IE 306 / HOMEWORK 1 / FATMANUR YAMAN / 2019402204 / 24.10.2022

1. Q1

$x(t) \;=\; The\ amount\ of\ the\ money\ in\ time\ t\ (Balance)$

The balance is the stock variable whereas the money gained from interest and transferred per day into another account are flow variables.

The Model Equation

$$\frac{dx}{dt} = i \times x(t) - a = 0.0036 \times x(t) - 360$$

The Simulation Equation

$$\frac{dx}{dt} \approx \frac{x(t+dt)-x(t)}{dt} \approx 0.0036 \times x(t) - 360$$

In [1]:

```python
import pandas as pd
def hw1_simulation(initial, i, deducted, time, dt):
    balance_list = []
    flow_list = []
    flow_time_list = []
    time_list = []
    t_initial = 0
    net_flow = 0
    balance = initial
    while t_initial <= time:
        balance += net_flow
        flow = (balance*i-deducted)
        net_flow = flow*dt
        time_list.append(t_initial)
        flow_list.append(flow)
        flow_time_list.append(net_flow)
        t_initial = dt + t_initial
        balance_list.append(balance)

    dict_list = {'Time' : time_list, 'Balance' : balance_list, 'Total Flow': flow_list , 'Net Flow': flow_time_list}
    df = pd.DataFrame(dict_list)
    return df
hw1_simulation(306306, 0.036, 306, 3, 0.5)
```

Out[1]:

	Time	Balance	Total Flow	Net Flow
0	0.0	306306.000000	10721.016000	5360.508000
1	0.5	311666.508000	10913.994288	5456.997144
2	1.0	317123.505144	11110.446185	5555.223093
3	1.5	322678.728237	11310.434217	5655.217108
4	2.0	328333.945345	11514.022032	5757.011016
5	2.5	334090.956361	11721.274429	5860.637214
6	3.0	339951.593576	11932.257369	5966.128684

2. Q2: A Single Waiting Line, Two Servers Queueing System

Interarrival Time U(6,16)
Service Time for Server1 U(14,20)
Service Time for Server2

Service Time (min)	Pdf	Cdf
4	0.18	0.18
15	0.38	0.56
23	0.34	0.90
32	0.10	1

Every customer has an arrival time, service time spent with the server, and departure time and these numbers are determined by the random numbers above and probabilistic distributions that are given in the question.

Interarrival Time = (16-6)*r+6
Server1 = (20-14)*r+14
Server2 = P(r)

A = Arrival Time, D = Departure Time, Tia = Interarrival Time, Ta = Arrival Time, Tsb = Service Beginning Time, Tser = Service Time, Td = Departure Time, Ts = Time Spent in the System, Tq = Time Spent in the Queue.

Customer	A	D	Tia	Ta	Tsb	Server	Tser	Td	Ts	Tq
1		0,497	0	0	0	1	16,982	16,982	16,982	0
2	0,380	0,862	9,8	9,8	9,8	2	23	32,8	23	0
3	0,020	0,975	6,2	16	16,982	1	19,85	36,832	20,832	0,982
4	0,391	0,480	9,91	25,91	32,8	2	15	47,8	21,89	6,89
5	0,005	0,959	6,05	31,96	36,832	1	19,754	56,586	24,626	4,872
6	0,360	0,593	9,6	41,56	47,8	2	23	(70,8)	(29,24)	6,24
7	0,744	0,069	1,.44	55	56,586	1	14,414	(71)	(16)	1,586
8	0,370	0,708	9,7	64,7	(70,8)	2	(23)	(93,8)	(29,1)	(6,1)

i. Average Number of Customers in The System :

[1*(9,8-0)+2*(16-9,8)+3*(16,982-16)+2*(19,85-16,982)+1*(23-16,982)+0*(25,91-23)+1*(31,96-25,91)+2*(41,56-31,96)+3*(47,8-41,56)+2*(55–47,8)+3*(56,586-55)+2*(64,7-56,586)+3*(70-64,7)]/70 = **1,8879**

ii. Average Time Spent in The Queue : $\frac{(0+0+0,982+6,89+4,872+6,24+1,586+5,3)}{8} = 2,8655$

iii. The Average Utilization of Each Server:

$Server\ 1: \frac{70}{70} = 1$

$Server\ 2: \frac{70-9,8}{70} = 0,86$

iv. Probability of Having One Customer In The Queue:

[1*(9,8-0) + 1*(23-16,982) + 1*(31,96-25,91)] / 70 = **0.3124**

3. Q3

a.

In [9]:
```
from scipy.stats import norm
import numpy as np
begin = 0.56
finish = 2.4
n = 1000
randx = np.random.uniform(begin,finish,n)
y = norm.pdf(randx,0,1)
montecarlo_integral = (finish-begin)*y.sum()/n
actual_integral = norm(0,1).cdf(2.4) - norm(0,1).cdf(0.56)
print('Monte Carlo Integral is: ' , montecarlo_integral)
print('Actual Integral is: ' , actual_integral)
```

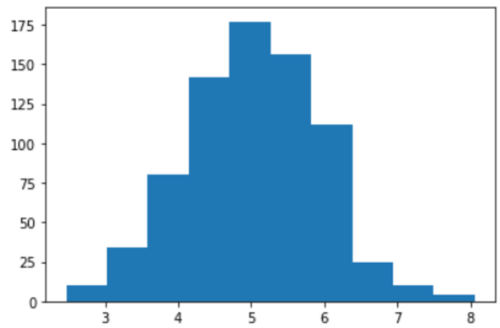
```
Monte Carlo Integral is:  0.2822211531449107
Actual Integral is:  0.2795421829244309
```

As can be seen in the above, although the approximation that is observed from Monte Carlo Simulation is not exactly equal to the true value of the integration, the results are very close to each other. ***Thus, we can say that the approximation is good.*** I generated 1000 random variates and if I generate more, the approximation will be closer to the actual value of the integral.

b.

In [14]:
```
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt
a = 0
b = 1
n = 10
Y_list = []
for i in range(750):
  randx = np.random.uniform(a,b,n)
  y = randx.sum()
  Y_list.append(y)
print('The mean of Y is :', np.mean(Y_list))
print('The variance of Y is:', np.var(Y_list))
plt.hist(Y_list, bins = 10)
plt.show()
```

```
The mean of Y is : 5.0399114482911225
The variance of Y is: 0.8274724746354722
```



The mean of Y should be 5 and I found 5.04. Also, the variance of Y should be 0.83 and I found 0.827. The values are very close to each other, ***thus I can say that the approximation is good.*** Like in the question 3a, if I generate more random variates, the results will be closer to the expected values of mean and variance. In addition to this, the histogram of Y looks like a normal distribution.