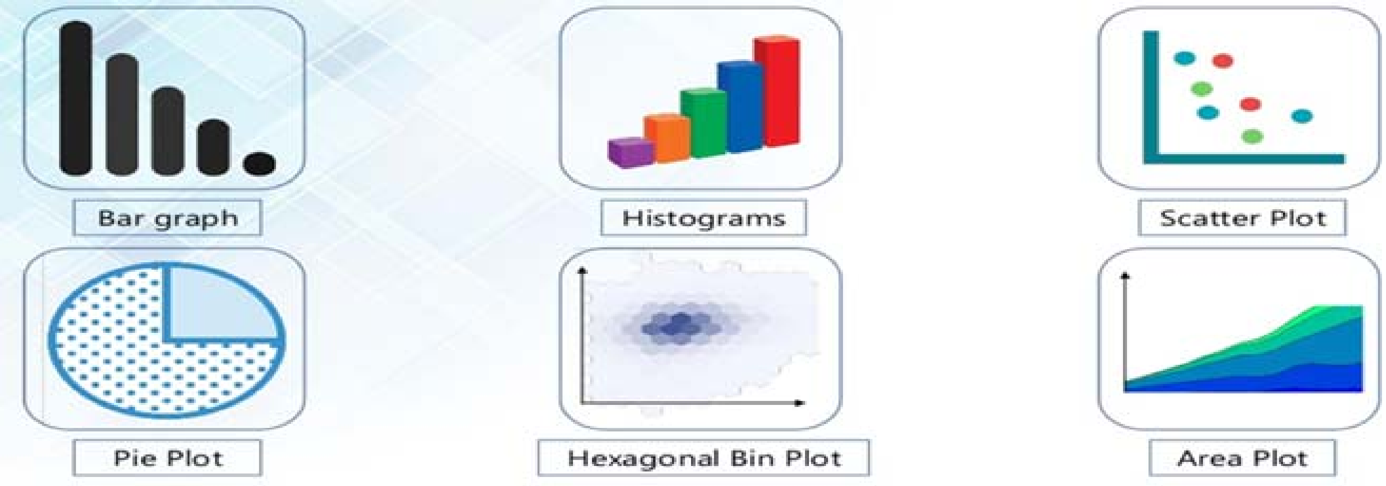
Python Library: Data Visualization

- Matplotlib is a low-level graph plotting library in Python that serves as a data visualization.
- Like Pandas, it is not directly related to Machine Learning.
- It particularly comes in handy when a programmer wants to visualize the patterns in the data.
- A module named pyplot makes it easy for programmers to plot as it provides features to control line styles, font properties, formatting axes, etc.
- It provides various graphs and plots for data visualization, viz,. histograms, error charts, bar charts, etc.



Dhirubhai Ambani University
Formerly known as DA-IICT

Python Library: Data Visualization

- **Installation**

```
!pip install matplotlib
```

- **Import**

```
import matplotlib
```

- **Most of the Matplotlib utilities lie under the pyplot submodule, and are usually imported under the plt alias:**

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

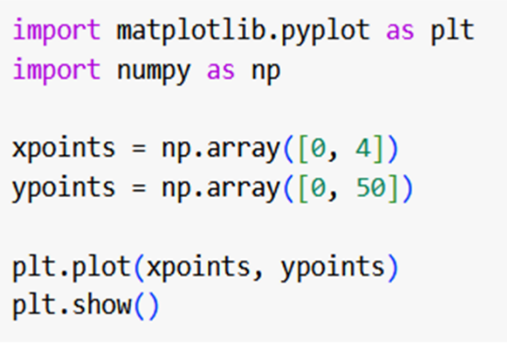
Python Library: Data Visualization

Plot( ) function:

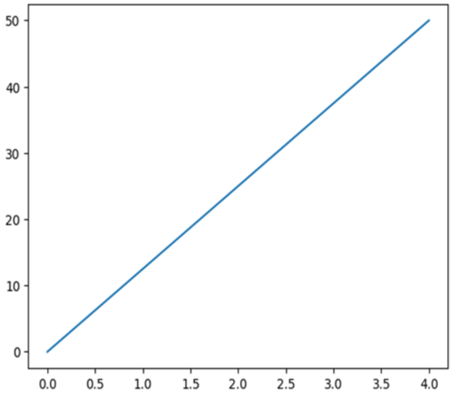
- The plot() function is used to draw points (markers) in a diagram.
- By default, the plot() function draws a line from point to point.
- The function takes parameters for specifying points in the diagram.
    - Parameter 1 is an array containing the points on the x-axis.
    - Parameter 2 is an array containing the points on the y-axis.

Example: Plotting x and y points - Line

Draw a line in a diagram from position (0,0) to position (4,50):



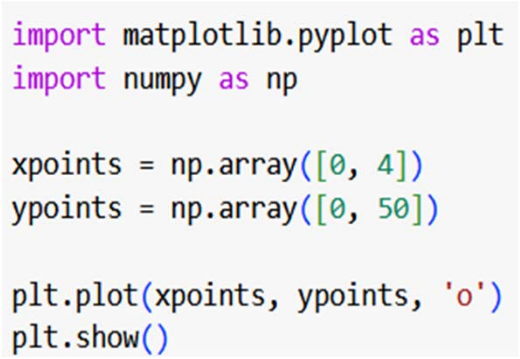
```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 4])
ypoints = np.array([0, 50])

plt.plot(xpoints, ypoints)
plt.show()
```



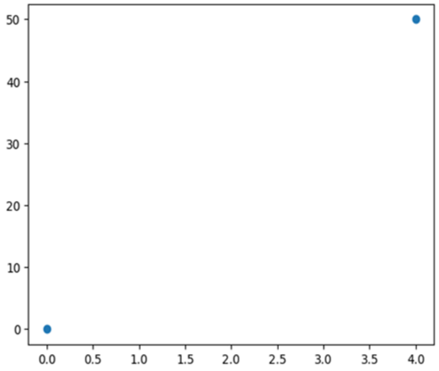
Python Library: Data Visualization

Example: Plotting x and y points - Without Line

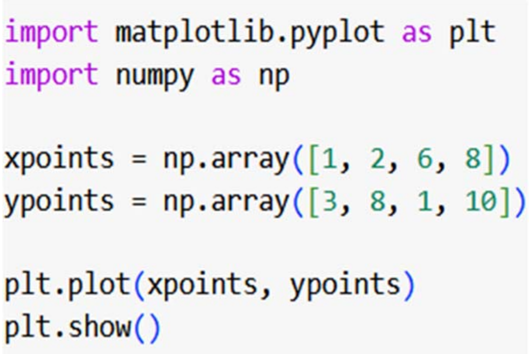
Draw a line in a diagram from position (0,0) to position (4,50):



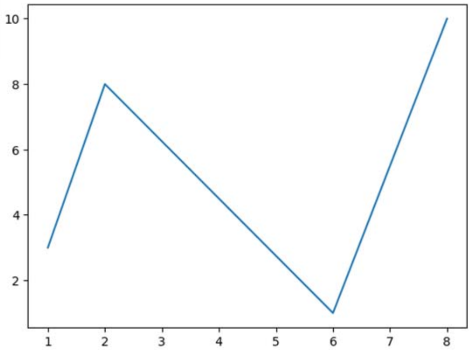
```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 4])
ypoints = np.array([0, 50])

plt.plot(xpoints, ypoints, 'o')
plt.show()
```



Example: Plotting multiple points



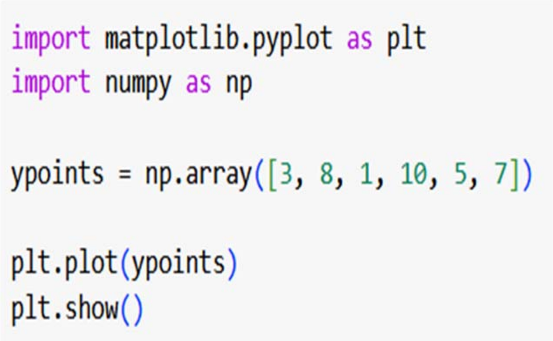
```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
plt.show()
```



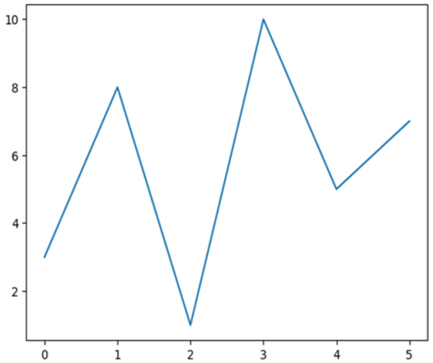
Python Library: Data Visualization

Example: Plotting with Default X-points

If we do not specify the points on the x-axis, they will get the default values 0, 1, 2, 3 etc., depending on the length of the y-points.

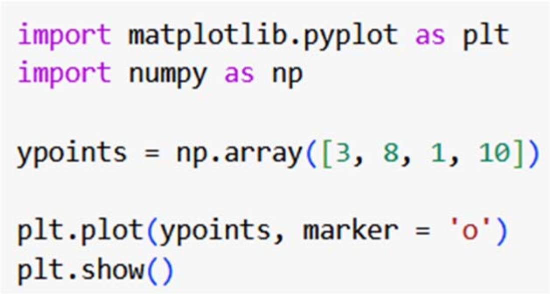


```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10, 5, 7])

plt.plot(ypoints)
plt.show()
```

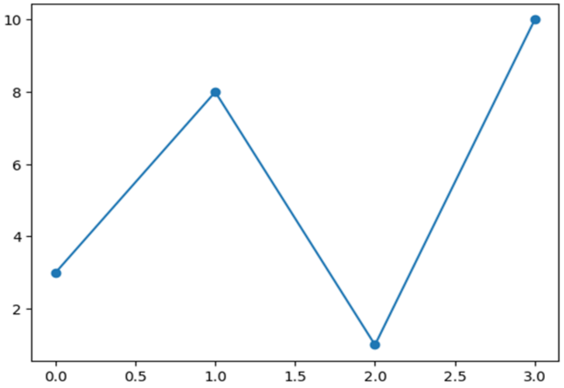


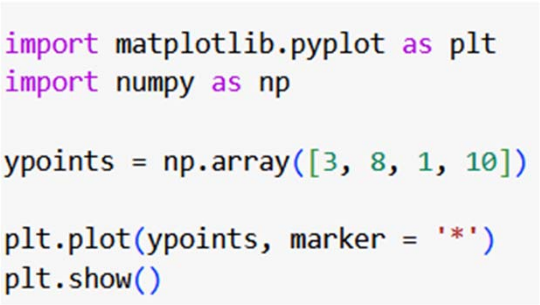
Python Library: Data Visualization

Example: Add Markers

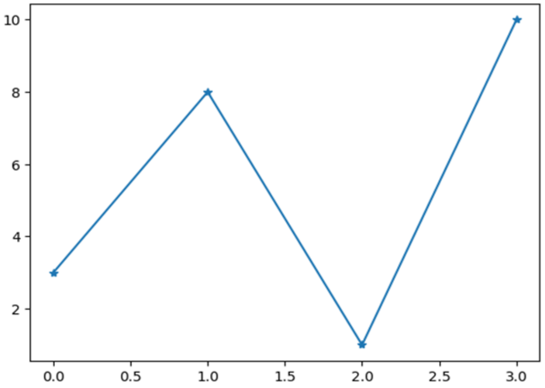


```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o')
plt.show()
```





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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = '*')
plt.show()
```



Marker	Description
'o'	Circle
'*'	Star
'.'	Point
','	Pixel
'x'	X
'X'	X(filled)
'+'	Plus
'P'	Plus (filled)
's'	Square
'D'	Diamond
'd'	Diamond (thin)
'p'	Pentagon

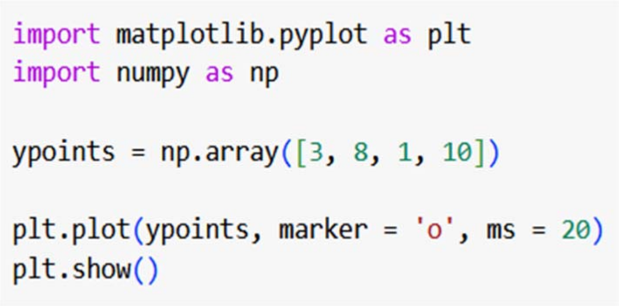
Marker	Description
'H'	Hexagon
'h'	Hexagon
'v'	Triangle Down
'^'	Triangle Up
'<'	Triangle Left
'>'	Triangle Right
'1'	Tri Down
'2'	Tri Up
'3'	Tri Left
'4'	Tri Right
'|'	Vline
'_'	Hline

Dhirubhai Ambani University
Formerly known as DA-IICT

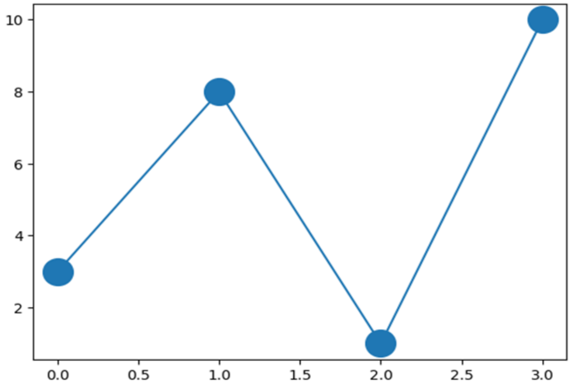
Python Library: Data Visualization

Example: Markers- Size

- Use the keyword argument markersize or the shorter version, ms to set the size of the markers

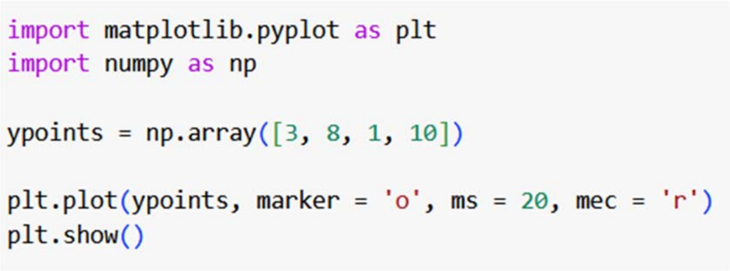


