Hand-Made Central Limit Theorem

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sts
import math
%matplotlib inline
```

Central Limit Theorem says that if we have a distribution $F(X)$ with expected value μ and dispersion σ^2, and we select n i.i.d. random variables $X_1, X_2, \ldots, X_n$, then their mean $\acute{X}$ is distributied normally with dispersion σ^2/n and mean μ:

$$\acute{X}_n \approx N(E\mu, \sigma^2/n)$$

Let us check if it really so for the case when $F(X)$ is a uniform distribution:

Define uniform distribution from a to b:

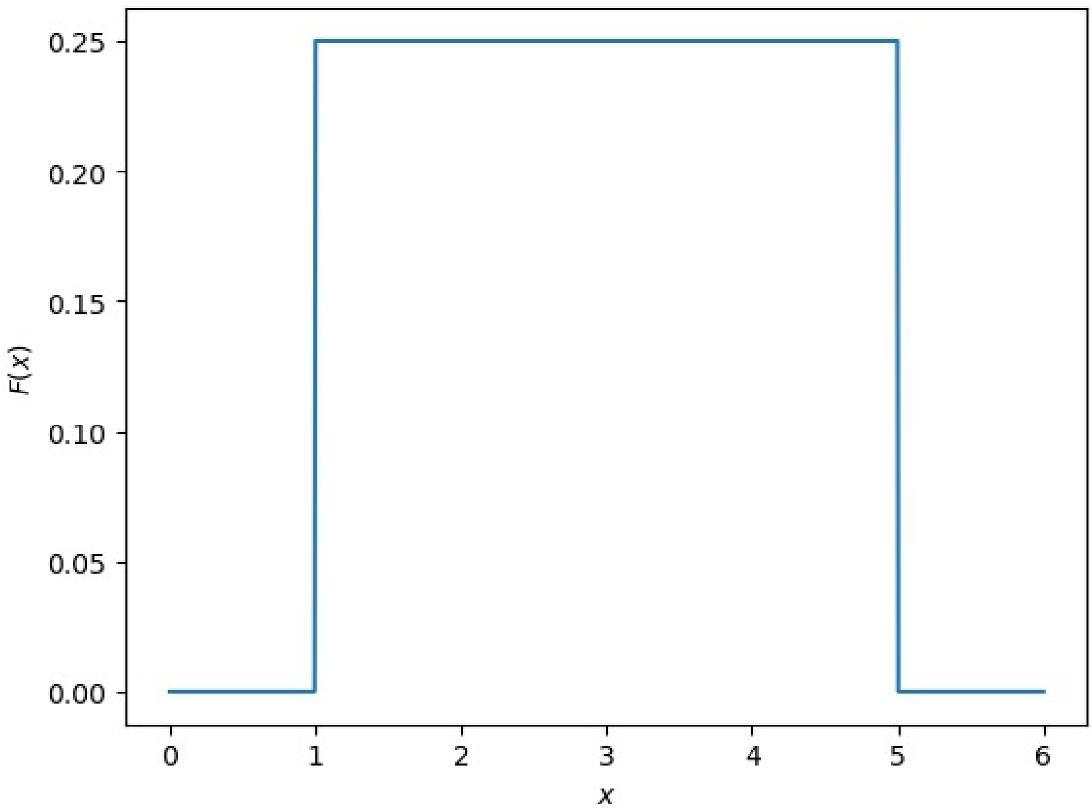
```
a = 1
b = 5

uniform_rv = sts.uniform(a, b-a)

uniform_rv.rvs(10)
```

```
array([2.58476614, 4.49957068, 4.20786148, 1.30622781, 4.03634175,
       4.19361475, 4.16440665, 4.58262524, 2.64970374, 2.09967057])
```

Task 1. Plot the probability density function of the uniform distribution:

```
#your code here

x = np.linspace(0,6,1000)
pdf = uniform_rv.pdf(x)
plt.plot(x, pdf)

plt.ylabel('$F(x)$')
plt.xlabel('$x$')

