

PROGRAM: Generate data for coordinates of a projectile and plot the trajectory. Determine the range, maximum height and time of flight for a projectile motion.

```
*untitled*
File Edit Format Run Options Window Help

import matplotlib.pyplot as plt
import numpy as np

def projectile_motion(v0, theta, g=9.8):
    # Time of flight
    t_flight = (2 * v0 * np.sin(theta)) / g

    # Range
    range = (v0**2) * np.sin(2 * theta) / g

    # Maximum height
    max_height = (v0**2) * np.sin(theta)**2 / (2 * g)

    # Time and displacement arrays
    t = np.linspace(0, t_flight, 100)
    x = v0 * np.cos(theta) * t
    y = v0 * np.sin(theta) * t - 0.5 * g * t**2

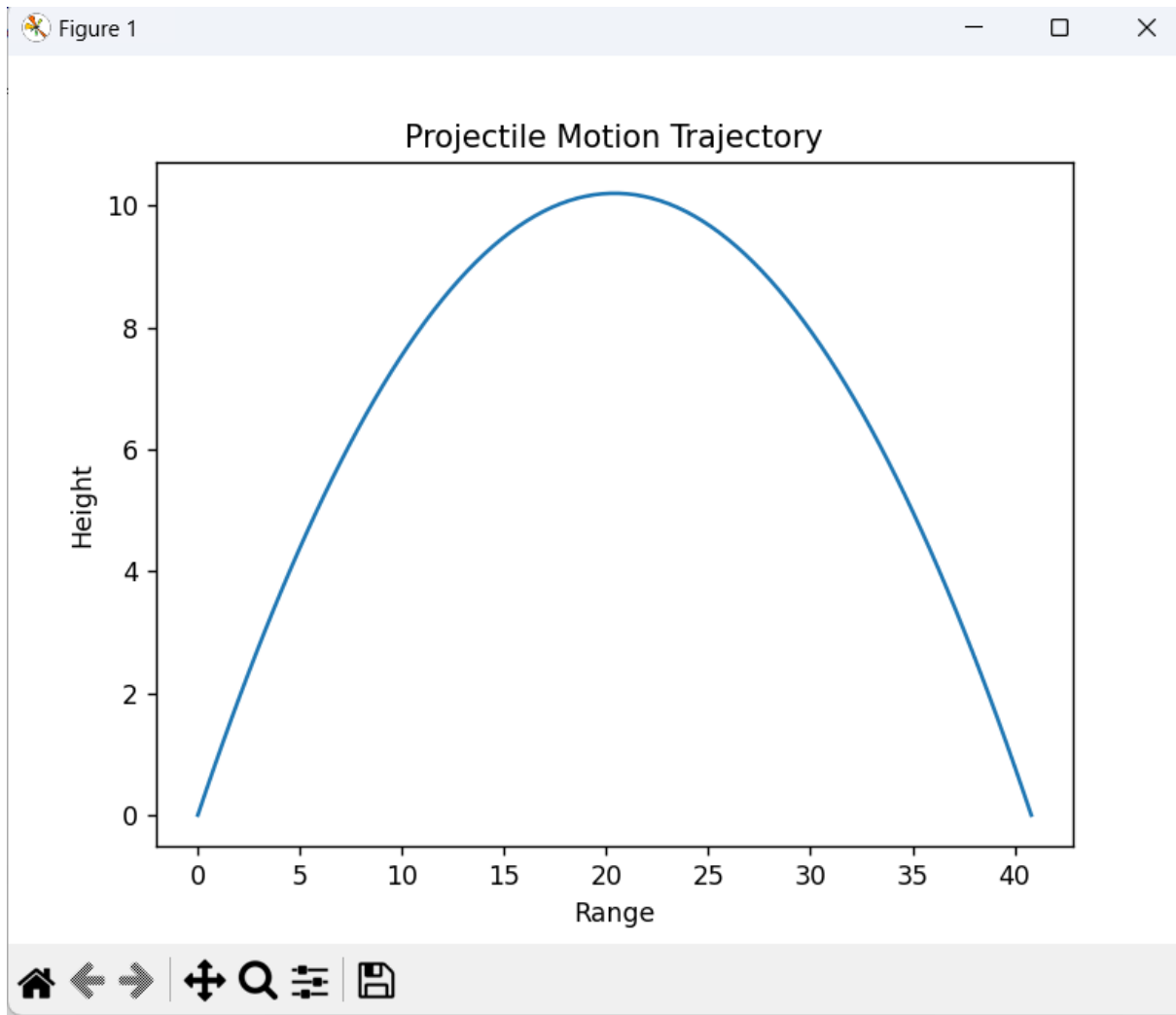
    return t, x, y, range, max_height, t_flight

# Example usage
v0 = 20
theta = np.pi / 4
t, x, y, range, max_height, t_flight = projectile_motion(v0, theta)

# Plotting the trajectory
plt.plot(x, y)
plt.xlabel('Range')
plt.ylabel('Height')
plt.title('Projectile Motion Trajectory')
plt.show()

# Printing the results
print('Range: ', range)
print('Maximum Height: ', max_height)
print('Time of Flight: ', t_flight)
```