```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20)
plt.show()
```

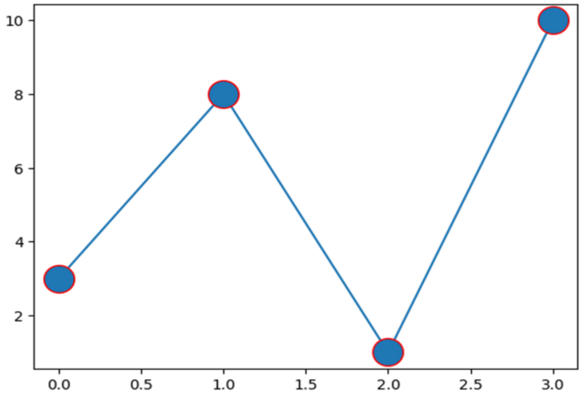


Example: Markers- Color

- Use the keyword argument markeredgecolor or the shorter mec to set the color of the edge of the markers.



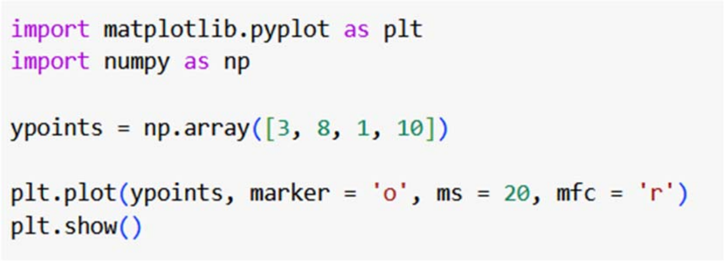
```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mec = 'r')
plt.show()
```



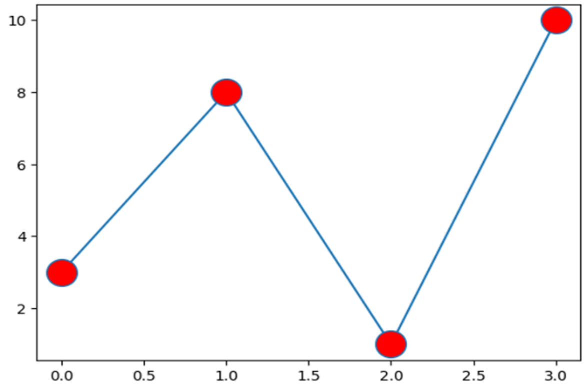
Python Library: Data Visualization

Example: Markers- Face Color

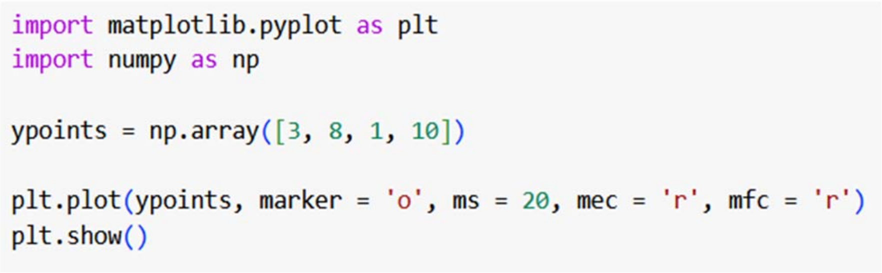
- Use the keyword argument markerfacecolor or the shorter mfc to set the color inside the edge of the markers:



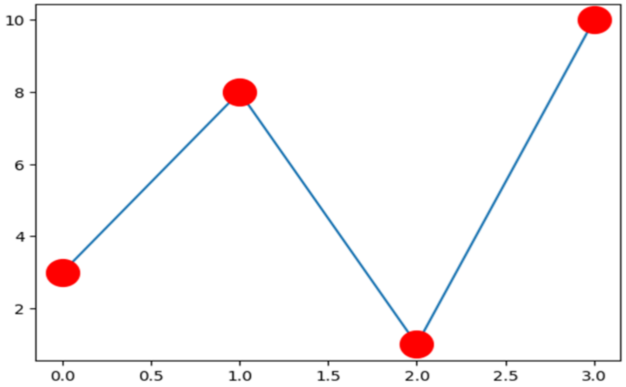
```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mfc = 'r')
plt.show()
```



Example: Markers- Edge and Face Color

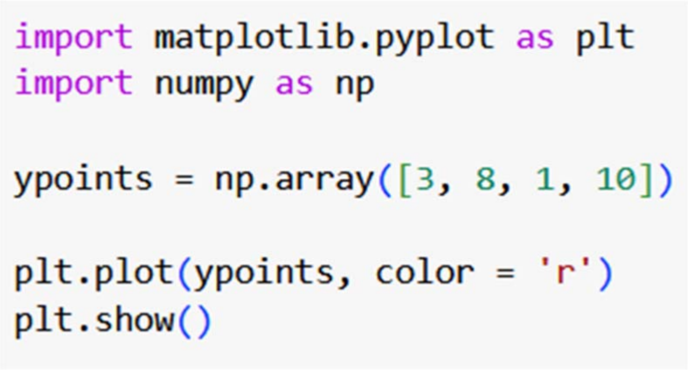


```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mec = 'r', mfc = 'r')
plt.show()
```

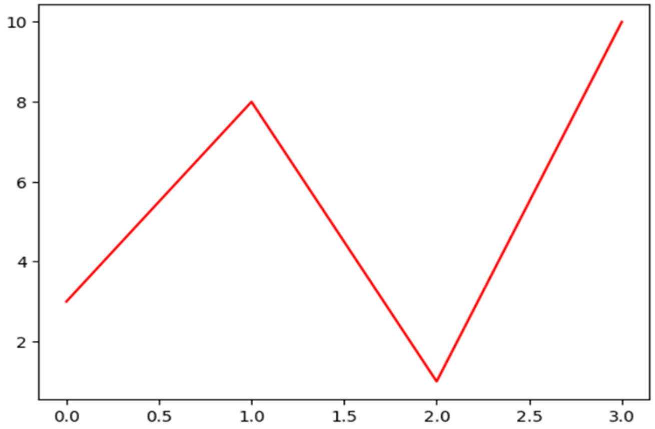


Python Library: Data Visualization

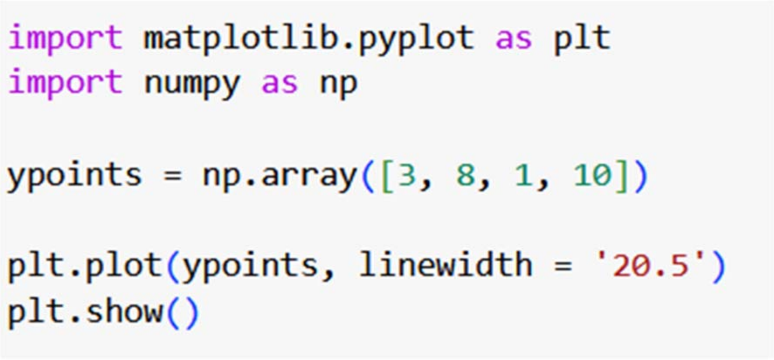
Example: Line Color



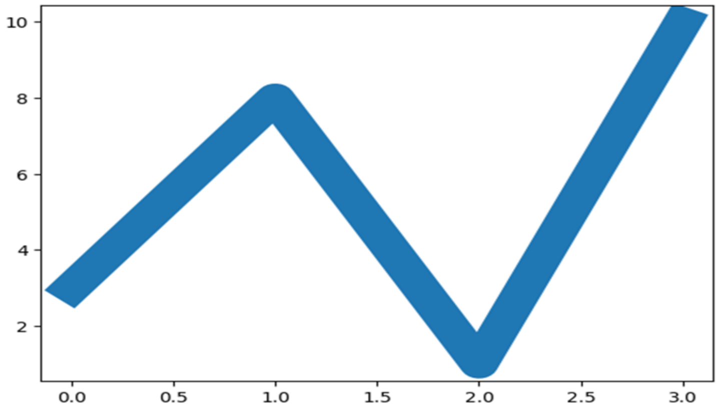
```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'r')
plt.show()
```



Example: Line Width



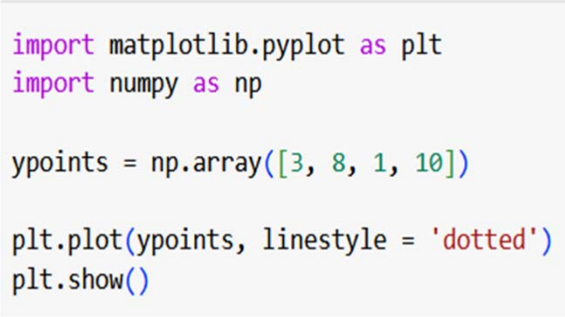
```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linewidth = '20.5')
plt.show()
```



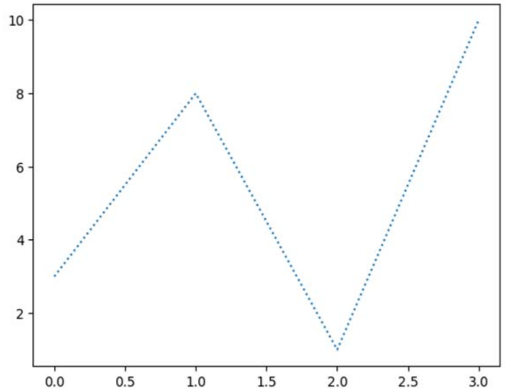
Python Library: Data Visualization

Example: Linestyle

- Use the keyword argument linestyle, or shorter ls, to change the style of the plotted line.



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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linestyle = 'dotted')
plt.show()
```

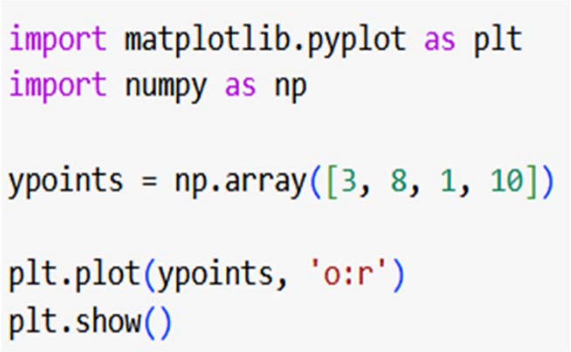


Style	Or
'solid' (default)	'-'
'dotted'	':'
'dashed'	'--'
'dashdot'	'-.'
'None'	'' or ' '

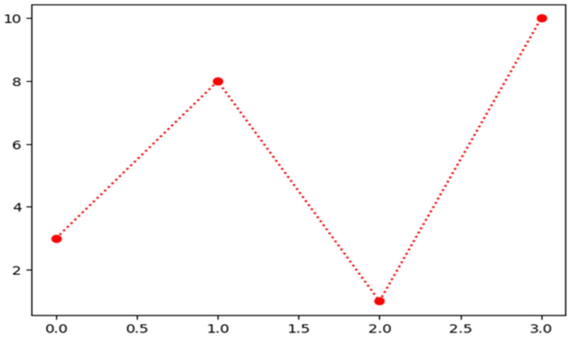
Python Library: Data Visualization

Example: Markers- Formats

- You can also use the shortcut string notation parameter to specify the marker.
- This parameter is also called fmt, and is written with this syntax:
  - marker|line|color



