish. This requires modifying the python scripts.