1/1/2023

Random Variables

EMP305
Engineering Mathematics (5)

كلية الهندسة بشبرا

FACULTY OF ENGINEERING AT SHOUBRA

Random Variables & Python Project

Engineering Mathematics (5)

EMP305

Submitted by:

S#	Student Name	Edu Email	Student ID	Marks		
				Report & Slides (30)	Implementation (50)	Presentation (20)
1	Ahmed Rafat	ahmed402812@feng.bu.edu.eg	231903661			
2	Ahmed Eslam Foad	ahmed402455@feng.bu.edu.eg	231903658			
3	Raghad Amr	raghad435083@feng.bu.edu.eg	231903551			
4	Yasmine Yahia Elmetwally	yasmin402552@feng.bu.edu.eg	221903092			
5	Mariam Mohamed	Maryam20889@feng.bu.edu.eg	221903111			
6	Ahmed Sherif	ahmed442274@feng.bu.edu.eg	231903592			
7	Mahmoud Mansour	mahmoud403355@feng.bu.edu.eg	231903562			

Supervised by:

Dr. Lamia Alrefaai

1. Introduction:

- Distribution function, mathematical expression that describes the probability that a system will take on a specific value or set of values, the classic examples are associated with games of chance.

- The random variables are used in many applications in our practical life, and through them we can determine many things, including determining the time of disasters and determining the validity period of many devices such as electronic devices.

2. Types of Random Variable

As discussed in the introduction, there are two random variables, such as:

- Discrete Random Variable

- Continuous Random Variable

- In our project we discuss some types of random variable for desecrate random variables
- we discuss Bernoulli, Binomial, Uniform, Geometric and Passion distribution.
- for continuous distribution we discuss Uniform, Exponential, and Gaussian distribution.
- And discuss some application and the importance of random variables in our real life.

➢ Discrete Random Variable:

A discrete random variable can take only a finite number of distinct values such as 0, 1, 2, 3, 4, ... and so on. The probability distribution of a random variable has a list of probabilities compared with each of its possible values known as probability mass function.

1. Bernoulli Distribution:

I. Definition

A Bernoulli random variable is a discrete random variable that takes on only two possible values, typically represented as "success" or "failure," or 1 or 0. It is a simple type of random variable that is used to model situations where there are only two possible outcomes.

II. Equation:

PMF pX (0) = 1−p pX (1) =p

CDF F(x)= $p^x\ (1-p)^{1-x}$

III. Properties:

X~Bernoulli(p)

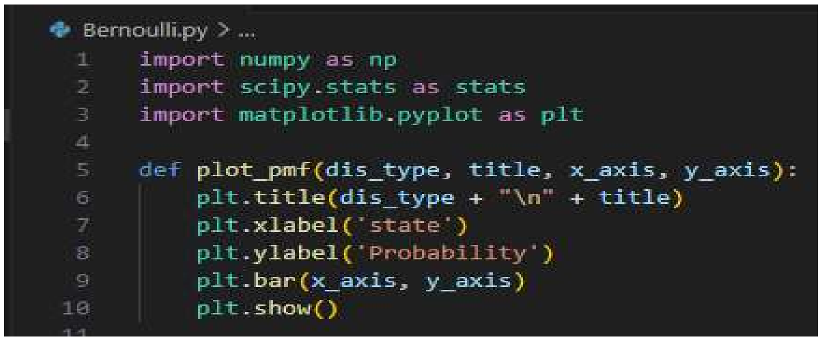
1- Mean: The expected value of a Bernoulli random variable X is **(**E [x] = P)

2- E [X^2] = P

3- Variance: Var [x] = P(1-P)

IV. Code

- **First:** we need to import needed libraries and define the plotting



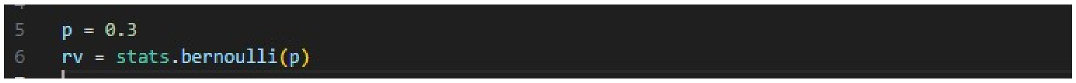
```python
Bernoulli.py > ...
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

def plot_pmf(dis_type, title, x_axis, y_axis):
    plt.title(dis_type + "\n" + title)
    plt.xlabel('state')
    plt.ylabel('Probability')
    plt.bar(x_axis, y_axis)
    plt.show()
```

➢ **"numpy"** as **np** for numerical operations, **"scipy,stats"** as **stats** for statistical functions, **"matplotlib.pyplot"** as **plt** for graphing the functions.

➢ For defining plotting function named **"plot_pmf"** and determining it's parameters from **"dis_type", "title", "x_axis", "y_axis"**

And creating bar plot using **"plt.bar"** and showing it using **"plt.show"**

- **Second**: setting up the Bernoulli distribution:

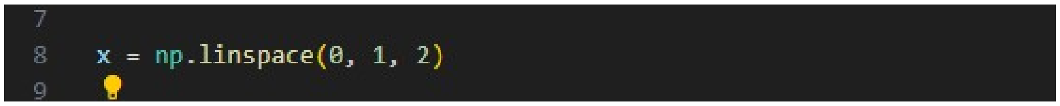


```python
p = 0.3
rv = stats.bernoulli(p)
```

Where **'P'** is the probability of success of the Bernoulli distrbution

In addition, **'stats.bernoulli (p)'** creates a Bernoulli random variable distribution with specified probability 'P'.

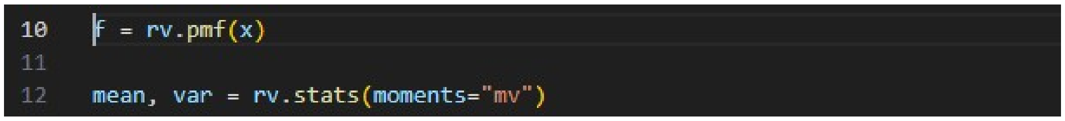
- **Third: Defining values for X (Possible outcomes)**:



```
x = np.linspace(0, 1, 2)
```

  - **'np.linespace (0, 1, 2)'** generates an array of two values between 0 and 1 In this case, **X** will be an array of [0 , 1].

- **Fourth:** calculating probability mass function (PFM), Mean and

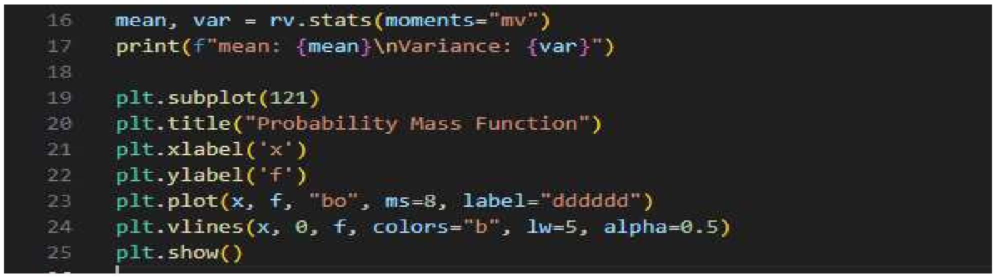


```
f = rv.pmf(x)

mean, var = rv.stats(moments="mv")
```

Variance:

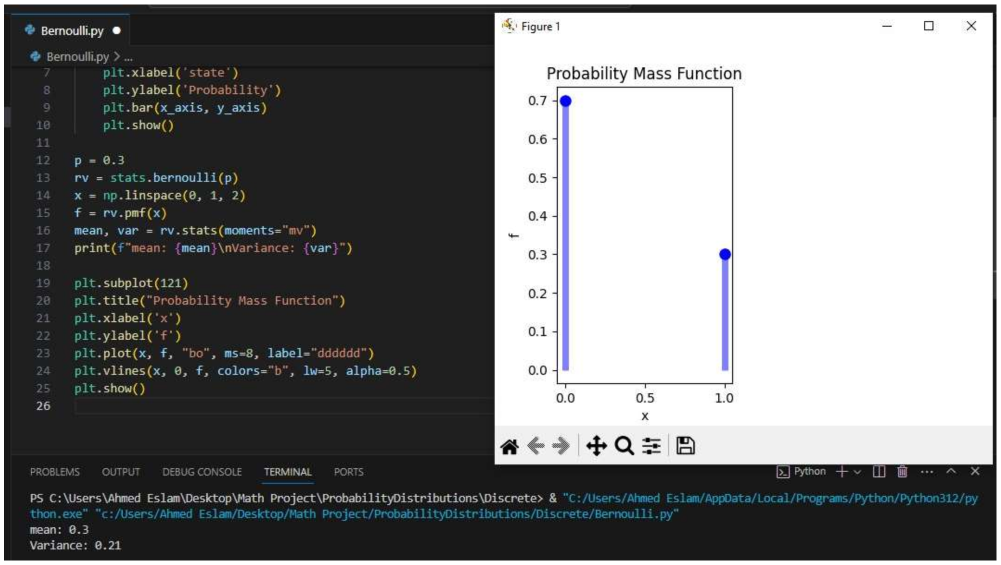
  - Here **'rv.pmf(x)'** calculates the probability mass function for the specified outcomes in **'X'** Function **'rv.stats(moments="mv")'** calculates mean and var of Bernoulli

- **Finally,** printing mean and variance also, plotting the PMF:



```
mean, var = rv.stats(moments="mv")
print(f"mean: {mean}\nVariance: {var}")

plt.subplot(121)
plt.title("Probability Mass Function")
plt.xlabel('x')
plt.ylabel('f')
plt.plot(x, f, "bo", ms=8, label="dddddd")
plt.vlines(x, 0, f, colors="b", lw=5, alpha=0.5)
plt.show()
```

V. Output



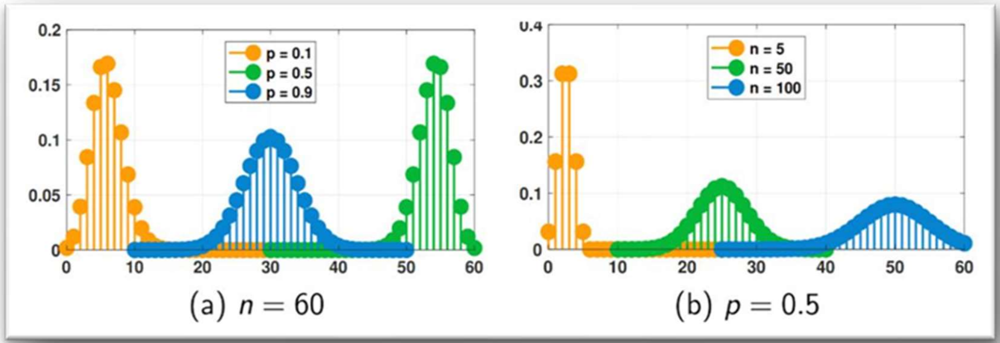
```python
        plt.xlabel('state')
        plt.ylabel('Probability')
        plt.bar(x_axis, y_axis)
        plt.show()

p = 0.3
rv = stats.bernoulli(p)
x = np.linspace(0, 1, 2)
f = rv.pmf(x)
mean, var = rv.stats(moments="mv")
print(f"mean: {mean}\nVariance: {var}")

plt.subplot(121)
plt.title("Probability Mass Function")
plt.xlabel('x')
plt.ylabel('f')
plt.plot(x, f, "bo", ms=8, label="dddddd")
plt.vlines(x, 0, f, colors="b", lw=5, alpha=0.5)
plt.show()
```

```
PS C:\Users\Ahmed Eslam\Desktop\Math Project\ProbabilityDistributions\Discrete> & "C:/Users/Ahmed Eslam/AppData/Local/Programs/Python/Python312/py
thon.exe" "c:/Users/Ahmed Eslam/Desktop/Math Project/ProbabilityDistributions/Discrete/Bernoulli.py"
mean: 0.3
Variance: 0.21
```

2. Binomial Distribution:

I. Definition:

A binomial distribution is a discrete probability distribution that models the number of successes in a fixed number of independent Bernoulli trials, where each trial has the same probability of success, denoted by "p." The distribution is characterized by two parameters: the number of trials, denoted by "n," and the probability of success, denoted by "p."



II. Equation:

PMF

$$p_X(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Where:

n: is number of trials

p: probability of success in one trial k: number of successes

CDF $$F_X(k) = \sum_{\ell=0}^{k} \binom{k}{\ell} p^{\ell}(1-p)^{k-\ell}.$$

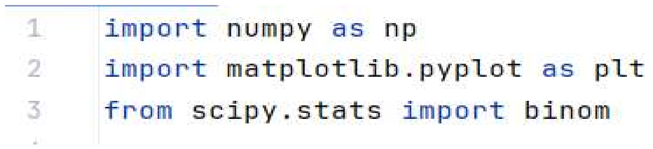
III. properties:

X~Binomial (n, p)

1-Mean: the expected value of X is(E[X] =np).
2-E $[X^2]$ = np (np+ (1−p)).
3-Variance: (Var[X] =np(1−p)).

