Session 12: Simulation Modeling I (with Solutions)

1. Generating Samples

```
[1]: from scipy.stats import norm
     import numpy as np
     np.random.seed(0)
     dist=norm(100,30)
     dist.rvs()
```

```
152.92157037902993
```

```
[2]: dist.rvs(size=3)
```

```
array([112.00471625, 129.36213952, 167.22679598])
```

```
[3]: import pandas as pd
     data=dist.rvs(size=10000)
     demand=pd.Series(data)
     demand.head()
```

```
0    156.026740
1     70.681664
2    128.502653
3     95.459284
4     96.903434
dtype: float64
```

```
[4]: demand.mean()
```

```
99.43350357671024
```

```
[5]: np.mean(data)
```

```
99.43350357671024
```

```
[6]: demand.std()
```

```
29.619606530616284
```

```
[7]: np.std(data)
```

```
29.618125513263394
```

```
[8]: (demand<100).mean()
```

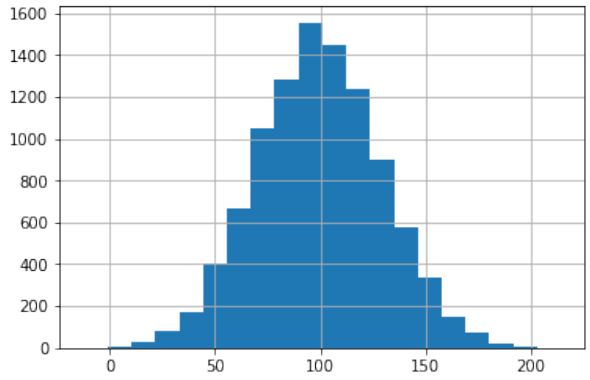
```
0.5109
```

```
[9]: np.mean(data<100)
```

```
0.5109
```

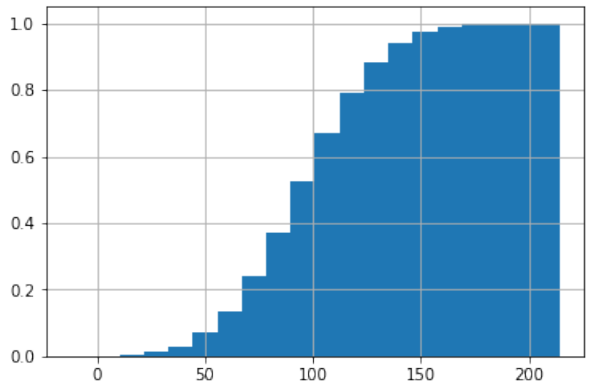
```
[33]: demand.hist(bins=20)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af82050f0>
```



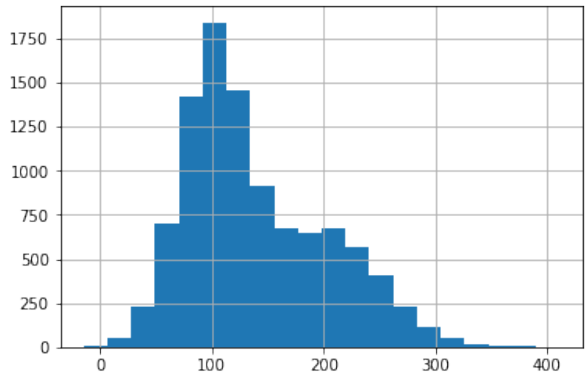
[11]:
```
demand.hist(bins=20,density=True,cumulative=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af88ecb38>
```



[12]:
```
distA=norm(200,50)
distB=norm(100,30)
from scipy.stats import bernoulli
distSegment=bernoulli(0.4)
data=[]
for i in range(10000):
    segment=distSegment.rvs()
    if segment==1:
        value=distA.rvs()
    else:
        value=distB.rvs()
    data.append(value)
demand2=pd.Series(data)
demand2.hist(bins=20)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af886ba20>
```



[13]:
```
values=[1,2.5,3.5]
probs=[0.3,0.5,0.2]
np.random.choice(values,p=probs)
```

