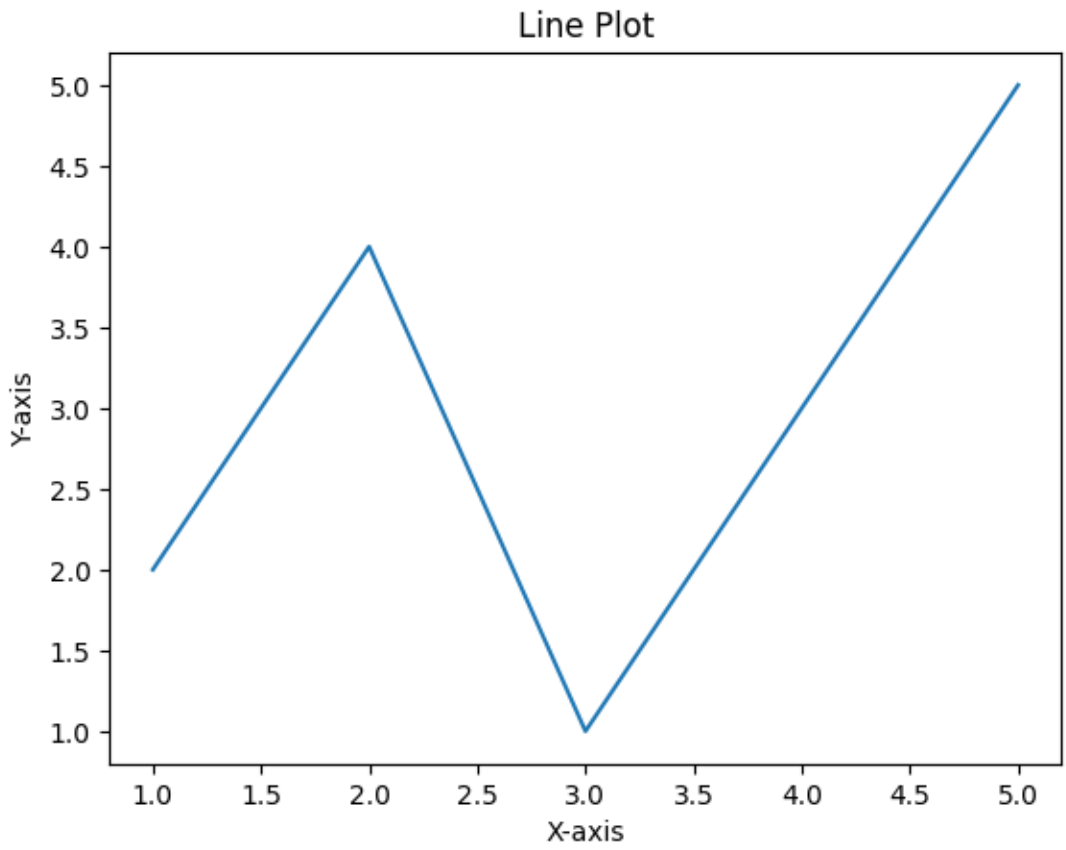
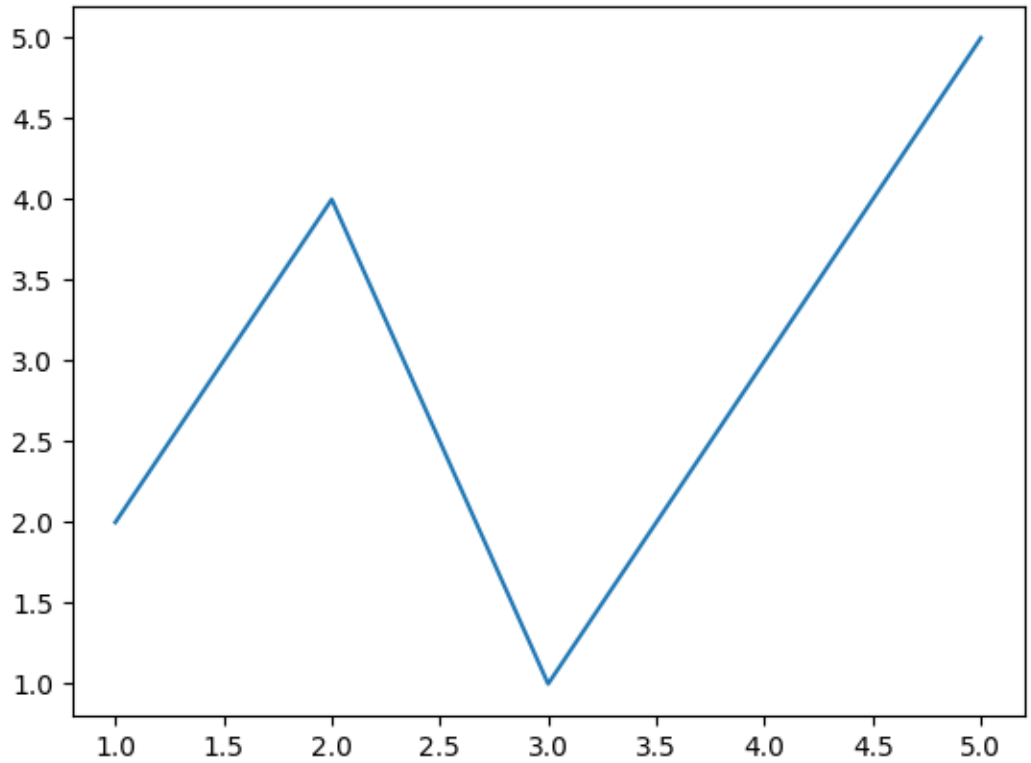
Data Visualization in python
matplotlib
line_chart

```
import matplotlib.pyplot as plt

import seaborn as sns


# Sample data use for :Trend dikhane ke liye, Comparisons ke liye,Continuous data ke liye

x = [1, 2, 3, 4, 5]

y = [2, 4, 1, 3, 5]


# Create a line plot

plt.plot(x, y)

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.title("Line Plot")

linestyle="solid"

plt.show()


# Or using Seaborn

sns.lineplot(x=x, y=y)

plt.show()
```


Line Plot
Y-axis
X-axis
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

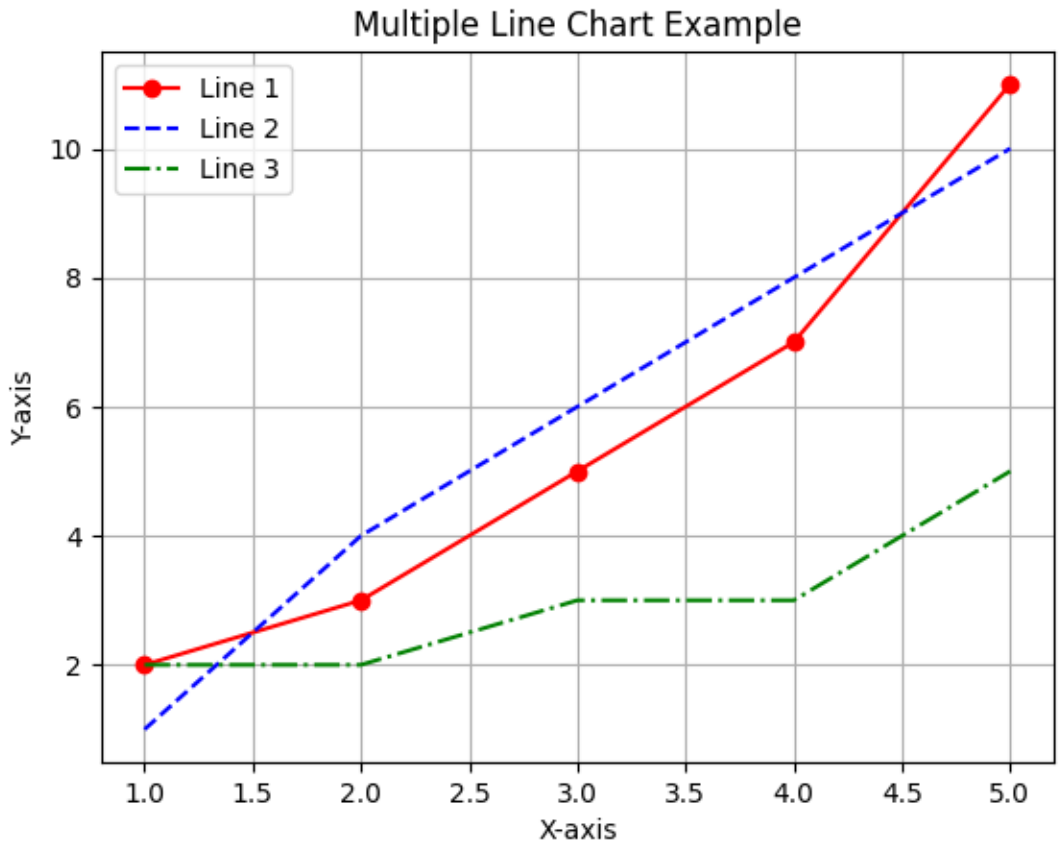

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

multiple line chart

```
import matplotlib.pyplot as plt


# Sample data
x = [1, 2, 3, 4, 5]
y1 = [2, 3, 5, 7, 11]
y2 = [1, 4, 6, 8, 10]
y3 = [2, 2, 3, 3, 5]

# Plotting multiple lines
plt.plot(x, y1, label='Line 1', color='red', marker='o')
plt.plot(x, y2, label='Line 2', color='blue', linestyle='--')
plt.plot(x, y3, label='Line 3', color='green', linestyle='-.')

# Adding title and labels
plt.title('Multiple Line Chart Example')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Show legend
plt.legend()

# Show grid
plt.grid(True)
```

 # Display plot

plt.show()



Contour laine

import matplotlib.pyplot **as** plt

import numpy **as** np

 # Correct function name is linspace, not linespace

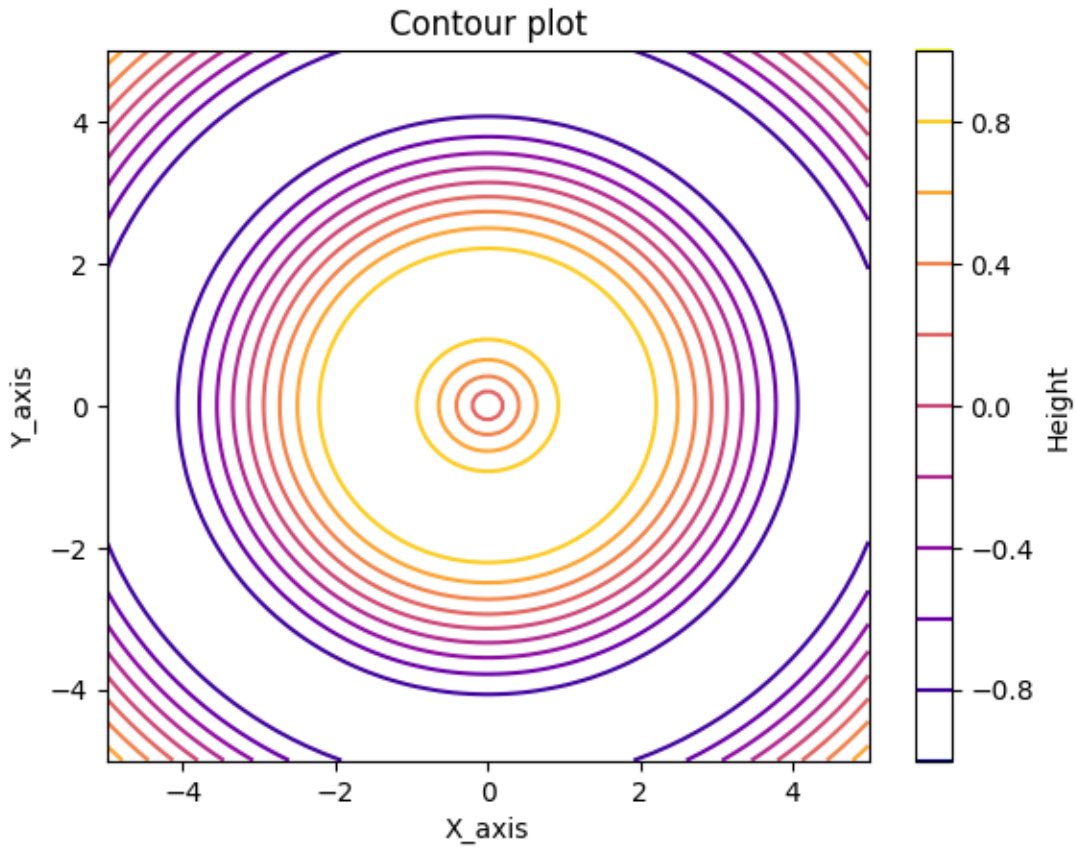
x = np.linspace(-5, 5, 100)

y = np.linspace(-5, 5, 100)

 # Correct function name is meshgrid, not npmeshgrid

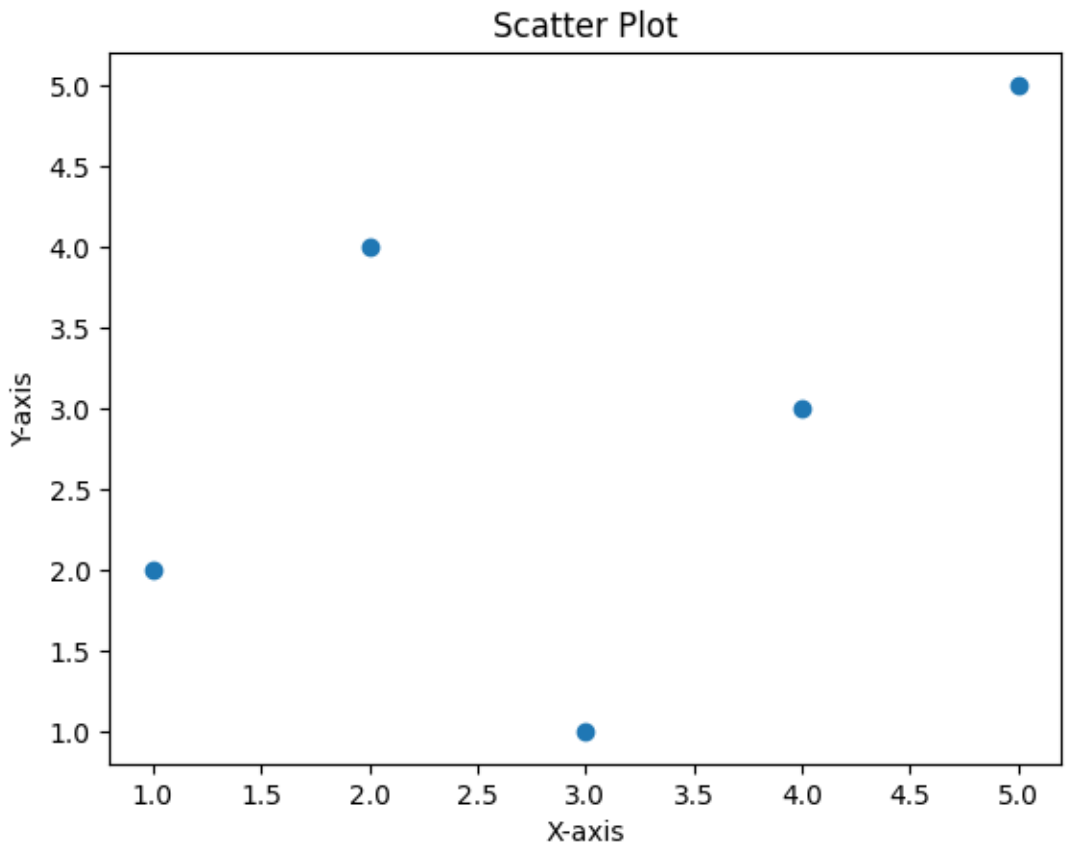
X, Y = np.meshgrid(x, y)