IV. Code

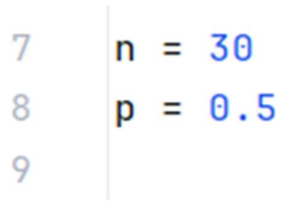
- **First:** We start by importing the necessary libraries:



```
1    import numpy as np
2    import matplotlib.pyplot as plt
3    from scipy.stats import binom
```

   - Line 1: **numpy** library helps in mathematical and logical operations on arrays
   - Line 2 : we use **matplotlib library** for used for plotting the PMF and CDF
   - Line 3: the **scipy.stats library** used for statistical and probabilistic operations

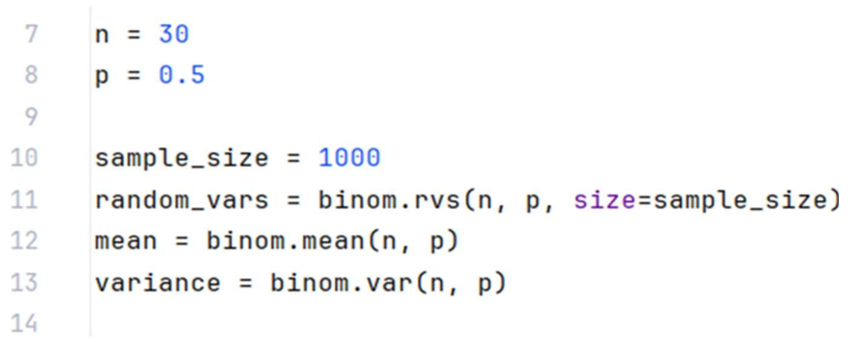
- After that we set the parameters of the Binomial Distribution



```
n = 30
p = 0.5
```

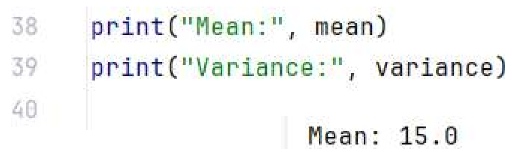
- N is number of trials while p is probability of success note that p should be a value between 0 and 1

- After that we generate the Mean, Variance and random variables
  The **binom.rvs()** , **binom.mean()** and **binom.var()** functions are associated with the **scipy.stats library** and used for calculating mean and variance and random variable using n and p 'as in the context of the code'



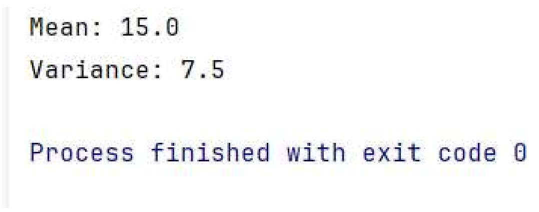
```python
n = 30
p = 0.5

sample_size = 1000
random_vars = binom.rvs(n, p, size=sample_size)
mean = binom.mean(n, p)
variance = binom.var(n, p)

```

- We use the following code to print mean and variance:



```python
print("Mean:", mean)
print("Variance:", variance)

```

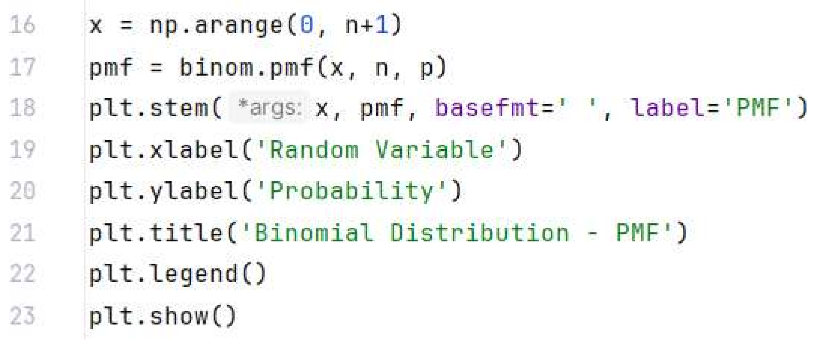
➢ **Output**:



```
Mean: 15.0
Variance: 7.5

Process finished with exit code 0
```

- After we will all that we will plot the PMF and CDF:

  PDF:



```python
x = np.arange(0, n+1)
pmf = binom.pmf(x, n, p)
plt.stem( *args: x, pmf, basefmt=' ', label='PMF')
plt.xlabel('Random Variable')
plt.ylabel('Probability')
plt.title('Binomial Distribution - PMF')
plt.legend()
plt.show()
```

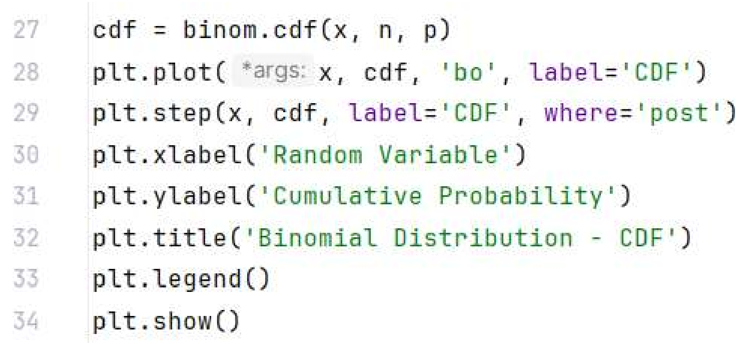
- For the previous code:

➢ **np.arange()** This function creates an array x using **numpy's** `arange` function. The array contains values from 0 to n

➢ **binom.pmf()** this function calculates the PMF values using X array and n and p which are previously initiated

- **plt.stem** function used to create the stem plot pf PMF X array is the X axis While PMF array is the Y axis , we used **basefmt** to clear the base line
- **plt.xlabel()** and **plt.ylabel()** sets the labels of x-axis and y-axis
- **plt.title()** displays the title of the plot
- **plt.legend()** displays the legend of the plot
- **plt.show()** displays the plot on the screen

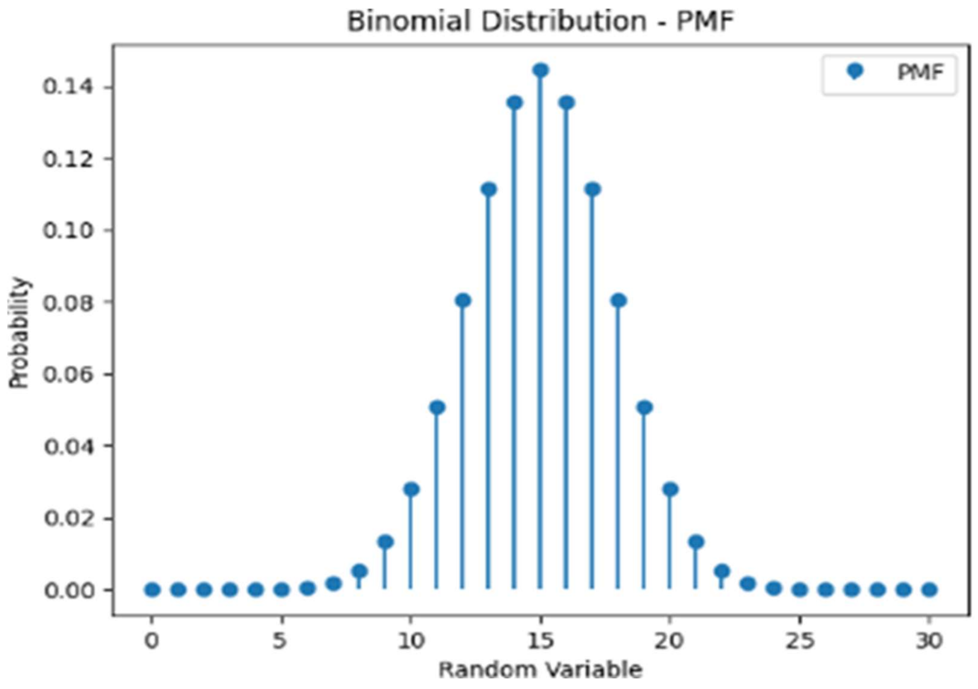
- **CDF:**

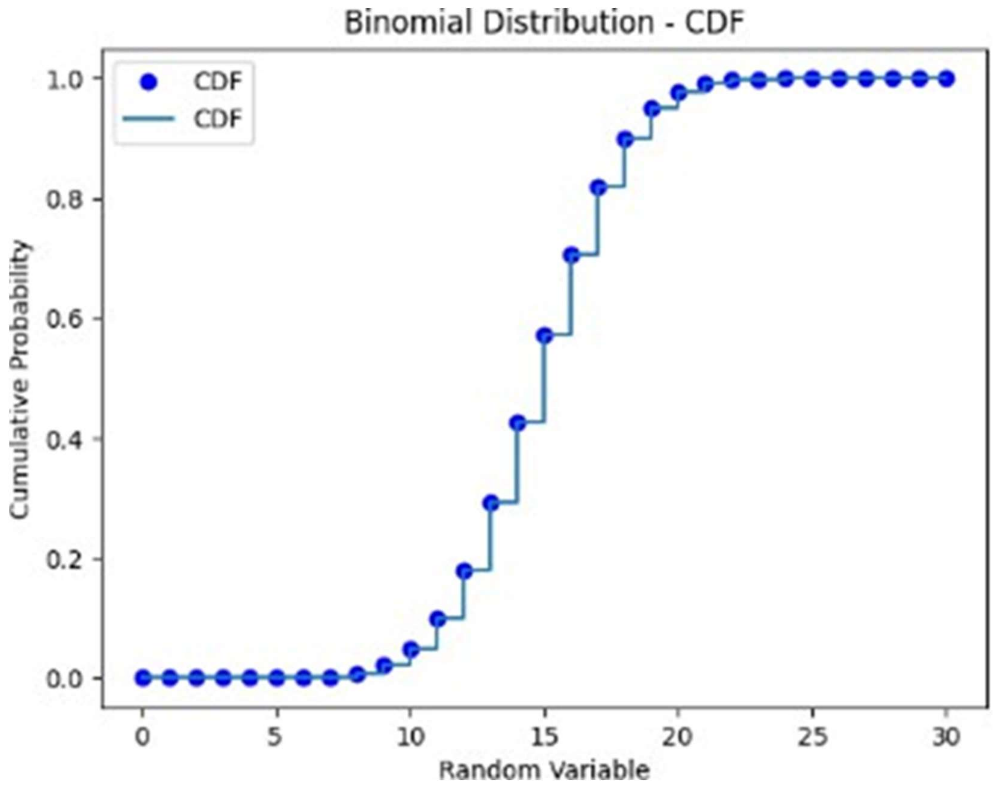


```
cdf = binom.cdf(x, n, p)
plt.plot( *args: x, cdf, 'bo', label='CDF')
plt.step(x, cdf, label='CDF', where='post')
plt.xlabel('Random Variable')
plt.ylabel('Cumulative Probability')
plt.title('Binomial Distribution - CDF')
plt.legend()
plt.show()
```

- for plotting CDF we will do the same as in PMF and just create a new array 'cdf'
- **binom.cdf()** to create array 'cdf' and calculating CDF values by using x and n and p
- **plt.plot()** to plot CDF as dot by using (**'bo'**) we may not use it and just use **plt.step()**
- **plt.step()** to plot CDF as a step.
- The last 5 lines are the same as the ones used in plotting PMF

V. Results


Binomial Distribution - PMF
PMF
Probability
Random Variable
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0
5
10
15
20
25
30


Binomial Distribution - CDF
CDF
CDF
Cumulative Probability
Random Variable
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30

3. Geometric Distribution:

I. Definition

The Geometric random variable is a discrete random variable function that is used when one is modelling a series of experiments that have one of two possible outcomes – success or failure – 1 or 0.

II. Equation

$$P_x\,(k) = P\,(1 - P\,)^{k-1}$$

III. properties:

$$X \sim \text{Geometric}\ (p)$$

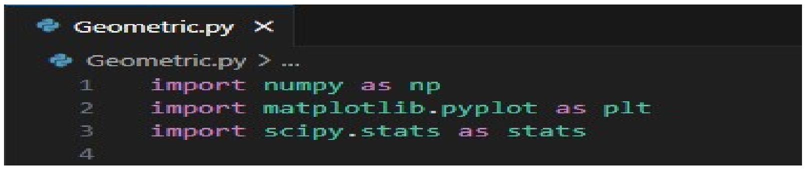
1- **Mean:** $E\,(\,x\,) = \frac{1}{P}$

2- $E\,(\;X^2\;)$: $\frac{2}{P^2} - \frac{1}{P}$

3- $Var\,(\,x\,)$: $\frac{1-P}{P^2}$

IV. Code

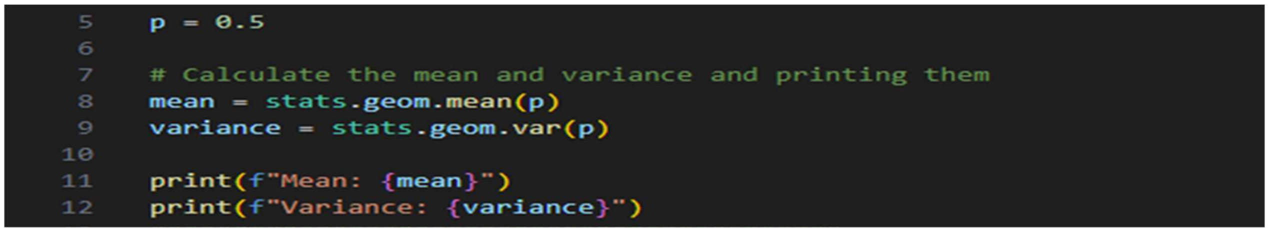
- **First:** Importing needed libiraries:



```
Geometric.py ×
Geometric.py > ...
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
```

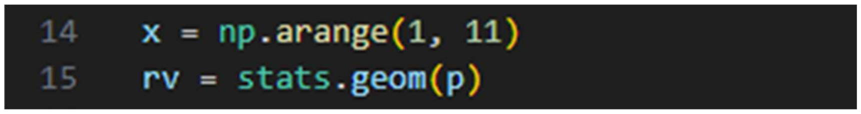
➢ **"numpy"** as **np** for numerical operations, **"scipy,stats"** as **stats** for statistical functions, **"matplotlib.pyplot"** as **plt** for graphing the functions.