OUTPUT:



```
=====
Range:  40.816326530612244
Maximum Height:  10.204081632653063
Time of Flight:  2.8861501272920305
```

PROGRAM: To plot the displacement-time and velocity-time graph for an un-damped , underdamped , critically damped and over damped oscillator using matplotlib using given formulae.

```
*GARIMASINGHPROGRAMS.py - C:/Users/thedy/AppData/Local/Programs/Python/Python311/GARIMASINGHPROGRAMS.py (3.11.1)*
File Edit Format Run Options Window Help

import matplotlib.pyplot as plt
import numpy as np

# Function to plot displacement-time graph for a damped oscillator
def plot_displacement(omega, alpha, t_start, t_end, title):
    t = np.linspace(t_start, t_end, 1000)
    x = np.exp(-alpha * t) * np.cos(omega * t)
    plt.plot(t, x)
    plt.title(title)
    plt.xlabel("Time (s)")
    plt.ylabel("Displacement (m)")
    plt.show()

# Function to plot velocity-time graph for a damped oscillator
def plot_velocity(omega, alpha, t_start, t_end, title):
    t = np.linspace(t_start, t_end, 1000)
    v = -omega * np.exp(-alpha * t) * np.sin(omega * t) - alpha * np.exp(-alpha * t) * np.cos(omega * t)
    plt.plot(t, v)
    plt.title(title)
    plt.xlabel("Time (s)")
    plt.ylabel("Velocity (m/s)")
    plt.show()

# Plot displacement-time graph for undamped oscillator
plot_displacement(1, 0, 0, 10, "Displacement-Time Graph for Undamped Oscillator")

# Plot velocity-time graph for undamped oscillator
plot_velocity(1, 0, 0, 10, "Velocity-Time Graph for Undamped Oscillator")

# Plot displacement-time graph for underdamped oscillator
plot_displacement(1, 0.1, 0, 10, "Displacement-Time Graph for Underdamped Oscillator")

