ANALYSIS OF DISCRETE RANDOM VARIABLES

PROBLEM STATEMENT

- Genrate a list of values drawn from the distribution of Uniform random variable.
- Compute the sample mean and sample variance and compare it to the mean, μ and variance, σ_X^2. Estimate the (sample) variance of the sample mean (based on the Central Limit Theorem) for N=100 and N=10000.
- Repeat the same step with N=100 50 times to create a set of sample means and verify if they follow the normal distribution.
- Also check whether the consecutive values of uniform random numbers are independent or not

THEORETICAL EXPLORATION

- $\textbf{Uniform random variable}$:For a uniform random variable X ~ U(a,b) the equations of mean and variance are as follows:
  - $E[X] = \mu = \frac{(a+b)}{2}$
  - $var(X) = \sigma_X^2 = \frac{a^2+ab+b^2}{12}$
  - Thus here we have a = 0 and b = 1 our values of μ and σ_X^2 are as follows:
    - $\mu = \frac{(0+1)}{2} = 0.5$
    - $\sigma_X^2 = \frac{0^2+0*1+1^2}{12} = 0.0833$
- $\textbf{Central Limit Theorem}$: The central limit theorem states that if we have a population with mean μ and variance σ_X^2 then if we take random samples from that population then the sample means of those samples would be normally distributed with mean μ and variance $\frac{\sigma_X^2}{n}$ where n is the size of each sample.
- $\textbf{Covariance and Correlation}$:
  - COV(X,Y) = E[X,Y] - E[X]*E[Y]
  - CORR(X,Y) = $\frac{COV(X,Y)}{\sigma_{X}\sigma_{Y}}$
  - Covariance and Correlation are measures through which we can see if two random variables are Dependent on each other.

SIMULATION METHODOLOGY

- The first part of the program generates a uniform random sample of size 100 and 10000 using $\textbf{numpy.random.uniform}$.
- The second part calculates the mean and variance of samples with size 100 and 10000. I have also generated a uniform random pdf for N=100 and N=10k and plotted it to make point that, more data means better approximation of the ture pdf.
- The third part includes creating sampling distribution with total number of samples being 50 with sample size N=100. To give better visualization of the normality of sampling distribution another sampling distribution which has more samples than 50 is also created. Both the sampling distributions are plotted using $\textbf{matplotlib.pyplot}$.
- Finally using the same numpy library, covariance is found. To be sure of the independence, correlation is also calculated.

EXPERIMENTS AND RESULTS

1.) X ~ U (0,1) , evaluate the mean, μ and variance, σ_x^2 .

In [17]:

```
import numpy as np
import sklearn as sk
import scipy.stats as sci
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
import matplotlib.mlab as mlab
%matplotlib inline
fontP = FontProperties()
fontP.set_size('small')
```

In [18]:

```
########### Calculating mean of Uniform pdf##################
true_mean = (0+1)/2
```

```
true_variance = (1)/12
true_std = np.sqrt(true_variance)
print("Mean of true uniform random variable is: ",true_mean)
print("Variance of true uniform random variable is: ",true_variance)
```

```
Mean of true uniform random variable is:  0.5
Variance of true uniform random variable is:  0.08333333333333333
```

In [19]:

```
##############Generate Uniform Random variable###############
np.random.RandomState(seed=42)
X = np.random.uniform(0,1,1000)
```

- $E[X] = \mu = \frac{(a+b)}{2}$
- $var(X) = \sigma_X^2 = \frac{a^2+ab+b^2}{12}$
- Thus here we have a = 0 and b = 1 our values of μ and σ_X^2 are as follows:
  - μ = $\frac{(0+1)}{2} = 0.5$
  - σ_X^2 = $\frac{0^2+0*1+1^2}{12} = 0.0833$

μ	σ_X^2
0.5	0.0833

2.) Generate a sequence of N = 100 random numbers between [0,1] and compute the sample mean,sample variance and compare to μ and variance, σ_x^2. Also estimate the (sample) variance of the sample mean (based on the Central Limit Theorem).Repeat for N =10,000.