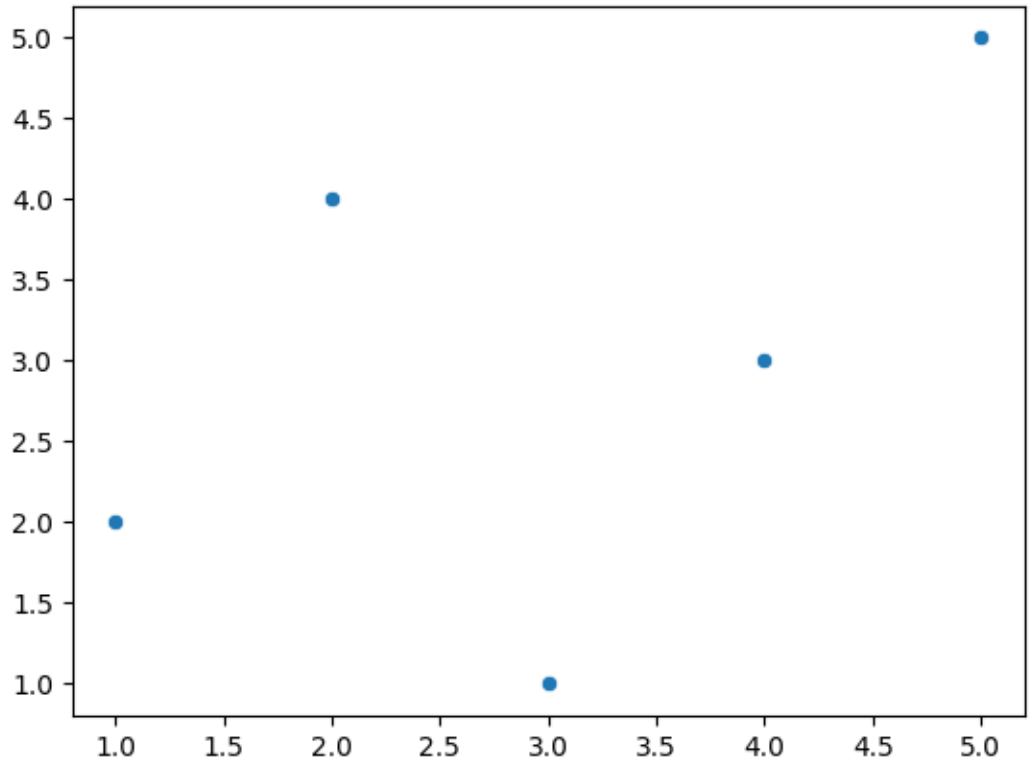
```
Z = np.sin(np.sqrt(X**2 + Y**2))


# Typo in 'camp', correct keyword is 'cmap' for color map
plt.contour(X, Y, Z, levels=10, cmap="plasma")


plt.colorbar(label="Height")
plt.title("Contour plot")
plt.xlabel("X_axis")
plt.ylabel("Y_axis")
plt.show()
```



SCATTAR PLOT

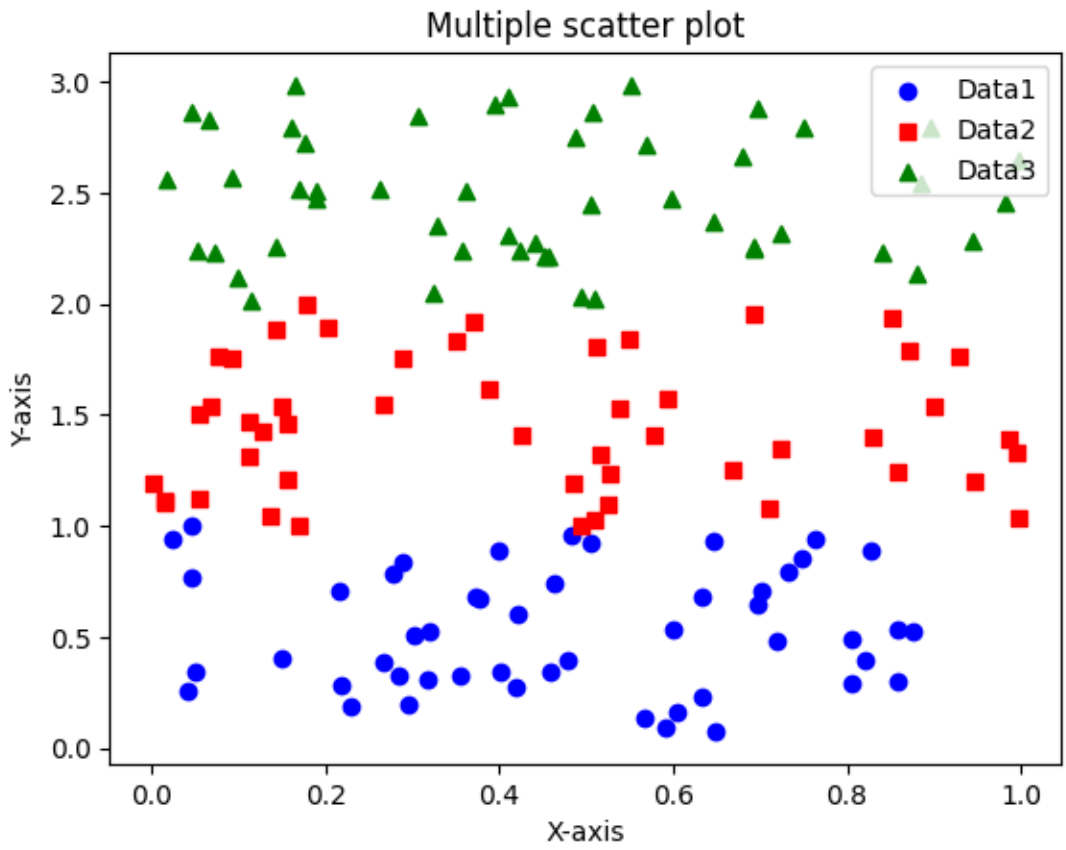
```
import matplotlib.pyplot as plt
import seaborn as sns

# Sample data use for: Correlation check karne ke liye,Outliers (ajeeb data points) identify karne ke liye
#Patterns ya trends dhoondhne ke liye,Relationship dekhne ke liye
x = [1, 2, 3, 4, 5]
y = [2, 4, 1, 3, 5]

# Create a scatter plot
plt.scatter(x, y)
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatter Plot")
plt.show()

# Or using Seaborn
sns.scatterplot(x=x, y=y)
plt.show()
```


Scatter Plot
Y-axis
X-axis
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

Multiple Scatter plot

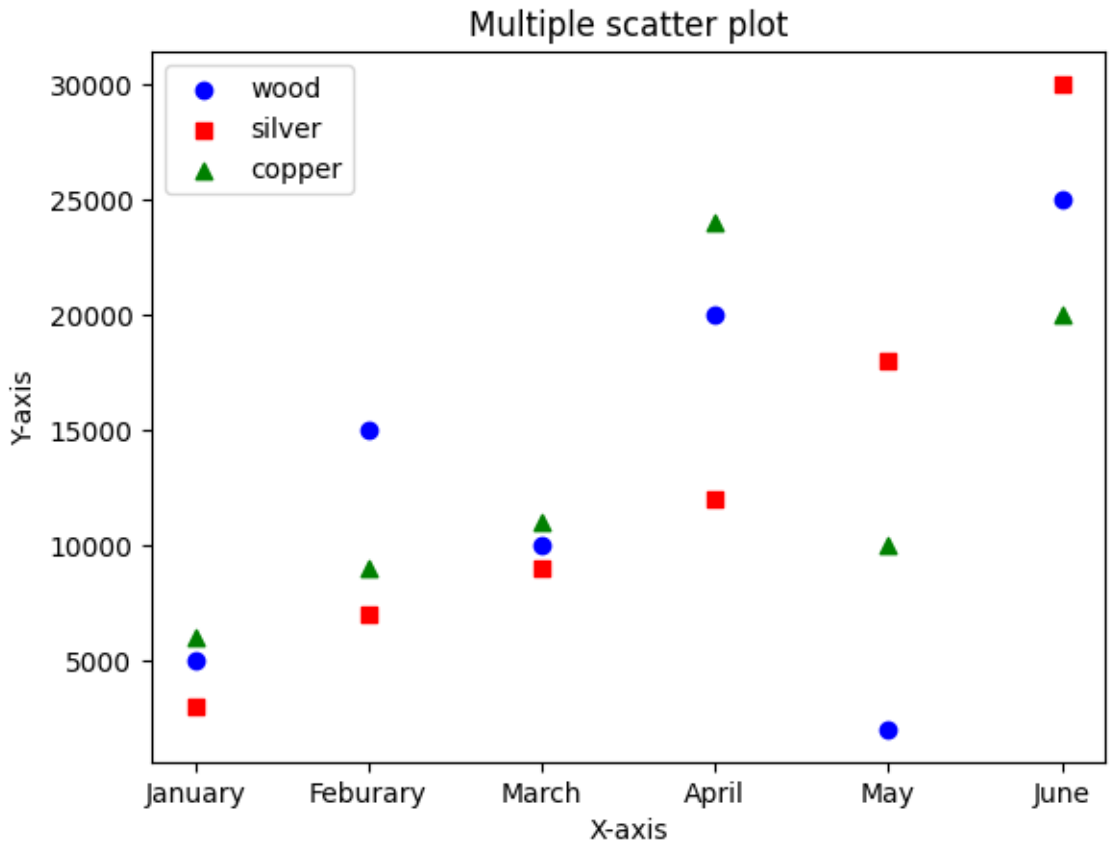
```
import matplotlib.pyplot as plt

import numpy as np

x1=np.random.rand(50)

y1=np.random.rand(50)

x2=np.random.rand(50)

y2=np.random.rand(50)+1  #shiffted for visibility

x3=np.random.rand(50)

y3=np.random.rand(50)+2  #shiffted for visibility

plt.scatter(x1,y1,label='Data1',marker='o',color='blue')

plt.scatter(x2,y2,label='Data2',marker='s',color='red')

plt.scatter(x3,y3,label='Data3',marker='^',color='green')

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.title("Multiple scatter plot")

plt.legend()

plt.show()
```




```
import matplotlib.pyplot as plt

import numpy as np

Months = ['January', 'Feburary', 'March', 'April','May','June']

Sale1 = [5000, 15000, 10000, 20000,2000,25000]

Sale2 = [3000, 7000, 9000, 12000,18000,30000]

Sale3 = [6000, 9000, 11000, 24000,10000,20000]

plt.scatter(Months,Sale1,label='wood',marker='o',color='blue')

plt.scatter(Months,Sale2,label='silver',marker='s',color='red')

plt.scatter(Months,Sale3,label='copper',marker='^',color='green')

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.title("Multiple scatter plot")
```

plt.legend()

plt.show()



Bar Chart

import matplotlib.pyplot **as** plt

import seaborn **as** sns

 # Sample data use for: Frequency dikhani ho,Compare karna ho,Categorical data ke liye

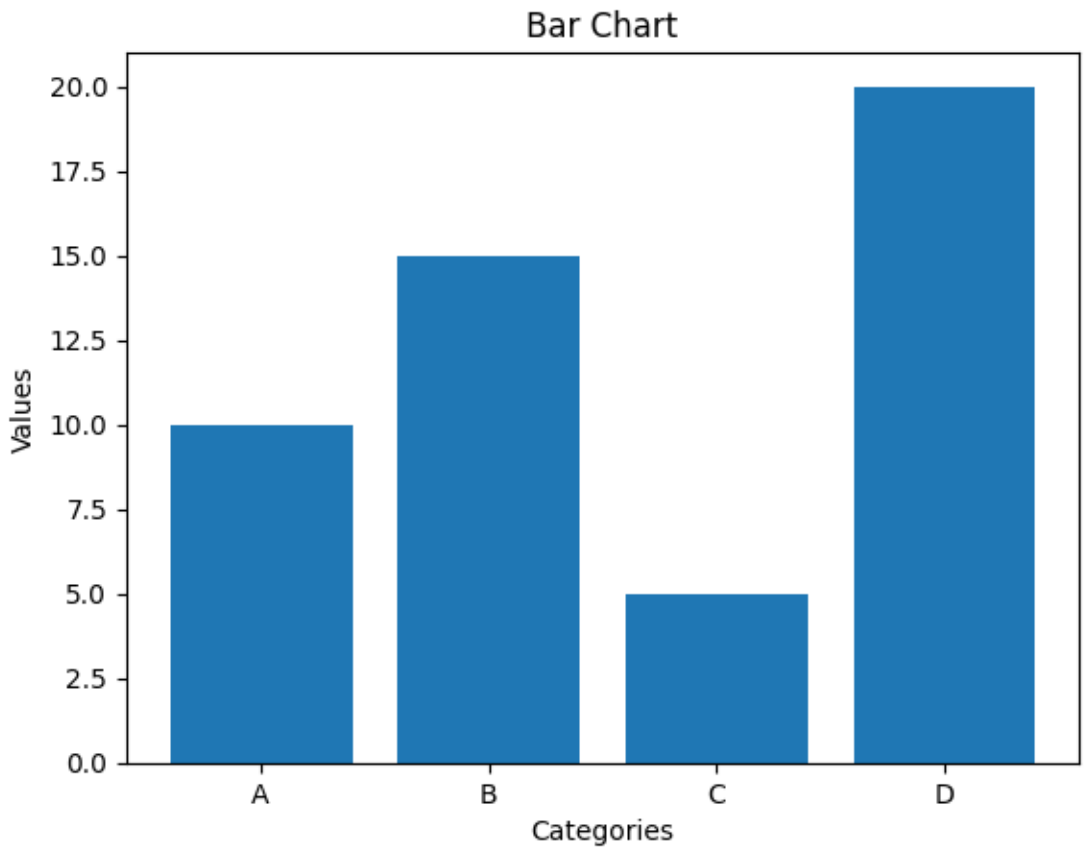
categories = ['A', 'B', 'C', 'D']

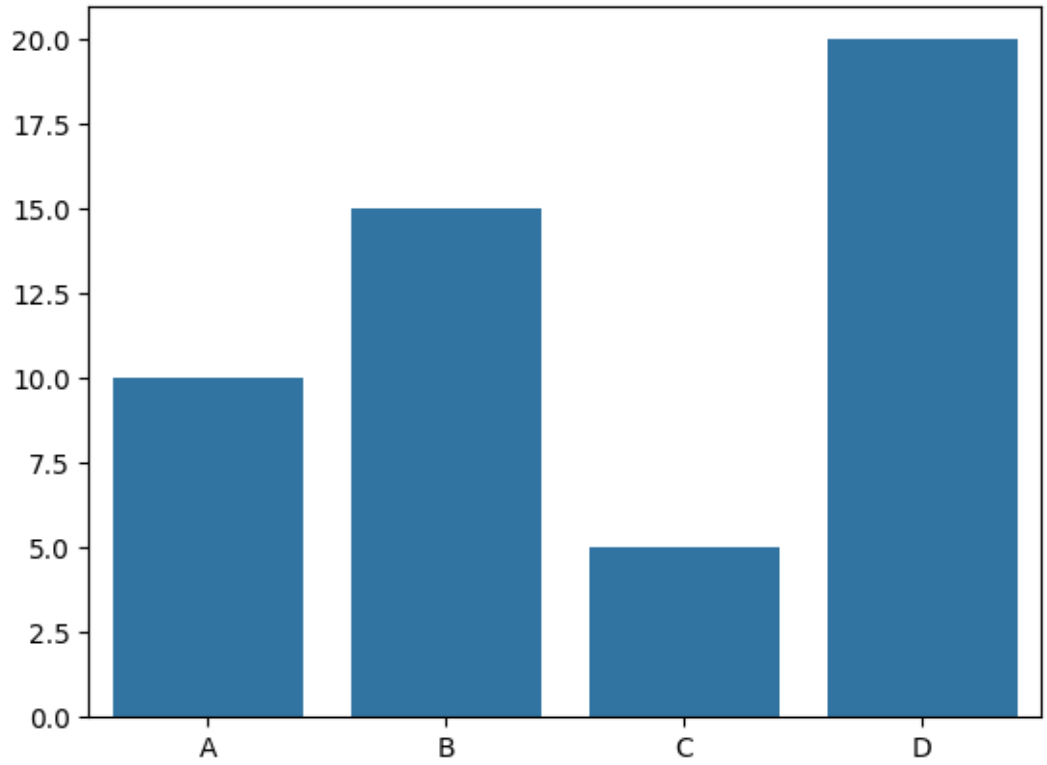
values = [10, 15, 5, 20]

 # Create a bar chart

plt.bar(categories, values)

```
plt.xlabel("Categories")

plt.ylabel("Values")

plt.title("Bar Chart")

plt.show()



# Or using Seaborn

sns.barplot(x=categories, y=values)

plt.show()
```