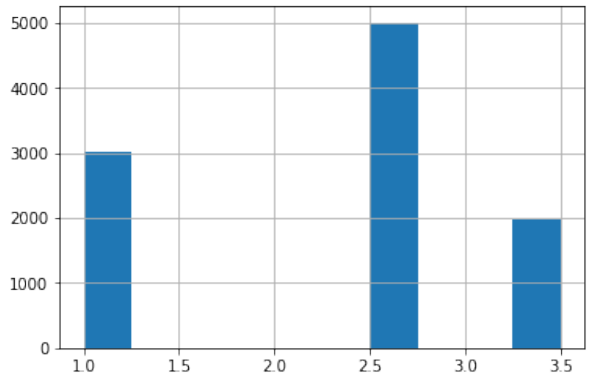
```
1.0
```

[14]:
```
np.random.choice(values,p=probs,size=10)
```

```
array([2.5, 3.5, 1. , 2.5, 1. , 2.5, 1. , 2.5, 1. , 1. ])
```

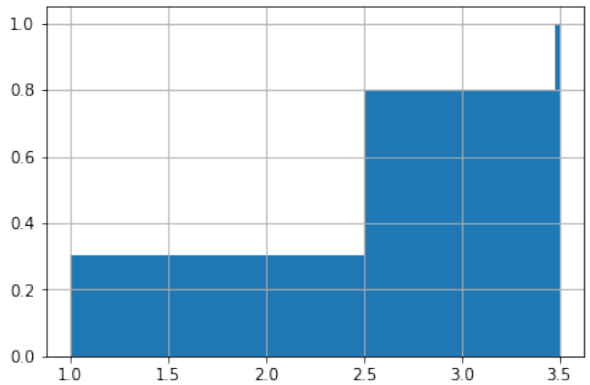
[15]:
```
samples=pd.Series(np.random.choice(values,p=probs,size=10000))
samples.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af892d710>
```



[16]:
```
samples.hist(bins=100,density=True,cumulative=True)
```

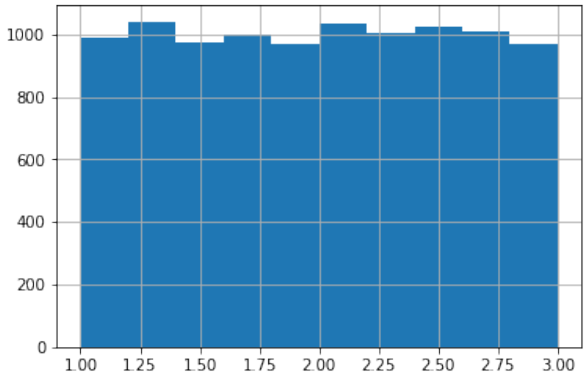
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af8715ac8>
```



Q1-a: Generate 10000 samples of a uniform distribution between 1 and 3 and plot the histogram, as well as the empirical CDF. Calculate the mean and standard deviation of the samples, as well as the proportion between 2 and 2.5 (inclusive).

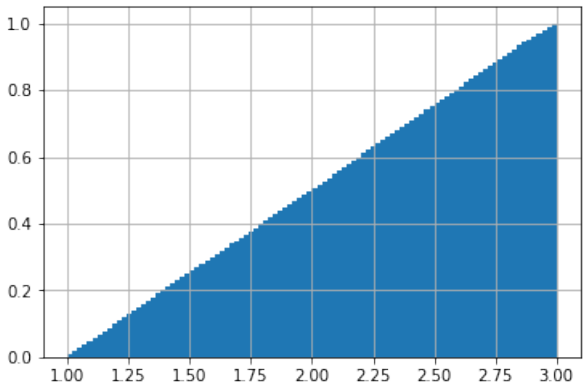
```
[17]: from scipy.stats import uniform
      s=pd.Series(uniform(1,2).rvs(10000))
      s.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af86bbac8>
```




```
[18]: s.hist(bins=100,density=True,cumulative=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af8857e48>
```