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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, 'o:r')
plt.show()
```



Line Reference

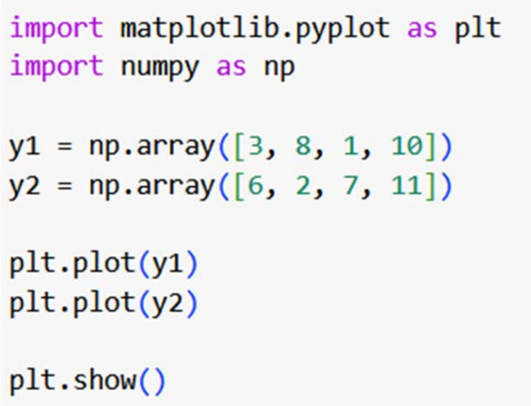
Line Syntax	Description
'-'	Solid line
':'	Dotted line
'--'	Dashed line
'-.'	Dashed / dotted line

Color Reference

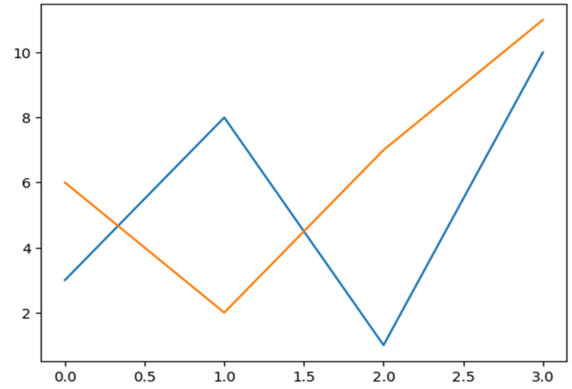
Color Syntax	Description
' r '	Red
' g '	Green
' b '	Blue
' c '	Cyan
' m '	Magenta
' y '	Yellow
' k '	Black
' W '	White

Python Library: Data Visualization

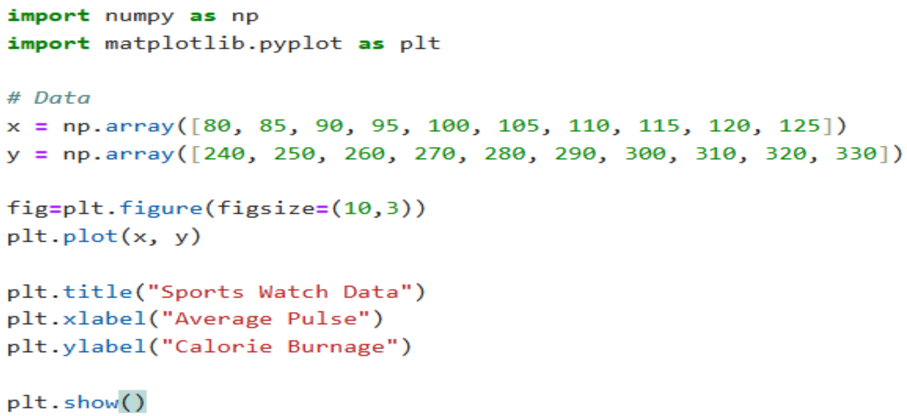
Example: Multiple Plots



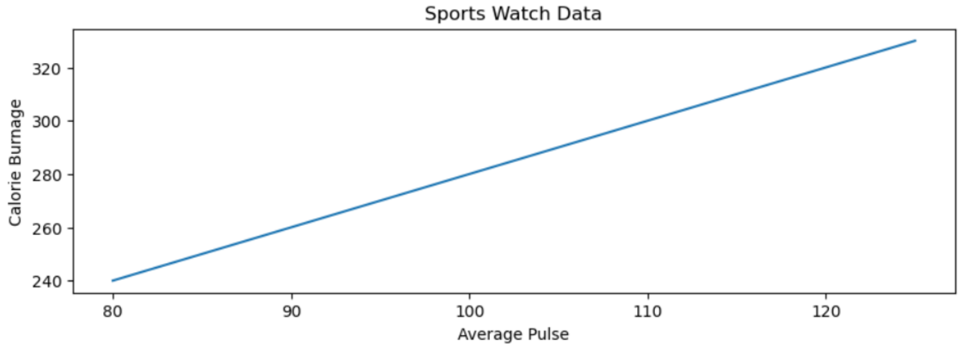
```
import matplotlib.pyplot as plt
import numpy as np

y1 = np.array([3, 8, 1, 10])
y2 = np.array([6, 2, 7, 11])

plt.plot(y1)
plt.plot(y2)

plt.show()
```



Example: Size of Figure

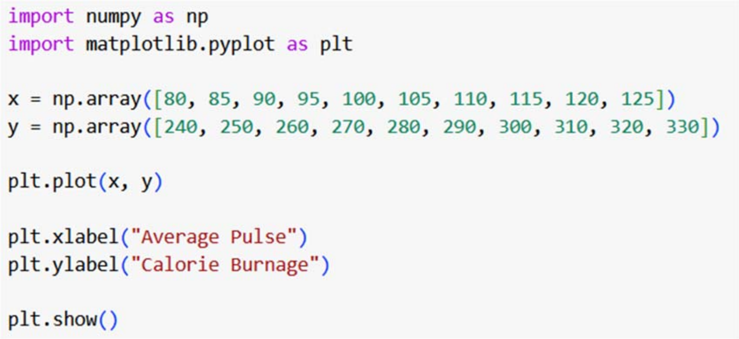


```
import numpy as np
import matplotlib.pyplot as plt

# Data
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

fig=plt.figure(figsize=(10,3))
plt.plot(x, y)

plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.show()
```

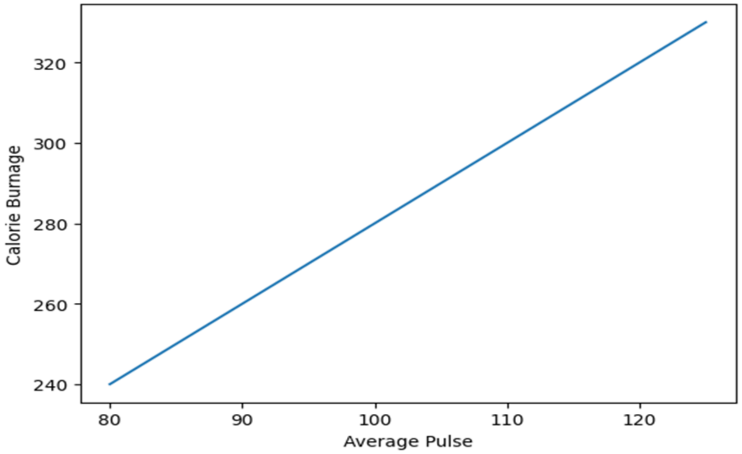


Python Library: Data Visualization

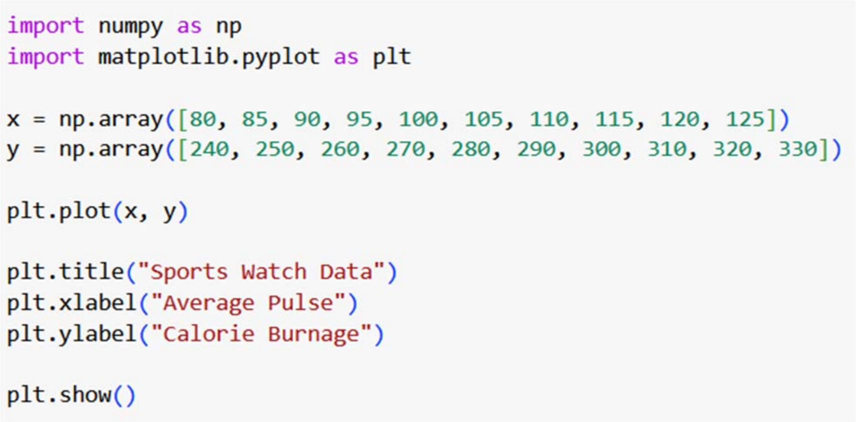
Example: Label



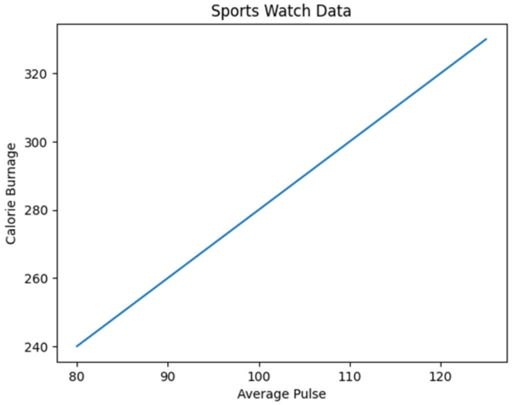
```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)

plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.show()
```



Example: Title

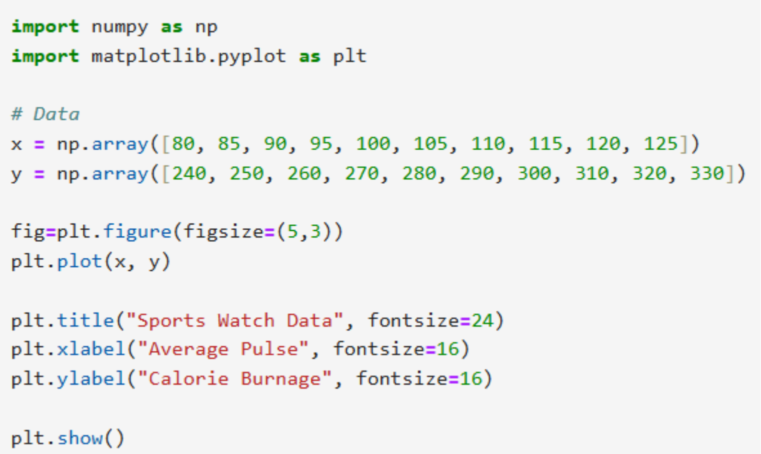


```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)

plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.show()
```

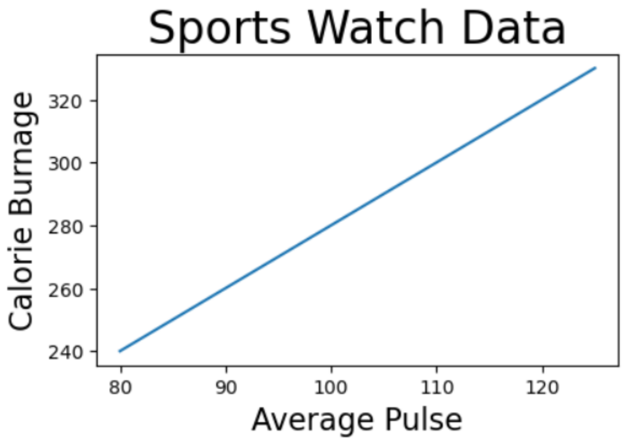


Python Library: Data Visualization

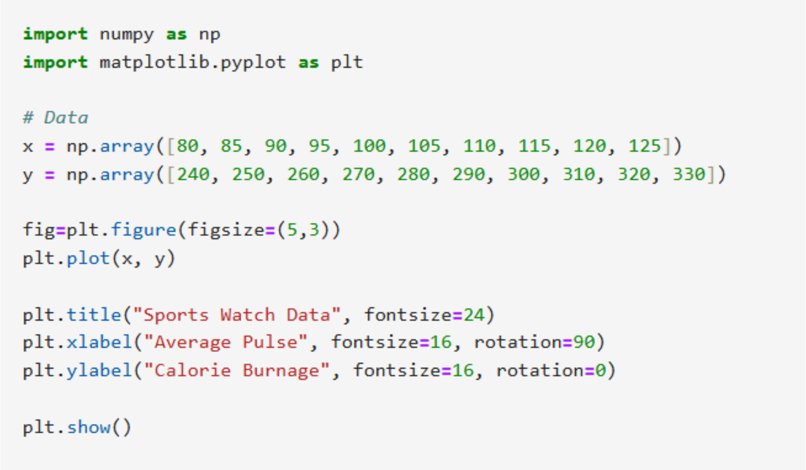
Example: Font size of Labels



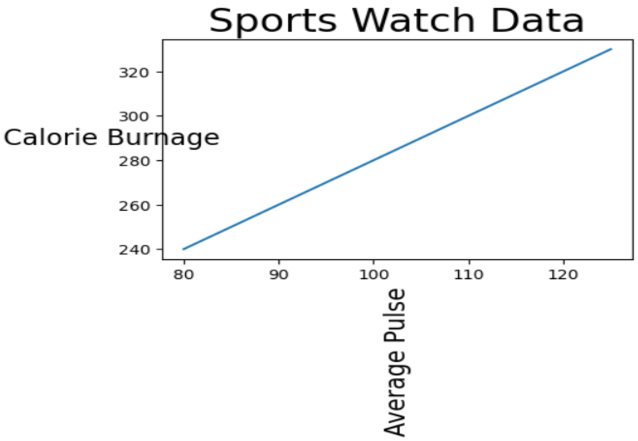
```
import numpy as np
import matplotlib.pyplot as plt