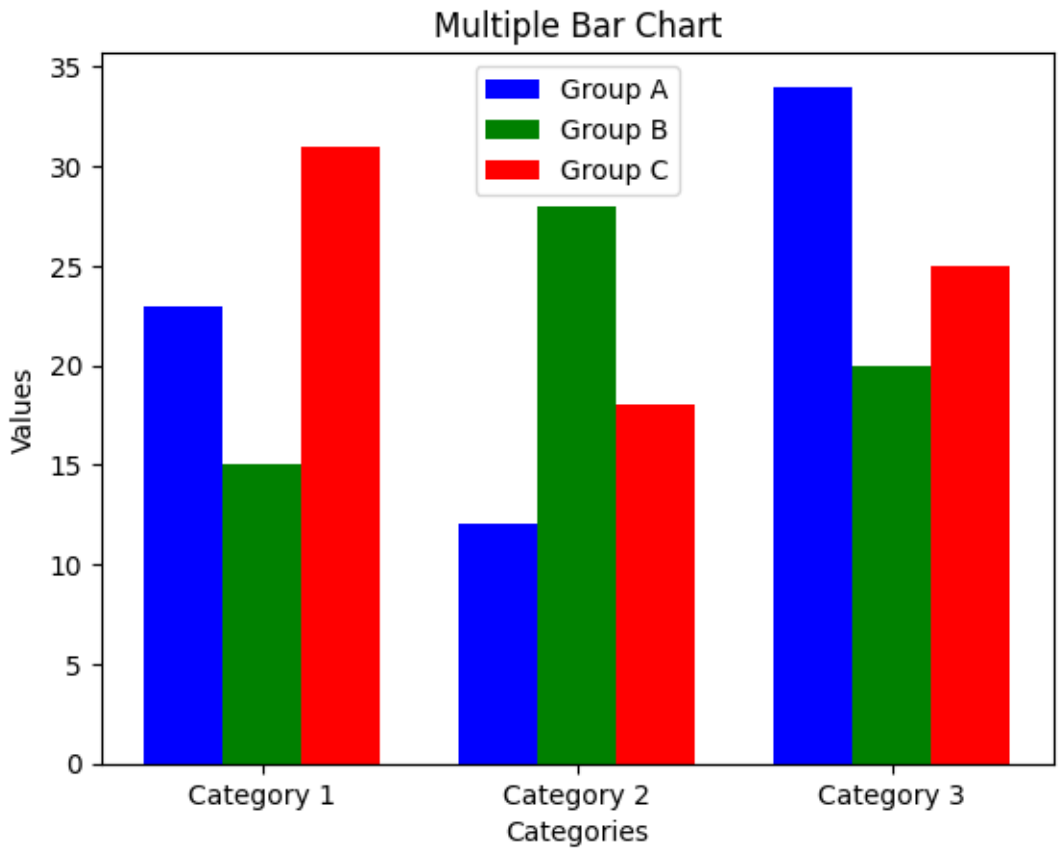




Multiple Bar Chart

```
import matplotlib.pyplot as plt

import numpy as np


# Sample data

groups = ['Group A', 'Group B', 'Group C']

categories = ['Category 1', 'Category 2', 'Category 3']

values_group_a = [23, 12, 34]

values_group_b = [15, 28, 20]

values_group_c = [31, 18, 25]


# Set the width of each bar

bar_width = 0.25
```

```
# Calculate the position of each bar on the x-axis
x_pos_group_a = np.arange(len(categories))
x_pos_group_b = [x + bar_width for x in x_pos_group_a]
x_pos_group_c = [x + bar_width for x in x_pos_group_b]

# Create the bar chart
plt.bar(x_pos_group_a, values_group_a, color='blue', width=bar_width, label='Group A')
plt.bar(x_pos_group_b, values_group_b, color='green', width=bar_width, label='Group B')
plt.bar(x_pos_group_c, values_group_c, color='red', width=bar_width, label='Group C')

# Set the x-axis tick positions and labels
plt.xticks([x + bar_width for x in range(len(categories))], categories)

# Add labels, title, and legend
plt.xlabel("Categories")
plt.ylabel("Values")
plt.title("Multiple Bar Chart")
plt.legend()

# Show the plot
plt.show()
```



Component Bar Chart

```
import matplotlib.pyplot as plt

import numpy as np

#sample data

City=['Faisalabad','isb','toba tek singh']

wheat=[23,12,34]

cotton=[15,28,20]

maize=[31,18,25]

#create a component bar chart

plt.bar(City,wheat,color='blue',label='wheat')

plt.bar(City,cotton,bottom=wheat,color='green',label='cotton')

plt.bar(City,maize,bottom=np.array(wheat)+np.array(cotton),color='red',label='maize')
```

#add label ,title and legend

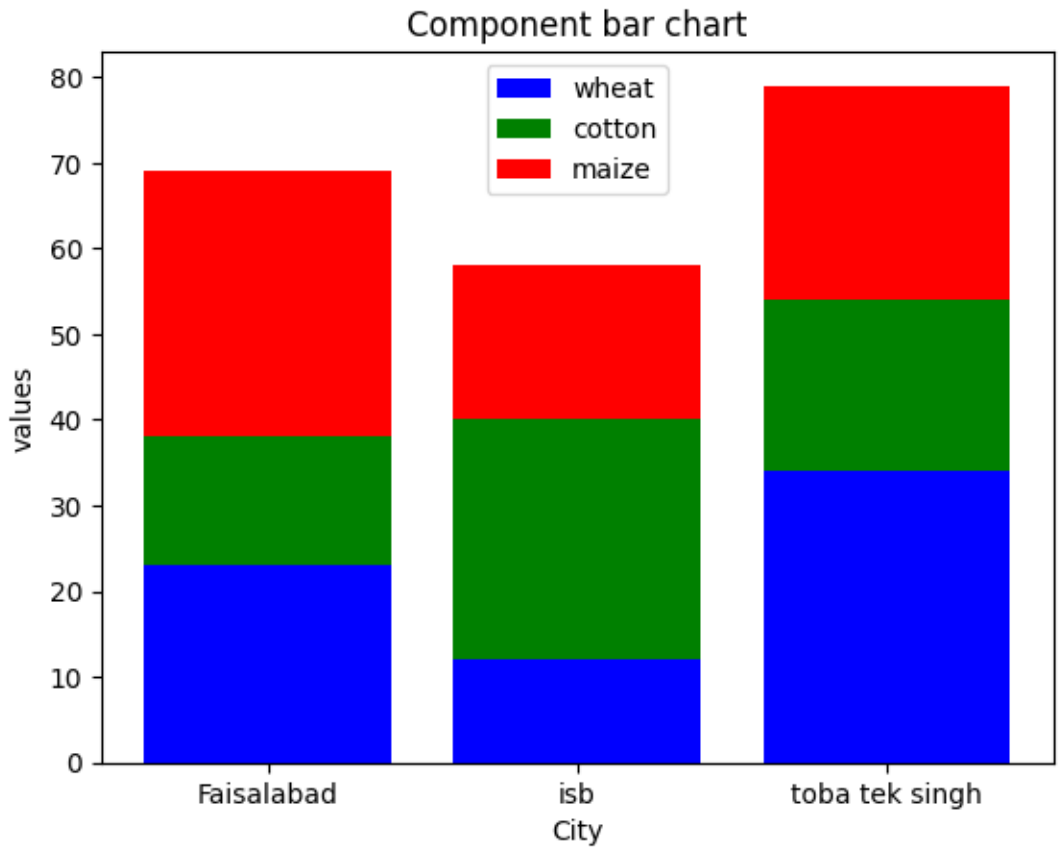
plt.xlabel("City")

plt.ylabel("values")

plt.title("Component bar chart")

plt.legend()

plt.show()



Pie Chart

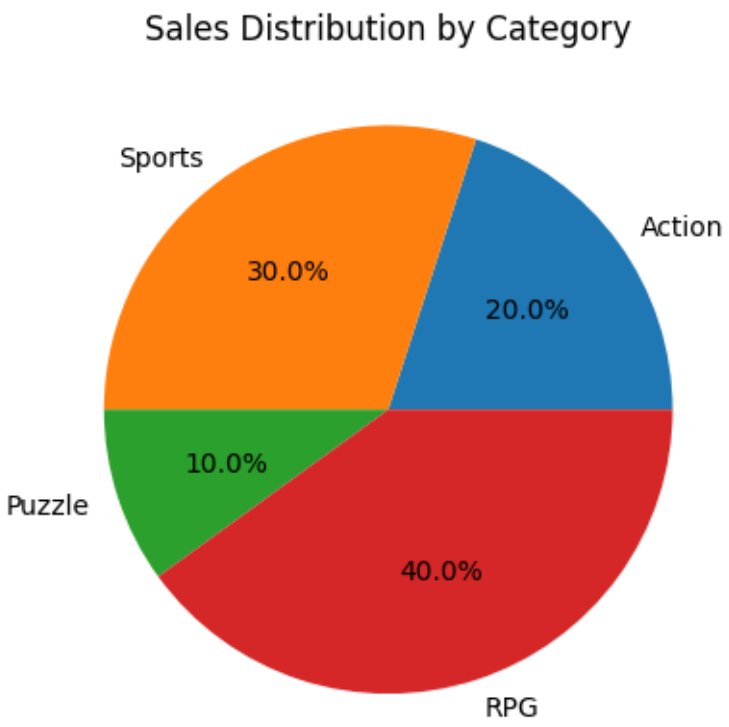
import pandas **as** pd

import matplotlib.pyplot **as** plt

 # Create your own data

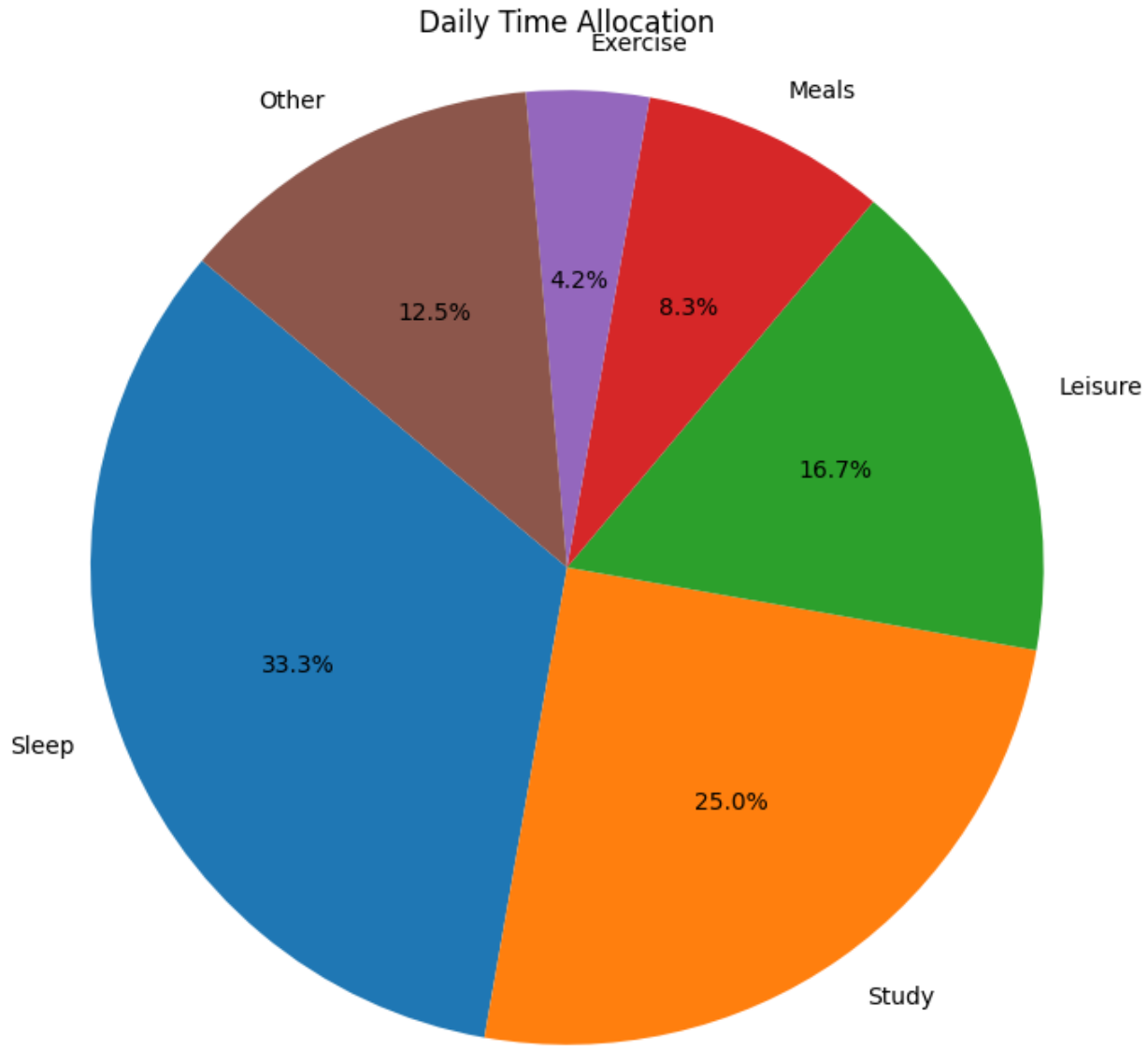
data **=** {

```
    'Category': ['Action', 'Sports', 'Puzzle', 'RPG'],
    'Sales': [100, 150, 50, 200]
}

file = pd.DataFrame(data)

# Plot
plt.pie(file['Sales'], labels=file['Category'], autopct='%1.1f%%')
plt.title('Sales Distribution by Category')
plt.show()
```



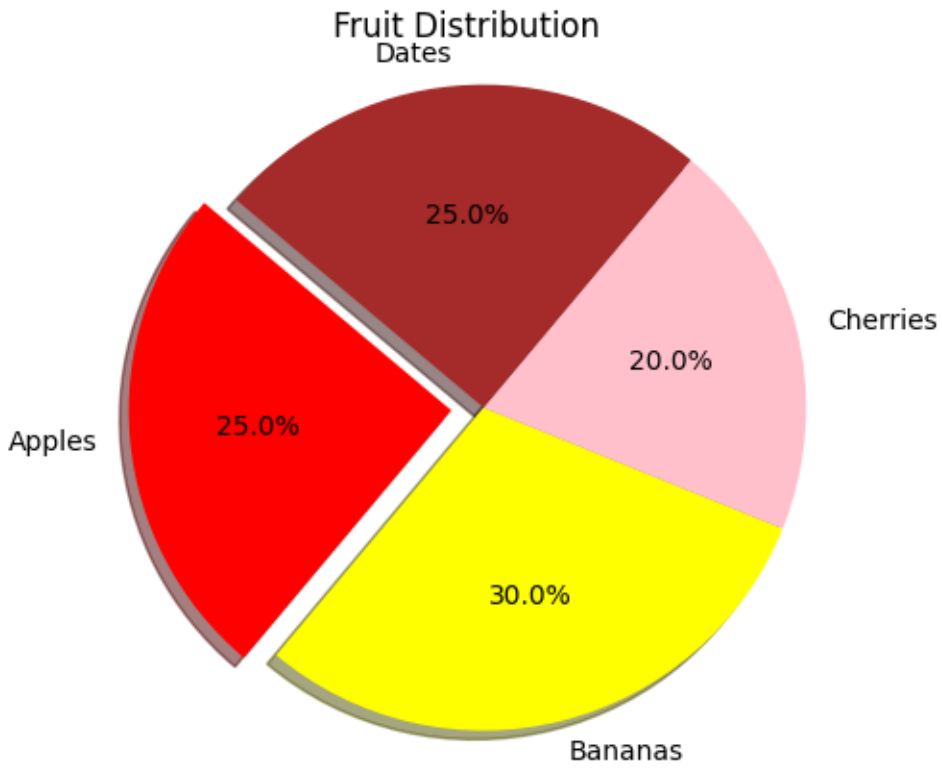

```
import matplotlib.pyplot as plt

# Data
activities = ['Sleep', 'Study', 'Leisure', 'Meals', 'Exercise', 'Other']
```

```
hours = [8, 6, 4, 2, 1, 3]


# Create pie chart
plt.figure(figsize=(8, 8))
plt.pie(hours, labels=activities, autopct='%1.1f%%', startangle=140)
plt.title('Daily Time Allocation')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

# Show plot
plt.show()
```




```
import matplotlib.pyplot as plt


# Data to plot
labels = ['Apples', 'Bananas', 'Cherries', 'Dates']
sizes = [25, 30, 20, 25]  # percentage values
colors = ['red', 'yellow', 'pink', 'brown']
explode = (0.1, 0, 0, 0)  # "explode" the 1st slice
```