# Plot velocity-time graph for underdamped oscillator
plot_velocity(1, 0.1, 0, 10, "Velocity-Time Graph for Underdamped Oscillator")

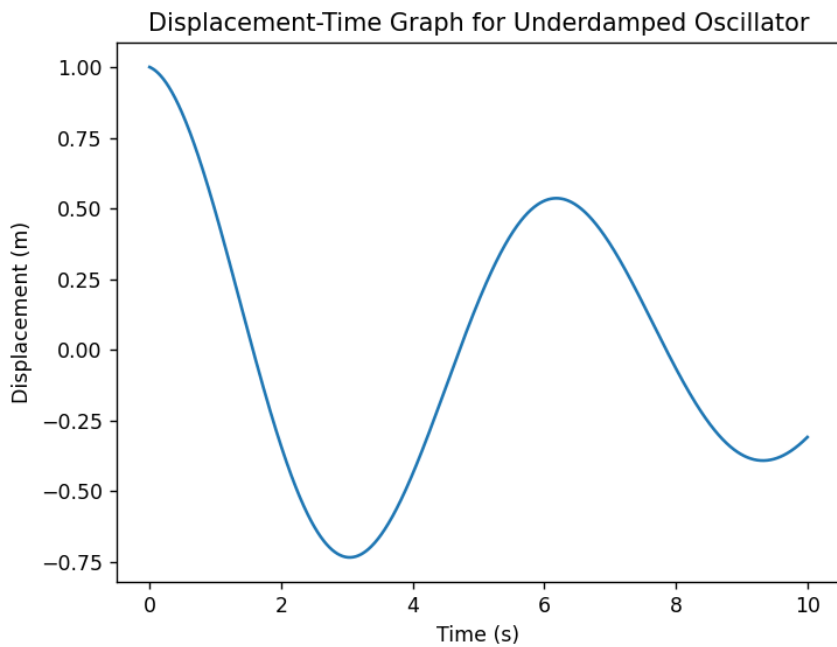
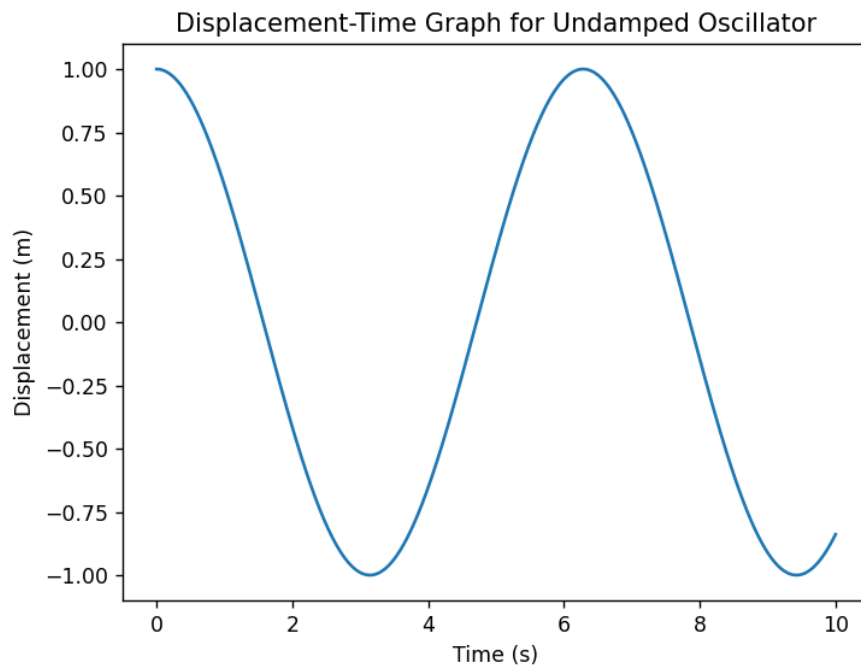
# Plot displacement-time graph for critically damped oscillator
plot_displacement(1, 1, 0, 10, "Displacement-Time Graph for Critically Damped Oscillator")

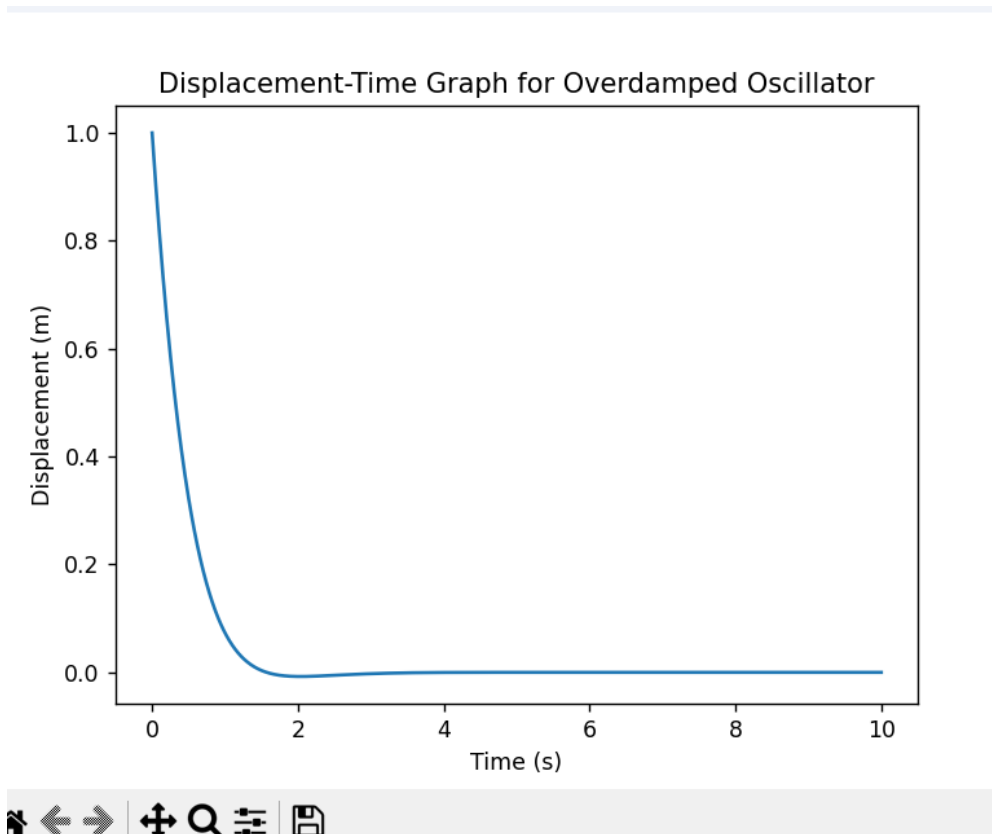
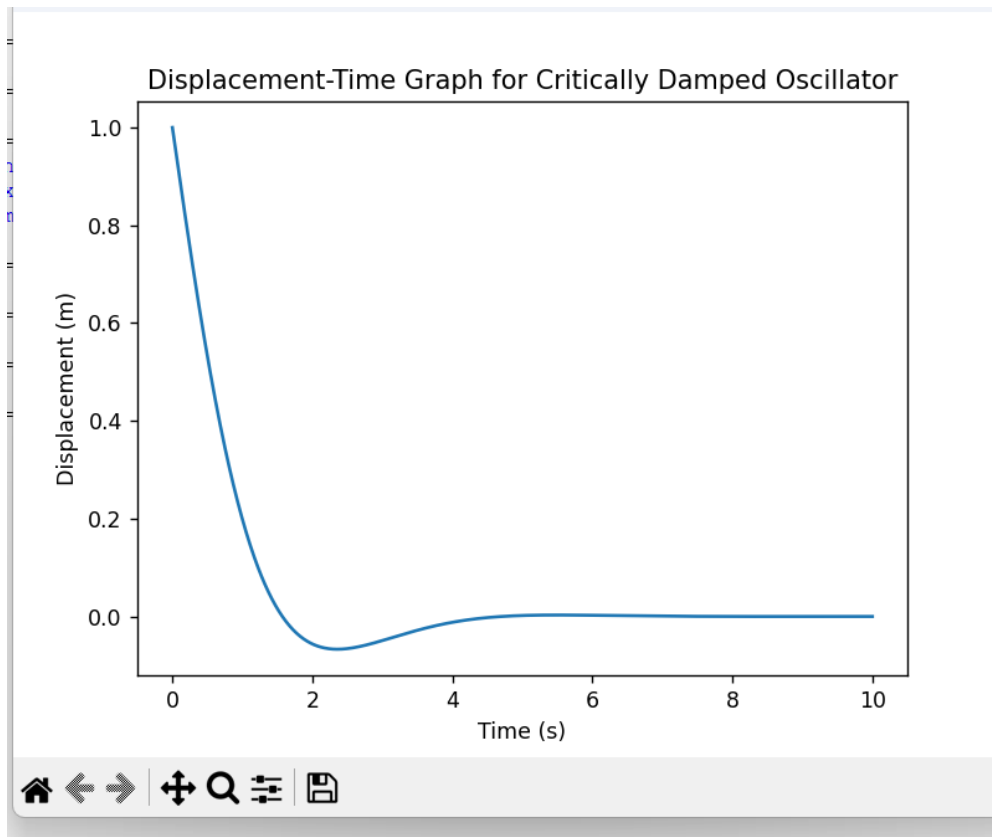
# Plot velocity-time graph for critically damped oscillator
plot_velocity(1, 1, 0, 10, "Velocity-Time Graph for Critically Damped Oscillator")

# Plot displacement-time graph for overdamped oscillator
plot_displacement(1, 2, 0, 10, "Displacement-Time Graph for Overdamped Oscillator")

# Plot velocity-time graph for overdamped oscillator
plot_velocity(1, 2, 0, 10, "Velocity-Time Graph for Overdamped Oscillator")
```