- **Second:** Defining the probability and calculating Mean and Variance



```
p = 0.5

# Calculate the mean and variance and printing them
mean = stats.geom.mean(p)
variance = stats.geom.var(p)

print(f"Mean: {mean}")
print(f"Variance: {variance}")
```

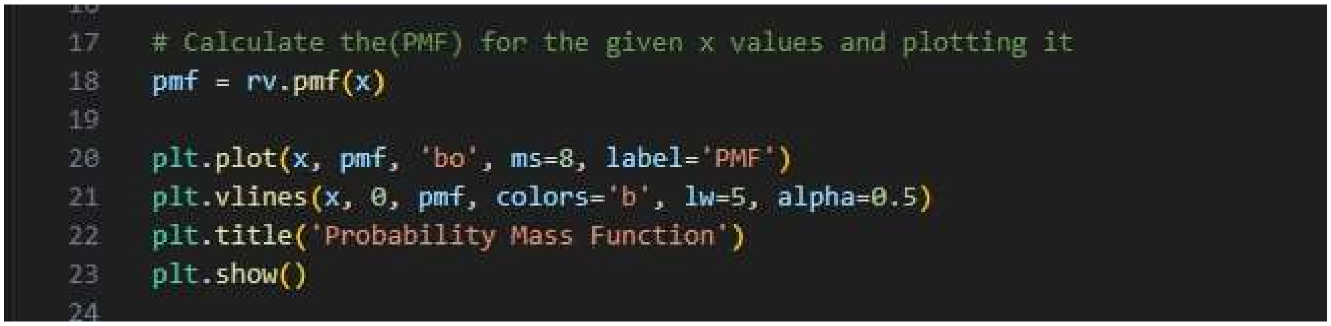
- Here "p" has 0.5 value that represent probability of success in geometric distribution Moreover, using "stats.geom.mean" and "stats.geom.var" from "scipy" libirary to calculate Mean and Variance for "p"
- **Third:** Creating values for x and Geometric Distribution



```python
x = np.arange(1, 11)
rv = stats.geom(p)
```

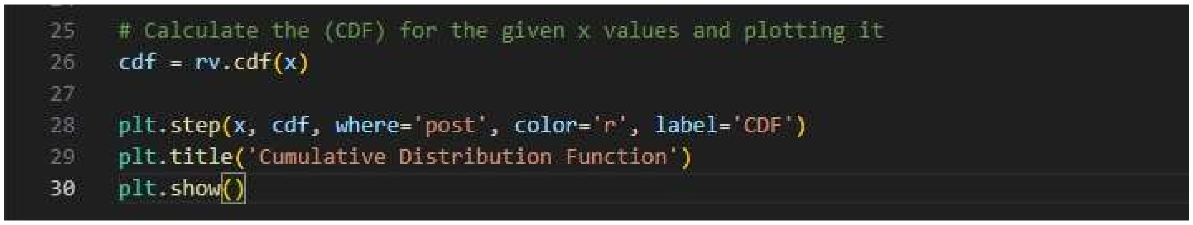
- ➢ Here "x" is an array from 1 to 10 and "rv" is a Geometric Distribution for probability 'p'.

- **Fourth:** Calculating Probability mass function for "x" and plotting it



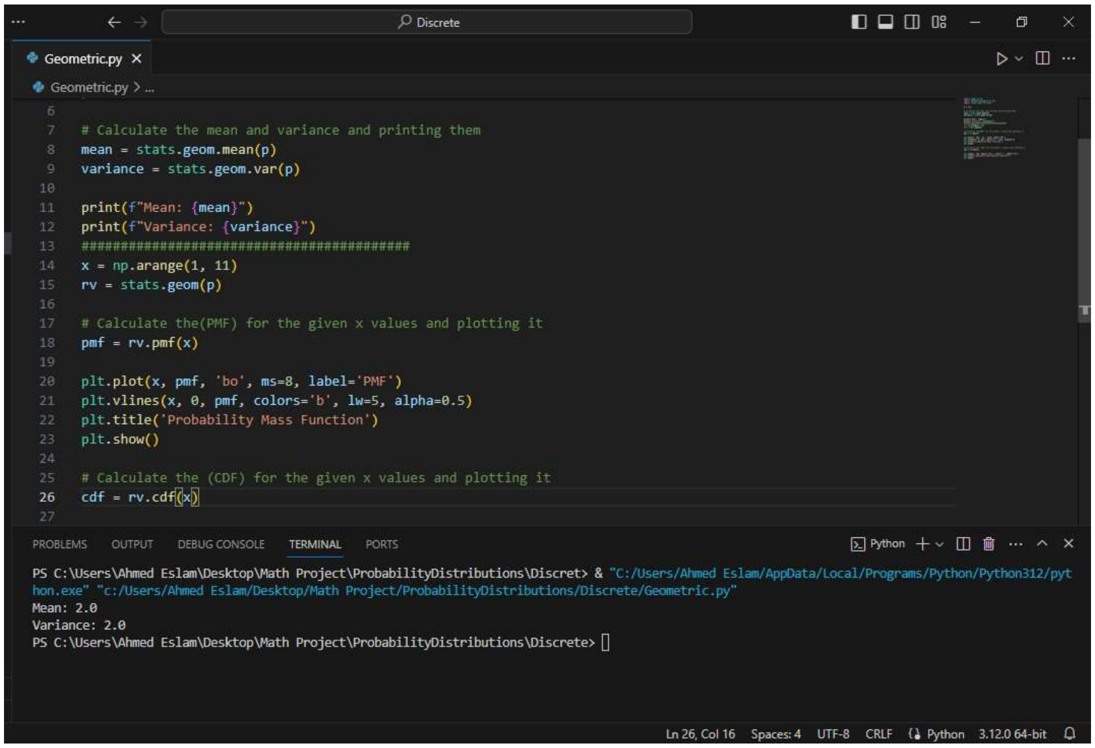
```python
# Calculate the(PMF) for the given x values and plotting it
pmf = rv.pmf(x)

plt.plot(x, pmf, 'bo', ms=8, label='PMF')
plt.vlines(x, 0, pmf, colors='b', lw=5, alpha=0.5)
plt.title('Probability Mass Function')
plt.show()
```

- **Fifth**: Calculating cumulative density function for "x" and plotting it



```python
# Calculate the (CDF) for the given x values and plotting it
cdf = rv.cdf(x)

plt.step(x, cdf, where='post', color='r', label='CDF')
plt.title('Cumulative Distribution Function')
plt.show()
```

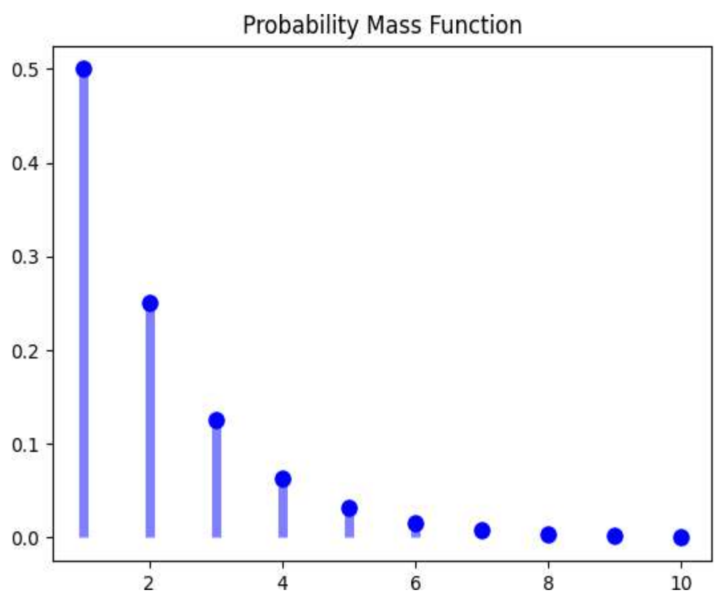
V. Results:

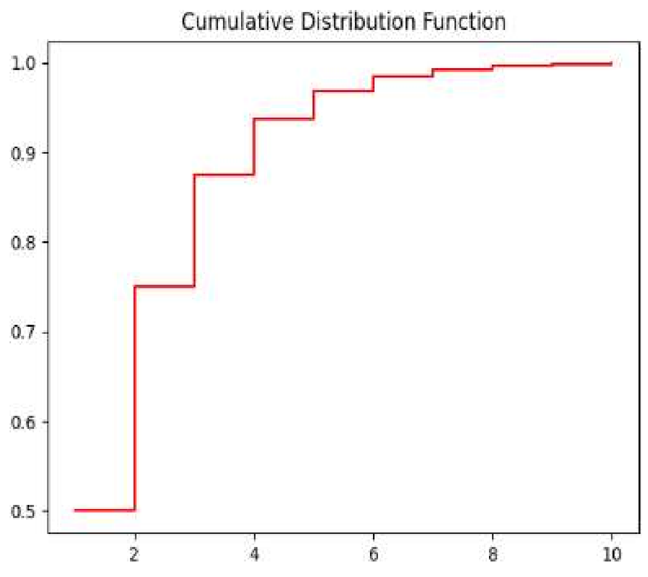


```python

# Calculate the mean and variance and printing them
mean = stats.geom.mean(p)
variance = stats.geom.var(p)

print(f"Mean: {mean}")
print(f"Variance: {variance}")
##########################################
x = np.arange(1, 11)
rv = stats.geom(p)

# Calculate the(PMF) for the given x values and plotting it
pmf = rv.pmf(x)

plt.plot(x, pmf, 'bo', ms=8, label='PMF')
plt.vlines(x, 0, pmf, colors='b', lw=5, alpha=0.5)
plt.title('Probability Mass Function')
plt.show()

# Calculate the (CDF) for the given x values and plotting it
cdf = rv.cdf(x)

```

```
PS C:\Users\Ahmed Eslam\Desktop\Math Project\ProbabilityDistributions\Discret> & "C:/Users/Ahmed Eslam/AppData/Local/Programs/Python/Python312/pyt
hon.exe" "c:/Users/Ahmed Eslam/Desktop/Math Project/ProbabilityDistributions/Discrete/Geometric.py"
Mean: 2.0
Variance: 2.0
PS C:\Users\Ahmed Eslam\Desktop\Math Project\ProbabilityDistributions\Discrete> 
```

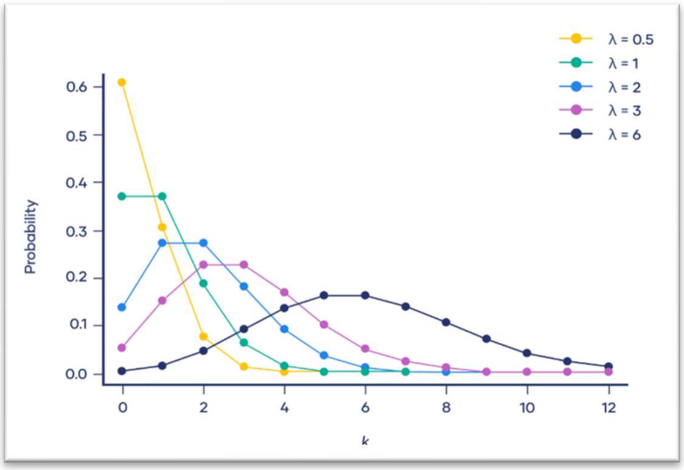
Mean and Variance





4- Poisson Distribution:

I. Definition: Is a discrete probability distribution used to model the number of occurrences of a random event in a fixed interval of time or space.



II. Formula

Poisson distribution formula is:

$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

- Where:

  - X is a random variable following a Poisson distribution
  - k is the number of times an event occurs
  - P (X = k) Is the probability that an event will occur k times
  - e is Euler's constant (approximately 2.718)
  - λ is the average number of times an event occurs
  - ! is the factorial function

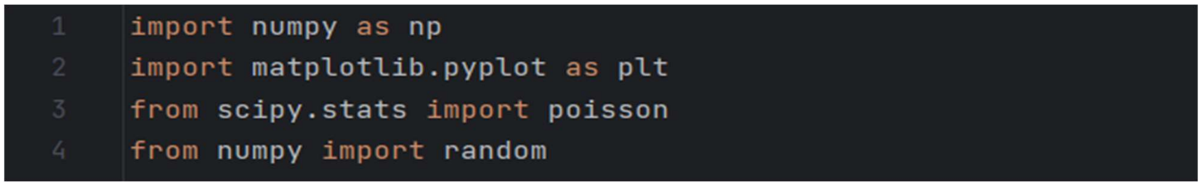
III. Properties of Poisson Distribution

X ~ Poisson (λ)

1. **Mean:** $E(x) = \lambda$
2. $E\,(x^2) = \lambda + \lambda^2$
3. **variance**: $Var\,(x) = \lambda$

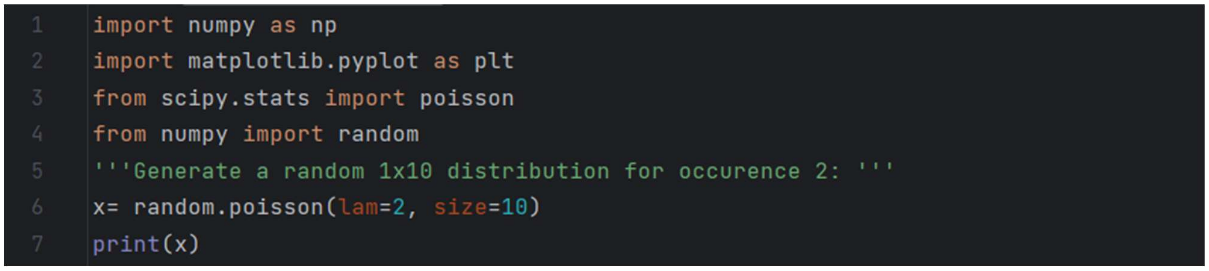
IV. Code

- First, we start by importing the required libraries.