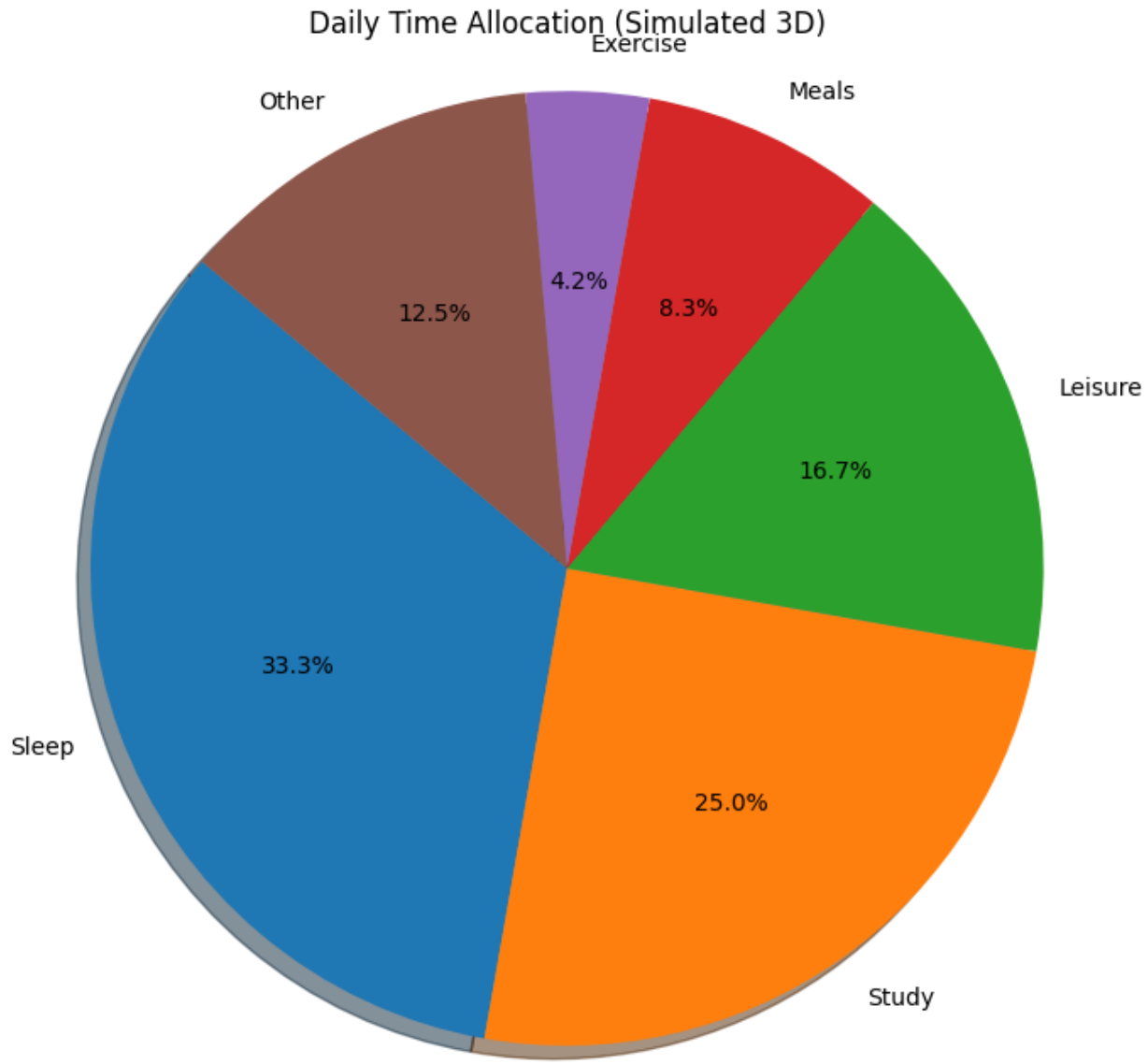
```
# Create pie chart
plt.pie(sizes, labels=labels, colors=colors, explode=explode,
        autopct='%1.1f%%', shadow=True, startangle=140)

plt.title('Fruit Distribution')
plt.axis('equal')  # Equal aspect ratio ensures the pie chart is circular.
plt.show()
```



3D pie

```
import matplotlib.pyplot as plt

# Data
activities = ['Sleep', 'Study', 'Leisure', 'Meals', 'Exercise', 'Other']
hours = [8, 6, 4, 2, 1, 3]
```

```
# Create pie chart with shadow to simulate 3D
plt.figure(figsize=(8, 8))
plt.pie(hours, labels=activities, autopct='%1.1f%%', startangle=140, shadow=True)
plt.title('Daily Time Allocation (Simulated 3D)')
plt.axis('equal')

plt.show()
```



```
import plotly.express as px

# Data
activities = ['Sleep', 'Study', 'Leisure', 'Meals', 'Exercise', 'Other']
hours = [8, 6, 4, 2, 1, 3]

# Create pie chart
fig = px.pie(names=activities, values=hours, title='Daily Time Allocation (3D Style)',
             hole=0.2)  # hole=0.2 gives it a 3D-like appearance
fig.update_traces(textinfo='percent+label', pull=[0.05]*6)

fig.show()
```

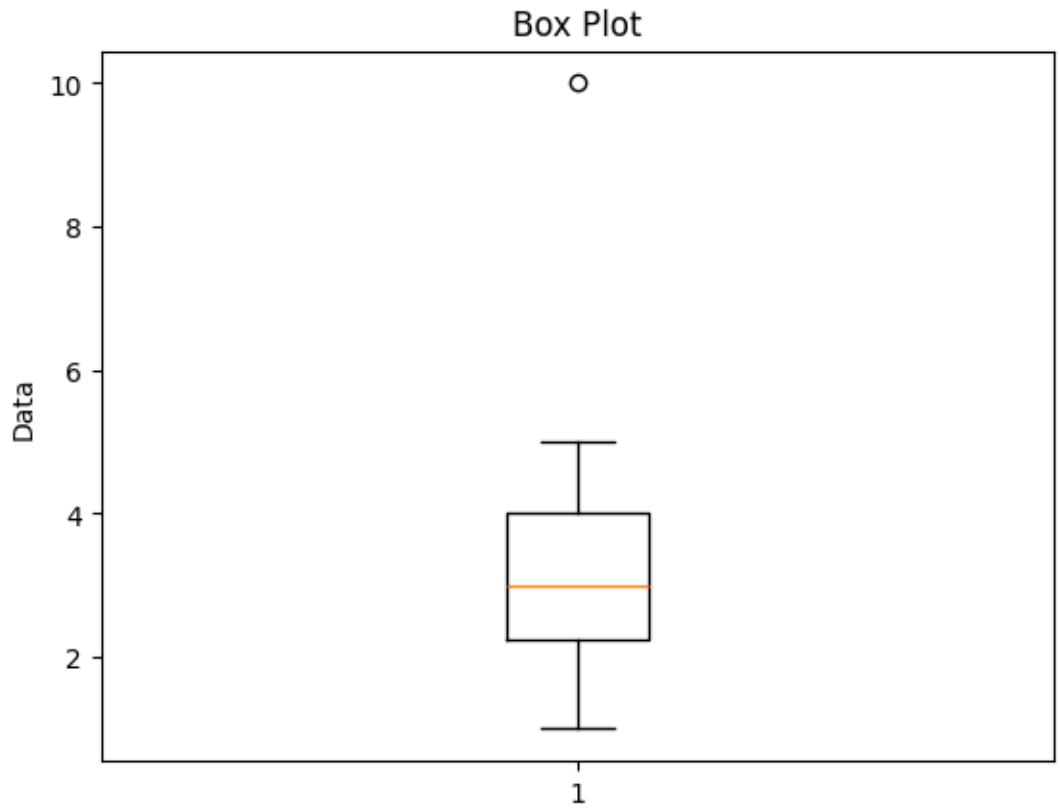
Box plot

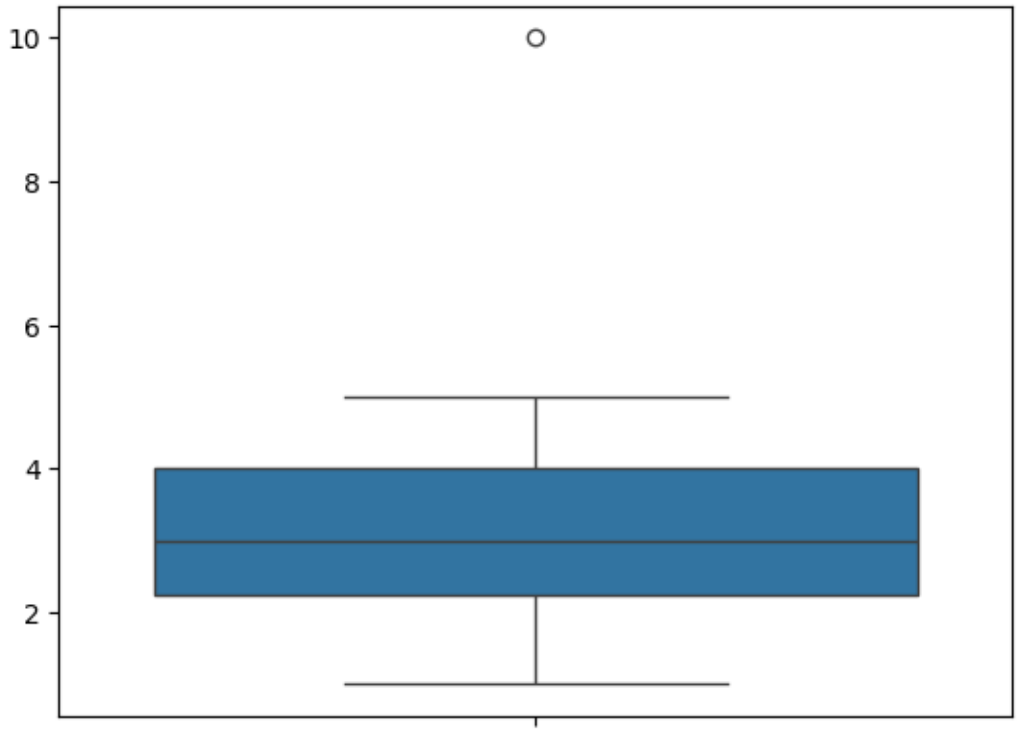
```
import matplotlib.pyplot as plt
import seaborn as sns

# Sample data
data = [1, 2, 2, 3, 3, 3, 4, 4, 5, 10]  # 10 is an outlier

# Create a box plot
plt.boxplot(data)
plt.ylabel("Data")
plt.title("Box Plot")
plt.show()
```

 # Or using Seaborn

sns.boxplot(y=data) *# Use 'y' for vertical orientation*

plt.show()

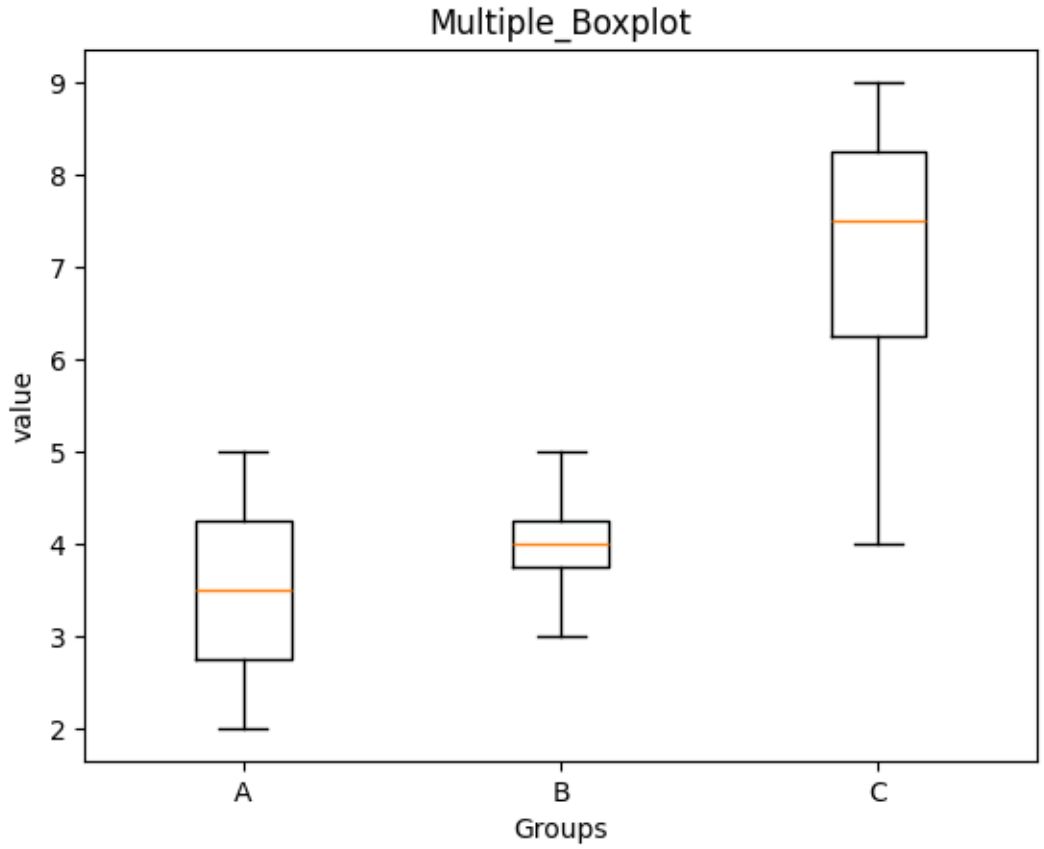




multiple box plot

```
import matplotlib.pyplot as plt

import seaborn as sns

data_a = [2,3,4,5]

data_b = [3,4,4,5]

data_c = [4,7,8,9]

labels = ['A','B','C']

data = [data_a, data_b, data_c]


plt.boxplot(data, tick_labels=labels)# delta che da kam tick lafzl dee da sirf jptr da para de pcm kii simple labels=labels dee

plt.title('Multiple_Boxplot')

plt.xlabel('Groups')

plt.ylabel('value')
```

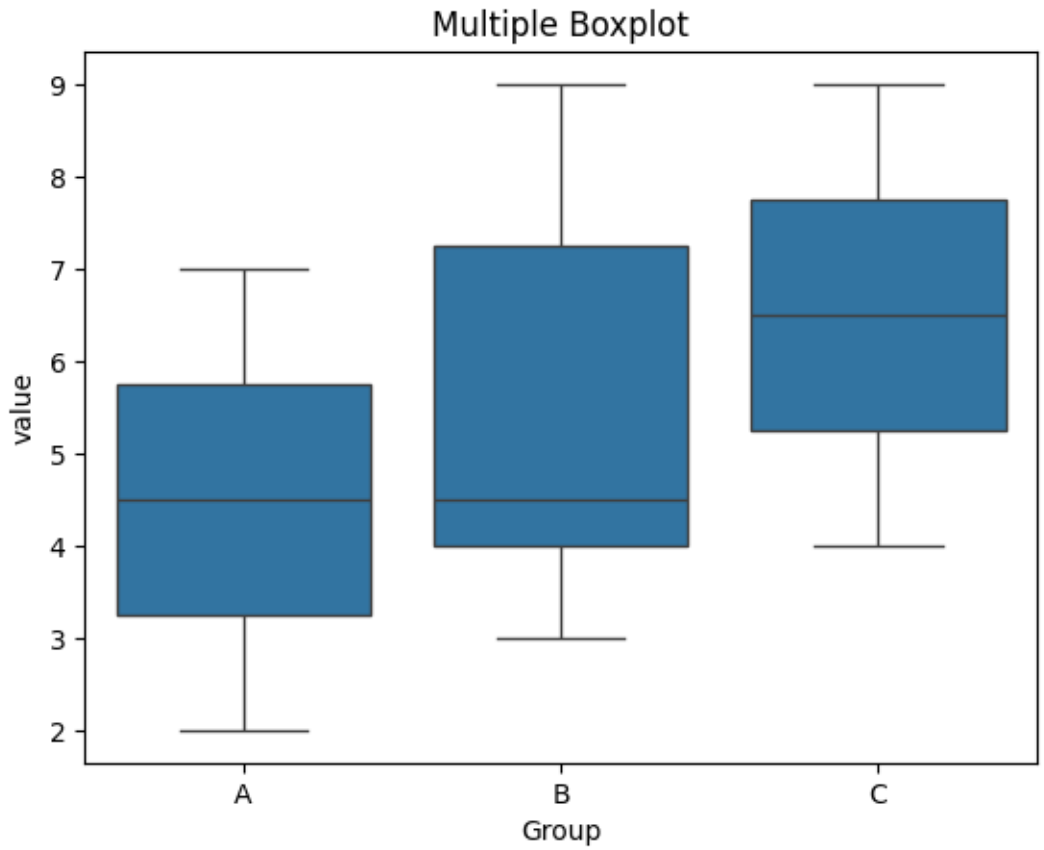
plt.show()



Multiple Boxplot in Seaborn

```
import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd


data_a = [2, 3, 4, 5, 6, 7]

data_b = [3, 4, 4, 5, 8, 9]

data_c = [4, 7, 8, 9, 5, 6]


# Create a DataFrame

data = pd.DataFrame({
```