- There are two things we need to take care of when we are considering a particular statistic of a random variable:
  - Population parameters: These parameters are mean(μ) and variance(σ_X^2). These polupation parameters capture the real underlying mean and variance of the population. Generally speaking they are unknown to us as we will have to take the data of all the entities in order to calculate the true parameters.
    - For example: If we consider the weights of all the males in United states, we will have to measure the weights of each and every persons' weight in order to get the true mean and true variance.
  - Point Estimates: As mentioned above instead of measuring the value of each and every entity and wasting our resources we can rather sample randomly from the population and measure the mean and variance of that sample mean. This is known as sample mean(m) and sample variance(s^2). In the example that I mentioned above a sample can be a finite or small number of males selected independently.
    - The property of this sample mean and variance is that it tends to vary around the true mean(μ) and true variance(σ_X^2). Infact there is a trend in which if the sample size is less then the sample mean(m) tends to be far from the true mean(μ) and if the sample size is more then the sample mean(m) tends to be very close to the true mean(μ). This this is evident from the experiment that we carried above in which we used a small sample of size 100 and then a small sample of size 10000.

In [20]:

```
#############Calculating mean var for N=100#######################
X_100 = np.random.uniform(0, 1,100)
m_100 = np.sum(X_100)/len(X_100)
var_100 = np.sum((X_100-m_100)*(X_100-m_100))/(len(X_100)-1)
std_100 = np.sqrt(var_100)
#print('Mean of 100 random sample is: ',m_100)
#print('Variance of 100 random sample is: ',var_100)
#print('Sample variance of sample mean of size 100 is: ',var_100/100)
```

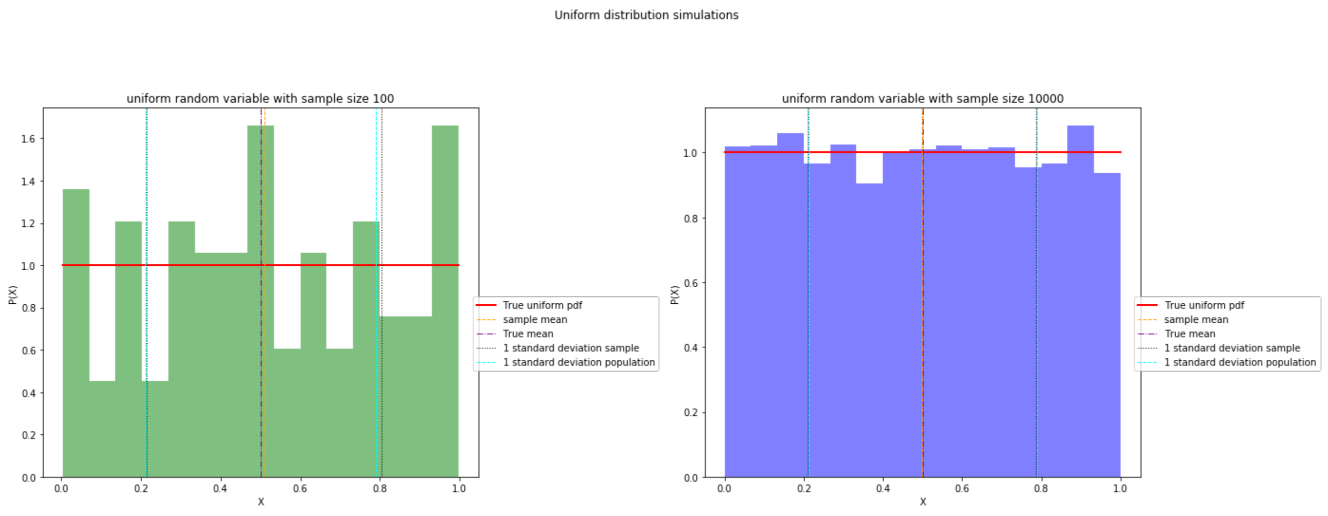
In [21]:

```
#############Calculating mean var for N=10000#####################
X_10k = np.random.uniform(0, 1,10000)
m_10k = np.sum(X_10k)/len(X_10k)
var_10k = np.sum((X_10k-m_10k)*(X_10k-m_10k))/(len(X_10k)-1)
std_10k = np.sqrt(var_10k)
#print('Mean of 10000 random sample is: ',m_10k)
#print('Variance of 10000 random sample is: ',var_10k)
#print('Sample variance of sample mean of size 10k is: ',var_100/10000)
```

```
#print('sample variance of sample mean of size 10k is: ',var_100/10000)
```