```
[19]: s.mean()
```

```
2.0000049590260542
```

```
[20]: s.std()
```

```
0.5748514236036787
```

```
[21]: ((s>=2) & (s<=2.5)).mean()
```

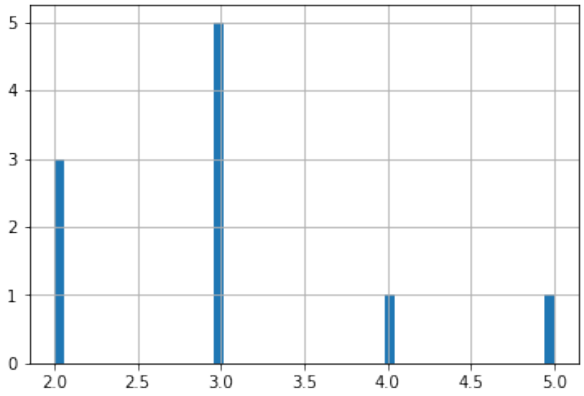
```
0.2561
```

Q1-b: Generate 100 samples of a binomial distrubution with $n = 10$ and $p = 0.3$. Calculate the mean and standard deviation of the sample and compare with what it should be from the formula. Plot a histogram with 50 bins. Repeat with 10,000 samples.

[22]:
```
from scipy.stats import binom
import numpy as np
n,p=10,0.3
dist=binom(n,p)
s=pd.Series(dist.rvs(10))
print('Sample mean:',s.mean())
print('Sample std:',s.std())
print('Theoretical mean:',n*p)
print('Theoretical std:',np.sqrt(n*p*(1-p)))
s.hist(bins=50)
```

```
Sample mean: 3.0
Sample std: 0.9428090415820634
Theoretical mean: 3.0
Theoretical std: 1.4491376746189437
```

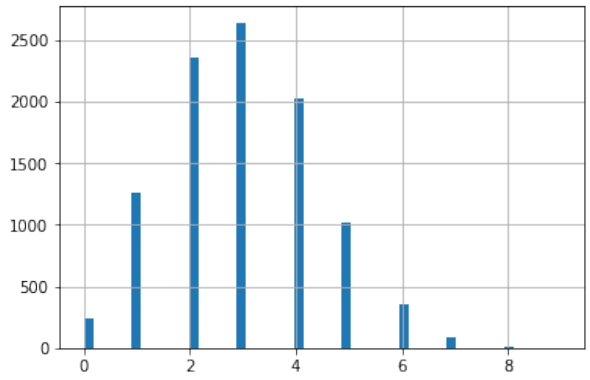
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af841b588>
```



[23]:
```
s=pd.Series(dist.rvs(10000))
print('Sample mean:',s.mean())
print('Sample std:',s.std())
print('Theoretical mean:',n*p)
print('Theoretical std:',np.sqrt(n*p*(1-p)))
s.hist(bins=50)
```

```
Sample mean: 2.9942
Sample std: 1.4376971368496672
Theoretical mean: 3.0
Theoretical std: 1.4491376746189437
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af8394ef0>
```



Q2: You are tasked with forecasting demand for a new product. Based on past data and your knowledge of the product, you estimate that the product quality will be Amazing with probability 0.1, Mediocre with probability 0.5, and Terrible with probability 0.4. You model the demand as normally distributed, with mean and standard deviation depending on the product quality as follows.

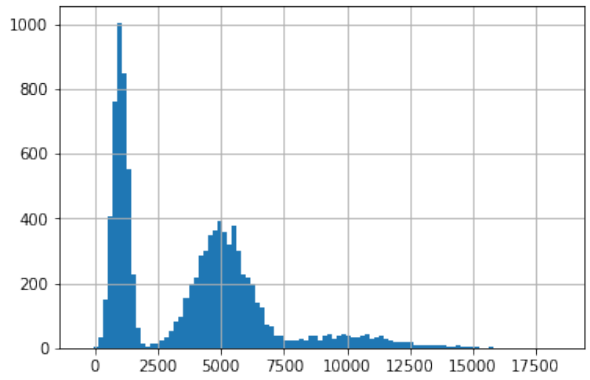
Prod. Quality:	Amazing	Mediocre	Terrible
μ	10000	5000	1000
σ	2000	1000	300

Create a `Series` called "forecast" with 10,000 samples of the demand forecast, and compute the mean and standard deviation of the samples, as well as the probability that demand is more than 6000. Finally, plot a histogram of the samples with 100 bins, as well as the empirical CDF.

```
[24]: from scipy.stats import binom
      import numpy as np
      data=[]
      for i in range(10000):
          quality=np.random.choice(['A','M','T'],p=[0.1,0.5,0.4])
          if quality=='A':
              sample=norm(10000,2000).rvs()
          elif quality=='M':
              sample=norm(5000,1000).rvs()
          else:
              sample=norm(1000,300).rvs()
          data.append(sample)
      forecast=pd.Series(data)
      print('Sample mean:',forecast.mean())
      print('Sample standard deviation:',forecast.std())
      print('Probability demand more than 6000:',(forecast>6000).mean())
      forecast.hist(bins=100)
```