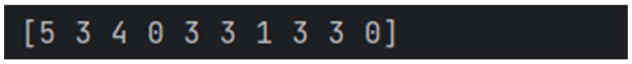
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import poisson
from numpy import random
```

- Now, to calculate Poisson distribution we need 2 parameters which are λ & k. In our code, λ is μ or our expectation. We need to generate random variable for occurrence 2, we use Poisson's function which is **"random.Poisson( )"** takes $\lambda = 2$ & $k = 10$.
- Then, the code produces a different set of random numbers each time it is executed.



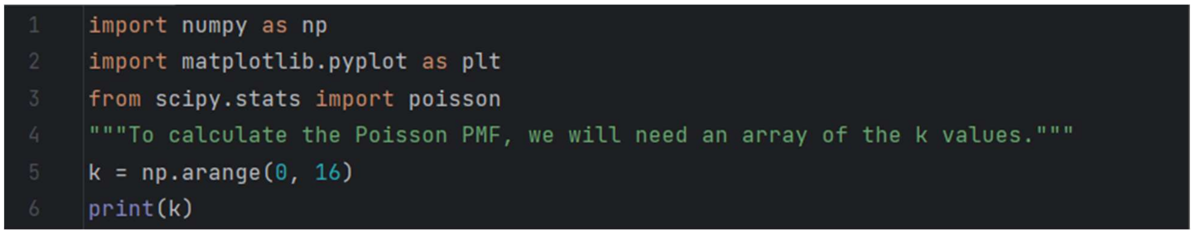
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import poisson
from numpy import random
'''Generate a random 1x10 distribution for occurence 2: '''
x= random.poisson(lam=2, size=10)
print(x)
```

➢ Output:



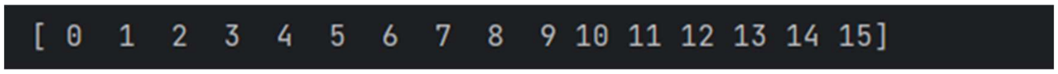
```
[5 3 4 0 3 3 1 3 3 0]
```

- Here, we need an array of the **k** values. We use **"np.arange ( )"** function and gave it numbers from 0 to 16 and print it.
  ➢ create an array with these values:



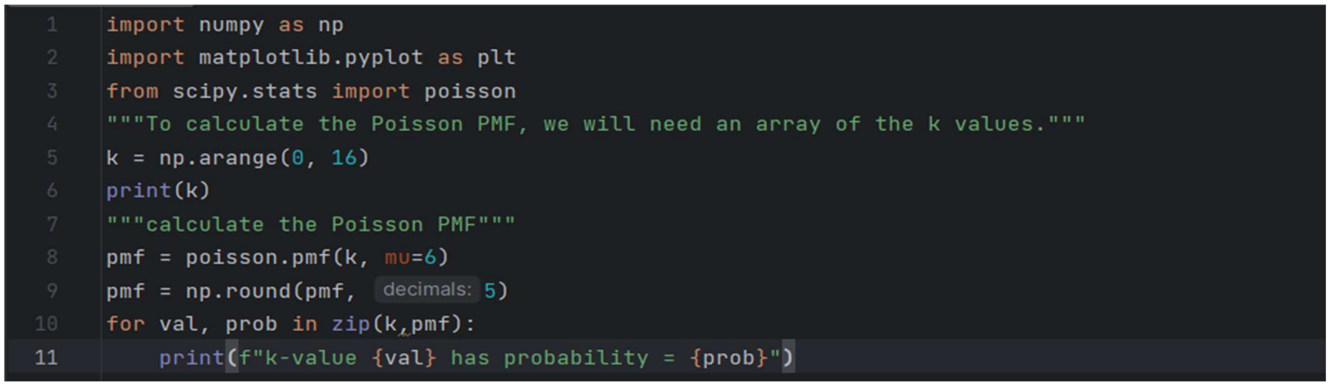
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import poisson
"""To calculate the Poisson PMF, we will need an array of the k values."""
k = np.arange(0, 16)
print(k)
```

➢ Output:



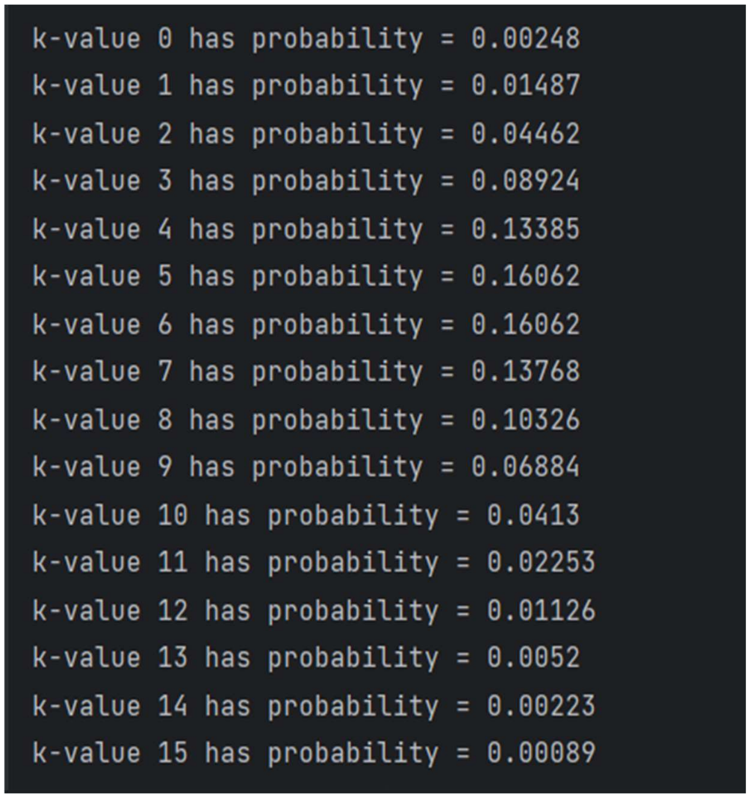
```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
```

- to calculate the Poisson PMF, we will use the **"poisson.pmf ( )"** method of the "**Scipy.poisson**" generator. It will need two parameters:
  - ➢ k value (the **k** array that we created).
  - ➢ μ value (which we will set to μ=6 as in our example).
  - ➢ **"np.round ( )" function :** to print 5 digits following the PMF's decimal.
  - ➢ We make **for loop** takes 2 parameters which are the value and the probability, then after substitute in Poisson's equation, it will give me the probability of each value.



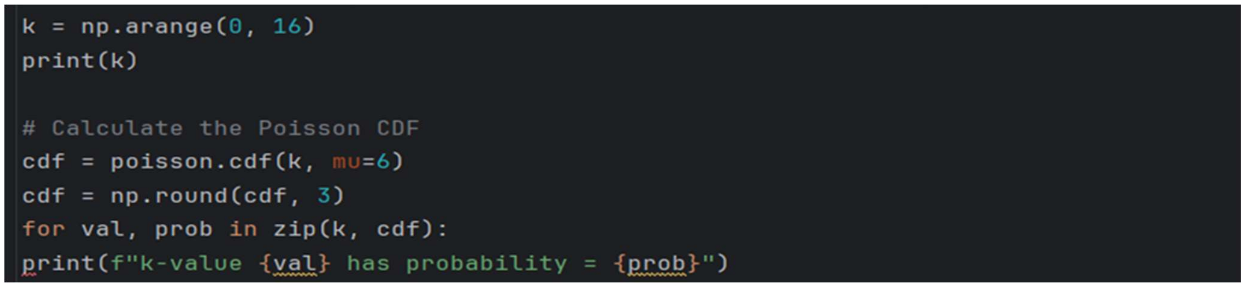
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import poisson
"""To calculate the Poisson PMF, we will need an array of the k values."""
k = np.arange(0, 16)
print(k)
"""calculate the Poisson PMF"""
pmf = poisson.pmf(k, mu=6)
pmf = np.round(pmf, decimals: 5)
for val, prob in zip(k,pmf):
    print(f"k-value {val} has probability = {prob}")
```

➢ **Output:**



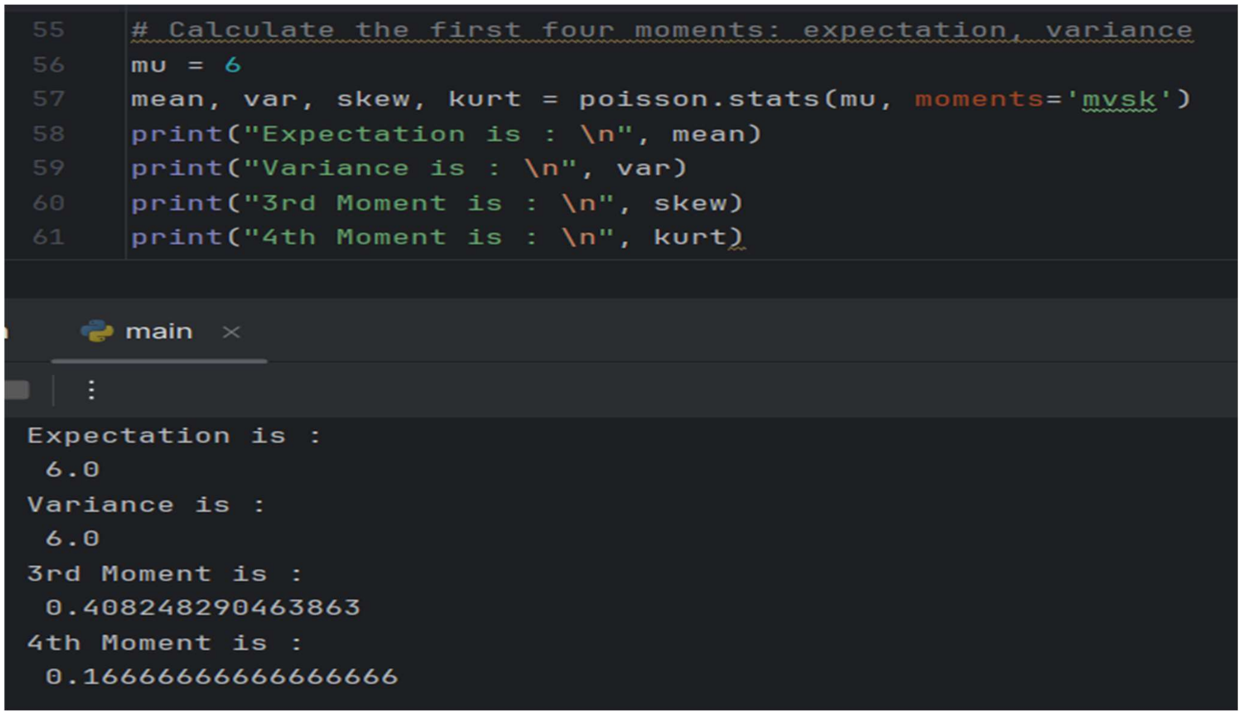
```
k-value 0 has probability = 0.00248
k-value 1 has probability = 0.01487
k-value 2 has probability = 0.04462
k-value 3 has probability = 0.08924
k-value 4 has probability = 0.13385
k-value 5 has probability = 0.16062
k-value 6 has probability = 0.16062
k-value 7 has probability = 0.13768
k-value 8 has probability = 0.10326
k-value 9 has probability = 0.06884
k-value 10 has probability = 0.0413
k-value 11 has probability = 0.02253
k-value 12 has probability = 0.01126
k-value 13 has probability = 0.0052
k-value 14 has probability = 0.00223
k-value 15 has probability = 0.00089
```

- The same steps as PMF exactly except the function which is **“poisson.cdf ()”.**
  - **“np.round()”:**This is a NumPy function that rounds each element to three decimal places.



```
k = np.arange(0, 16)
print(k)

# Calculate the Poisson CDF
cdf = poisson.cdf(k, mu=6)
cdf = np.round(cdf, 3)
for val, prob in zip(k, cdf):
print(f"k-value {val} has probability = {prob}")
```