In [22]:

```
###################Hist for N=100###################################
f, (ax1, ax2) = plt.subplots(1, 2,figsize=(18,8))
f.suptitle('Uniform distribution simulations')

count, bins, ignored = ax1.hist(X_100,color='green',alpha=0.5,bins=15,density=True)
ax1.plot(bins, np.ones_like(bins), linewidth=2, color='r',label='True uniform pdf')
ax1.axvline(m_100, color='orange', linestyle='dashed', linewidth=1,label='sample mean')
ax1.axvline(true_mean, color='purple', linestyle='-.', linewidth=1,label='True mean')
ax1.axvline(m_100-std_100, color='black', linestyle=':', linewidth=1,label='1 standard deviation
sample')
ax1.axvline(m_100+std_100, color='black', linestyle=':', linewidth=1)
ax1.axvline(true_mean-true_std, color='cyan', linestyle='dashed', linewidth=1,label='1 standard
deviation population')
ax1.axvline(true_mean+true_std, color='cyan', linestyle='dashed', linewidth=1)
ax1.set(xlabel="X",ylabel="P(X)")
ax1.legend(loc=9, bbox_to_anchor=(1.2, 0.5))
ax1.set_title('uniform random variable with sample size 100')

###################Hist for N=10000###################################
count, bins, ignored = ax2.hist(X_10k,color='blue',alpha=0.5,bins=15,density=True)
ax2.plot(bins, np.ones_like(bins), linewidth=2, color='r',label='True uniform pdf')
ax2.axvline(m_10k, color='orange', linestyle='dashed', linewidth=1,label='sample mean')
ax2.axvline(true_mean, color='purple', linestyle='-.', linewidth=1,label='True mean')
ax2.axvline(m_10k-std_10k, color='black', linestyle=':', linewidth=1,label='1 standard deviation
sample')
ax2.axvline(m_10k+std_10k, color='black', linestyle=':', linewidth=1)
ax2.axvline(true_mean-true_std, color='cyan', linestyle='dashed', linewidth=1,label='1 standard
deviation population')
ax2.axvline(true_mean+true_std, color='cyan', linestyle='dashed', linewidth=1)
ax2.set(xlabel="X",ylabel="P(X)")
ax2.legend(loc=9, bbox_to_anchor=(1.2, 0.5))
ax2.set_title('uniform random variable with sample size 10000')
plt.subplots_adjust(left=0.01, wspace=0.52, top=0.8)
plt.show()
```



Results:

sample size	m	s^2	sample variance
100	0.525	0.088	0.00085
10000	0.498	0.084	$8.5* 10^{-6}$
True value	μ = 0.5	σ_X^2 = 0.083	

- As you can see from the figures above the distribution of sample with bigger size(N=10k) is almost equal to the true distribution whereas the distribution with smaller sample size has its mean and standard deviation away from the actual population parameters. This makes sense because as we take in more and more data, the values tend to average themselves and finally when we get all the data the final distribution would be the true underlying one which in our case is the uniform distribution X ~ (0,1).