# Data
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

fig=plt.figure(figsize=(5,3))
plt.plot(x, y)

plt.title("Sports Watch Data", fontsize=24)
plt.xlabel("Average Pulse", fontsize=16)
plt.ylabel("Calorie Burnage", fontsize=16)

plt.show()
```



Example: Rotation of Axis text



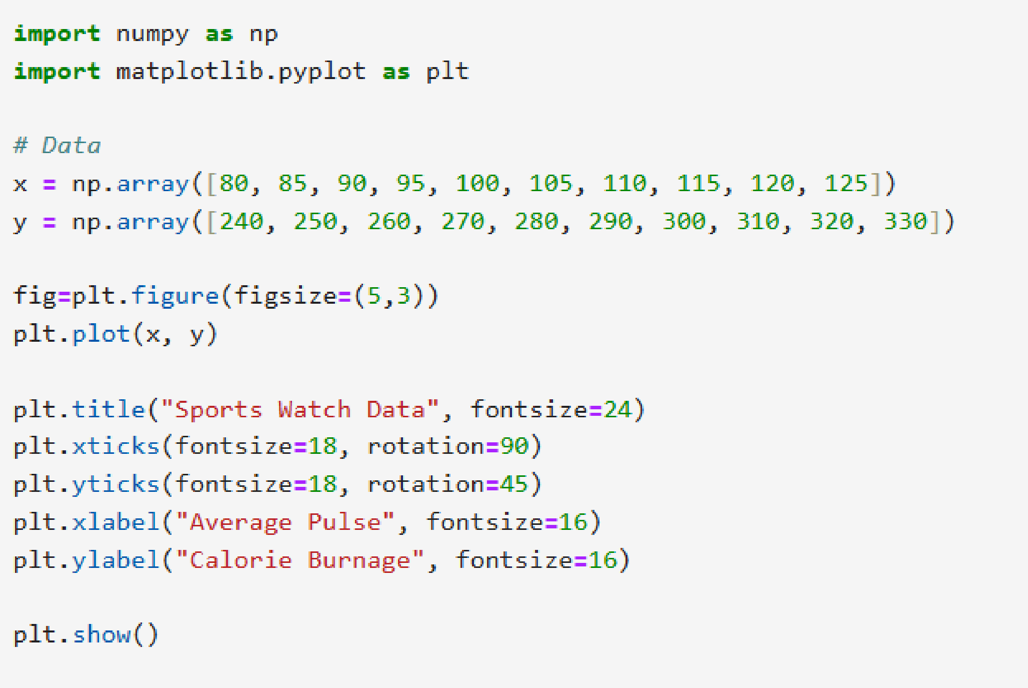
```
import numpy as np
import matplotlib.pyplot as plt

# Data
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

fig=plt.figure(figsize=(5,3))
plt.plot(x, y)

plt.title("Sports Watch Data", fontsize=24)
plt.xlabel("Average Pulse", fontsize=16, rotation=90)
plt.ylabel("Calorie Burnage", fontsize=16, rotation=0)

plt.show()
```



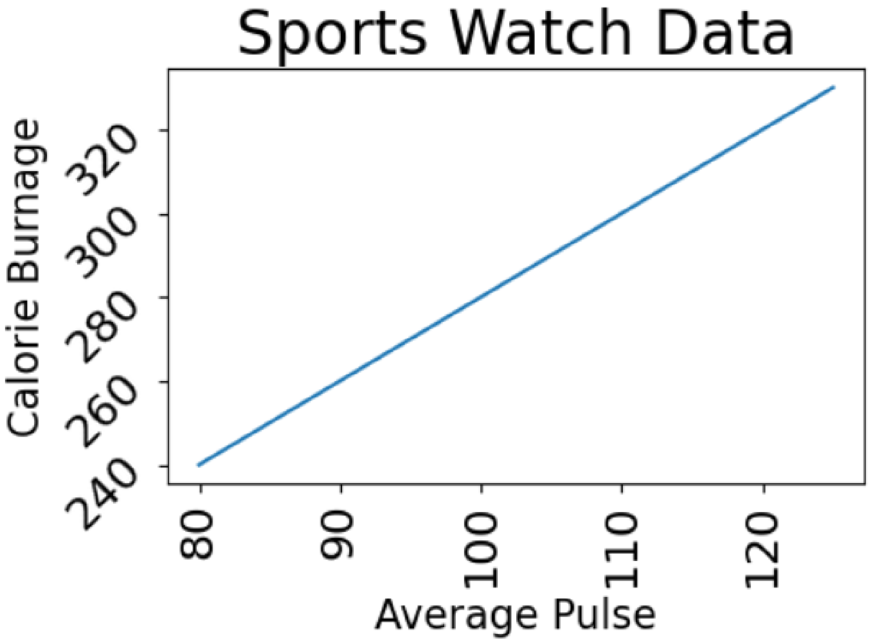
Dhirubhai Ambani University
Formerly known as DA-IICT

Python Library: Data Visualization

Example: Size of axis text

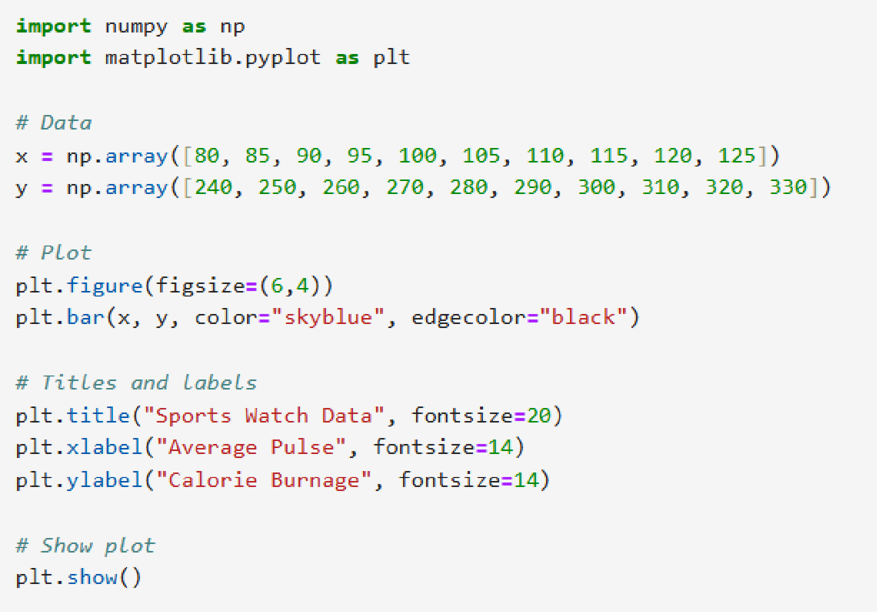


```
import numpy as np
import matplotlib.pyplot as plt

# Data
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

fig=plt.figure(figsize=(5,3))
plt.plot(x, y)

plt.title("Sports Watch Data", fontsize=24)
plt.xticks(fontsize=18, rotation=90)
plt.yticks(fontsize=18, rotation=45)
plt.xlabel("Average Pulse", fontsize=16)
plt.ylabel("Calorie Burnage", fontsize=16)

plt.show()
```

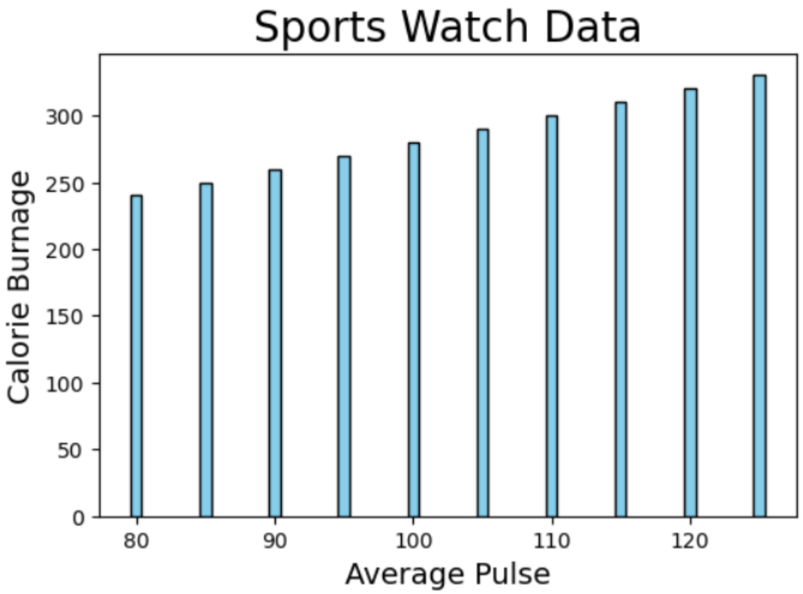


Python Library: Data Visualization

Bar Plot: **A bar plot (or bar chart) is a type of graph that uses rectangular bars to represent and compare categorical data (discrete groups).**

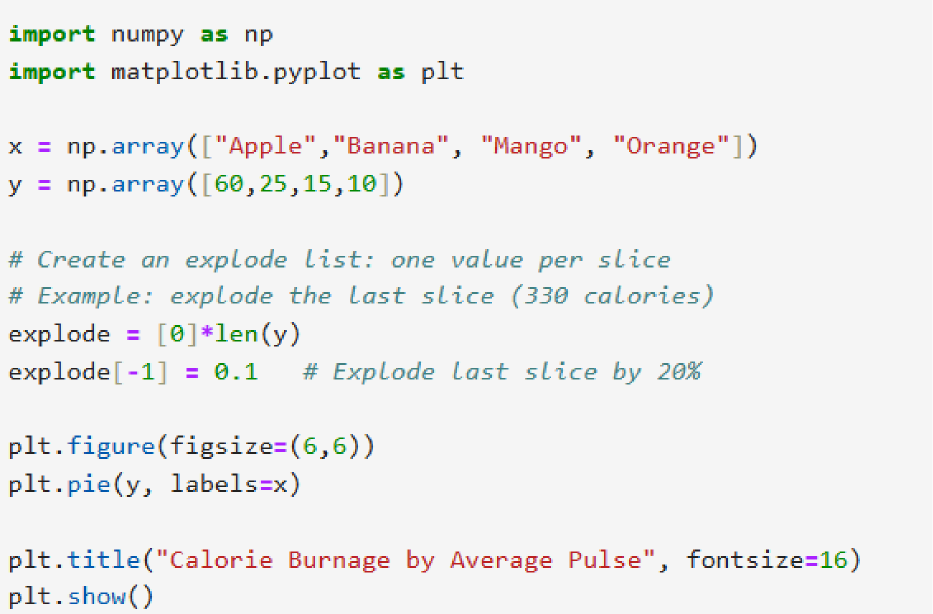


```
import numpy as np
import matplotlib.pyplot as plt

# Data
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

# Plot
plt.figure(figsize=(6,4))
plt.bar(x, y, color="skyblue", edgecolor="black")

# Titles and labels
plt.title("Sports Watch Data", fontsize=20)
plt.xlabel("Average Pulse", fontsize=14)
plt.ylabel("Calorie Burnage", fontsize=14)

# Show plot
plt.show()
```

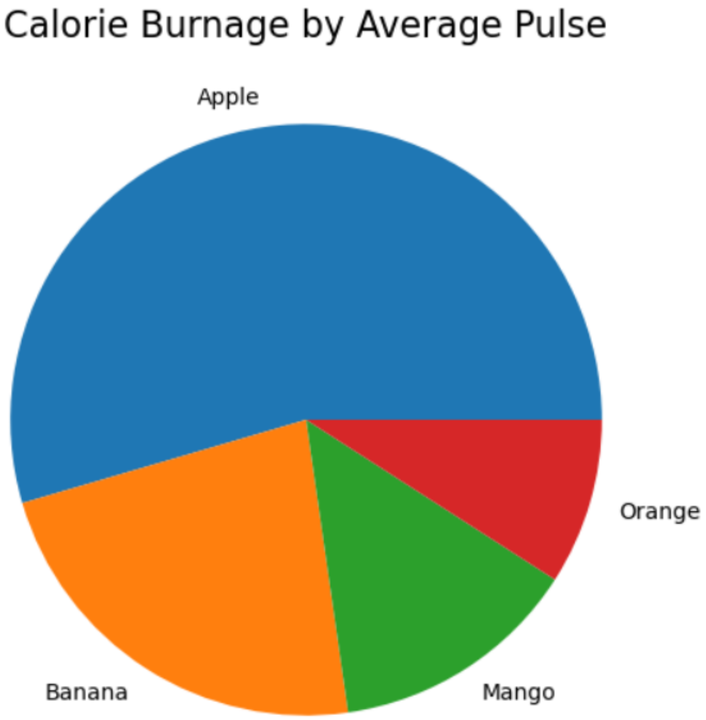


Python Library: Data Visualization

Pie Plot : **A pie plot (or pie chart) is a circular chart divided into slices, where each slice represents a proportion of the whole. It's mainly used to show the percentage or relative contribution of categories within a dataset.**

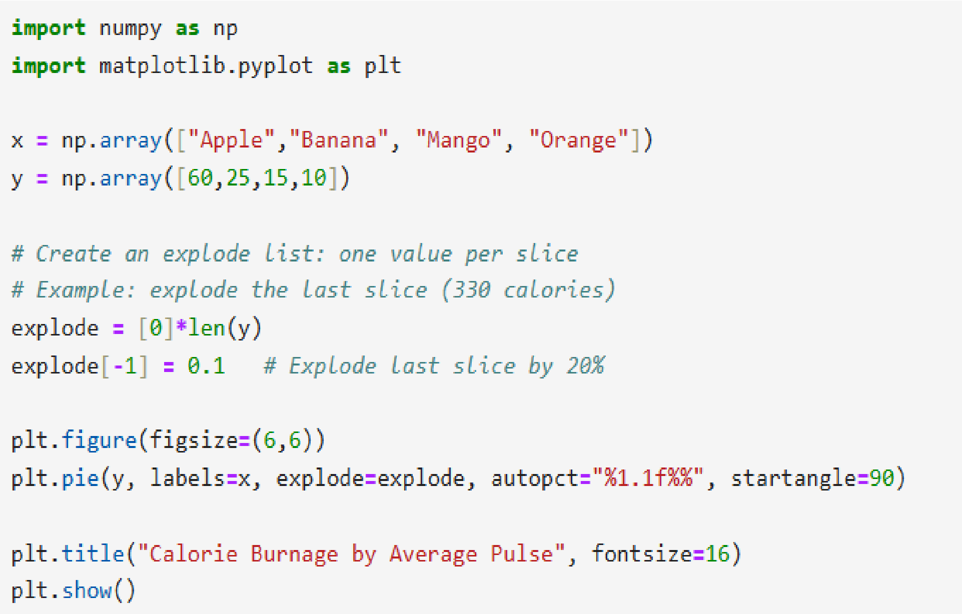