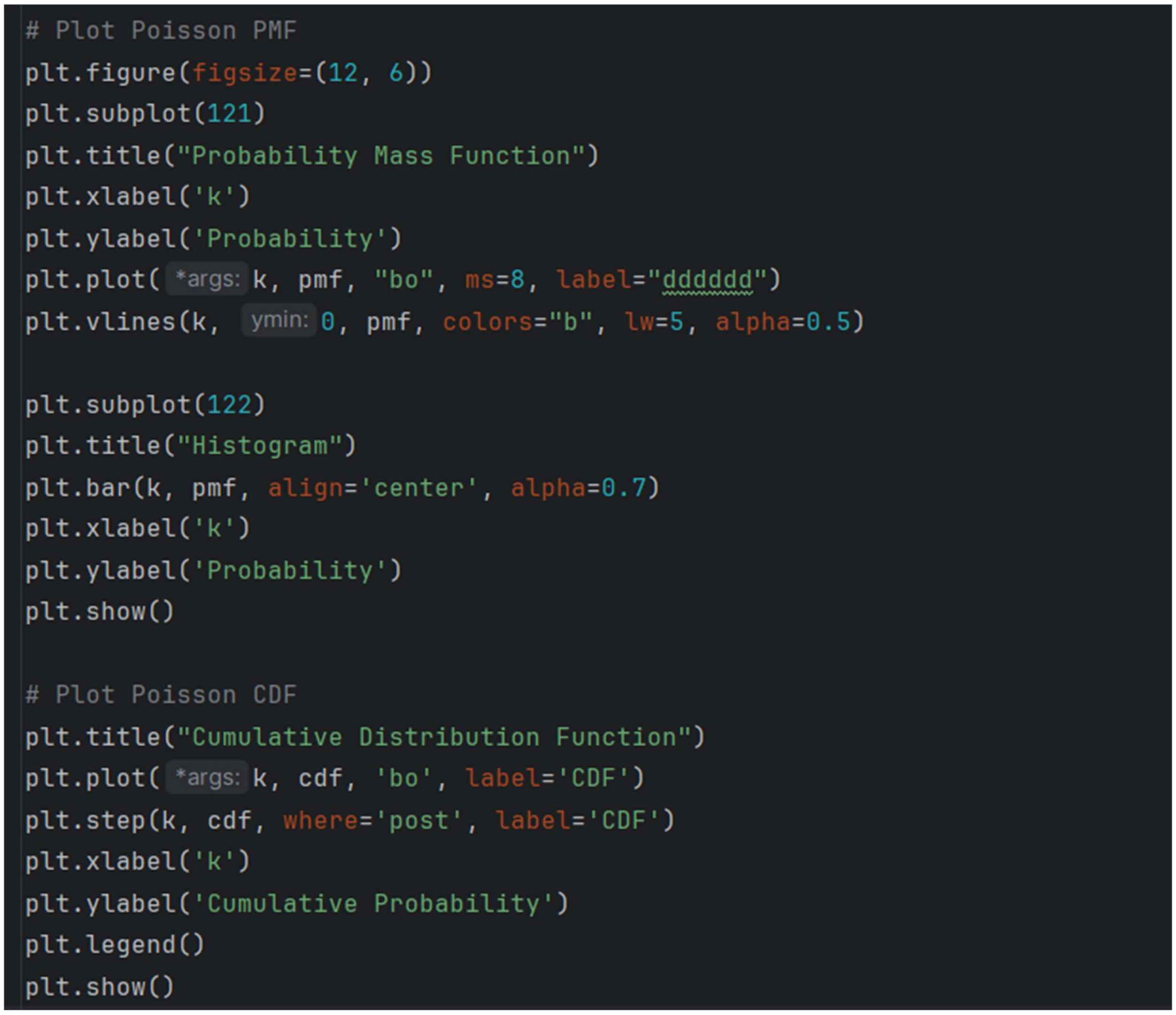
- In this code, we need to calculate the first four moment.
  - Our first moment is the expectation, we put it constant (μ=6) from the first code.
  - Second moment is the variance which is equal to the expectation.
  - Then $3^{rd}$and 4^{th} moments.



```
# Calculate the first four moments: expectation, variance
mu = 6
mean, var, skew, kurt = poisson.stats(mu, moments='mvsk')
print("Expectation is : \n", mean)
print("Variance is : \n", var)
print("3rd Moment is : \n", skew)
print("4th Moment is : \n", kurt)
```

```
Expectation is :
 6.0
Variance is :
 6.0
3rd Moment is :
 0.408248290463863
4th Moment is :
 0.16666666666666666
```

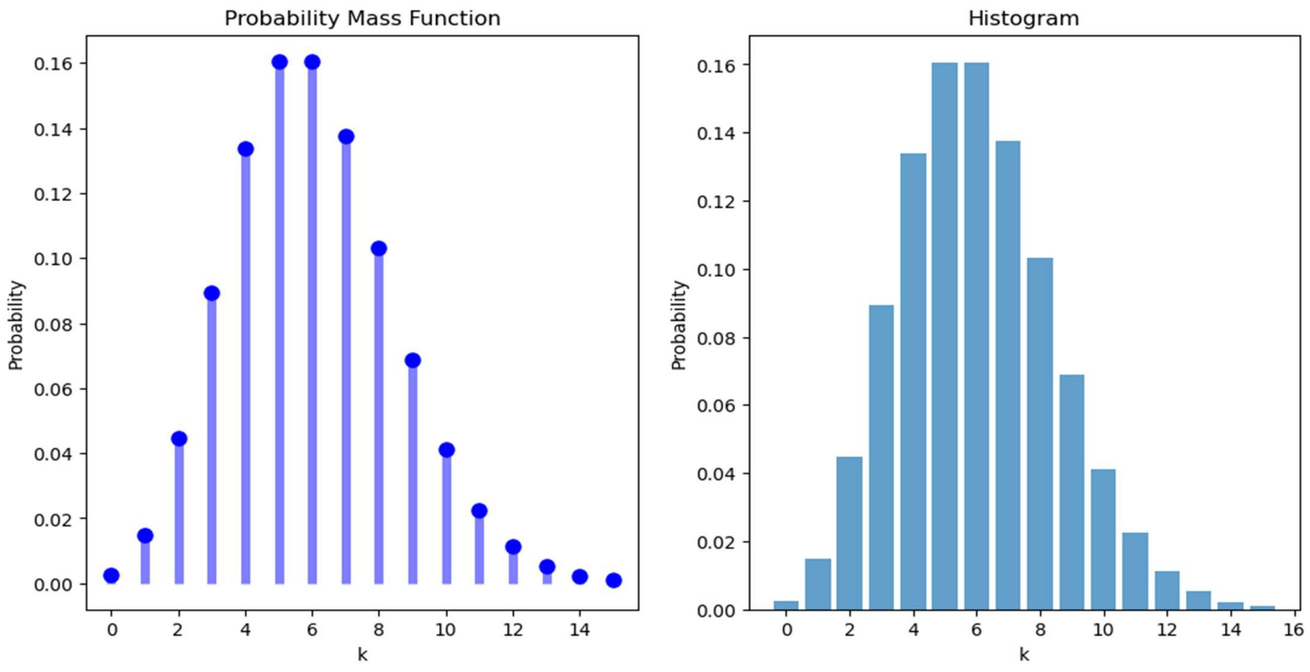
- We will need the **k** values array that we created earlier as well as the **PMF** values array in this step.
  - ➢ By using **matplotlib library**.
  - ➢ We need to plot PMF graph so, we use **"plt.bar( )"** function takes PMF value and k.
  - ➢ Then, we add label to X-axis and Y-axis by using **"plt.xlabel ( )"** and **"plt.ylabel ( )".**
- The same steps as PMF exactly except the function which is **"poisson.cdf ()".**



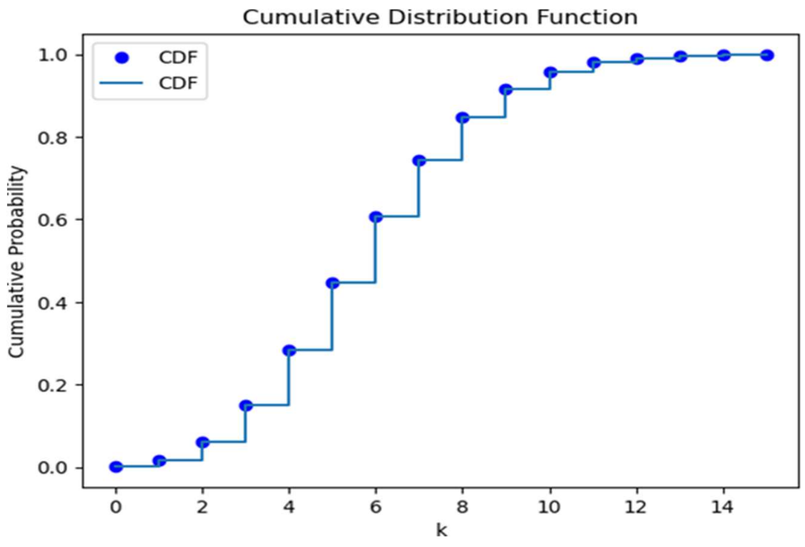
```
# Plot Poisson PMF
plt.figure(figsize=(12, 6))
plt.subplot(121)
plt.title("Probability Mass Function")
plt.xlabel('k')
plt.ylabel('Probability')
plt.plot( *args: k, pmf, "bo", ms=8, label="dddddd")
plt.vlines(k,  ymin: 0, pmf, colors="b", lw=5, alpha=0.5)

plt.subplot(122)
plt.title("Histogram")
plt.bar(k, pmf, align='center', alpha=0.7)
plt.xlabel('k')
plt.ylabel('Probability')
plt.show()

# Plot Poisson CDF
plt.title("Cumulative Distribution Function")
plt.plot( *args: k, cdf, 'bo', label='CDF')
plt.step(k, cdf, where='post', label='CDF')
plt.xlabel('k')
plt.ylabel('Cumulative Probability')
plt.legend()
plt.show()
```

V. Results

- By the above codes, finally we can be able to calculate the expectation and variance of PMF and CDF and plot the histogram of both.



(The PMF plot)



(The CDF plot)

5- Uniform Distribution:

I. Definition

The discrete uniform distribution occurs when there are a finite number (m) of equally likely outcomes possible.

II. Equation:

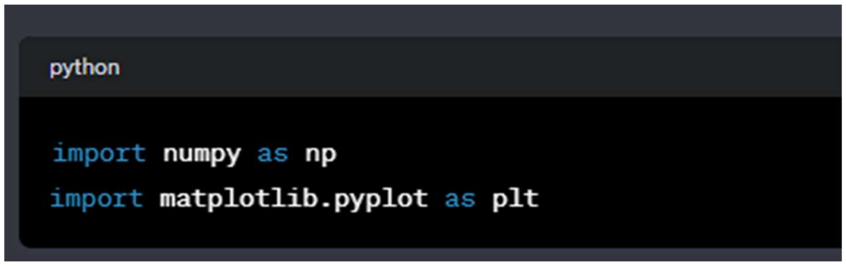
PMF $p(x) = 1 / m$, where $x=1,2,\dots,m$

III. Properties:

X~Uniform(m)

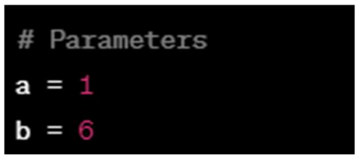
1- Mean: The expected value of a Uniform random variable X is ($E[x]= (m+1) /2$)
2- Variance: $Var[x] = (m^2 - 1) / 12$

IV. Code



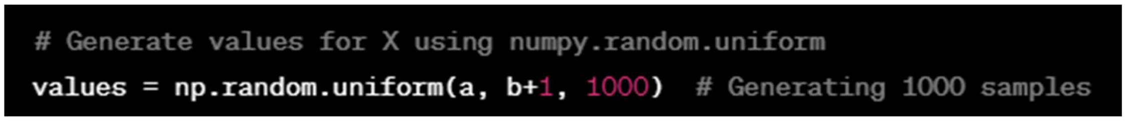
```python
import numpy as np
import matplotlib.pyplot as plt
```

- This section imports the necessary libraries: **numpy** for numerical operations and **matplotlib.pyplot** for plotting.



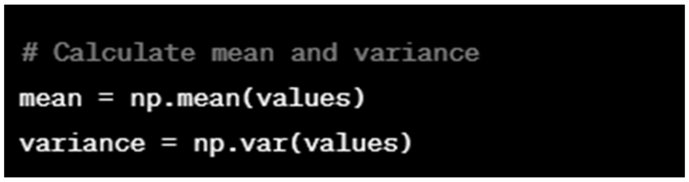
```
# Parameters
a = 1
b = 6
```

- Here, you define the parameters **a** and **b**, which represent the range of values for a discrete uniform distribution.



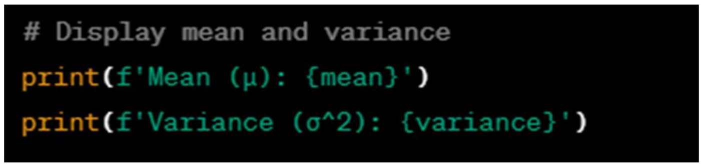
```
# Generate values for X using numpy.random.uniform
values = np.random.uniform(a, b+1, 1000)  # Generating 1000 samples
```

- This line generates 1000 random samples from a uniform distribution in the range **[a, b+1]** using **numpy.random.uniform**. These samples represent the random variable X.



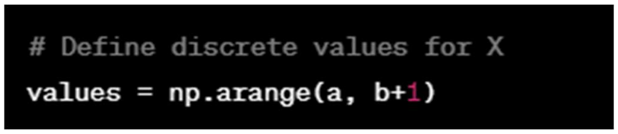
```
# Calculate mean and variance
mean = np.mean(values)
variance = np.var(values)
```

- Here, you calculate the mean and variance of the generated samples using **numpy.mean** and **numpy.var**.



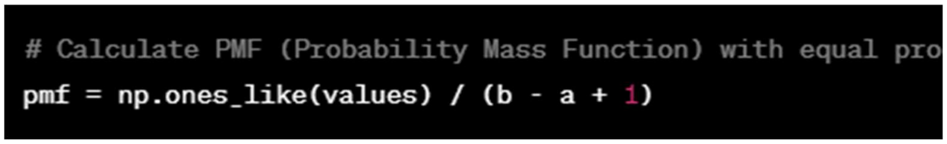
```
# Display mean and variance
print(f'Mean (μ): {mean}')
print(f'Variance (σ^2): {variance}')
```

- This prints the calculated mean and variance to the console.