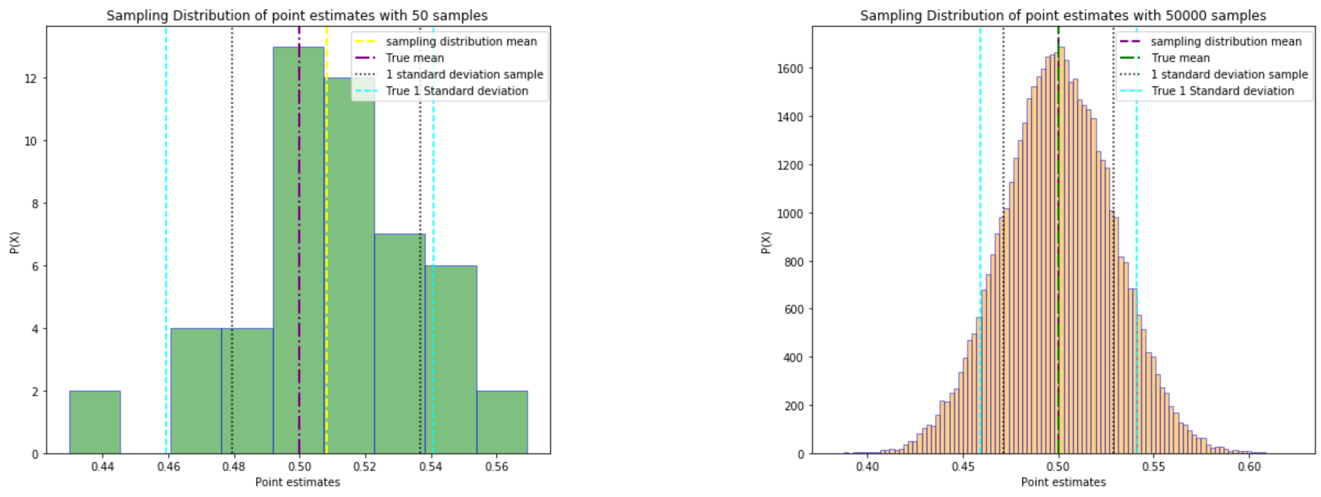
3.) Verify the central limit theorem

In [23]:

```
######################Calculating Point Estimates for N=50 samples########################
point_est = []
for i in range(50):
    X = np.random.uniform(0, 1,100)
    m = np.sum(X)/len(X)
    point_est.append(m)

point_est = np.array(point_est)
mean = np.sum(point_est)/len(point_est)
var = np.sum(np.square(point_est-mean))/(len(point_est)-1)
std = np.sqrt(var)
true_sample_var = true_variance/50
true_sample_std = np.sqrt(true_sample_var)

###########################Plotting distribution for 50 samples###########
f, (ax1, ax2) = plt.subplots(1, 2,figsize=(18,8))
f.suptitle('Sampling Distributions: ')


weight = point_est/np.sum(point_est)
count,bins, ignored = ax1.hist(point_est,bins=9,alpha = 0.5,color = 'green',edgecolor='blue')
ax1.axvline(mean, color='yellow', linestyle='dashed', linewidth=2,label='sampling distribution
mean')
ax1.axvline(true_mean, color='purple', linestyle='-.', linewidth=2,label='True mean')
ax1.axvline(mean-std, color='black', linestyle=':', linewidth=1.5,label='1 standard deviation
sample')
ax1.axvline(mean+std, color='black', linestyle=':', linewidth=1.5)
l1 = 'True 1 Standard deviation'
ax1.axvline(true_mean-true_sample_std, color='cyan', linestyle='dashed', linewidth=1.5,label=l1)
ax1.axvline(true_mean+true_sample_std, color='cyan', linestyle='dashed', linewidth=1.5)
ax1.set(xlabel="Point estimates",ylabel="P(X)")
ax1.legend(loc=9, bbox_to_anchor=(0.8, 1))
ax1.set_title('Sampling Distribution of point estimates with 50 samples')

#print('Mean of point estimate is with samples 50 is: ',mean)
#print('Variance of point estimate with samples 50 is: ',var)

######################Calculating Point Estimates for N=100 samples########################
point_est = []
for i in range(50000):
    X = np.random.uniform(0, 1,100)
    m = np.sum(X)/len(X)
    point_est.append(m)

point_est = np.array(point_est)
mean = np.sum(point_est)/len(point_est)
var = np.sum(np.square(point_est-mean))/(len(point_est)-1)
std = np.sqrt(var)
true_sample_var = true_variance/50
true_sample_std = np.sqrt(true_sample_var)

weight = point_est/np.sum(point_est)

###########################Plotting distribution for 50k samples###########
count,bins, ignored = ax2.hist(point_est,bins=100,alpha = 0.5,color = 'orange',edgecolor='blue')
ax2.axvline(mean, color='purple', linestyle='dashed', linewidth=2,label='sampling distribution
mean')
ax2.axvline(true_mean, color='green', linestyle='-.', linewidth=2,label='True mean')
ax2.axvline(mean-std, color='black', linestyle=':', linewidth=1.5,label='1 standard deviation
sample')
ax2.axvline(mean+std, color='black', linestyle=':', linewidth=1.5)
l2 = 'True 1 Standard deviation'
ax2.axvline(true_mean-true_sample_std, color='cyan', linestyle='dashed', linewidth=1.5,label=l2)
ax2.axvline(true_mean+true_sample_std, color='cyan', linestyle='dashed', linewidth=1.5)
ax2.set(xlabel="Point estimates",ylabel="P(X)")
ax2.legend(loc=9, bbox_to_anchor=(0.8, 1))
ax2.set_title('Sampling Distribution of point estimates with 50000 samples')


#print('Mean of point estimate with sample 50k is: ',mean)
```

```
#print('Variance of point estimate with samples 50k: ',var)

plt.subplots_adjust(left=0.01, wspace=0.52, top=0.8)
plt.show()
```

```
Sampling Distributions:
```