OUTPUT:





PROGRAM: To generate array of N random numbers drawn from a given distribution (uniform , binomial , poisson and gaussian) and plot them using matplotlib for increasing N to verify the distribution.

Normal (Gaussian) Distribution

It fits the probability distribution of many events, eg. IQ Scores, Heartbeat etc.

Use the `random.normal()` method to get a Normal Data Distribution.

It has three parameters:

loc - (Mean) where the peak of the bell exists.

scale - (Standard Deviation) how flat the graph distribution should be.

size - The shape of the returned array.

PROGRAM:

```
*GARIMASINGHPROGRAMS.py - C:/Users/thedy/AppData/Local/Programs/Python/F
File Edit Format Run Options Window Help
from numpy import random

x = random.normal(loc=50, scale=5, size=1000)

print(x)
```

GARIMASINGHPROGRAMS.py - C:/Users/thedy/AppData/Local/Programs/Python/Python311/GARIMASINGHPROGRAMS.py (3.11.1)

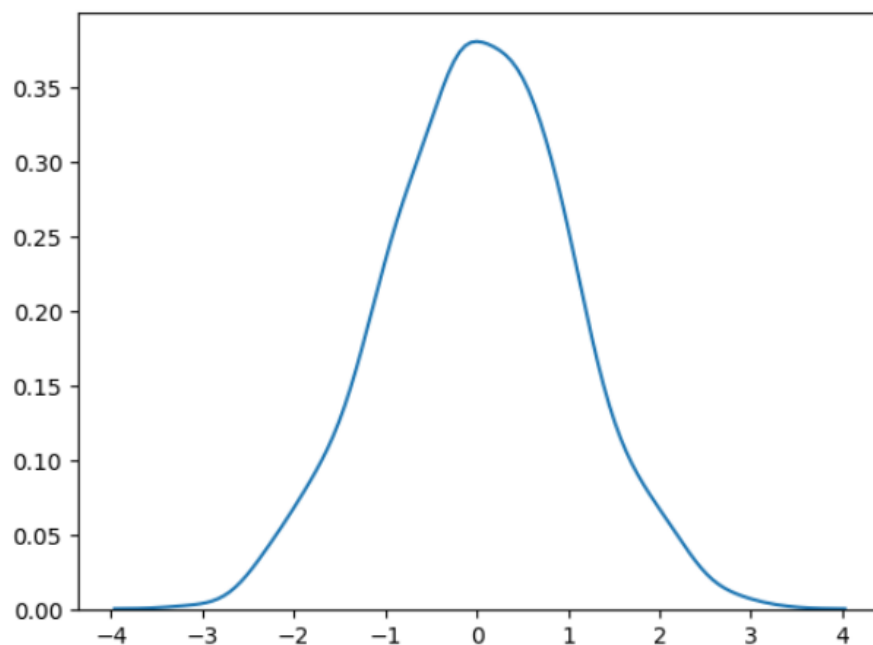
File Edit Format Run Options Window Help

```
from numpy import random
import matplotlib.pyplot as plt
import seaborn as sns
```

```
sns.distplot(random.normal(loc=50, scale=5, size=100), hist=False, label='normal')|
```