#you should obtain something like this:
```

```
Text(0.5, 0, '$x$')
```



We are going to create *samples_no = 1000* sets of the size *n = 50* and calculate mean $\acute{X}$ of each sample and store the results in the vector *sample_means*:

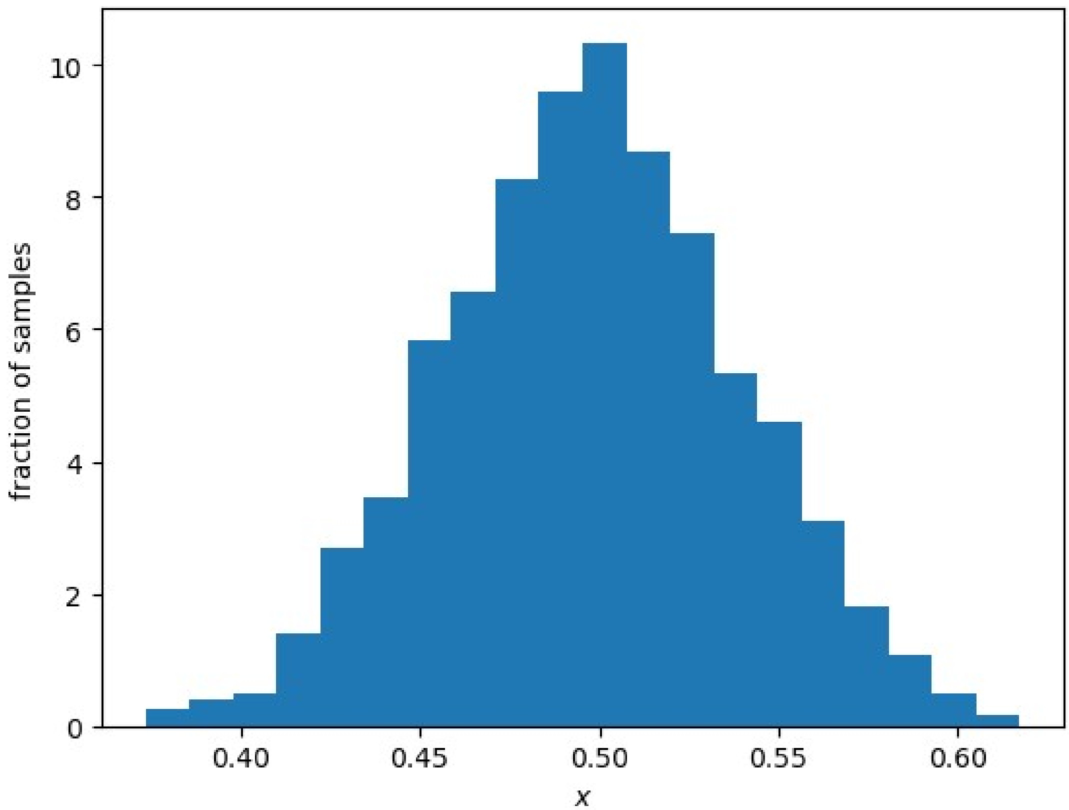
```
samples_no = 1000

n = 50
sample_means = np.empty(samples_no)
```

Task 2: Create a cycle for *i* from 0 to *samples_no*, in which you create a sample of the size *n*, calculate the mean of the sample and store its value in the vector *sample_means*

```
#your code here
for i in range(samples_no):
    sample = np.random.sample(n)
    sample_means[i] = np.mean(sample)
```

Task 3: plot histogram of the vector sample_means. It should have a bell-like shape:

```
#your code here:

plt.hist(sample_means, bins = 20, density=True)
plt.ylabel('fraction of samples')
plt.xlabel('$x$')
#You should get something like this:

Text(0.5, 0, '$x$')
```

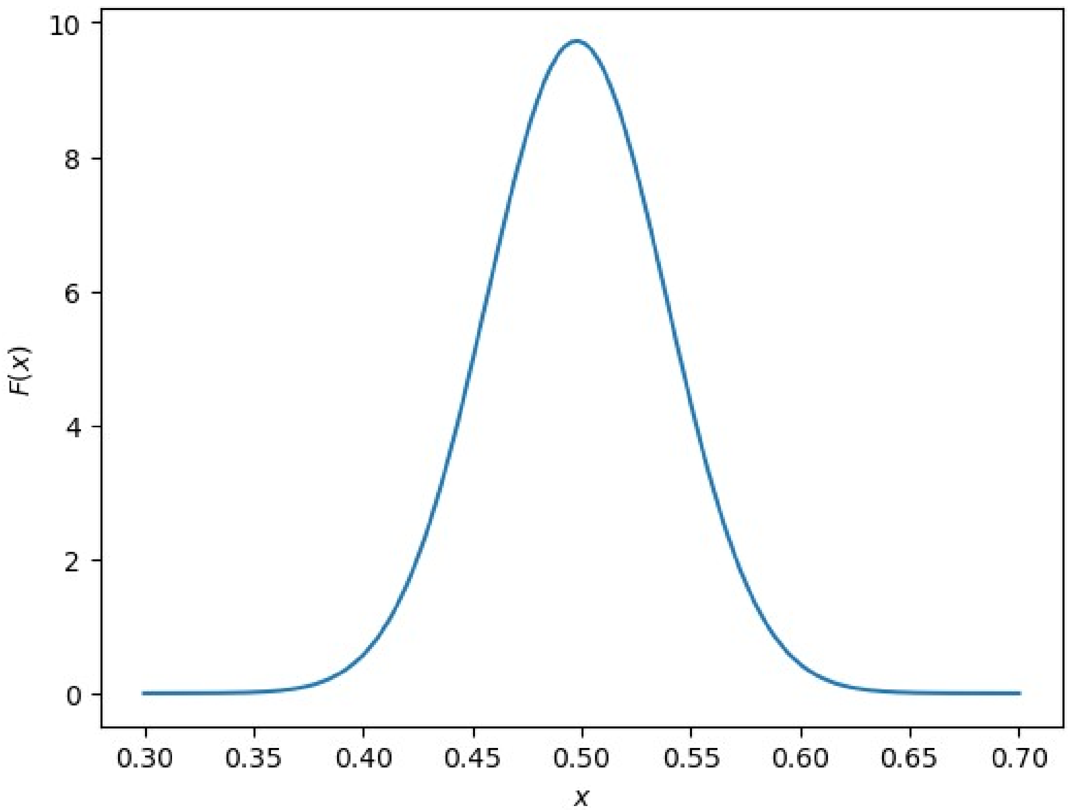


You are going to compare the obtained histogram from Task 3 with the normal distribution

Task 4: Create a normal distribution such, that it will be a distribution of the data you store in the vector *sample_means* and plot its density distribution function. Hint: to do that, you should know the **theoretical** expected value (i.e. mean) and dispersion (i.e. variance) of your initial uniform distribution. Look it up in Wikipedia.