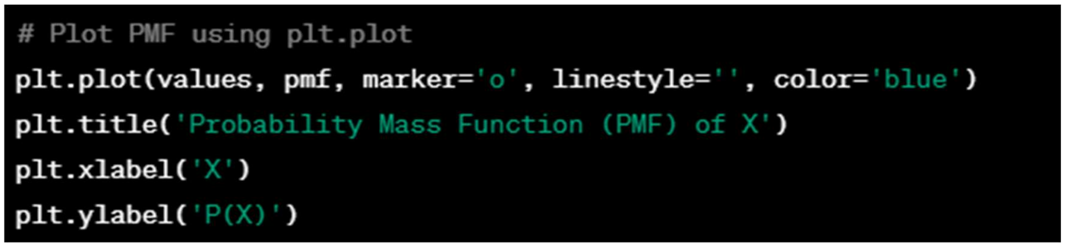
```
# Define discrete values for X
values = np.arange(a, b+1)
```

- This line creates an array of discrete values for X in the range **[a, b+1]** using **numpy.arange**.



```
# Calculate PMF (Probability Mass Function) with equal pro
pmf = np.ones_like(values) / (b - a + 1)
```

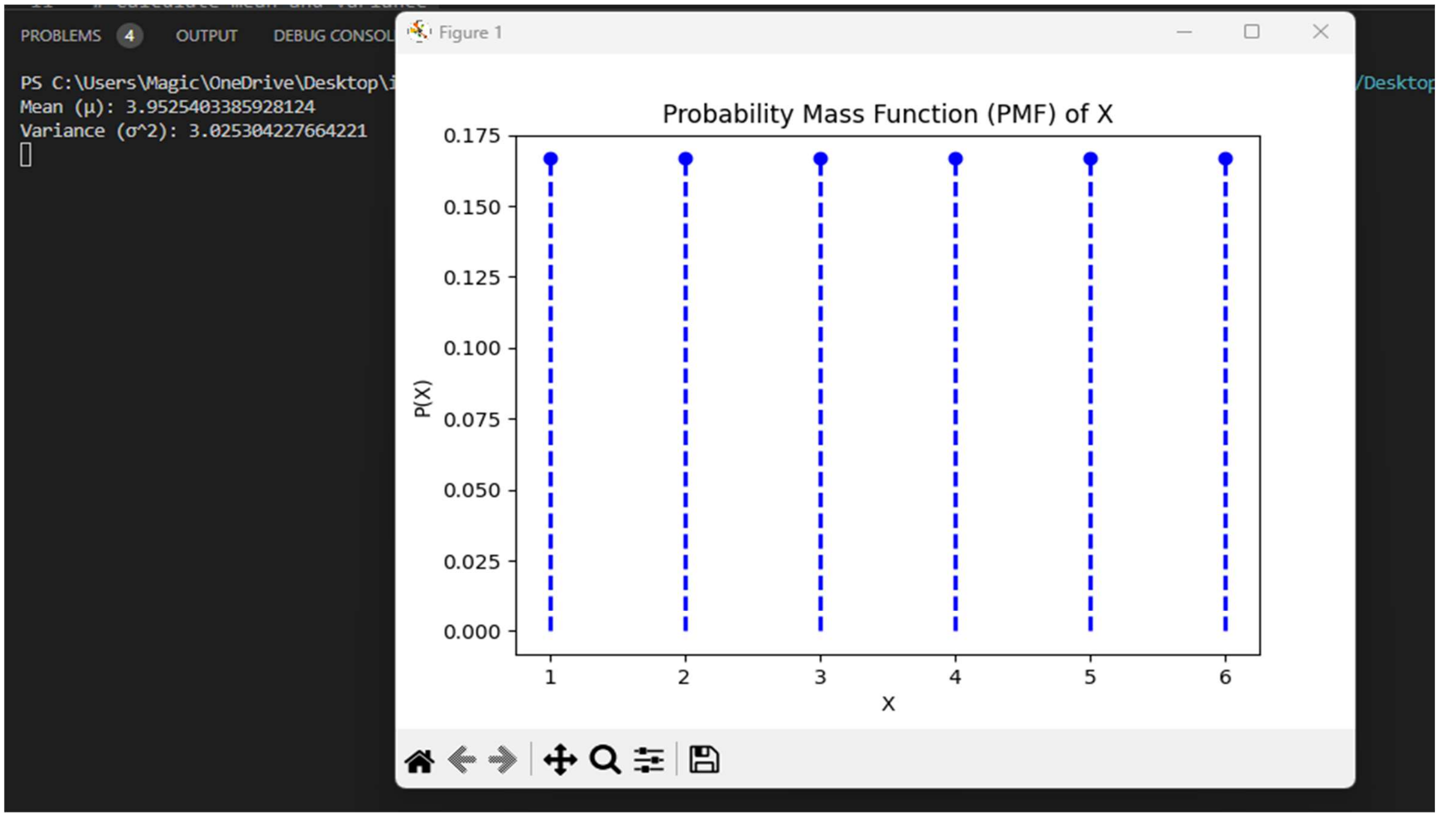
- Here, you calculate the Probability Mass Function (PMF) for the discrete values. In a discrete uniform distribution, each value has an equal probability, so the PMF is a constant value for each possible outcome.



```
# Plot PMF using plt.plot
plt.plot(values, pmf, marker='o', linestyle='', color='blue')
plt.title('Probability Mass Function (PMF) of X')
plt.xlabel('X')
plt.ylabel('P(X)')
```

- This block of code plots the PMF using **plt.plot**. The **marker='o'** option adds circular markers for each point.
- The **title**, **xlabel**, and **ylabel** functions add labels and a title to the plot.

V. Results



```
PS C:\Users\Magic\OneDrive\Desktop\i
Mean (μ): 3.9525403385928124
Variance (σ^2): 3.025304227664221
```

➢ Continuous Random Variable

Continuous random variable is a random variable that has only continuous values. Continuous values are uncountable and are related to real numbers

1. Continuous uniform distribution:

I. Definition:

The uniform distribution is a symmetric probability distribution where all outcomes have an equal probability of occurring. All values in the distribution have a constant probability, making them uniformly distributed.

II. Equation

$$\mathbb{P}[a \leq X \leq b] = \int_a^b f_X(x)dx.$$

III. Properties:

$$\text{Mean}(\mu) = (\text{E}(\text{x}) = \frac{a+b}{2}$$

$$\text{Variance } (\sigma^2) = \frac{(b-a)^2}{12}$$

PDF:

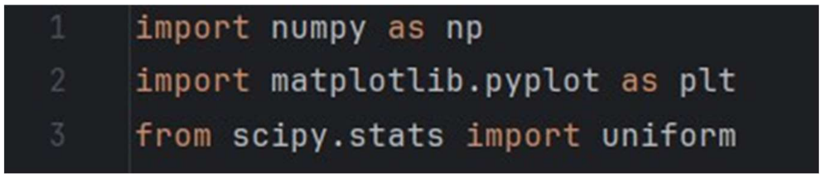
$$f_X(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b, \\ 0, & \text{otherwise}, \end{cases}$$

CDF:

$$F_X(x) = \begin{cases} 0, & x < a, \\ \frac{x-a}{b-a}, & a \leq x \leq b, \\ 1, & x > b. \end{cases}$$

IV. Code

- **First**: we import the libraries we need for executing the code

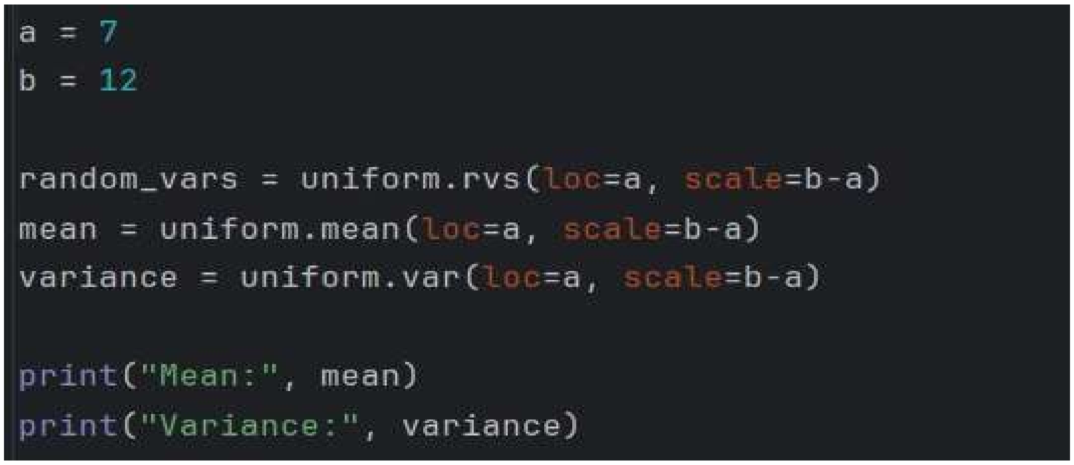


```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   from scipy.stats import uniform
```

➢ **Line 1: numpy** library helps in mathematical and logical operations on arrays

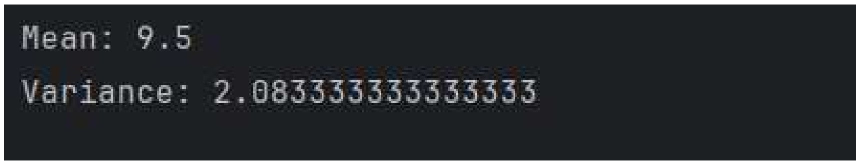
- **Line 2:** we use **matplotlib library** for used for plotting the pdf and cdf
- **Line 3:** the **scipy.stats library** used for statistical and probabilistic operations.

- **Second:** To calculate and plot the uniform distribution we should set two parameters : a is the lower bound, b is the upper bound
  and then generate the random variable, the mean and the variance
  - The **uniform.rvs()** , **uniform.mean()** and **uniform.var()** functions are associated with the **scipy.stats library**
  - The context of these functions contains (**loc=a**) which determines that the starting point is a and (**scale=b-a**) determines the width of the distribution



```
a = 7
b = 12

random_vars = uniform.rvs(loc=a, scale=b-a)
mean = uniform.mean(loc=a, scale=b-a)
variance = uniform.var(loc=a, scale=b-a)

print("Mean:", mean)
print("Variance:", variance)
```

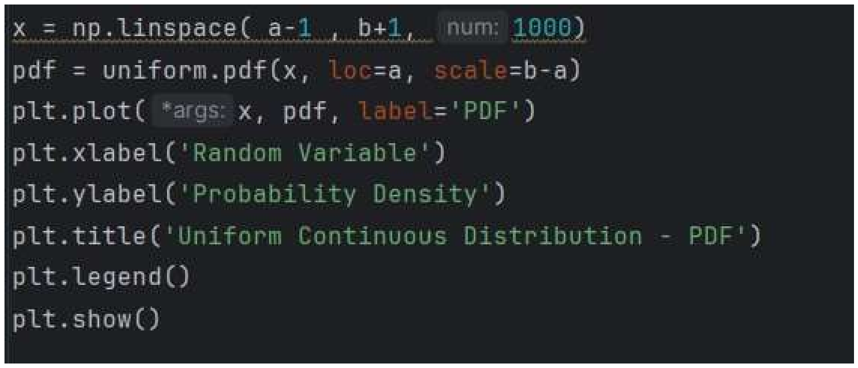
- **output** of printing Mean and Variance:



```
Mean: 9.5
Variance: 2.0833333333333333
```

- **Third:** After calculating mean and variance and generating the random variables we will plot PDF and CDF using the following codes:

- For PDF:



```
x = np.linspace( a-1 , b+1,  num: 1000)
pdf = uniform.pdf(x, loc=a, scale=b-a)
plt.plot( *args: x, pdf, label='PDF')
plt.xlabel('Random Variable')
plt.ylabel('Probability Density')
plt.title('Uniform Continuous Distribution - PDF')
plt.legend()
plt.show()
```

➢ The **np.linespace( )** function generates an array name x contains 1000 evenly distributed values spaced between a-1 and b+1

➢ the **uniform.pdf()** function used to calculate the PDF of the distribution of array x

➢ **plt.plot()** plots the PDF using x array values as x-axis and pdf values as y-axis

➢ **plt.xlabel()** and **plt.ylabel()** sets the labels of x-axis and y-axis

➢ **plt.title()** displays the title of the plot

➢ **plt.legend()** displays the legend of the plot

➢ **plt.show()** displays the plot on the screen

- **For CDF:**

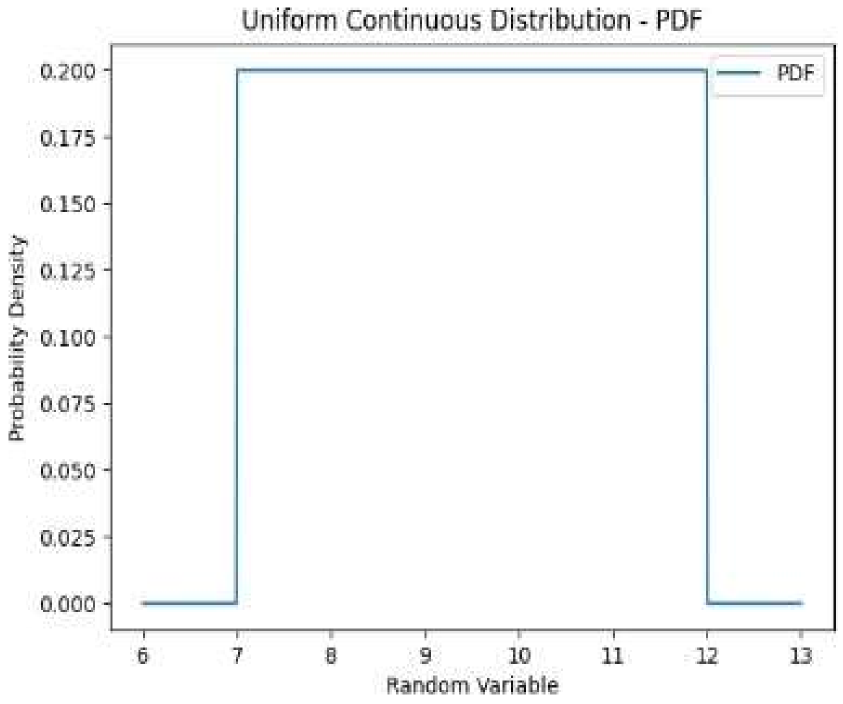
➢ The **np.linespace( )** function generates an array name x_cdf contains 1000 evenly distributed values spaced between a-1 and b+1 and this array will be used as x-axis in plotting