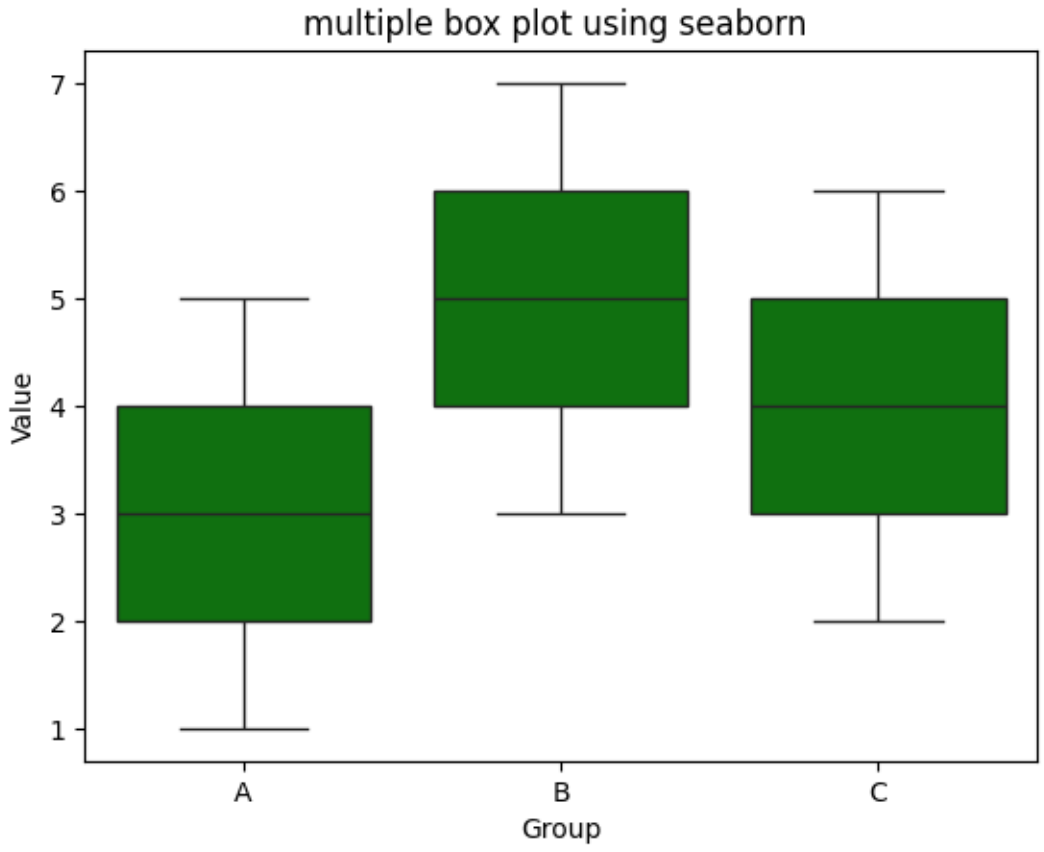
```
    'Group': ['A'] * len(data_a) + ['B'] * len(data_b) + ['C'] * len(data_c),
    'value': data_a + data_b + data_c
})

# Create the boxplot
sns.boxplot(x='Group', y='value', data=data)
plt.title('Multiple Boxplot')
plt.show()
```



multiple boxplot use seaborn

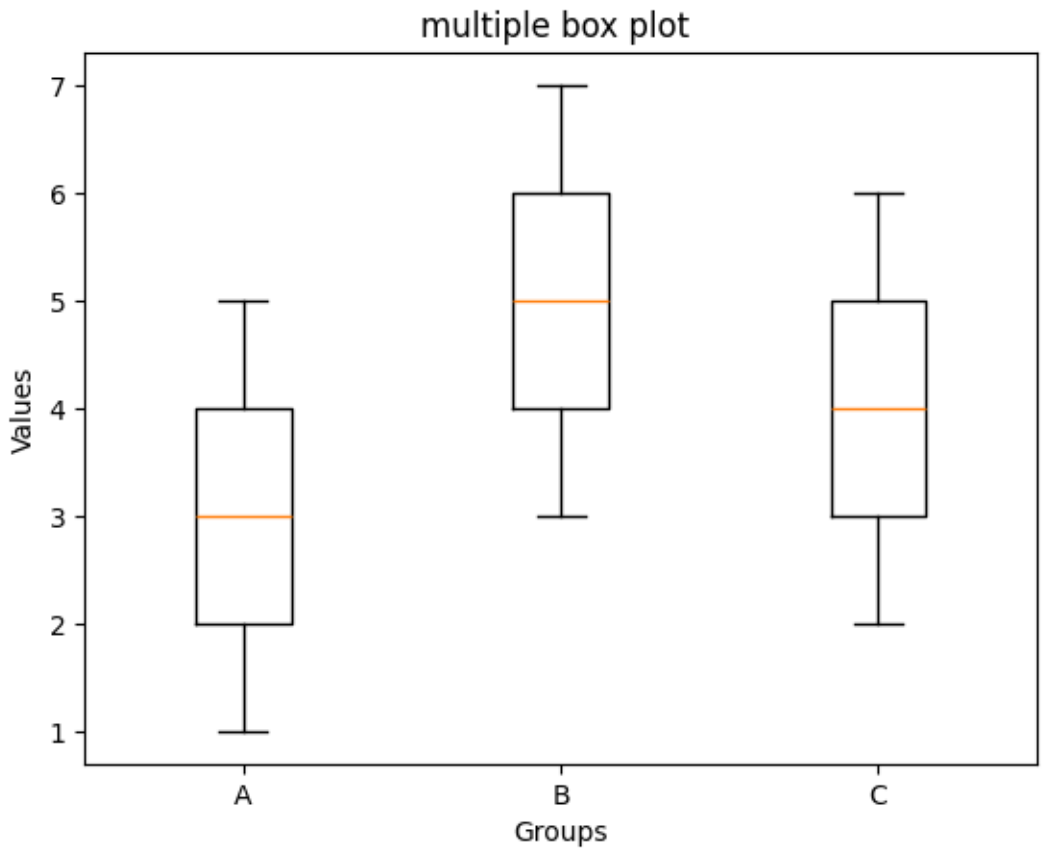
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

```
#sample data
data_a=[1,2,3,4,5];data_b=[3,4,5,6,7]
data_c=[2,3,4,5,6]
#Create a pandas data frame
data=pd.DataFrame({
    'Group':['A']*len(data_a)+['B']*len(data_b)+['C']*len(data_c),'Value':data_a+data_b+data_c
})
#create the box plot
sns.boxplot(x='Group',y='Value',data=data,color='green')
plt.title("multiple box plot using seaborn ")
plt.show()
```



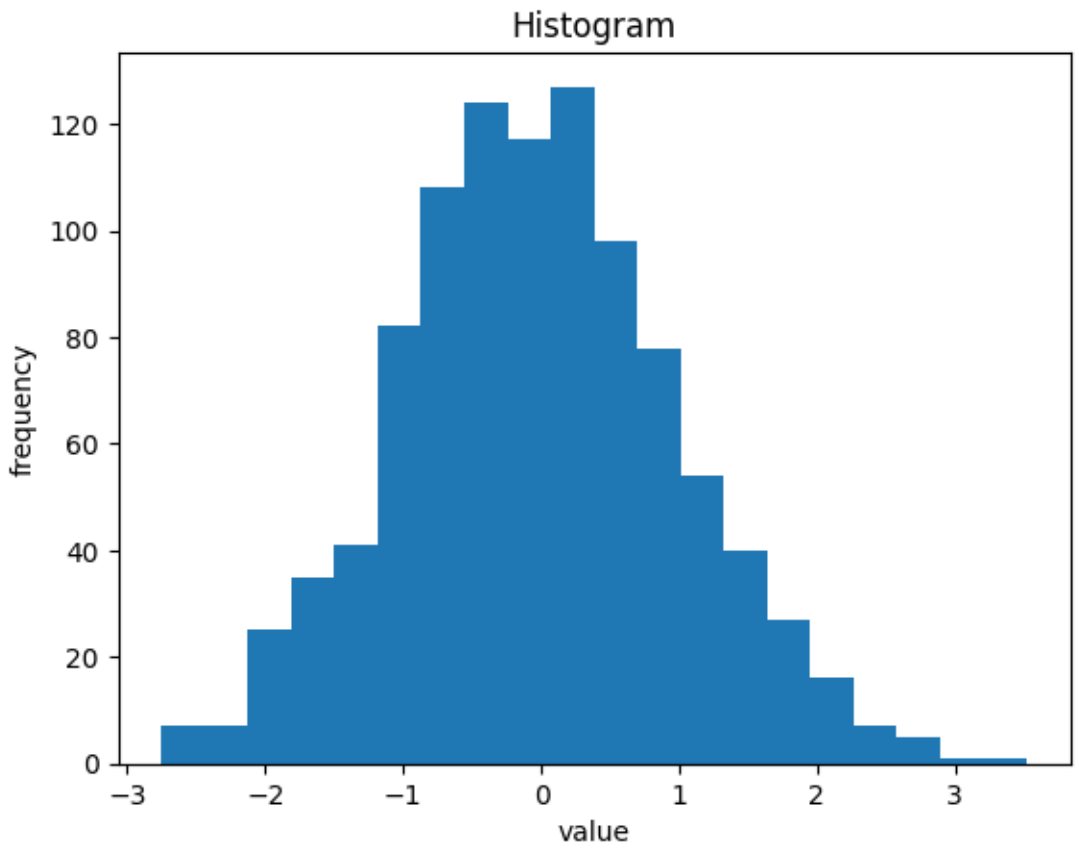

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
#sample data
data_a=[1,2,3,4,5];data_b=[3,4,5,6,7]
data_c=[2,3,4,5,6]
#combine data into a list of lists
data=[data_a,data_b,data_c]
#group label
labels=['A','B','C']
#create the box plot
plt.boxplot(data,labels=labels)
#ass titles and labels
plt.title("multiple box plot ")
plt.xlabel('Groups')
plt.ylabel('Values')
plt.show()
```



Simple_Histogram

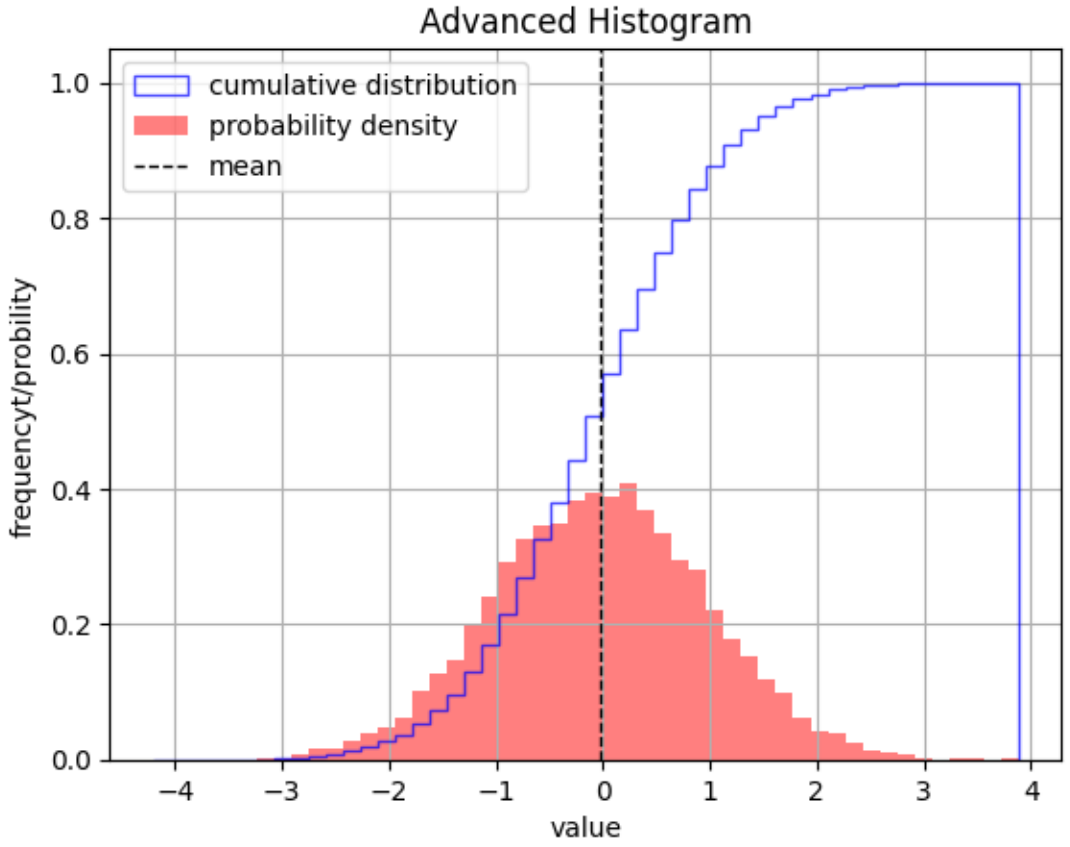
```
import matplotlib.pyplot as plt

import numpy as np


data=np.random.randn(1000)

plt.hist(data,bins=20)

plt.xlabel('value')

plt.ylabel('frequency')

plt.title('Histogram')

plt.show()#display the histogram
```



Advance_Histogram

```
import matplotlib.pyplot as plt

import numpy as np


data=np.random.randn(10000)

plt.hist(data,bins=50,density=True,cumulative=True,

        histtype='step',color='blue',alpha=0.7,

        label='cumulative distribution')

plt.hist(data,bins=50,density=True,

        histtype='bar',color='red',alpha=0.5,

        label='probability density')

#add a vertical line at the mean
```

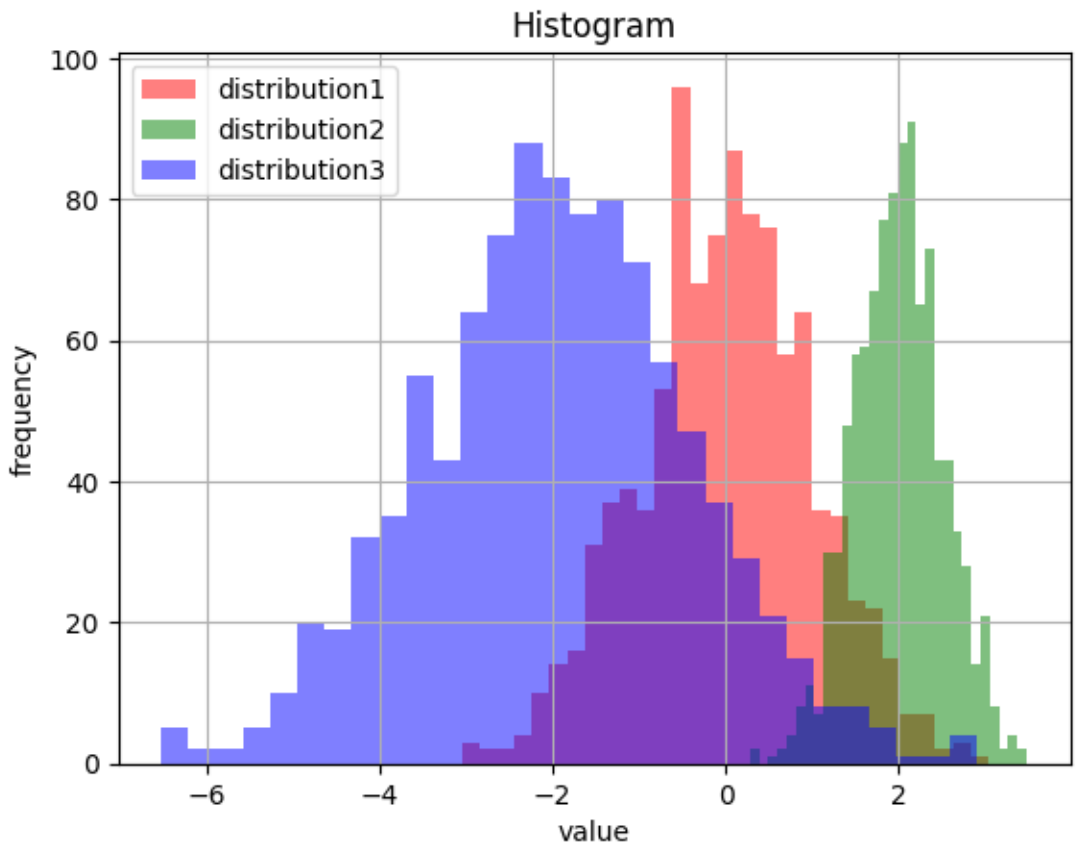
```
plt.axvline(data.mean(),color='k',linestyle='dashed',
        linewidth=1,label='mean')
plt.xlabel('value')
plt.ylabel('frequencyt/probility')
plt.title('Advanced Histogram')
plt.legend()
plt.grid(True)
plt.show()
```



Verstyle_Histogram_multiple

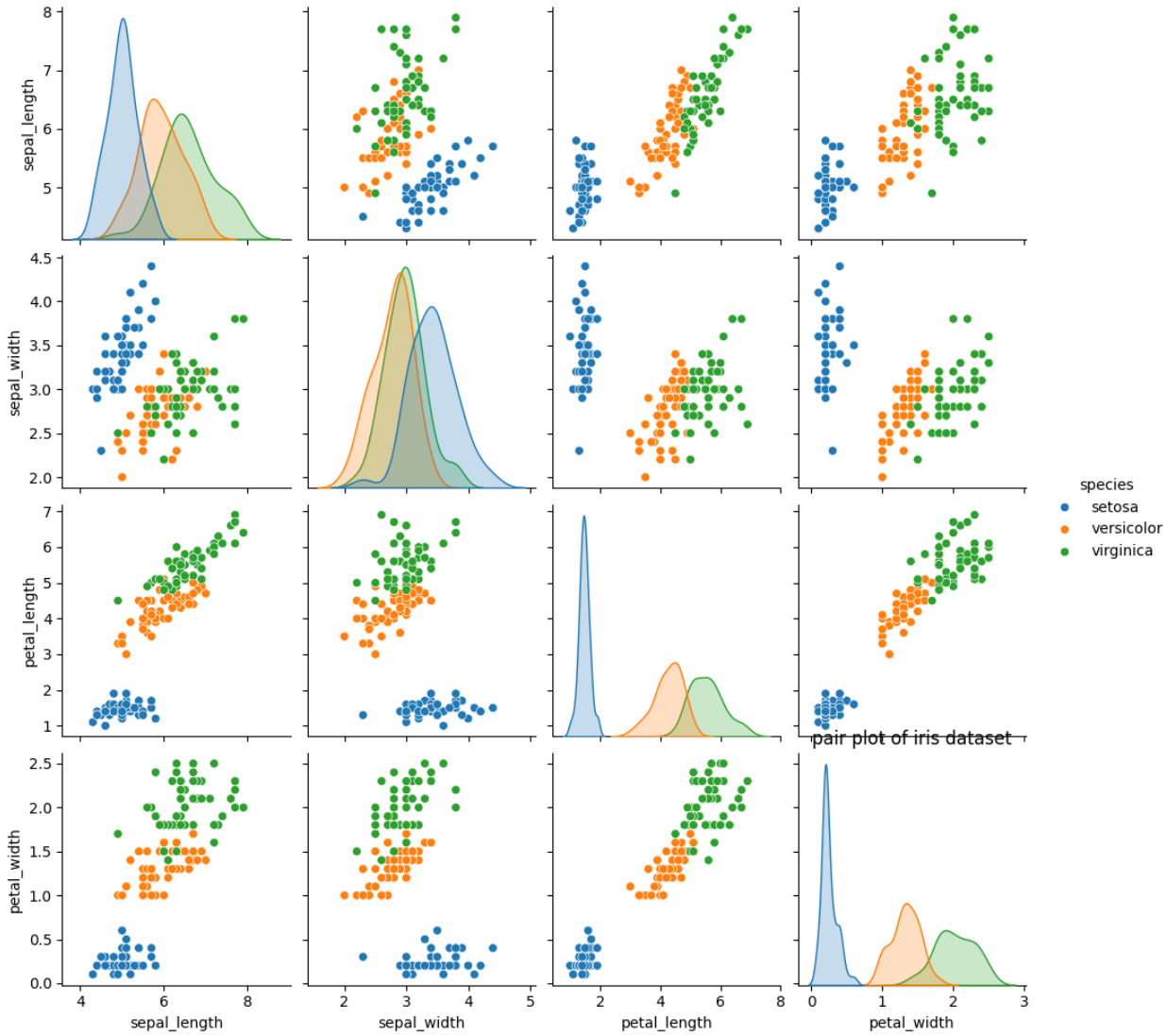
```
#verstyle histogram
import matplotlib.pyplot as plt
import numpy as np
```