- $\textbf{Insight on CLT}$: The sample mean of a population is distributed normally with mean equal to the population parameter and variance equal to the population varinace divided by the sample size.
  - This makes sense because each of the randomly chosen sample would be capturing some information about the data. Also when we say that a population has mean μ, it means that majority of the values associated to that random variable have value μ or have values around it. To prove this point we will look at the pdf of gaussian normal distribution which is given by: $$P_X(x) = \frac{1}{\sqrt(2\pi\sigma)}e^\frac{-(X-\mu)^2}{2\sigma^2}$$
    - We can see that when X = μ the exponent term is 0 and hence the pdf has maximum value where as for all values of X greater and smaller that μ the pdf reduces exponentially. This means that for any element randomly chosen from the population, its probability of being close to the true mean is highest. As a result of this from the n samples that we randomly choose from the population, most of them should have values close to μ. This means that the mean of these randomly sampled data would also be near the true mean. Thus if we keep on taking more and more samples we will get more and more samples whose mean value value near to μ. This finally results in a sampling distribution which has a bell curve and its parameters are μ and variance $\frac{\sigma_X^2}{n}$.
    - It should be noted that the sample variance reduces as the number of sample increases. This is because as we take more and more samples we include data from the population. Since we are talking about the sampling distribution of the means, as the number of samples increases we get get more point estimates which are near the true mean. Thus the variance of the sampling distribution decreases.

total samples(N = 100)	sampling dist mean	sampling dist variance
50	0.499	0.00090
50000	0.50	0.00083
Sampling values(N=100)	0.50	0.00085

- The samplling distribution with 50 samples and N=100 values each has a normal distribution.
- As we can see the values of mean and sample variance are similar to the ones calculated before. We calculated the mean and sample variance for N=100 before which are almost equal to the mean and variance of this sampling distribution.
- It is not evident from the first plot that the sampling distribution has a normal shape and hence I have simulated a sampling distribution which has 50000 samples each of size 100. Such a sample clearly shows us that as we increse the number of samples this distribution tends to go towards the normal distribution.

4.) We want to check whether there is any dependency between X_i and X_{i+1}

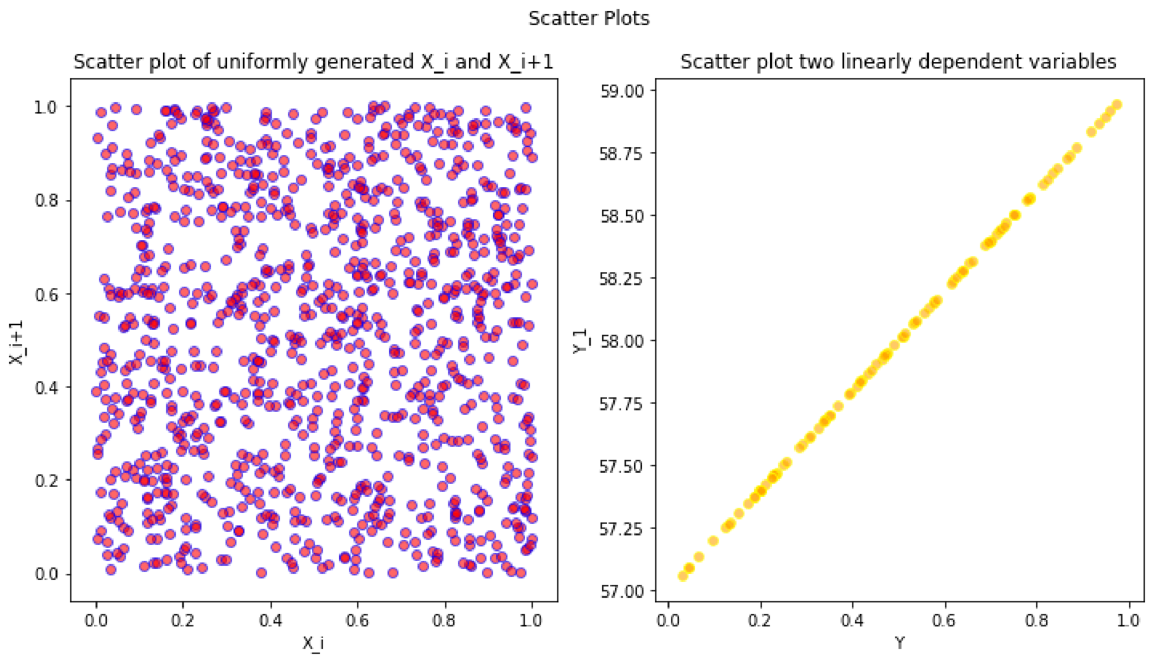
- We consider the samples to be equally likely.
- Since the sample is drawn of a uniform independent sample we expect that the covariance would be zero or very less. Covariance basically measures the linear relationship between two random variables. However two independent variables would have 0 covariance but a zero covariance does not imply that the random variables are independent. As a result of this we will also calculate the correlation whose range is between [-1,1]. If the value of correlation is 1 or -1 then there is linear