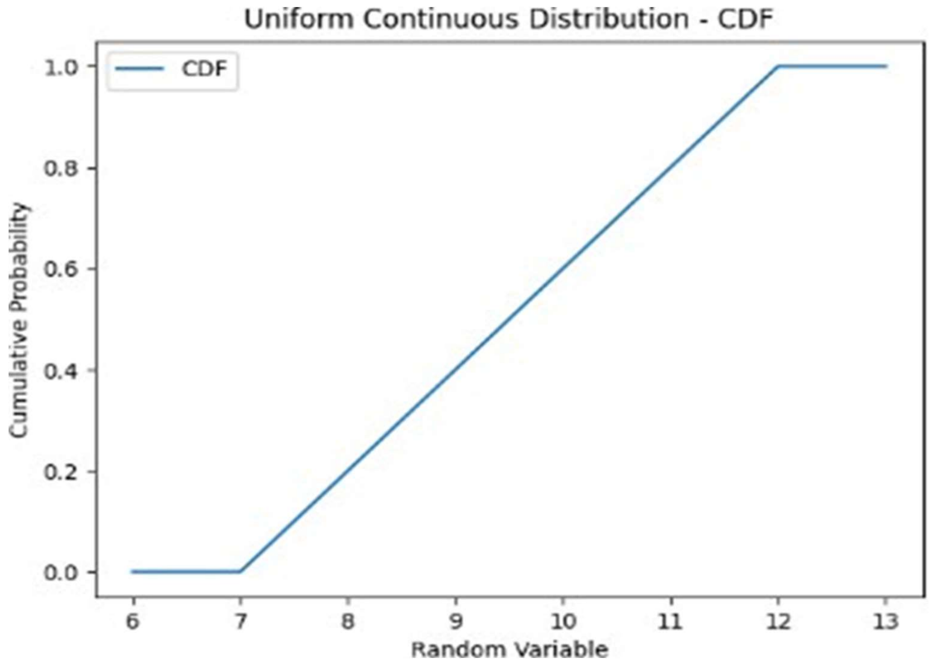
➢ the **uniform.pdf()** function used to calculate the PDF of the distribution of array x_cdf

➢ **plt.plot()** plots the CDF using x_cdf array values as x-axis and cdf array values as y-axis

➢ The next 5 lines function is the same as in PDF plotting

V. Results


Uniform Continuous Distribution - PDF
PDF
Probability Density
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
6
7
8
9
10
11
12
13
Random Variable


Uniform Continuous Distribution - CDF
CDF
Cumulative Probability
1.0
0.8
0.6
0.4
0.2
0.0
6
7
8
9
10
11
12
13
Random Variable

2. Exponential Continuous Distribution

I. Definition

The Exponential distribution is a continuous probability distribution that often concerns the amount of time until some specific event happens. It is a process in which events happen continuously and independently at a constant average rate.

II. Equation

PDF:

$$f_X(x|\lambda) = \begin{cases} \lambda e^{-\lambda x} & for\ x > 0 \\ 0 & for\ x \leq 0 \end{cases}$$

Where

λ is called the distribution rate.

CDF:

$$F_X(x) = \begin{cases} 0, & x < 0, \\ 1 - e^{-\lambda x}, & x \geq 0. \end{cases}$$

III. Properties:

Mean:

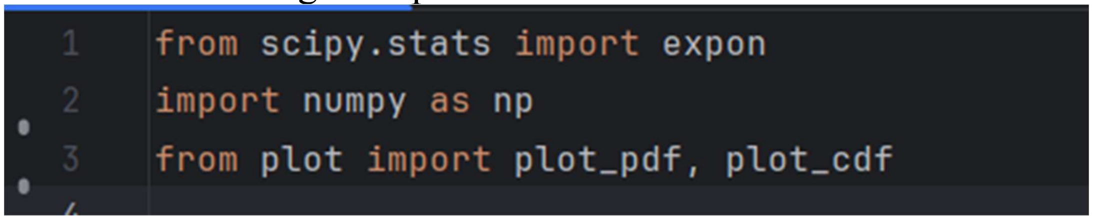
$$\mathbb{E}[X] = \frac{1}{\lambda},$$

Variance:

$$\mathrm{Var}[X] = \frac{1}{\lambda^2}.$$

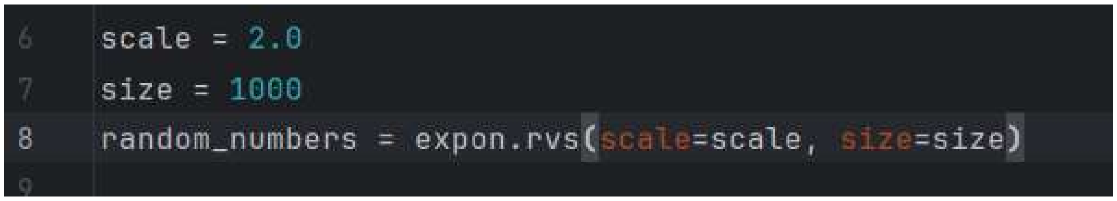
IV. Code

- The code imports the necessary libraries and modules for performing statistical analysis on data following an exponential distribution.



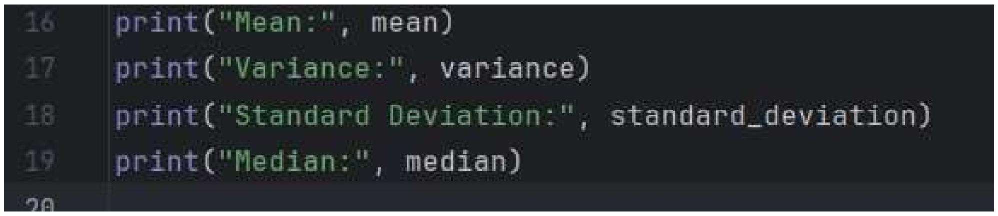
```
from scipy.stats import expon
import numpy as np
from plot import plot_pdf, plot_cdf
```

- Now, we set Random numbers it's size 1000
  - ➢ Exponential distribution using expon .rvs() function which scale is set to 2.0 ,which is λ value is 1/scale=0.5



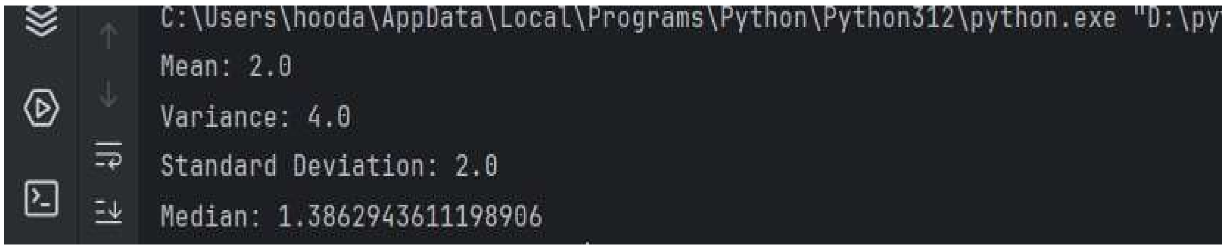
```
scale = 2.0
size = 1000
random_numbers = expon.rvs(scale=scale, size=size)
```

- Calculating statistics such as mean, variance, standard deviation and median using function from expon class.



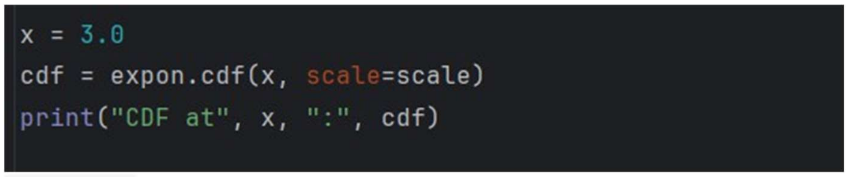
```
print("Mean:", mean)
print("Variance:", variance)
print("Standard Deviation:", standard_deviation)
print("Median:", median)
```

Output



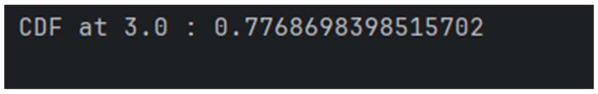
```
C:\Users\hooda\AppData\Local\Programs\Python\Python312\python.exe "D:\py
Mean: 2.0
Variance: 4.0
Standard Deviation: 2.0
Median: 1.3862943611198906
```

- Calculating the Cumulative distribution function (CDF) at a given value using functions from the expon



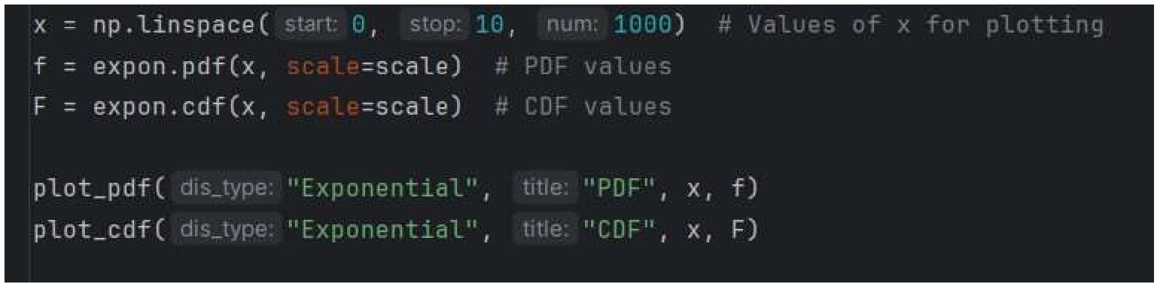
```
x = 3.0
cdf = expon.cdf(x, scale=scale)
print("CDF at", x, ":", cdf)
```

Output



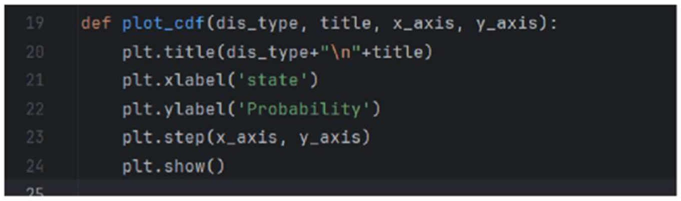
```
CDF at 3.0 : 0.7768698398515702
```

- The code generates plots of the probability density function (PDF) and cumulative distribution function (CDF) of the exponential distribution. It creates an array of x values using np.linspace() 1000 values between 0 and 10 and calculates the corresponding PDF and CDF values using expon.pdf() and expon.cdf().



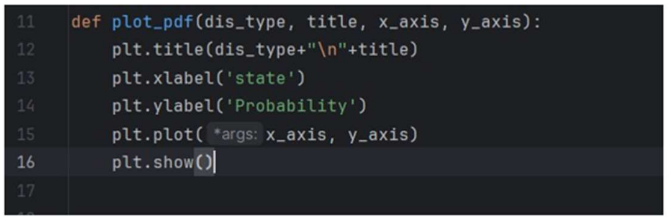
```
x = np.linspace( start: 0,  stop: 10,  num: 1000)  # Values of x for plotting
f = expon.pdf(x, scale=scale)  # PDF values
F = expon.cdf(x, scale=scale)  # CDF values

plot_pdf( dis_type: "Exponential",  title: "PDF", x, f)
plot_cdf( dis_type: "Exponential",  title: "CDF", x, F)
```

Function plot_cdf :



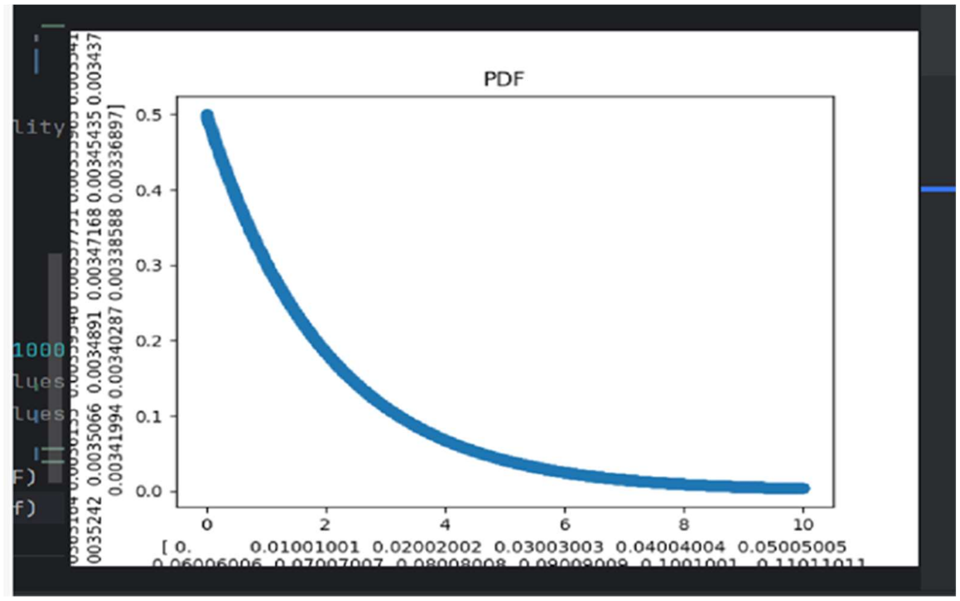
```
def plot_cdf(dis_type, title, x_axis, y_axis):
    plt.title(dis_type+"\n"+title)
    plt.xlabel('state')
    plt.ylabel('Probability')
    plt.step(x_axis, y_axis)
    plt.show()
```

Function plot_pdf :