OUTPUT:

```
>>> |===== RESTART: C:/Users/thedy/AppData/Local/Python/Python311/Python.exe
[51.63940237 56.99313732 41.74023342 54.39173663 56.41820581 54.50178687
 51.59470461 51.53082878 41.02124036 55.49129215 52.60781589 49.36164461
 45.51198514 50.99834761 45.75762907 47.36550051 49.99654407 47.15001822
 59.9394484 43.85392459 47.0299026 46.34526764 47.99938165 50.63366168
 48.47993345 49.56448912 42.19707005 48.14809566 59.95753318 54.20599788
 52.77613249 51.74253089 55.35051896 51.52603203 52.81330202 57.29170062
 39.33708782 43.34513271 43.9695511 48.48225615 58.46862898 48.17482716
 58.96147521 42.76400238 59.58098733 45.99802506 52.95825204 48.44648273
 42.74622717 50.03624814 53.28275681 50.49755857 47.11597146 46.49586324
 54.05114184 46.6264877 43.58724497 45.67916797 57.37041876 48.97616301
 53.86575943 53.95991787 52.22588772 48.11868618 55.97830142 48.45799322
 40.18540588 52.57070143 48.07997611 56.27710881 42.10291864 48.46677748
 43.27712641 43.9877325 45.36133201 51.35682772 46.61281708 55.68191808
 50.1580903 52.45756425 59.45853516 47.91615296 44.53073455 54.79779697
 40.03627754 44.39360401 51.44360388 46.8135857 47.49377955 51.52707368
 44.73683723 57.29287985 41.12585909 49.64166176 45.89303827 38.25693619
 43.23012103 55.84512409 50.74507569 51.31932392]
>>> |
```



Binomial Distribution

It describes the outcome of binary scenarios, e.g. toss of a coin, it will either be head or tails.


It has three parameters:

n - number of trials.

p - probability of occurrence of each trial (e.g. for toss of a coin 0.5 each).

size - The shape of the returned array.

PROGRAM:


 GARIMASINGHPROGRAMS.py - C:/Users/thedy/AppData/Local/Programs/Python/Python311/GARIMASINGHPROGRAMS.py (3)

File Edit Format Run Options Window Help

```
from numpy import random

x = random.binomial(n=10, p=0.5, size=100)

print(x)
|
```

 *GARIMASINGHPROGRAMS.py - C:/Users/thedy/AppData/Local/Programs/Python/Python311/GARIMASINGHPROGRAMS.py (3)