```
data1=np.random.normal(loc=0,scale=1,size=1000)
data2=np.random.normal(loc=2,scale=0.5,size=1000)
data3=np.random.normal(loc=-2,scale=1.5,size=1000)

plt.hist(data1,bins=30,color='red',alpha=0.5,label='distribution1')
plt.hist(data2,bins=30,color='green',alpha=0.5,label='distribution2')
plt.hist(data3,bins=30,color='blue',alpha=0.5,label='distribution3')

plt.xlabel('value')
plt.ylabel('frequency')
plt.title('Histogram')
plt.legend()
plt.grid(True)
plt.show() #display the histogram
```



pair plot

```
import matplotlib.pyplot as plt

import seaborn as sns

data=sns.load_dataset('iris')

##create pair plot


sns.pairplot(data,hue='species')

plt.title('pair plot of iris dataset')

plt.show()
```



jion plot

import matplotlib.pyplot **as** plt

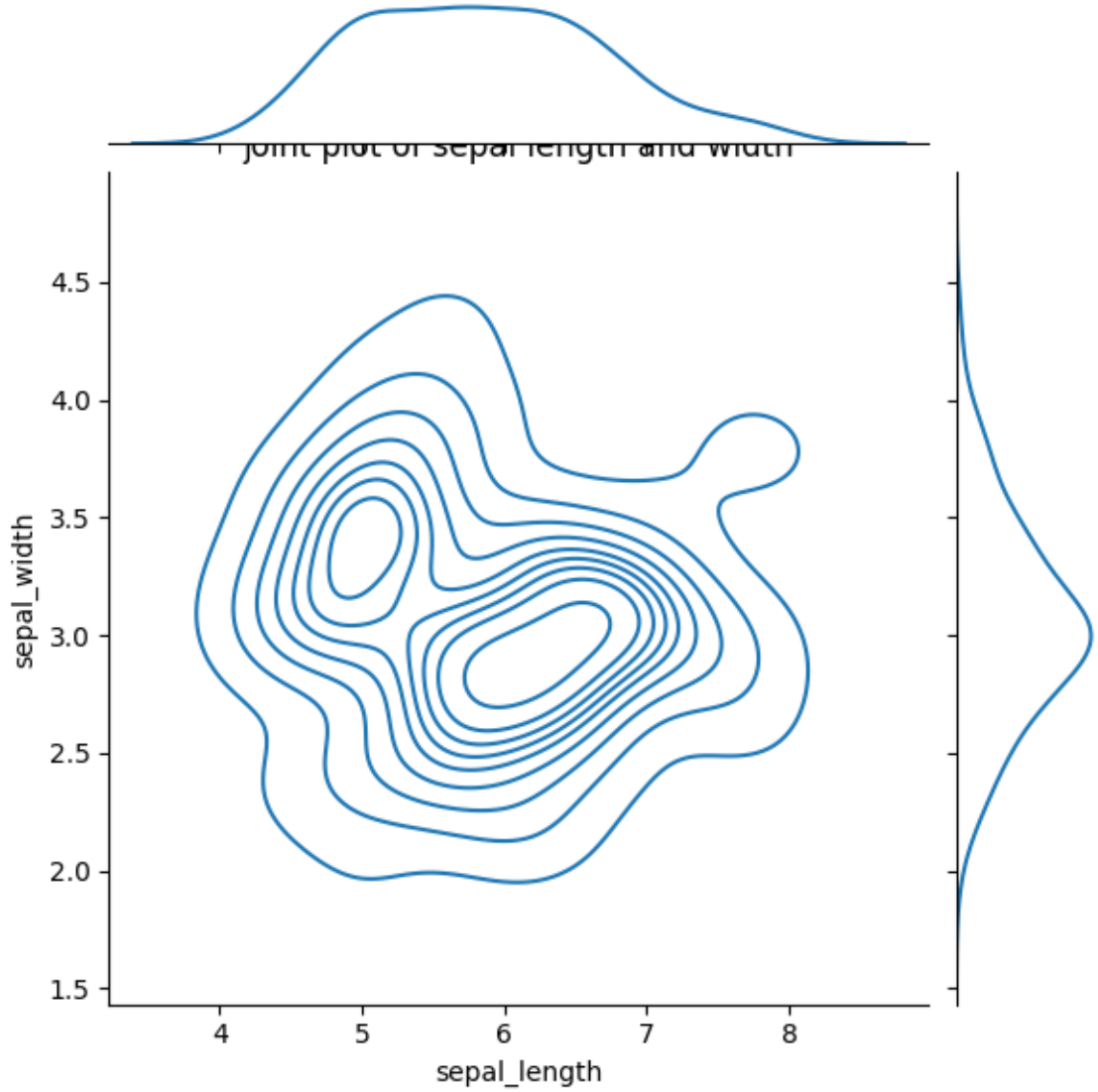
import seaborn **as** sns

#sample data(replace with your data)

data=sns.load_dataset('iris')

#create joint plot

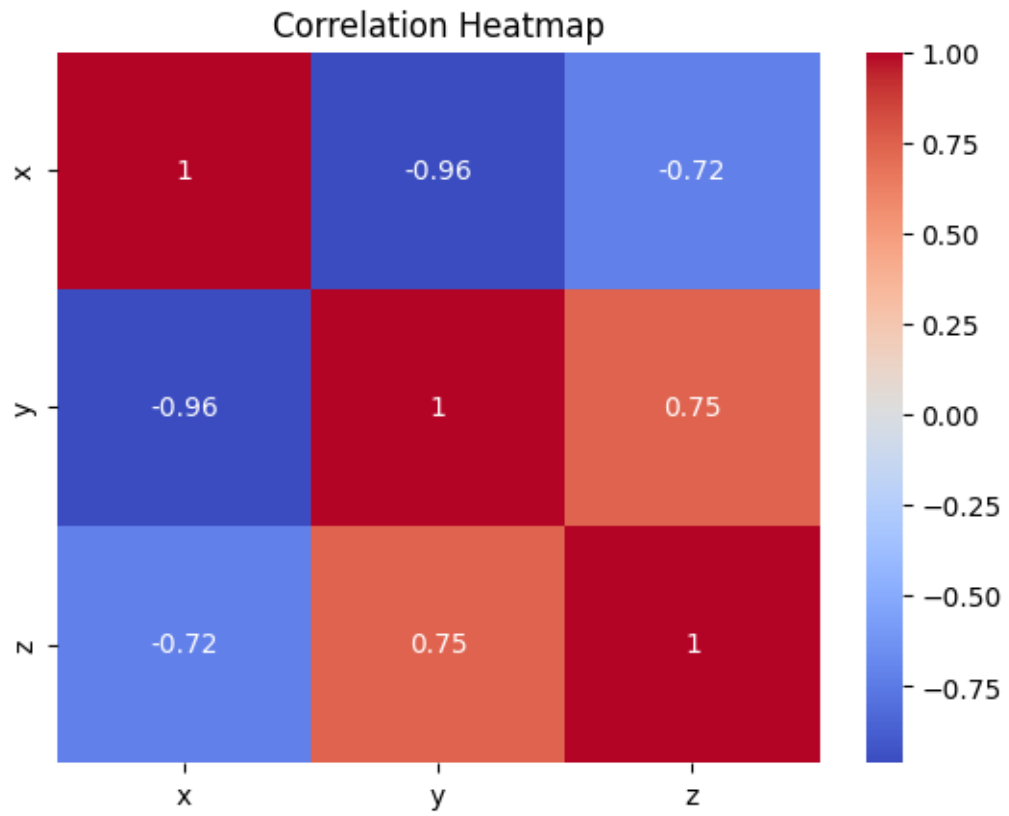
```
sns.jointplot(x='sepal_length',y='sepal_width',data=data,
              kind='kde')

plt.title('joint plot of sepal length and width')
plt.show()
```



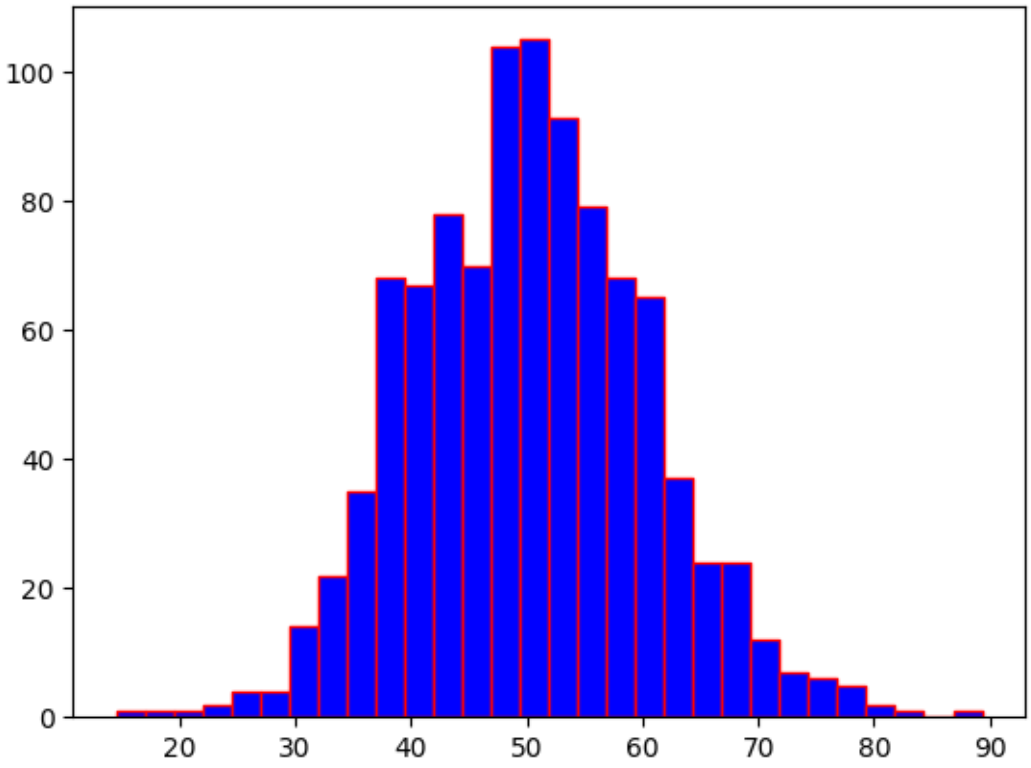
Correlation Heatmap

```
import matplotlib.pyplot as plt
```

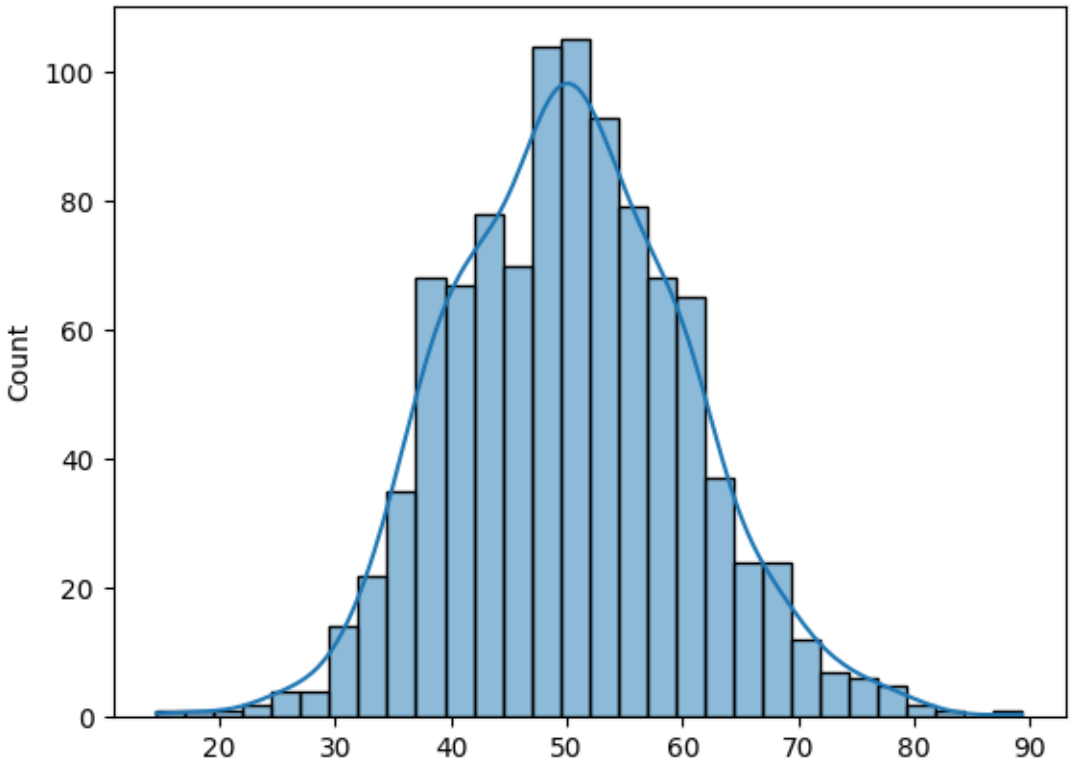
```
import numpy as np
import seaborn as sns
import pandas as pd
data = pd.DataFrame({
   "x": [1, 2, 3, 4, 5],
   "y": [66, 44, 39, 2, 1],
   "z": [244, 29, 35, 3, 33],
})

# Correlation matrix
corr = data.corr()

# Heatmap
sns.heatmap(corr,
            annot=True,
            cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```




```
data = np.random.normal(loc=50, scale=10, size=1000)


plt.hist(data, bins=30, color='blue', edgecolor='red' )

plt.show()
```