```
import numpy as np
import matplotlib.pyplot as plt

x = np.array(["Apple","Banana", "Mango", "Orange"])
y = np.array([60,25,15,10])

# Create an explode list: one value per slice
# Example: explode the last slice (330 calories)
explode = [0]*len(y)
explode[-1] = 0.1   # Explode last slice by 20%

plt.figure(figsize=(6,6))
plt.pie(y, labels=x)

plt.title("Calorie Burnage by Average Pulse", fontsize=16)
plt.show()
```

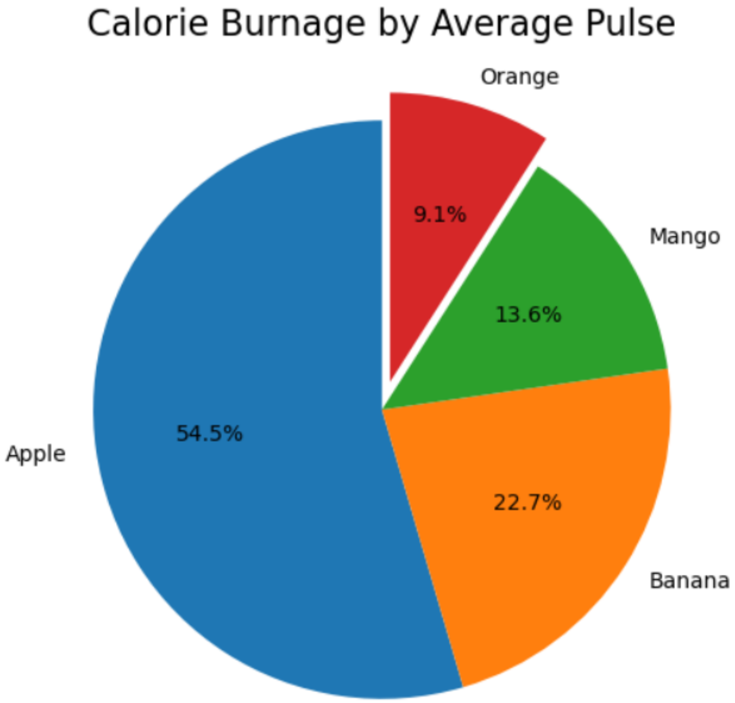


Python Library: Data Visualization

Pie Plot

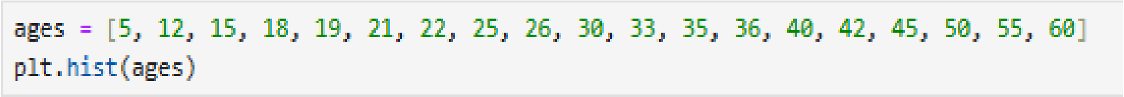


```
import numpy as np
import matplotlib.pyplot as plt

x = np.array(["Apple","Banana", "Mango", "Orange"])
y = np.array([60,25,15,10])

# Create an explode list: one value per slice
# Example: explode the last slice (330 calories)
explode = [0]*len(y)
explode[-1] = 0.1   # Explode last slice by 20%

plt.figure(figsize=(6,6))
plt.pie(y, labels=x, explode=explode, autopct="%1.1f%%", startangle=90)

plt.title("Calorie Burnage by Average Pulse", fontsize=16)
plt.show()
```

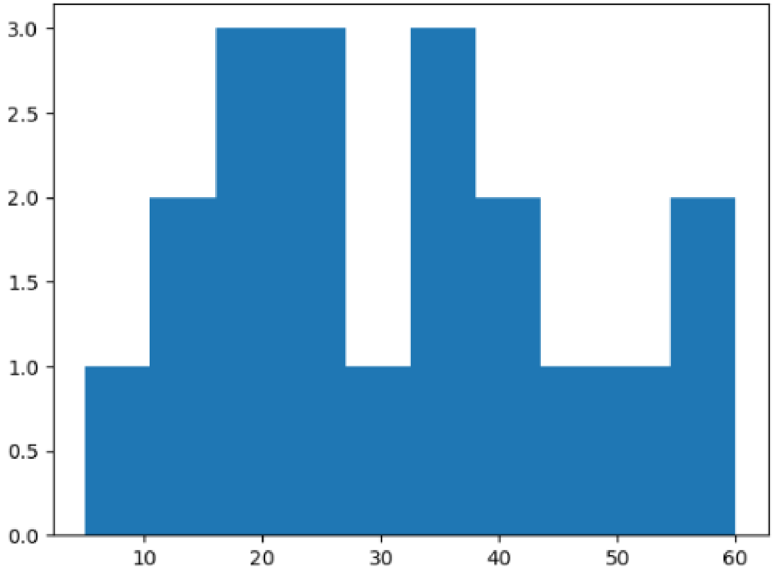


Python Library: Data Visualization

Histogram Plot: **A histogram plot is a type of chart that shows the distribution of numerical data by dividing the values into intervals (called bins) and counting how many values fall into each bin.**

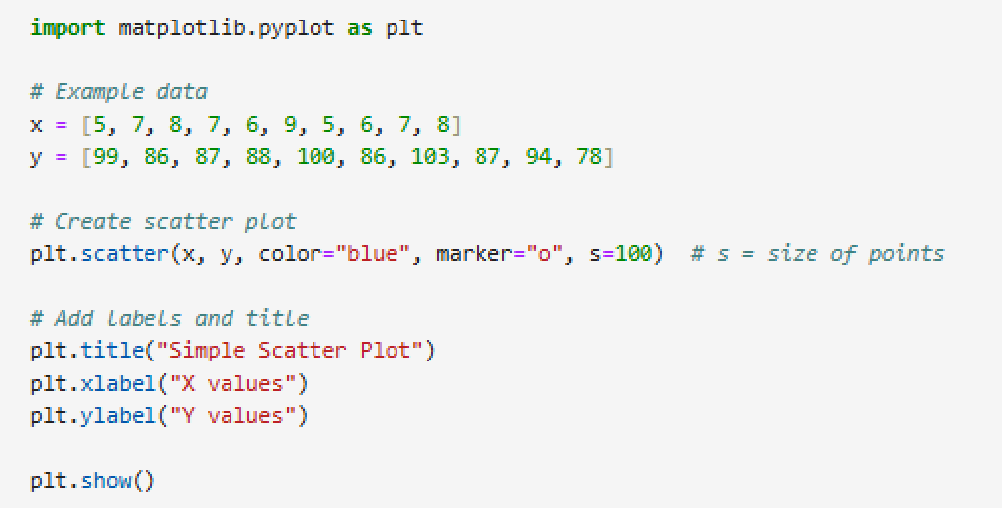


```
ages = [5, 12, 15, 18, 19, 21, 22, 25, 26, 30, 33, 35, 36, 40, 42, 45, 50, 55, 60]
plt.hist(ages)
```

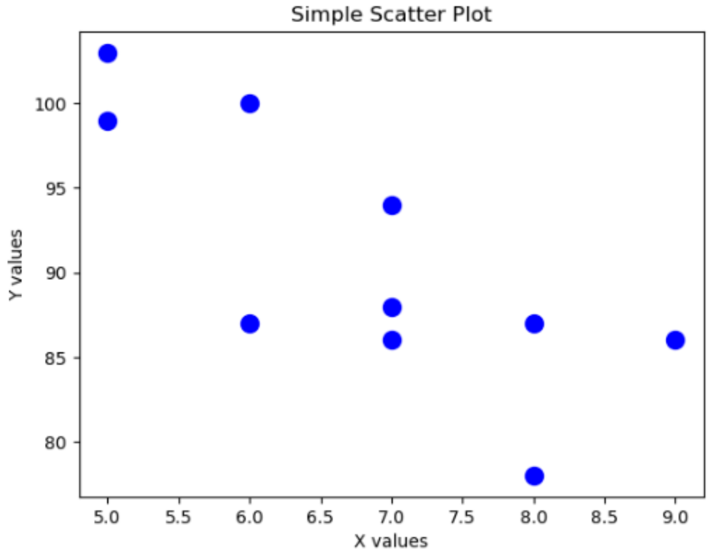


Python Library: Data Visualization

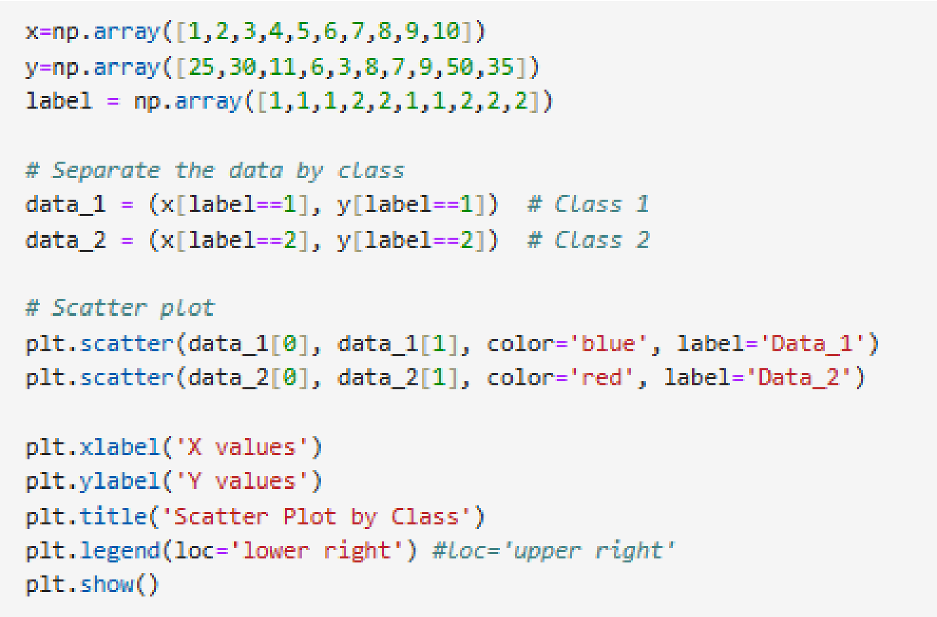
Scatter Plot: **A scatter plot is a type of plot that shows the relationship between two numerical variables using points on a 2D plane. Each point's position represents one observation.**



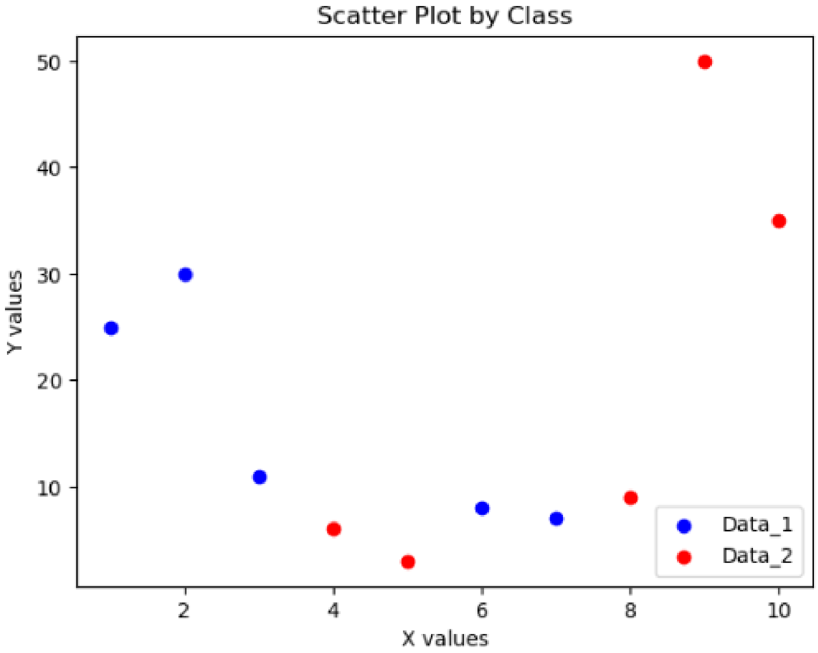
```
import matplotlib.pyplot as plt