we will also calculate the correlation whose range is between [-1,1]. If the value of correlation is 1 or -1 then there is linear relationship between the two random variables and if the value is 0 then there is no linear relationship between the random variables.

- COV(X_i,X_{i+1}) = E[X_i,X_{i+1}] - E[X_i]* E[X_{i+1}]
- COV(X_i,X_{i+1}) = $\sum_{N=1}^{\infty} \frac{X_i*X_{i+1}}{N}$ - $\sum_{N=1}^{\infty} \frac{X_i}{N}* \sum_{N=1}^{\infty} \frac{X_{i+1}}{N}$
- CORR(X_i,X_{i+1}) = $\frac{COV(X_i,X_{i+1})}{\sigma_{X_i}\sigma_{X_{i+1}}}$

In [24]:

```python
#######Calculating Covariance and Correlation############
X = np.random.uniform(0,1,1001)
A=0
for i in range(1000):
    A = A + X[i]*X[i+1]
A_final = A/1000
B = (np.sum(X)-X[1000])/1000
C = (np.sum(X)-X[0])/1000
Z = A_final - (B*C)
X_i = X[0:1000]
X_i1 = X[1:1001]
std_Xi = np.std(X_i)
std_Xi_1 = np.std(X_i1)
#print("value of Covariance Z is: ",Z)
#print("Correlation is :",(Z)/(std_Xi*std_Xi_1))

#Show linear relationship
Y = np.random.uniform(0,1,100)
Y_1 = 2*Y + 57

#################Plotting scatter plots########################
f, (ax1, ax2) = plt.subplots(1, 2,figsize=(12,6))
f.suptitle('Scatter Plots')
ax1.scatter(X_i,X_i1,color='red',alpha=0.6,edgecolors='blue')
ax1.set(xlabel='X_i', ylabel='X_i+1')
ax1.set_title('Scatter plot of uniformly generated X_i and X_i+1')

ax2.scatter(Y,Y_1,color='orange',alpha=0.6,edgecolors='yellow')
ax2.set(xlabel='Y', ylabel='Y_1')
ax2.set_title('Scatter plot two linearly dependent variables')

plt.show()
```



Results:

covariance(Z)	correlation
0.001	0.016

- In order to show that our sample is highly uncorrelated and is independent we have created a graph of a variable which is Linearly dependent on the other.
  - The equation is: Y_1 = 2*Y + 57
  - Here Y is a uniform random sample and Y_1 is linearly dependent on Y as per the above equation.
  - As we can see from the above graphs the linearly dependent variable has a linear relationship as it is dependent on the other variable Y. The variables that we have considered i.e the random variables that take consecutive values of uniform random variable have their scatter plots all over the place which suggests that these random variables are highly uncorrelated and independent.
- As we can see from the above results the correlation and covariance, both are very near to zero. Thus we confirm our fact that the random number generator indeed generates numbers which are independent of each other and have no liner relationship between them.

Conclusion

- The accuracy of capturing the population parameters is directly proportional to the size N of the sample.
- The sample mean and sample variance follow a normal distribution and the value of the mean of those sample means is equal to the value of the population mean.
- The random number generator used by python generates values which are not correlated to each other.