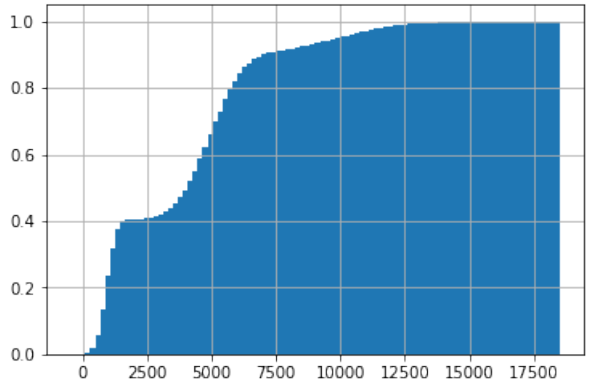
```
Sample mean: 3881.302260377957
Sample standard deviation: 2939.8432697511016
Probability demand more than 6000: 0.1781
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af882eb70>
```



[25]:
```
forecast.hist(bins=100,density=True,cumulative=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af80b4438>
```



Q3: Nadeem is a car salesperson who faces the following incentive scheme at the dealership where he works. For each month, there is a "target profit" that the dealership sets for the month. If he makes more profit for the dealership that month than the target, then he receives a 20% bonus on the amount of profit over the target. However, if he does not meet the target, he receives no bonus. For example, if the target is 80,000 and he makes 100,000 of profit, then he receives a 4,000 bonus that month. However, if he makes 70,000, then he receives zero bonus that month. Nadeem would like to understand the distribution of his monthly bonus.

Nadeem estimates that the number of cars he sells is binomial distributed with $n = 200$ and $p = 0.2$. On every car he sells, the amount of profit he makes for the dealership is normally distributed with $\mu = 3000$ and $\sigma = 1000$, and the profit from each car is independent of another.

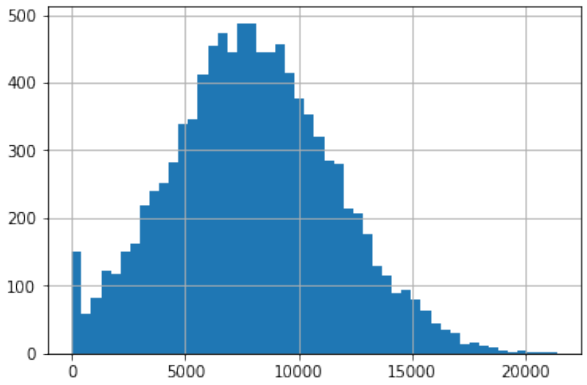
Create a `Series` called "monthlyBonus" with 10,000 samples of his monthly bonus. Compute the mean, the standard deviation, the probability the bonus is less than 5000, and plot a histogram with 50 bins as well as the empirical CDF.

[26]:
```
from scipy.stats import binom, norm
data=[]
target=80000
for i in range(10000):
    cars=binom(200,0.2).rvs()
    sales=np.sum(norm(3000,1000).rvs(size=cars))
    if sales>target:
        bonus=(sales-target)*0.2
    else:
```

```
            bonus=0
        data.append(bonus)
    monthlyBonus=pd.Series(data)
    print('Mean is',monthlyBonus.mean())
    print('Standard deviation is',monthlyBonus.std())
    print('Probability less than 5000 is',(monthlyBonus<5000).mean())
    monthlyBonus.hist(bins=50)
```

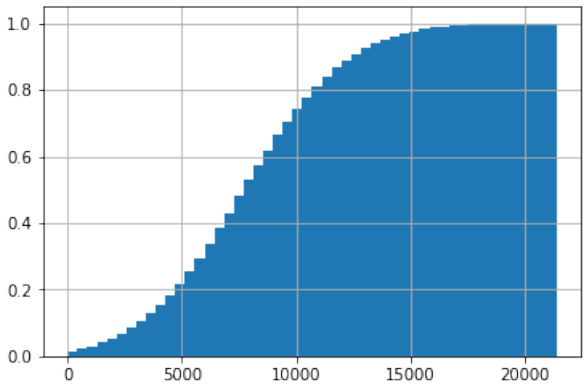
```
Mean is 7966.691715092367
Standard deviation is 3595.714686711149
Probability less than 5000 is 0.2058
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af7f98e80>
```