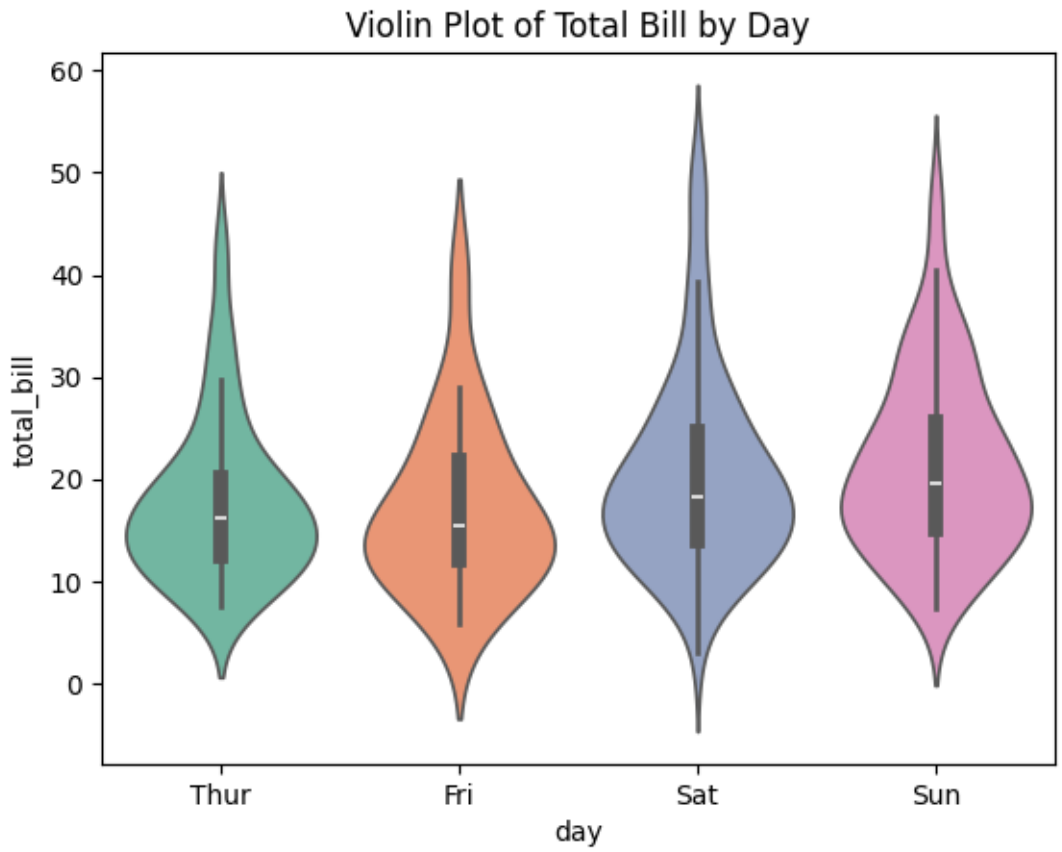



```
sns.histplot(data, bins=30, kde=True)

plt.show()
```



Voilin Chart

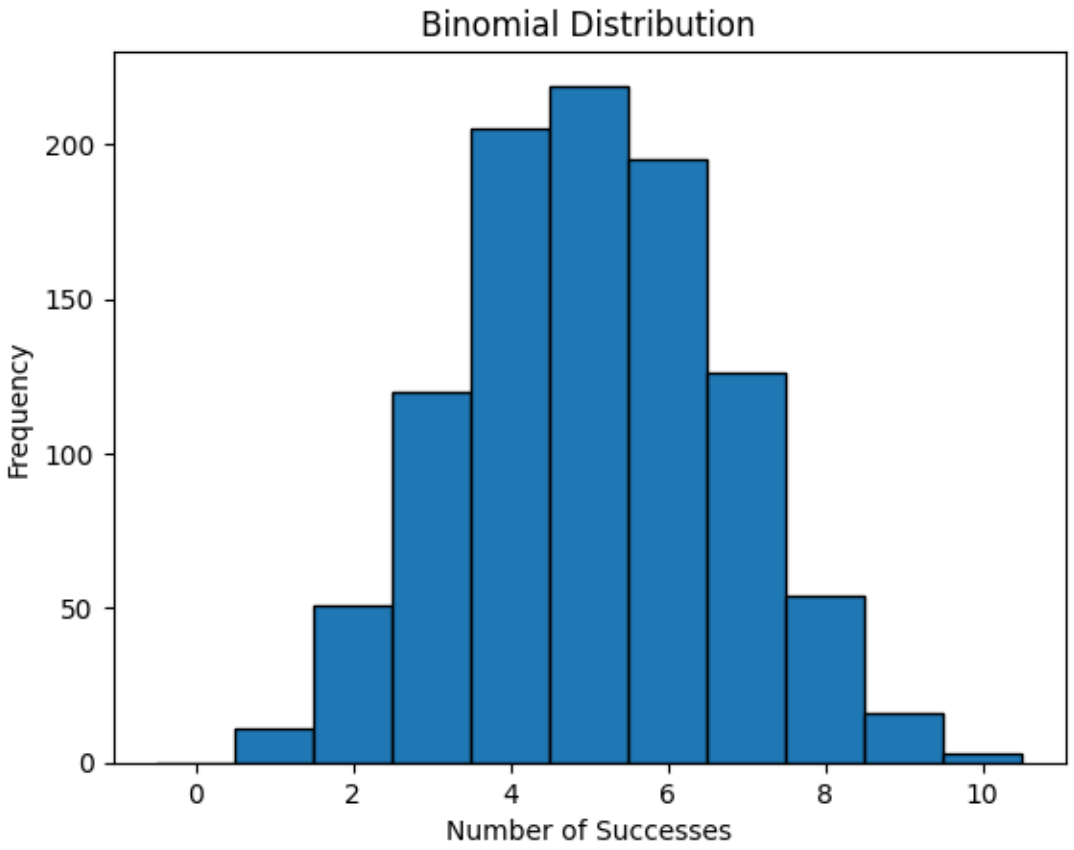
```
import seaborn as sns

import matplotlib.pyplot as plt


# Load the example dataset

tips = sns.load_dataset("tips")


# Violin plot with hue set to x-variable and legend disabled

sns.violinplot(x="day", y="total_bill", data=tips, inner="box", palette="Set2", hue="day", legend=False)


plt.title("Violin Plot of Total Bill by Day")

plt.show()
```



Binomial Distribution

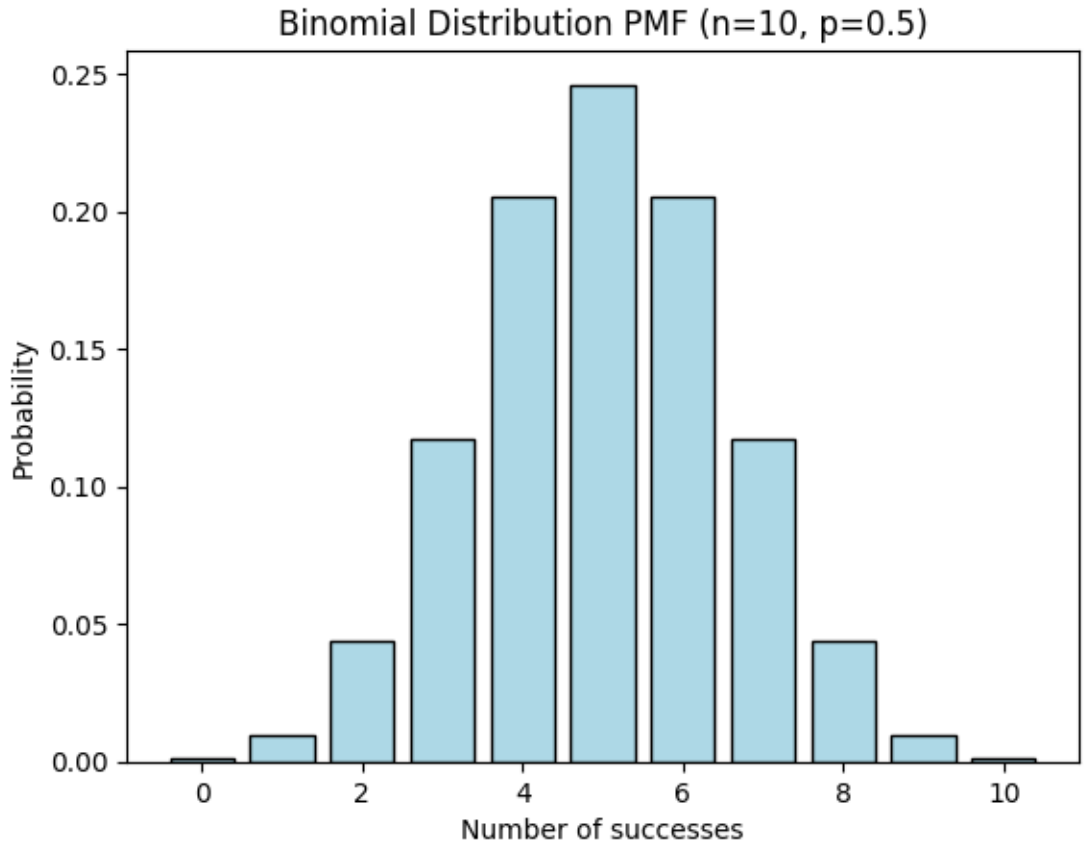
```
import numpy as np


# Parameters: n = number of trials, p = probability of success, size = number of experiments
n = 10        # number of trials
p = 0.5       # probability of success
size = 1000   # simulate 1000 experiments


# Generate binomial distribution samples
data = np.random.binomial(n, p, size)


# Optional: visualize
```

```
import matplotlib.pyplot as plt

plt.hist(data, bins=range(n+2), edgecolor='black', align='left')

plt.title("Binomial Distribution")

plt.xlabel("Number of Successes")

plt.ylabel("Frequency")

plt.show()
```




```
import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

import pandas as pd

from scipy.stats import binom

n = 10
```

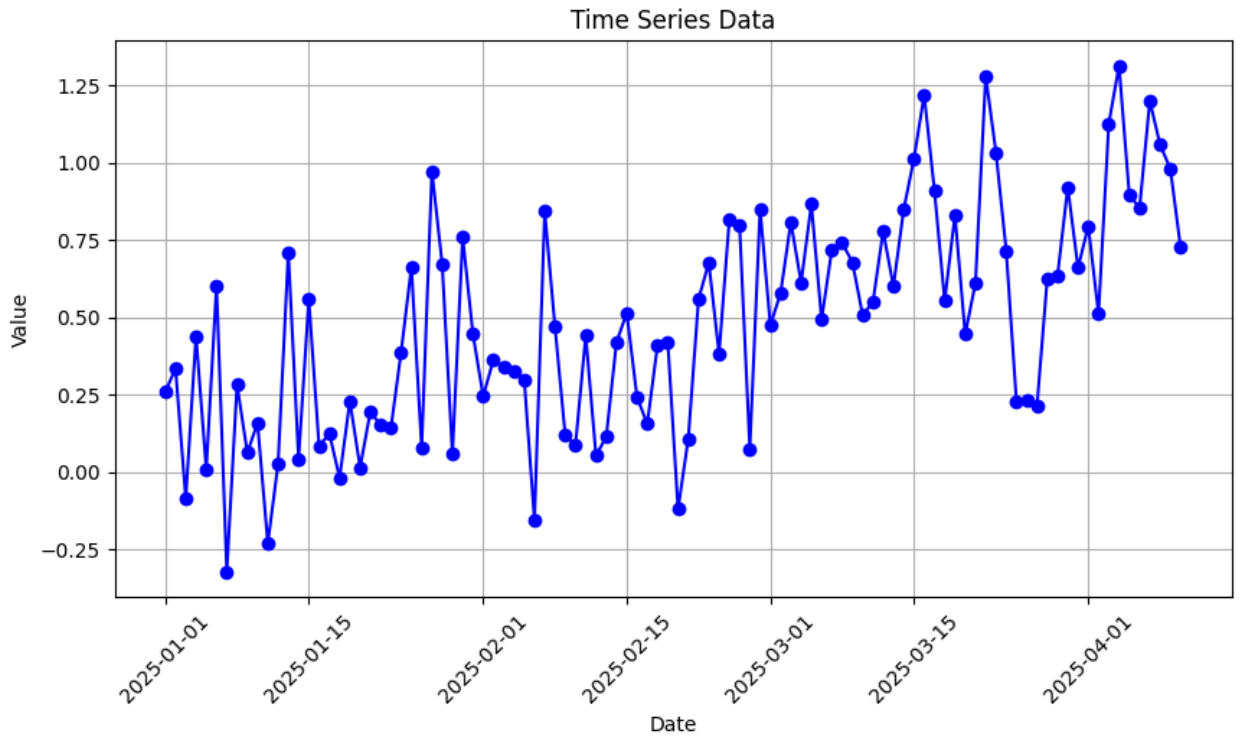
```
p = 0.5


x = np.arange(0, n+1)
y = binom.pmf(x, n, p)


plt.bar(x,y,color='lightblue', edgecolor='black')
plt.title('Binomial Distribution PMF (n=10, p=0.5)')
plt.xlabel('Number of successes')
plt.ylabel('Probability')
plt.show()
```



Time Series Data

```
dates = pd.date_range('2025-01-01', periods=100, freq='D')
```

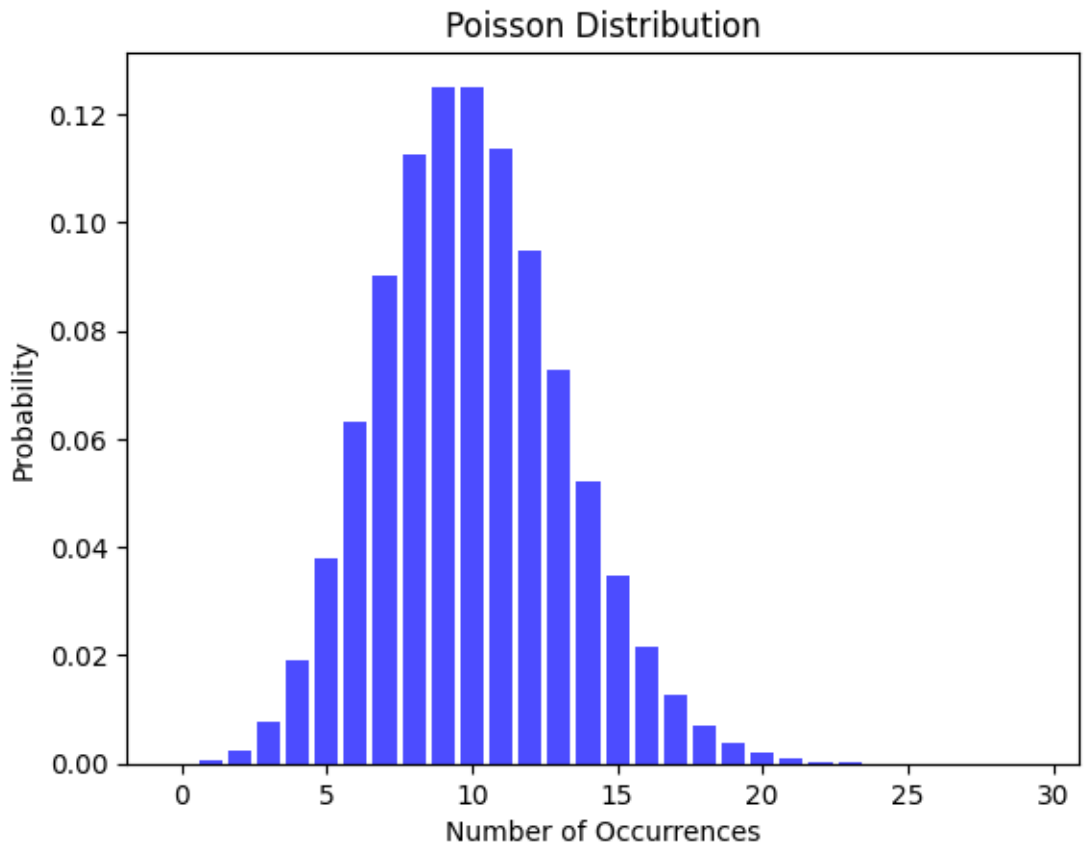
```
values = np.sin(np.linspace(0,1,100)) + np.random.normal(scale=0.3, size=100)
df = pd.DataFrame({'Date': dates, 'Value': values})

plt.figure(figsize=(10, 5))

plt.plot(df['Date'], df['Value'], marker='o', linestyle='-', color='blue')
plt.title('Time Series Data')
plt.xlabel('Date')
plt.ylabel('Value')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



paission distribution