# Example data
x = [5, 7, 8, 7, 6, 9, 5, 6, 7, 8]
y = [99, 86, 87, 88, 100, 86, 103, 87, 94, 78]

# Create scatter plot
plt.scatter(x, y, color="blue", marker="o", s=100)  # s = size of points

# Add labels and title
plt.title("Simple Scatter Plot")
plt.xlabel("X values")
plt.ylabel("Y values")

plt.show()
```



Multi-class Data Plot

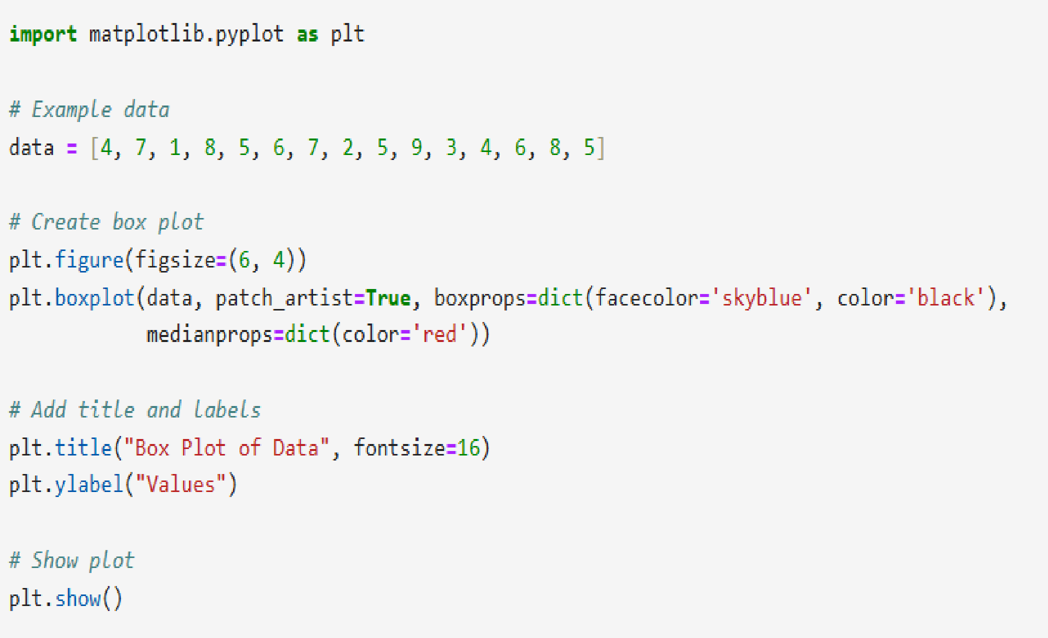


```python
x=np.array([1,2,3,4,5,6,7,8,9,10])
y=np.array([25,30,11,6,3,8,7,9,50,35])
label = np.array([1,1,1,2,2,1,1,2,2,2])

# Separate the data by class
data_1 = (x[label==1], y[label==1])  # Class 1
data_2 = (x[label==2], y[label==2])  # Class 2

# Scatter plot
plt.scatter(data_1[0], data_1[1], color='blue', label='Data_1')
plt.scatter(data_2[0], data_2[1], color='red', label='Data_2')

plt.xlabel('X values')
plt.ylabel('Y values')
plt.title('Scatter Plot by Class')
plt.legend(loc='lower right') #loc='upper right'
plt.show()
```

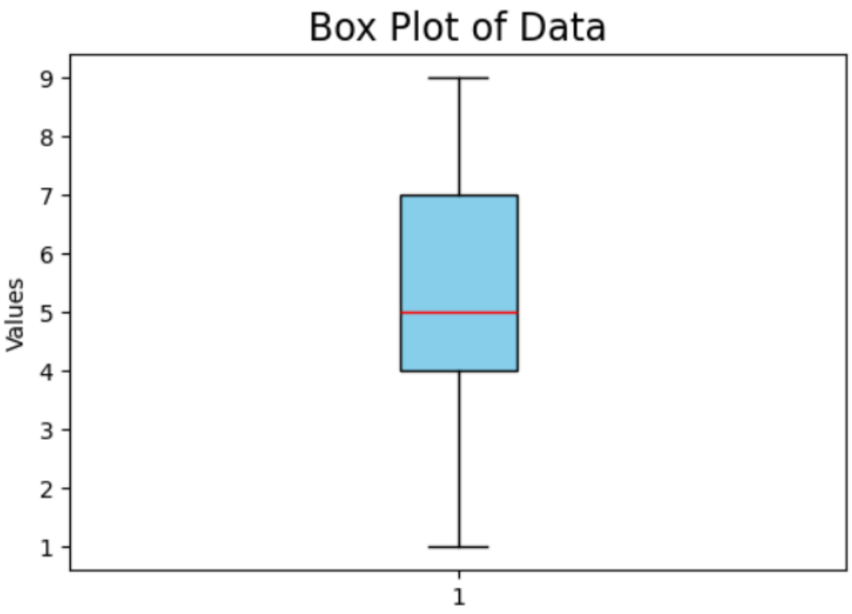


Python Library: Data Visualization

Box Plot : **A box plot (or box-and-whisker plot) is a graphical representation of a dataset that displays its median, quartiles, and potential outliers, summarizing the distribution and spread of the data.**



```python
import matplotlib.pyplot as plt

# Example data
data = [4, 7, 1, 8, 5, 6, 7, 2, 5, 9, 3, 4, 6, 8, 5]

# Create box plot
plt.figure(figsize=(6, 4))
plt.boxplot(data, patch_artist=True, boxprops=dict(facecolor='skyblue', color='black'),
            medianprops=dict(color='red'))

# Add title and labels
plt.title("Box Plot of Data", fontsize=16)
plt.ylabel("Values")

# Show plot
plt.show()
```



Python Library: Data Visualization

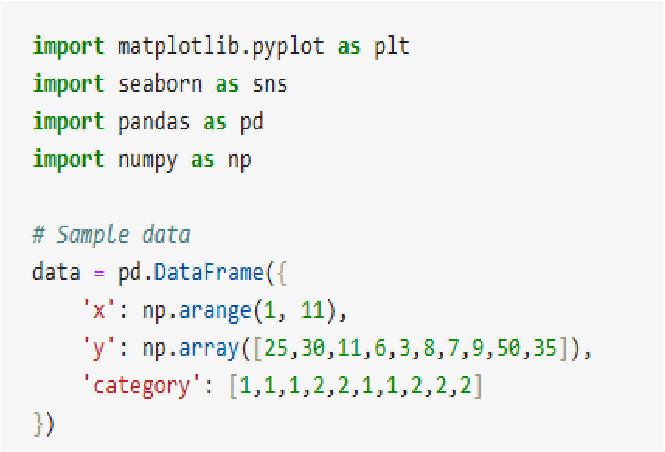
- **Seaborn** is a Python data visualization library built on top of Matplotlib.
- Its main purpose is to create statistically informative and visually attractive plots with minimal code.
- It works seamlessly with Pandas Dataframes, allowing you to plot columns directly without converting them to arrays.
- It includes built-in themes, color palettes, and support for complex visualizations like distributions, categorical plots, heatmaps, and regression plots.

Differences Between Seaborn and Matplotlib

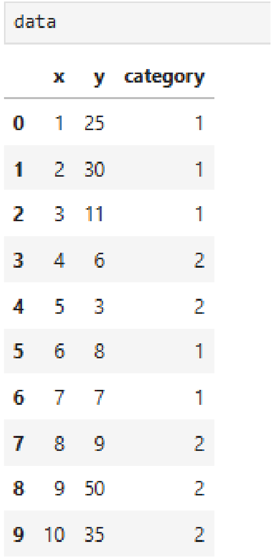
Feature	Matplotlib	Seaborn
Level	Low-level plotting library	High-level interface on top of Matplotlib
Syntax	More lines of code for styling & color	Cleaner, concise syntax for common plots
Data Handling	Works with lists, arrays	Works natively with Pandas DataFrames
Statistical Plots	Basic (line, bar, scatter, histogram)	Built-in plots for distributions, categories, regressions, heatmaps, violin/box plots
Styling & Aesthetics	Manual customization required	Attractive default themes, grids, and palettes

Python Library: Data Visualization

Scatter Plot

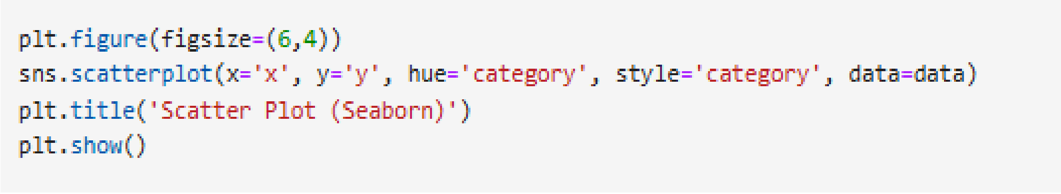


```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

# Sample data
data = pd.DataFrame({
    'x': np.arange(1, 11),
    'y': np.array([25,30,11,6,3,8,7,9,50,35]),
    'category': [1,1,1,2,2,1,1,2,2,2]
})
```

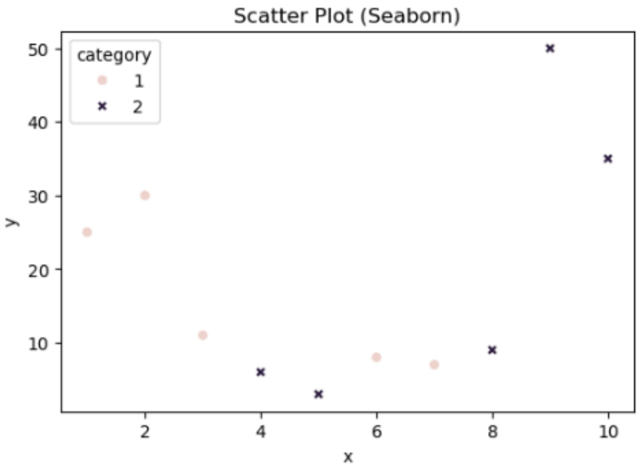


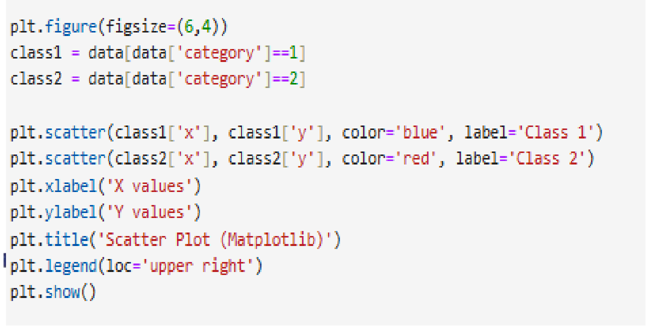
```
data
```

	x	y	category
0	1	25	1
1	2	30	1
2	3	11	1
3	4	6	2
4	5	3	2
5	6	8	1
6	7	7	1
7	8	9	2
8	9	50	2
9	10	35	2

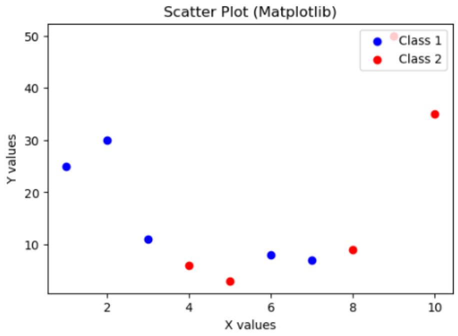