[27]: `monthlyBonus.hist(bins=50,density=True,cumulative=True)`

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af7df9a20>
```



(Bonus) This question asks you to illustrate the Central Limit Theorem (CLT) by example. Consider the following distribution,

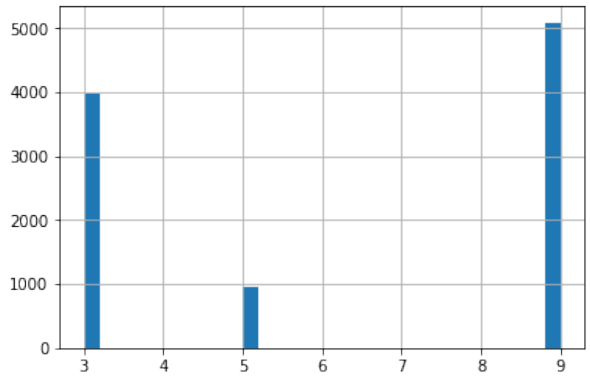
$$X = \begin{cases} 3 & \text{with probability } 0.4, \\ 5 & \text{with probability } 0.1, \\ 9 & \text{with probability } 0.5. \end{cases}$$

Define Y_n to be the sum of n independent random variables with the above distribution. Create 10000 samples of Y_1, Y_5, Y_{30}, Y_{100}, and Y_{1000} and plot their histograms (with 30 bins). You should be able to see the histograms converging to a Bell curve as n increases.

The Central Limit Theorem (CLT) says that this phenomenon always happens, regardless of the distribution of X. Moreover, it still takes place even if the n independent random variables do not have the same distribution, as long as each term in the sum is "small" relative to the whole. (For a precise mathematical formulation of the CLT in the case with non-identical random variables, search for Lyapunov or Lindeberg CLT on Wikipedia.)

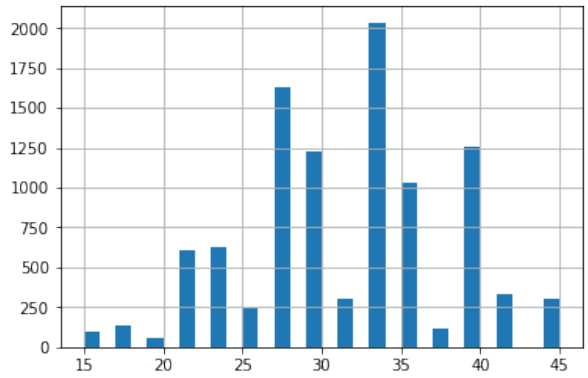
[28]:
```
import numpy as np
def simulateCLT(n,T=10000):
    x=[3,5,9]
    p=[0.4,0.1,0.5]
    data=[]
    for i in range(T):
        value=np.sum(np.random.choice(x,p=p,size=n))
        data.append(value)
    samples=pd.Series(data)
    return samples

simulateCLT(1).hist(bins=30)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af7d9cbe0>
```



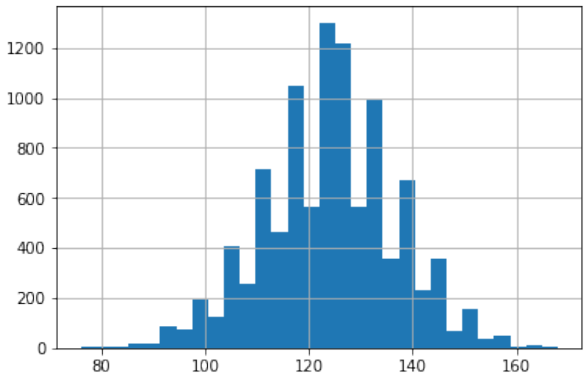
[29]:
```
simulateCLT(5).hist(bins=30)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af7ce4ef0>
```



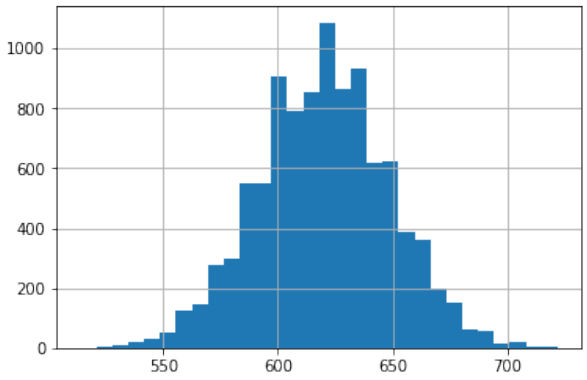
[30]:
```
simulateCLT(20).hist(bins=30)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af7cf7be0>
```



[31]: `simulateCLT(100).hist(bins=30)`

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af7cb5eb8>
```



[32]: `simulateCLT(1000).hist(bins=30)`

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af7b648d0>
```