```
import numpy as np
```

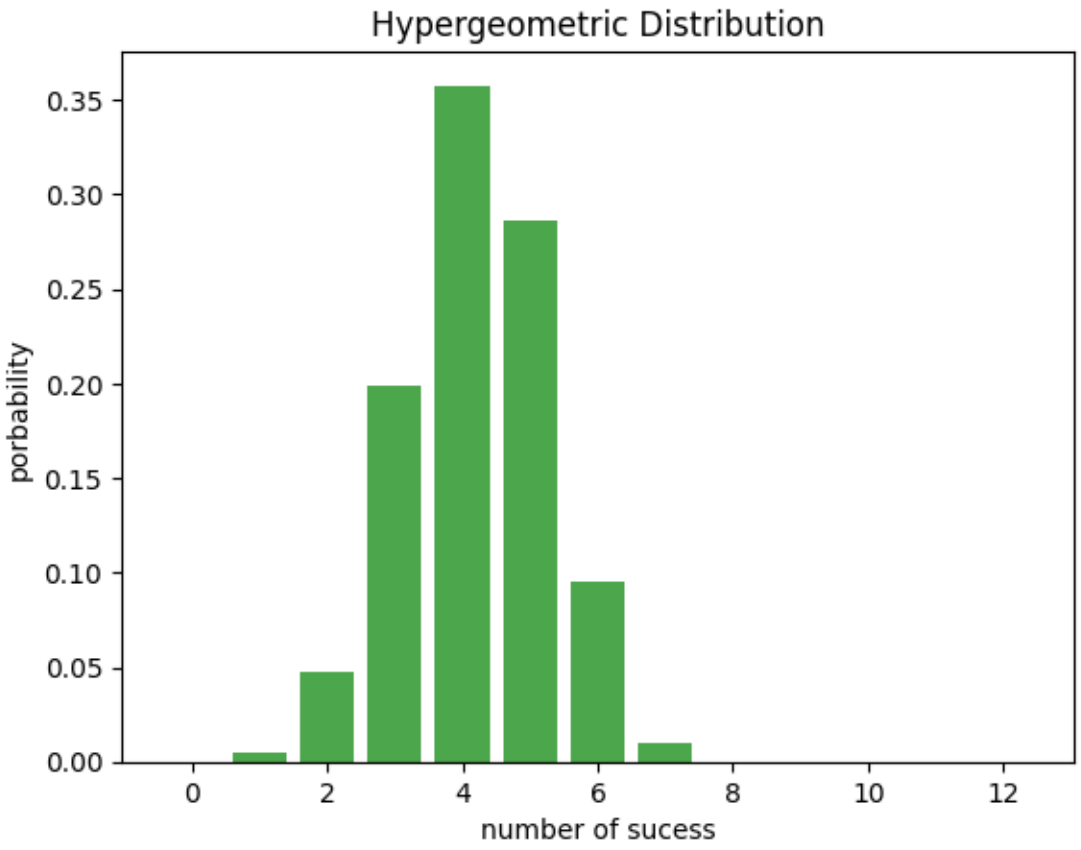
```
import matplotlib.pyplot as plt
import scipy.stats as stats

# Parameters
mu = 10
x = np.arange(0, 30)

# Poisson PMF
pmf = stats.poisson.pmf(x, mu)

# Plot
plt.bar(x, pmf, color="blue", alpha=0.7)
plt.xlabel("Number of Occurrences")
plt.ylabel("Probability")
plt.title("Poisson Distribution")
plt.show()
```



Hypergeometric Distribution

```
import numpy as np

import matplotlib.pyplot as plt

import scipy.stats as stats

N = 20 # population size

K = 7 # number of sucess stats in the populatin

n = 12 # number of draws

x = np.arange(0, n+1)

pmf = stats.hypergeom.pmf(x, N, K, n)

plt.bar(x,pmf, color="green",alpha=0.7)

plt.xlabel("number of sucess")

plt.ylabel("porbability")
```

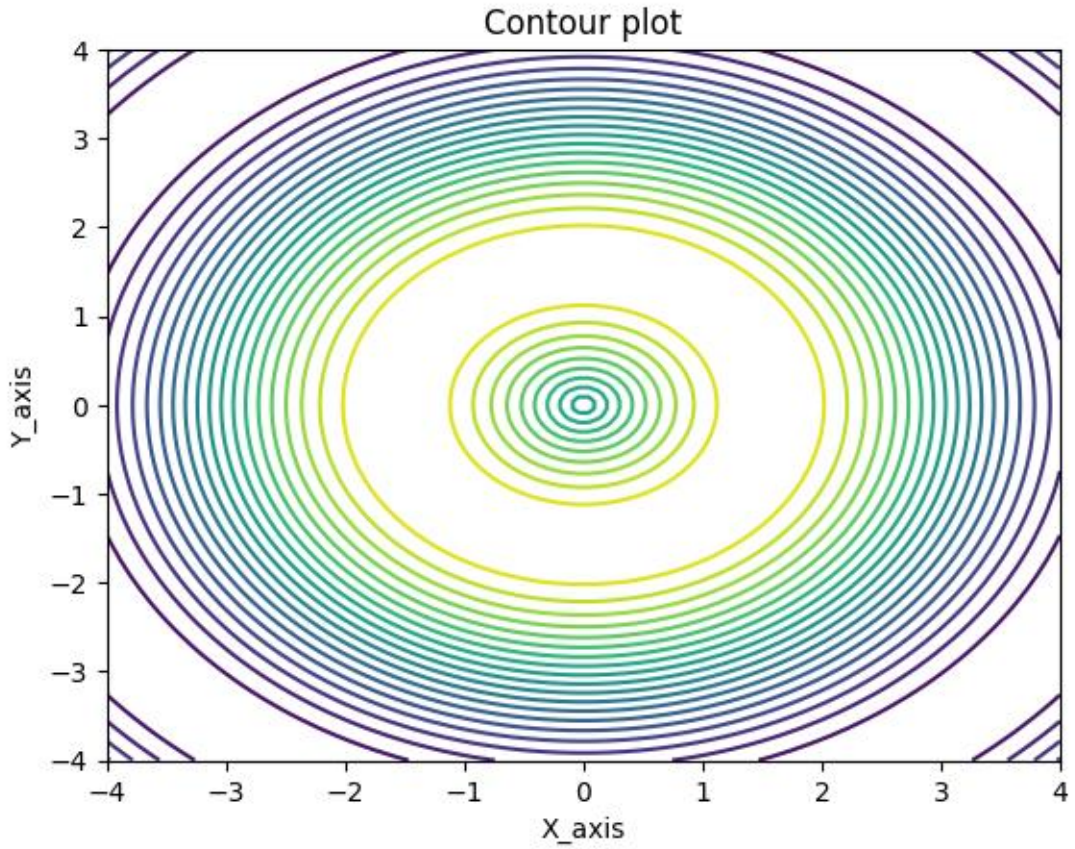
```
plt.title("Hypergeometric Distribution")

plt.show()
```



contoure plot

```
import numpy as np

import matplotlib.pyplot as plt  # Fixed: matplot → matplotlib

x = np.linspace(-4, 4, 100)       # Fixed: linespace → linspace, corrected syntax

y = np.linspace(-4, 4, 100)

X, Y = np.meshgrid(x, y)

Z = np.sin(np.sqrt(X**2 + Y**2))


plt.contour(X, Y, Z, levels=20, cmap='viridis')  # Fixed: camp → cmap

plt.title("Contour plot")                    # Removed duplicate plt.contour()
```

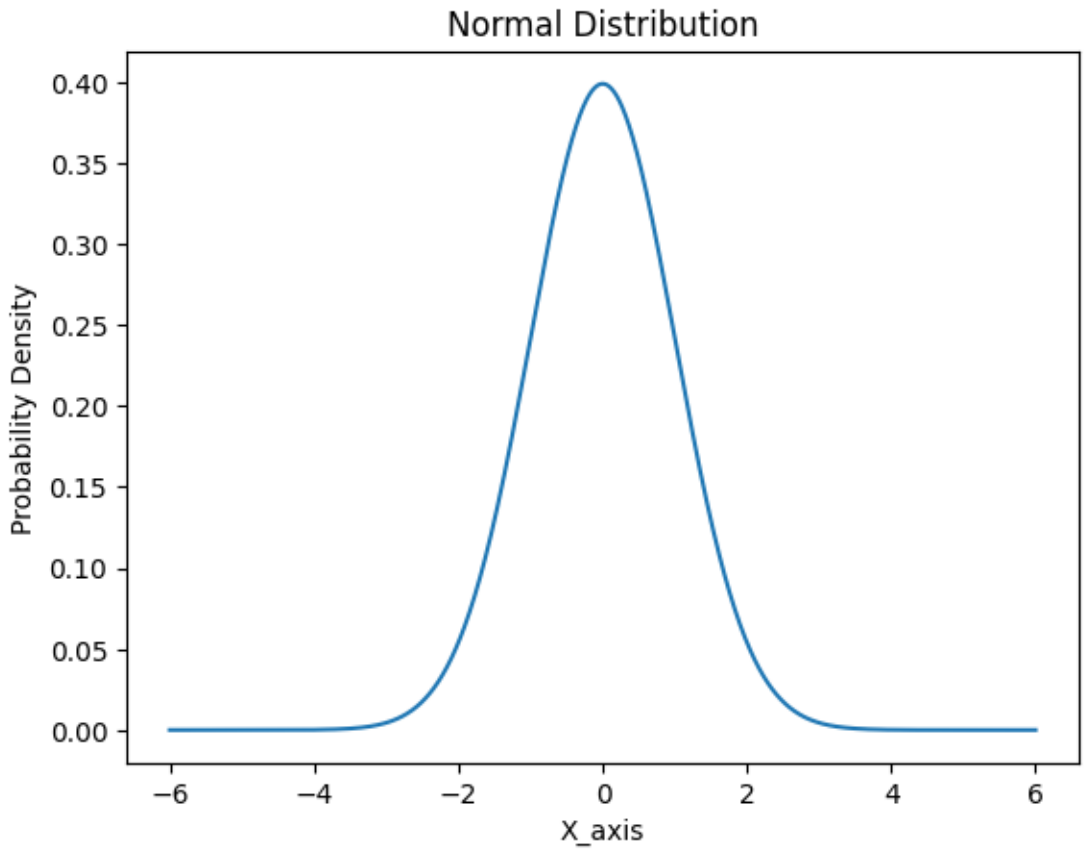
```
plt.xlabel("X_axis")

plt.ylabel("Y_axis")

plt.show()
```



Normal Distribution

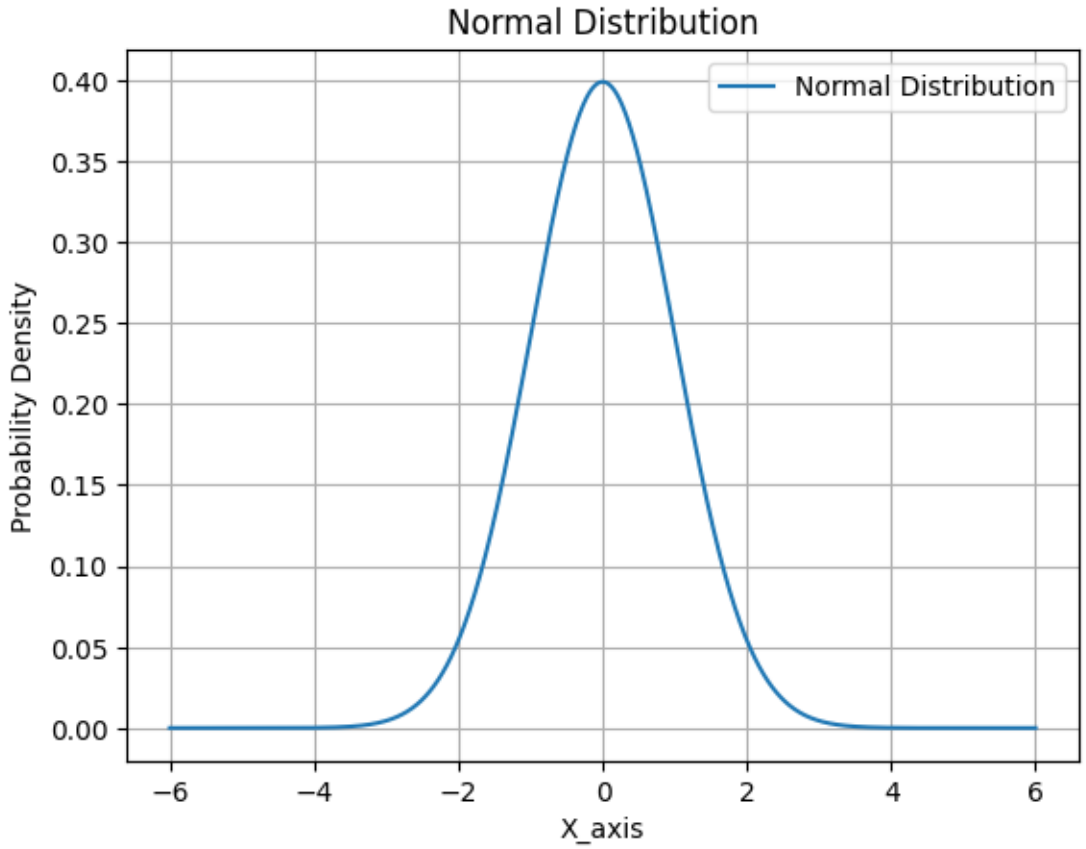
```
import numpy as np

import matplotlib.pyplot as plt

import scipy.stats as stats


x = np.linspace(-6, 6, 1000)

y = stats.norm.pdf(x, loc=0, scale=1.0)

plt.plot(x, y, label="Normal Distribution")

plt.xlabel("X_axis")
```

```
plt.ylabel("Probability Density")

plt.title("Normal Distribution")

plt.show()
```




```
import numpy as np

import matplotlib.pyplot as plt

import scipy.stats as stats


x = np.linspace(-6, 6, 1000)  # Fixed: 'linespace' -> 'linspace' and proper syntax

y = stats.norm.pdf(x, loc=0, scale=1.0)  # Fixed: added 'stats.'


plt.plot(x, y, label="Normal Distribution")

plt.xlabel("X_axis")
```

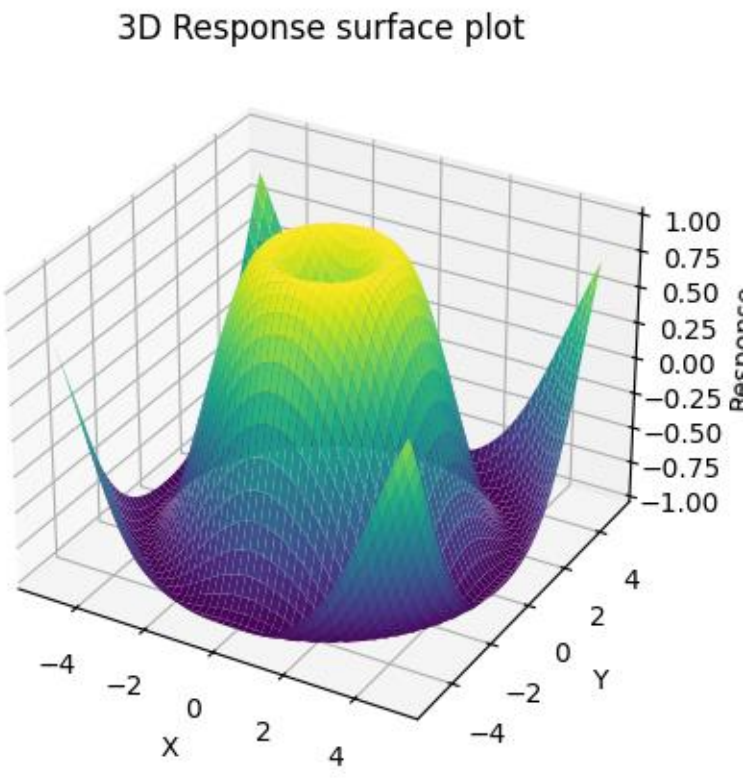
```
plt.ylabel("Probability Density")

plt.title("Normal Distribution")

plt.legend()  # Optional but recommended if using label

plt.grid(True)  # Optional: adds grid to the plot

plt.show()
```



3D response surface plot

```
import numpy as np

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D


X = np.linspace(-5, 5, 50)

Y = np.linspace(-5, 5, 50)
```

```
X, Y = np.meshgrid(X, Y)
Z = np.sin(np.sqrt(X**2 + Y**2))

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis')  # fixed 'camp' -> 'cmap'
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Response')
ax.set_title("3D Response surface plot")
plt.show()
```



Redar Chart

```
import matplotlib.pyplot as plt
import numpy as np
labels=np.array(["A","B","C","D","E"])
data=np.array([4,5,3,4,2])
angles = np.linspace(0,2*np.pi,len(labels),endpoint=False)
data=np.concatenate((data,[data[0]]))
angles=np.concatenate((angles,[angles[0]]))
plt.polar(angles,data,marker="o")
plt.fill(angles,data,alpha=0.25)
plt.title("Radar Chart")
plt.show()
```