```
# your code here:

mu = sample_means.mean()
sigma = sample_means.std() #np.var(sample_means)

norm_rv = sts.norm(loc=mu, scale=sigma)

x = np.linspace(0.3, 0.7, 130)
pdf = norm_rv.pdf(x)
plt.plot(x, pdf)
plt.ylabel('$F(x)$')
plt.xlabel('$x$')
```

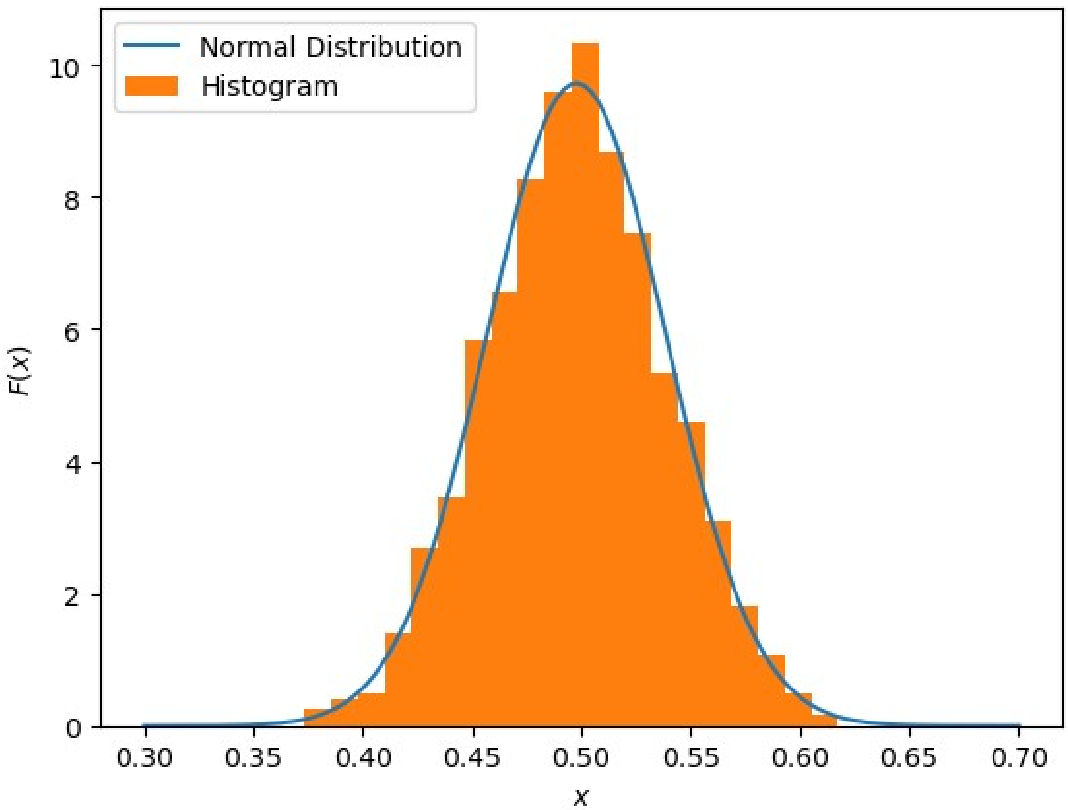
```
Text(0.5, 0, '$x$')
```



Task 5 Plot the histogram from Task 3 and the distribution density function from Task 4 *in the same Graph*. Do they look similar?

```
#your code here

x = np.linspace(0.3, 0.7, 130)
pdf = norm_rv.pdf(x)
plt.plot(x, pdf, label='Normal Distribution')

plt.hist(sample_means, bins = 20, density=True, label='Histogram')

plt.ylabel('$F(x)$')
plt.xlabel('$x$')
plt.legend(loc='upper left')

#you should obtain something like this:
```

```
<matplotlib.legend.Legend at 0x2906448f370>
```


Normal Distribution
Histogram
F(x)
10
8
6
4
2
0
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
x