```
plt.figure(figsize=(6,4))
sns.scatterplot(x='x', y='y', hue='category', style='category', data=data)
plt.title('Scatter Plot (Seaborn)')
plt.show()
```





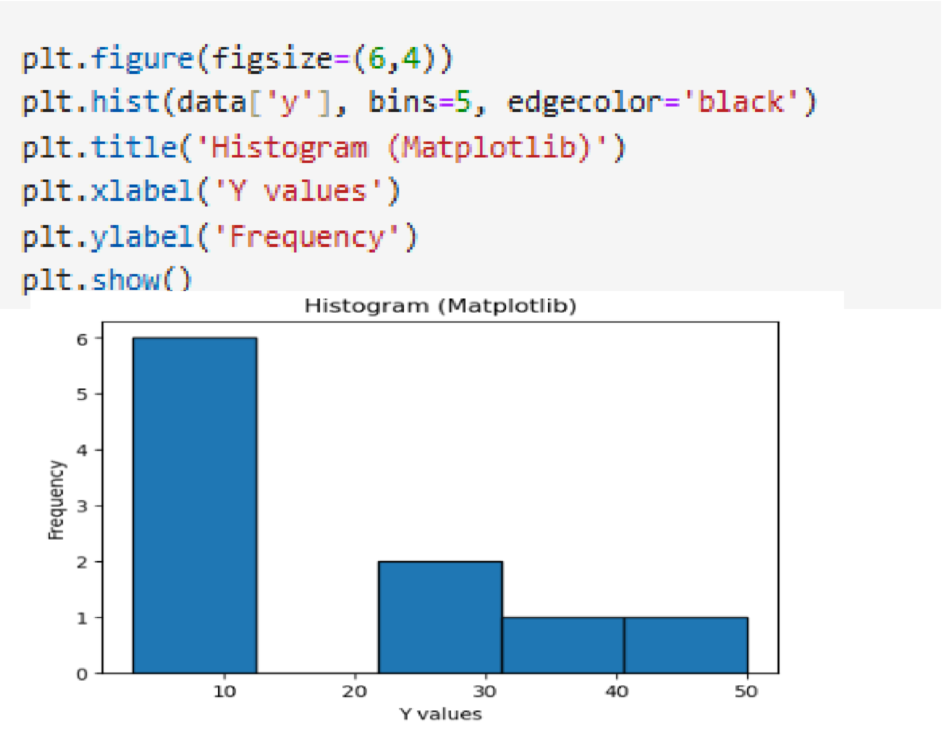
```
plt.figure(figsize=(6,4))
class1 = data[data['category']==1]
class2 = data[data['category']==2]

plt.scatter(class1['x'], class1['y'], color='blue', label='Class 1')
plt.scatter(class2['x'], class2['y'], color='red', label='Class 2')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.title('Scatter Plot (Matplotlib)')
plt.legend(loc='upper right')
plt.show()
```



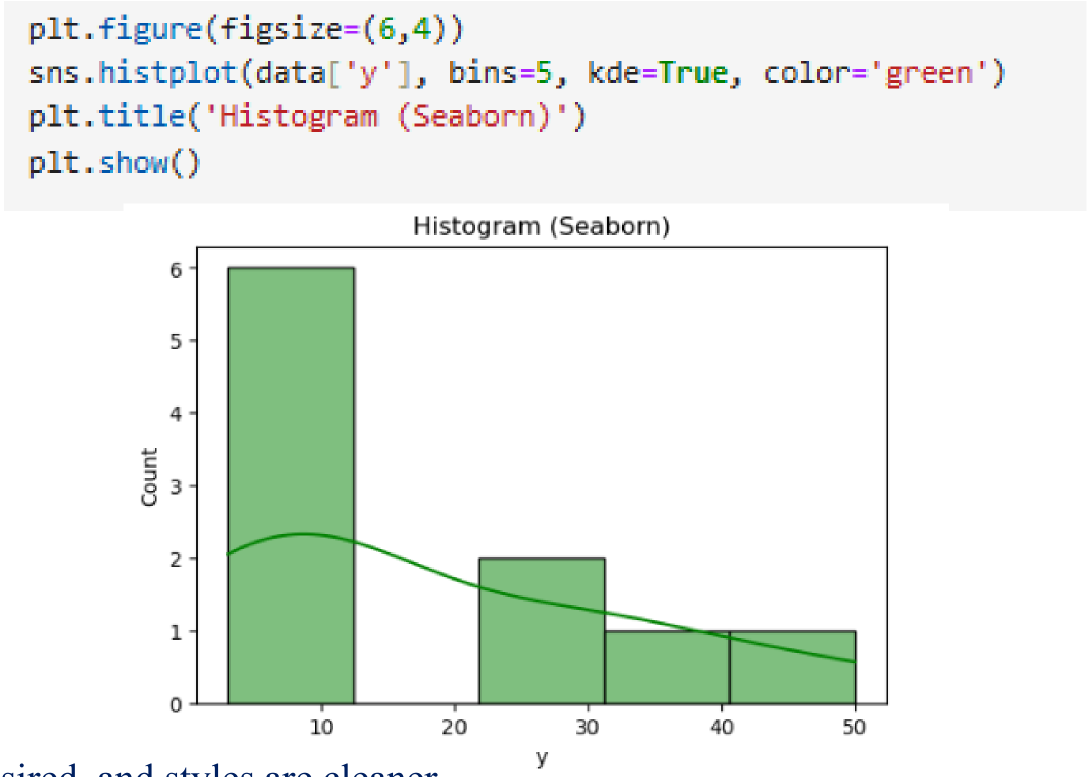
- hue for different colors
- style for different markers

Python Library: Data Visualization

Histogram Plot



```
plt.figure(figsize=(6,4))
plt.hist(data['y'], bins=5, edgecolor='black')
plt.title('Histogram (Matplotlib)')
plt.xlabel('Y values')
plt.ylabel('Frequency')
plt.show()
```



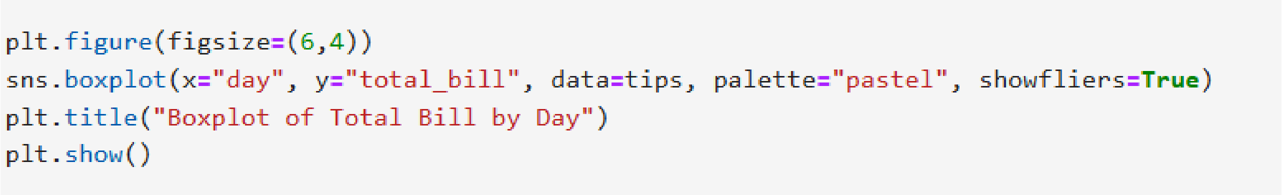
```
plt.figure(figsize=(6,4))
sns.histplot(data['y'], bins=5, kde=True, color='green')
plt.title('Histogram (Seaborn)')
plt.show()
```

- Seaborn adds a KDE curve automatically if desired, and styles are cleaner.
- Adds a Kernel Density Estimate (KDE) line over the histogram, which smooths the distribution curve.
- Divides the range of data into 5 intervals (bars).

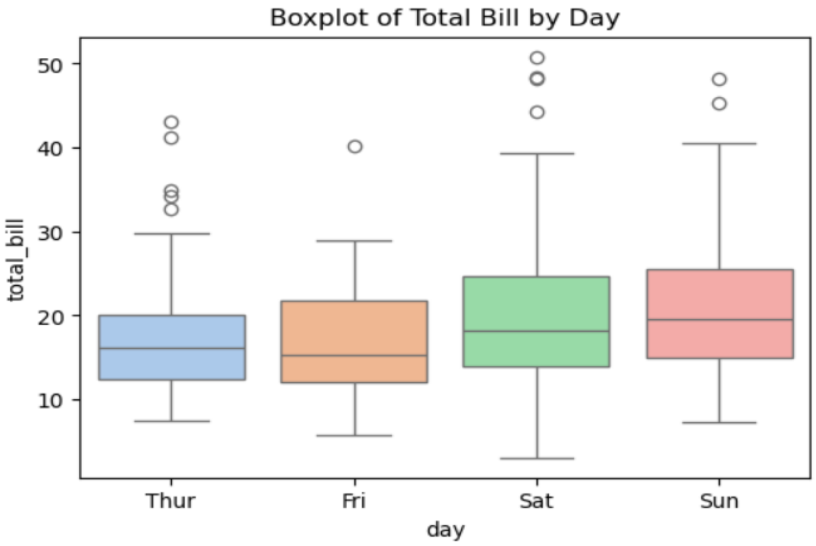
Dhirubhai Ambani University
Formerly known as DA-IICT

Python Library: Data Visualization

Box Plot using seaborn: Distribution of dataset

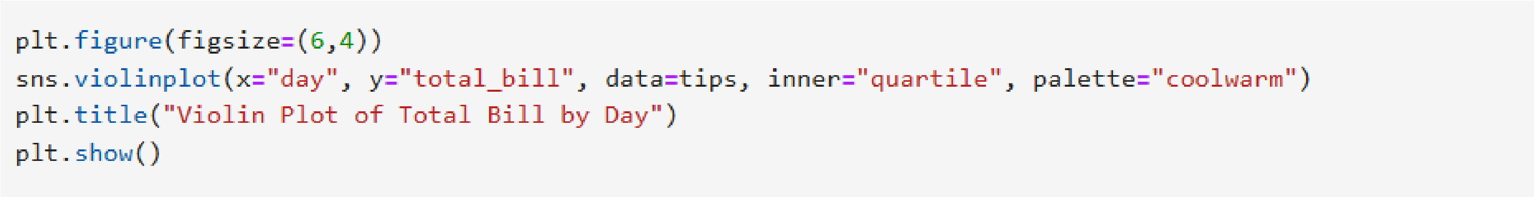


```
plt.figure(figsize=(6,4))
sns.boxplot(x="day", y="total_bill", data=tips, palette="pastel", showfliers=True)
plt.title("Boxplot of Total Bill by Day")
plt.show()
```

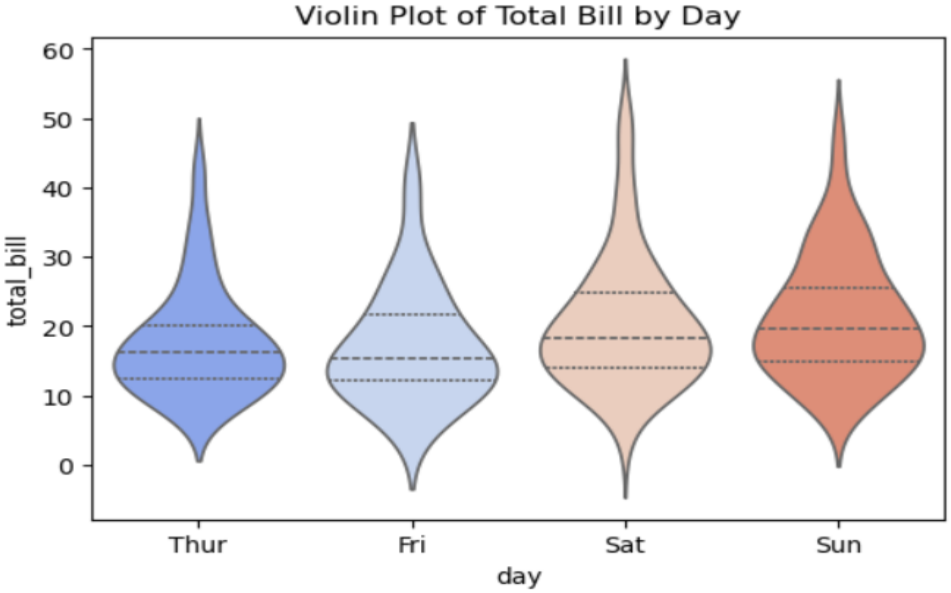


Python Library: Data Visualization

Violin Plot using seaborn : A violin plot is a statistical plot that combines a box plot with a kernel density estimate (KDE), showing both the summary statistics (median, quartiles) and the full distribution shape of the data.

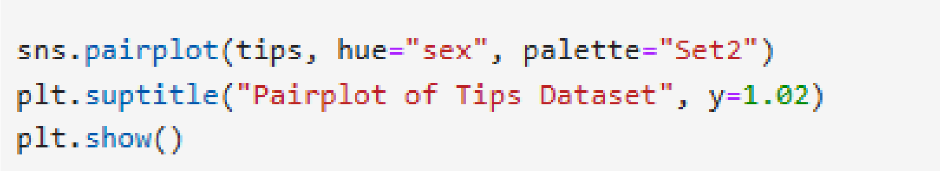


```
plt.figure(figsize=(6,4))
sns.violinplot(x="day", y="total_bill", data=tips, inner="quartile", palette="coolwarm")
plt.title("Violin Plot of Total Bill by Day")
plt.show()
```

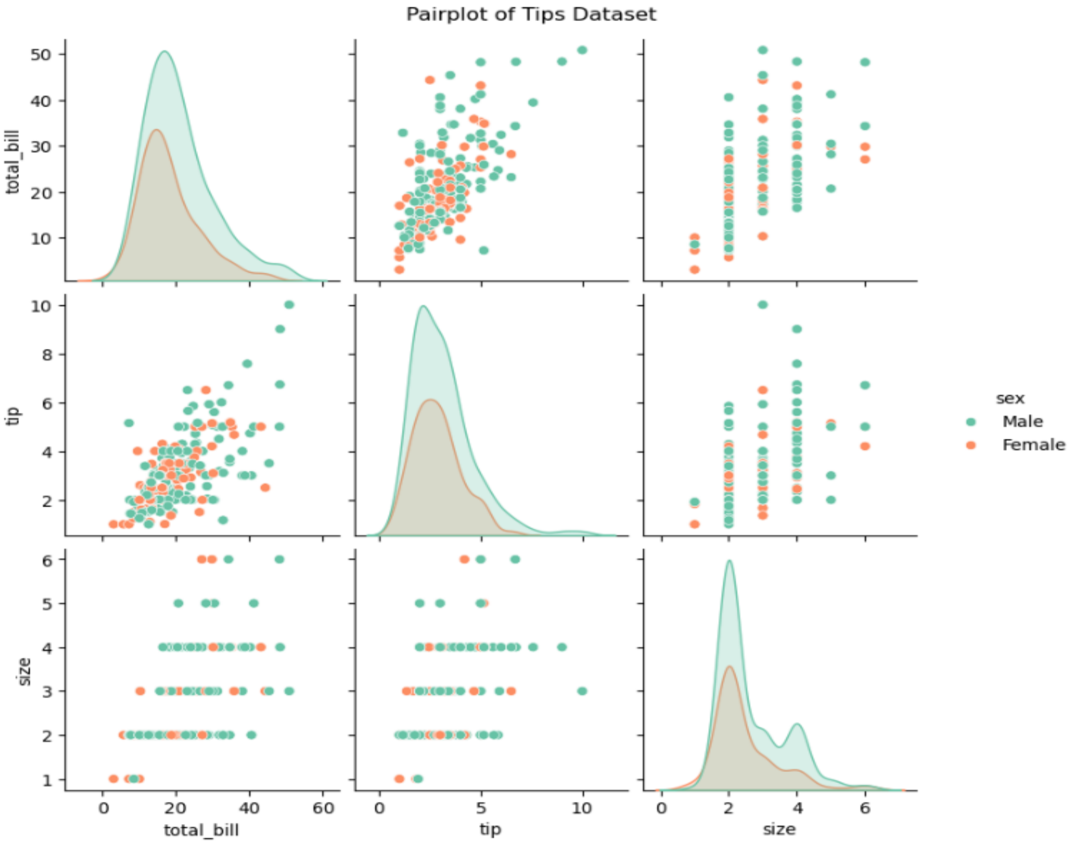


Python Library: Data Visualization

Pair Plot using seaborn