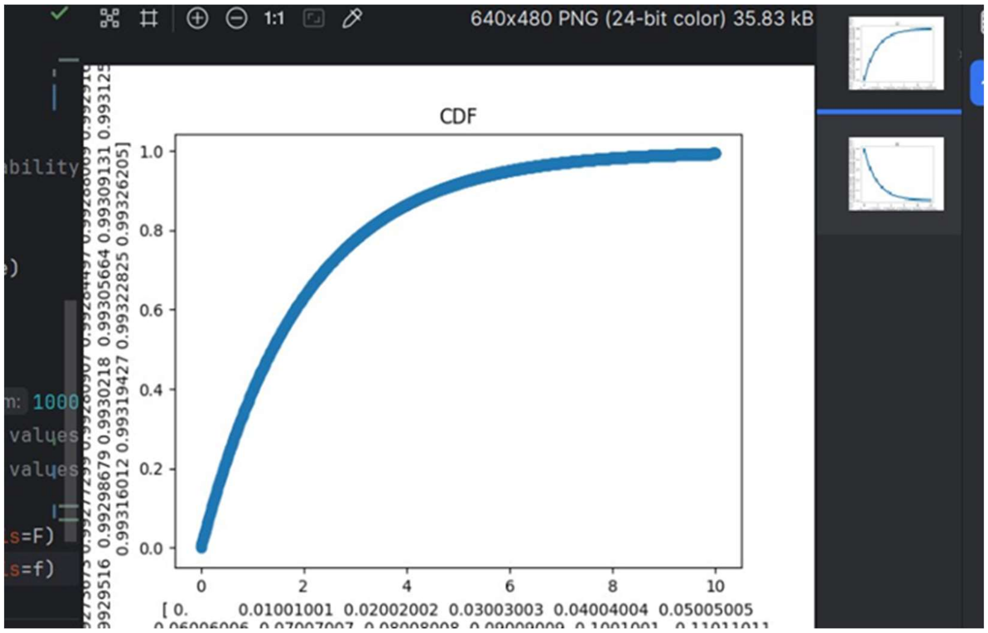


```
def plot_pdf(dis_type, title, x_axis, y_axis):
    plt.title(dis_type+"\n"+title)
    plt.xlabel('state')
    plt.ylabel('Probability')
    plt.plot( *args: x_axis, y_axis)
    plt.show()
```

V. Results



```
0035242 0.0035066 0.0034891 0.00347168 0.00345435 0.003437
 0.00341994 0.00340287 0.00338588 0.00336897]
[ 0.         0.01001001  0.02002002  0.03003003  0.04004004  0.05005005
```



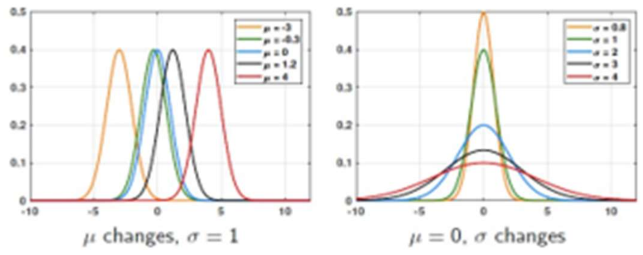
```
99929516 0.99298679 0.99302218 0.99305664 0.993309131 0.993125
 0.993316012 0.993319427 0.993322825 0.99326205]
[ 0.         0.01001001  0.02002002  0.03003003  0.04004004  0.05005005
```

3. Gaussian random variable

I. Definition:

A continuous random variable with probability density function (PDF) of the form $P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$

Where (μ, σ^2) are parameters of the distribution



II. Equation:

Let X be an Gaussian random variable. The PDF of X is

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The **CDF** of the standard Gaussian is defined as the $\Phi(\cdot)$ function

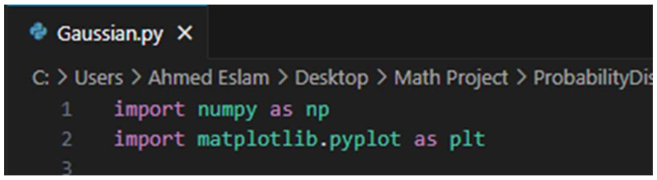
$$\Phi(x) \overset{\text{def}}{=} F_X(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt.$$

III. Properties:

$$\begin{aligned}
\text{mean} &= \mathbb{E}[X] \overset{\text{def}}{=} \mu, \\
\text{variance} &= \mathbb{E}\left[(X-\mu)^2\right] \overset{\text{def}}{=} \sigma^2, \\
\text{skewness} &= \mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] \overset{\text{def}}{=} \gamma, \\
\text{kurtosis} &= \mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] \overset{\text{def}}{=} \kappa.
\end{aligned}$$

IV. Code

- **Frist** Importing needed libraries



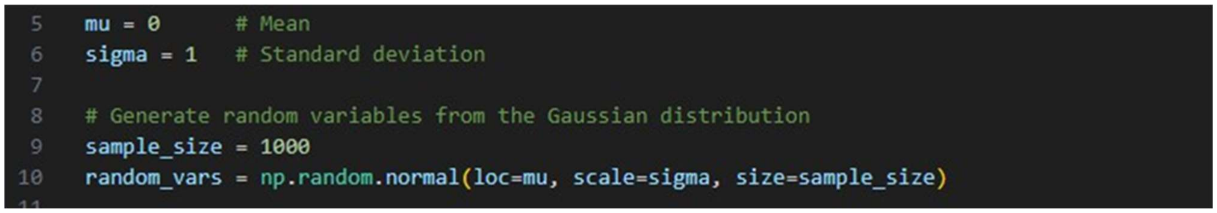
Gaussian.py ×

C: > Users > Ahmed Eslam > Desktop > Math Project > ProbabilityDis

```python
import numpy as np
import matplotlib.pyplot as plt
```

  - "numpy" as np for numerical operations, "matplotlib.pyplot" as plt for graphing the function

- **Second** Setting parameters for Gaussian and generating random variables

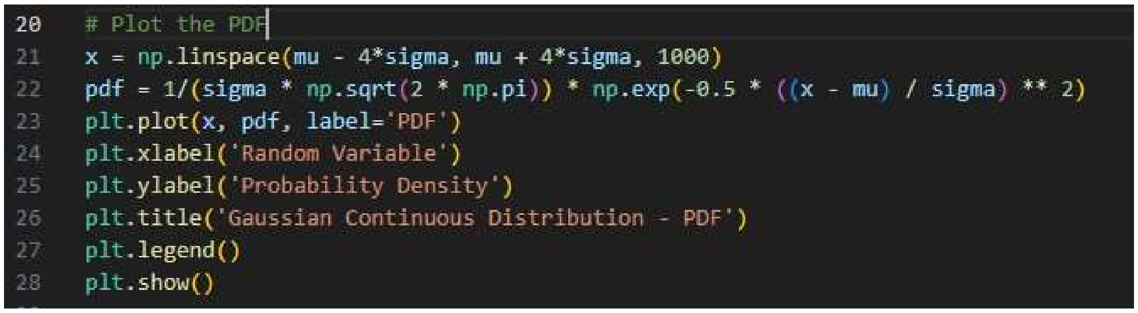


```python
mu = 0       # Mean
sigma = 1    # Standard deviation

# Generate random variables from the Gaussian distribution
sample_size = 1000
random_vars = np.random.normal(loc=mu, scale=sigma, size=sample_size)
```

  - The mean (**mu**) is the average value and standard deviation (sigma) is a measure of the spread of the distribution
  - **"np.random.normal"** generates a set of random number following a normal distribution.

- **Third** Calculating and printing Mean and Variance
  - **"np.mean","np.var"** calculate mean and variance of the generated random variables from step 2

- **Fourth** Plotting Probability Density Function (PDF)

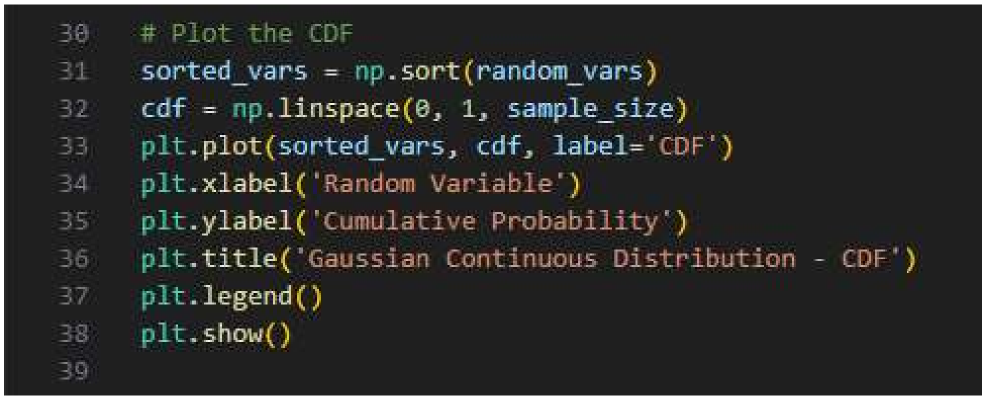


```python
# Plot the PDF
x = np.linspace(mu - 4*sigma, mu + 4*sigma, 1000)
pdf = 1/(sigma * np.sqrt(2 * np.pi)) * np.exp(-0.5 * ((x - mu) / sigma) ** 2)
plt.plot(x, pdf, label='PDF')
plt.xlabel('Random Variable')
plt.ylabel('Probability Density')
plt.title('Gaussian Continuous Distribution - PDF')
plt.legend()
plt.show()
```

  - **"np.linespace"** creates an array of evenly spaced values between two values

    PDF is calculated and plotted by **"plt.plot"** and displayed by **"plt.show()"**

- **Fifth** Plotting Cumulative Distribution Function

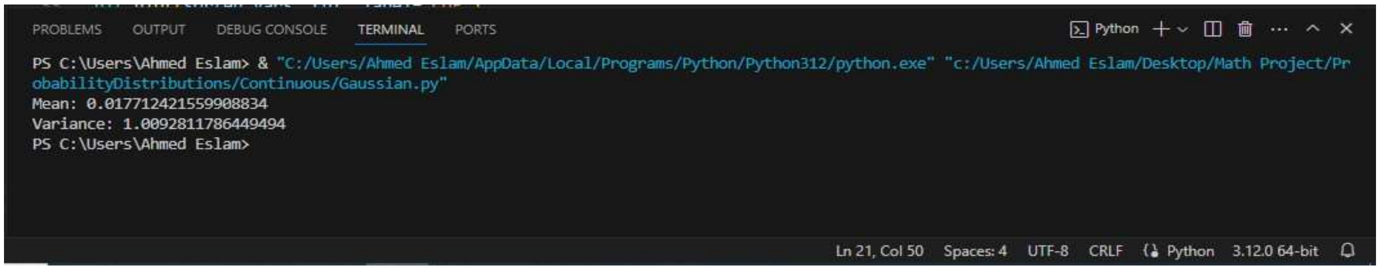


```
# Plot the CDF
sorted_vars = np.sort(random_vars)
cdf = np.linspace(0, 1, sample_size)
plt.plot(sorted_vars, cdf, label='CDF')
plt.xlabel('Random Variable')
plt.ylabel('Cumulative Probability')
plt.title('Gaussian Continuous Distribution - CDF')
plt.legend()
plt.show()
```

➢ **"np.sort"** sort the generated random variable in ascending order

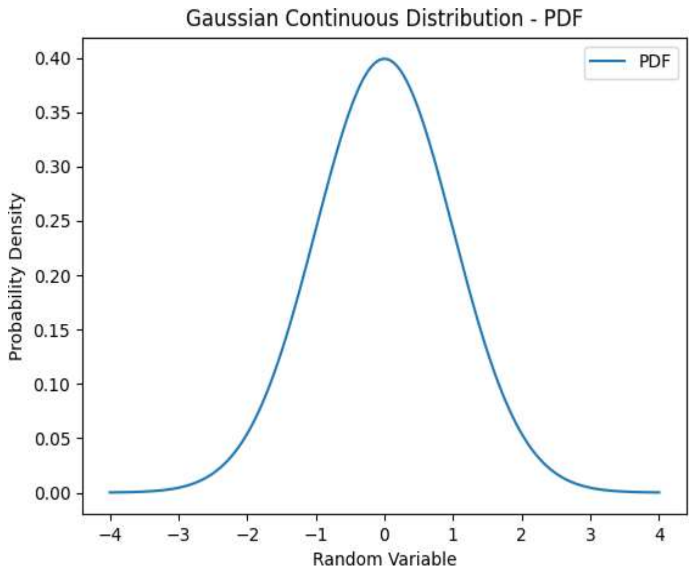
V. Results:

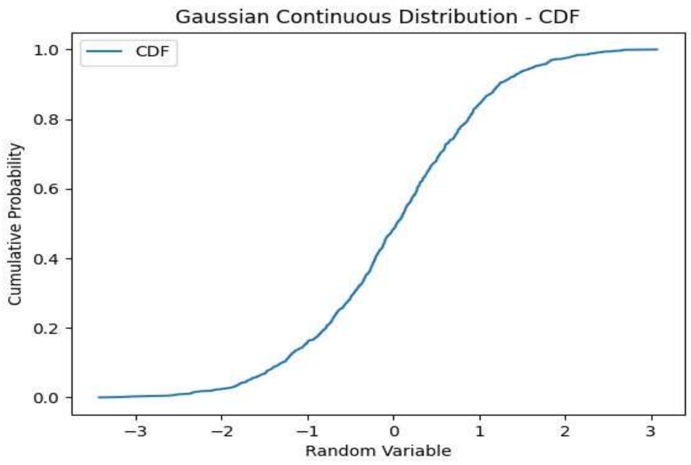
1- Mean and Variance



```
PS C:\Users\Ahmed Eslam> & "C:/Users/Ahmed Eslam/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Ahmed Eslam/Desktop/Math Project/ProbabilityDistributions/Continuous/Gaussian.py"
Mean: 0.017712421559908834
Variance: 1.0092811786449494
PS C:\Users\Ahmed Eslam>
```

2- PDF and CDF


Gaussian Continuous Distribution - PDF
PDF
Probability Density
Random Variable


Gaussian Continuous Distribution - CDF
CDF
Cumulative Probability
Random Variable