File Edit Format Run Options Window Help

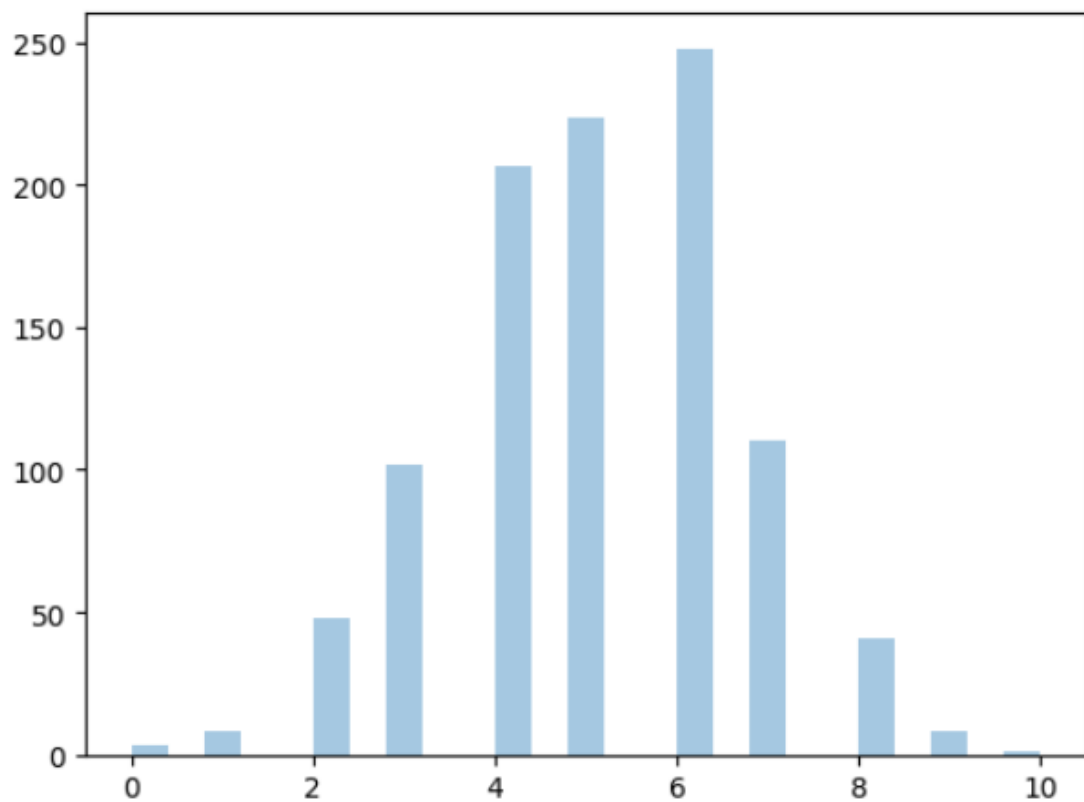
```
from numpy import random
import matplotlib.pyplot as plt
import seaborn as sns

sns.distplot(random.binomial(n=10, p=0.5, size=1000), hist=True, kde=False)

plt.show()
|
```


OUTPUT:

```
>>> ===== RESTART: C:/Users/thedy/AppDat
[4 4 6 5 5 6 5 3 7 5 5 6 3 5 6 4 6 6 4 4 7 5 5 4 3 4 6 4 5 7 4 4 5 6 5 4 7
 5 3 2 4 5 5 5 5 7 7 5 6 3 4 8 4 5 7 9 5 6 2 7 5 4 5 5 6 7 4 5 6 7 4 6 6 5
 4 6 3 4 7 5 3 5 4 7 7 4 6 3 6 5 4 7 4 3 7 4 3 5 5 8]
>>>
```



Poisson Distribution

Poisson Distribution is a *Discrete Distribution*.

It estimates how many times an event can happen in a specified time. e.g. If someone eats twice a day what is probability he will eat thrice?

It has two parameters:

lam - rate or known number of occurrences e.g. 2 for above problem.

size - The shape of the returned array.

PROGRAM:

GARIMASINGHPROGRAMS.py - C:/Users/thedy/AppData/Local/Programs/Python/Python311/GARIMASINGHPROGRAMS.py

File Edit Format Run Options Window Help

```
from numpy import random

x = random.poisson(lam=2, size=100)

print(x)
```

*GARIMASINGHPROGRAMS.py - C:/Users/thedy/AppData/Local/Programs/Python/Python311/GARIMASINGHPROGRAMS.py

File Edit Format Run Options Window Help

```
from numpy import random
import matplotlib.pyplot as plt
import seaborn as sns

sns.distplot(random.poisson(lam=2, size=1000), kde=False)

plt.show()
```

OUTPUT:

```
>>> ===== RESTART: C:/Users/thedy/AppData/Lo
[2 3 2 2 1 1 2 2 0 1 1 2 1 1 4 0 4 3 2 2 2 1 1 0 0 1 2 1 2 1 2 1 1 1 5 4 1
 4 1 0 1 2 1 2 3 1 3 0 2 4 0 5 3 1 1 1 2 1 0 2 2 2 3 2 1 3 5 4 2 3 2 2 0 2
 2 6 3 2 0 1 1 1 0 1 2 4 1 2 2 0 1 4 4 0 2 1 2 2 4 5]
>>>
```