```
sns.pairplot(tips, hue="sex", palette="Set2")
plt.suptitle("Pairplot of Tips Dataset", y=1.02)
plt.show()
```



Dhirubhai Ambani University
Formerly known as
DA-IICT

Python Library: Data Visualization

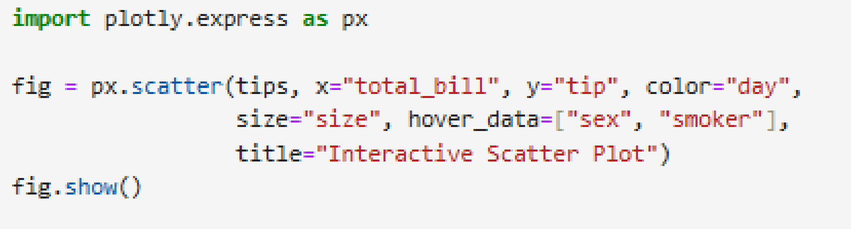
- **Plotly** is an interactive Python data visualization library.
- It allows to create interactive plots that can be zoomed, hovered over or updated dynamically.
- Supports a wide variety of plots: line, scatter, bar, histogram, box, violin, heatmap, 3D plots, maps and dashboards.
- Integrated with Pandas, Numpy, Dash and Web applications.

Differences Between Seaborn and Matplotlib

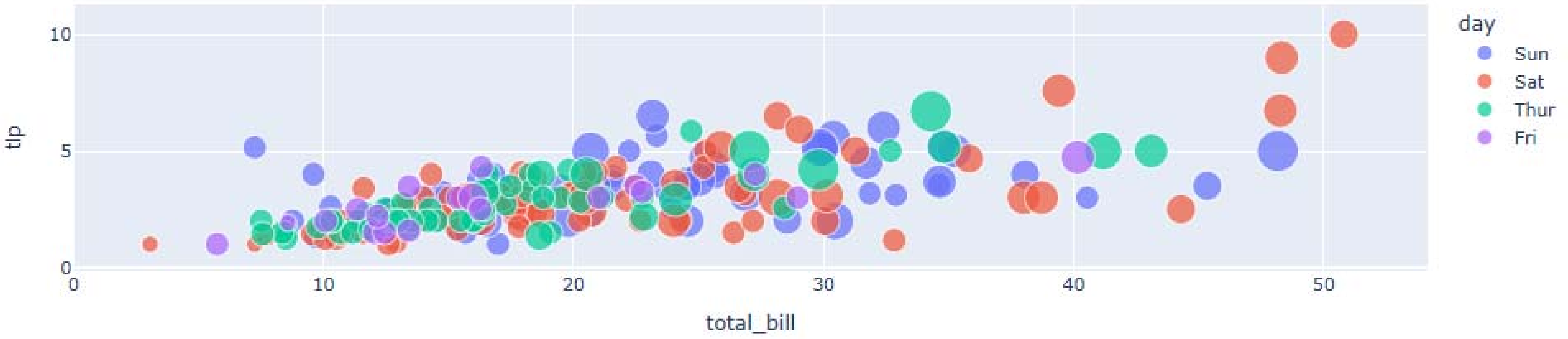
Feature	Matplotlib	Seaborn	Plotly
Interactivity	Static (basic interactivity via widgets)	Mostly static	Highly interactive (hover, zoom)
Ease of Use	Low-level, flexible	High-level, prettier defaults	High-level API, interactive by default
Data Handling	Arrays, lists	Pandas DataFrames	Pandas DataFrames, dictionaries, arrays
Statistical Plots	Basic	Built-in (box, violin, regression, KDE)	Limited built-in statistics, but can be combined with Pandas/NumPy
3D & Maps	Needs extra libraries	Very limited	Native support for 3D plots, choropleth maps, scatter maps
Styling	Manual	Attractive defaults	Customizable with themes, but interactive focus

Python Library: Data Visualization

Interactive Scatter plot : Plotly

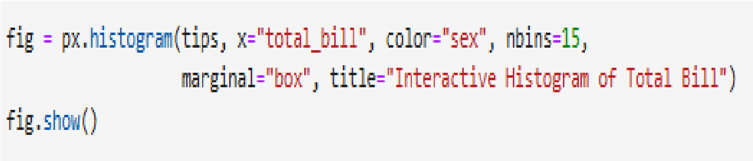


```
import plotly.express as px

fig = px.scatter(tips, x="total_bill", y="tip", color="day",
                 size="size", hover_data=["sex", "smoker"],
                 title="Interactive Scatter Plot")
fig.show()
```

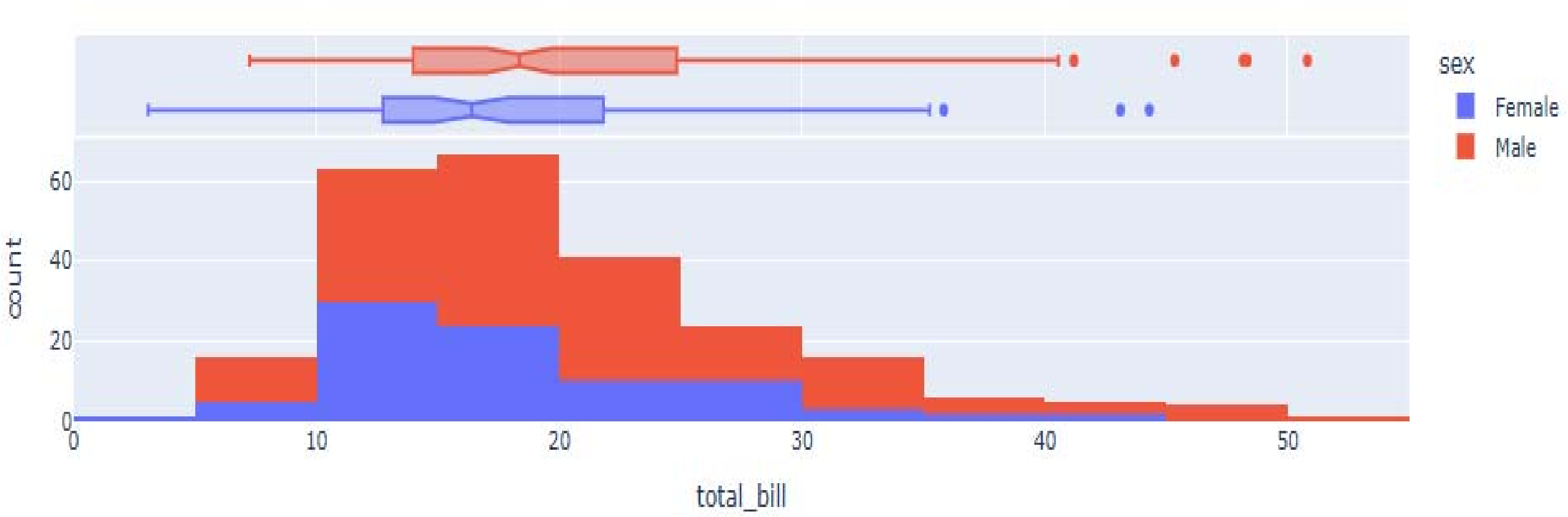


Python Library: Data Visualization

Interactive Histogram Plot



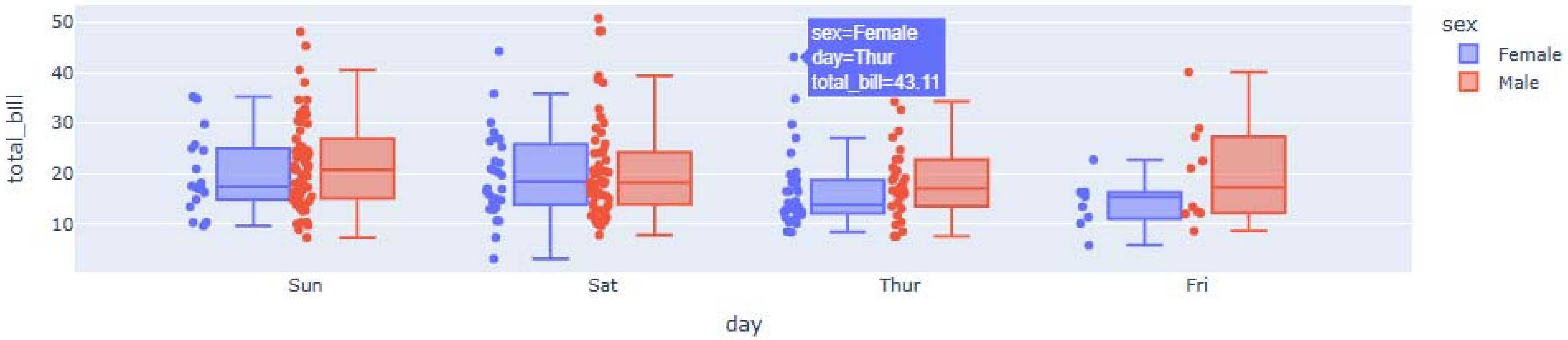
```
fig = px.histogram(tips, x="total_bill", color="sex", nbins=15,
                   marginal="box", title="Interactive Histogram of Total Bill")
fig.show()
```

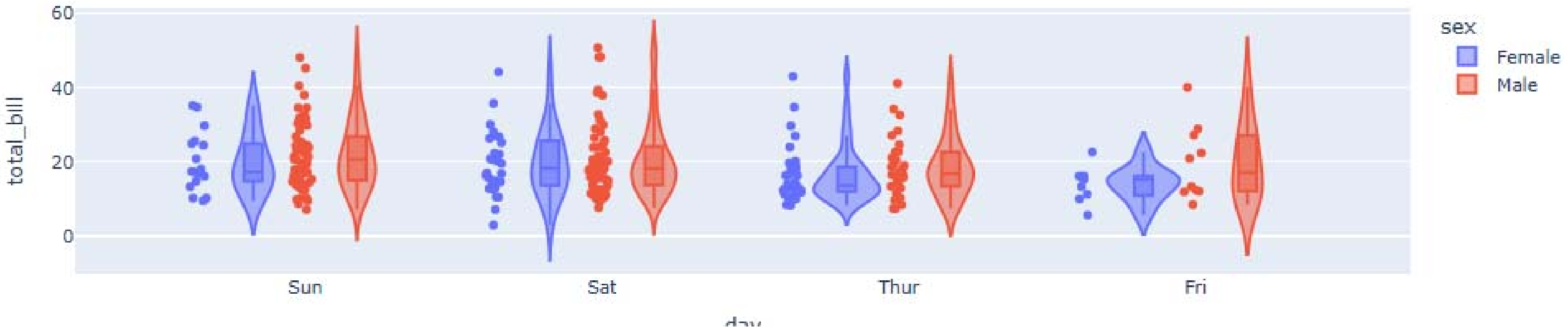


Python Library: Data Visualization

Interactive Box Plot and Violin Plot

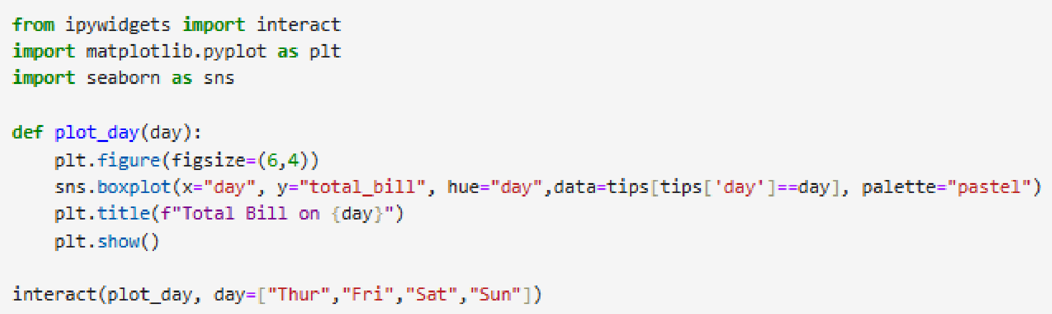
```
fig = px.box(tips, x="day", y="total_bill", color="sex", points="all", title="Interactive Box Plot")
fig.show()

fig2 = px.violin(tips, x="day", y="total_bill", color="sex", box=True, points="all", title="Interactive Violin Plot")
fig2.show()
```



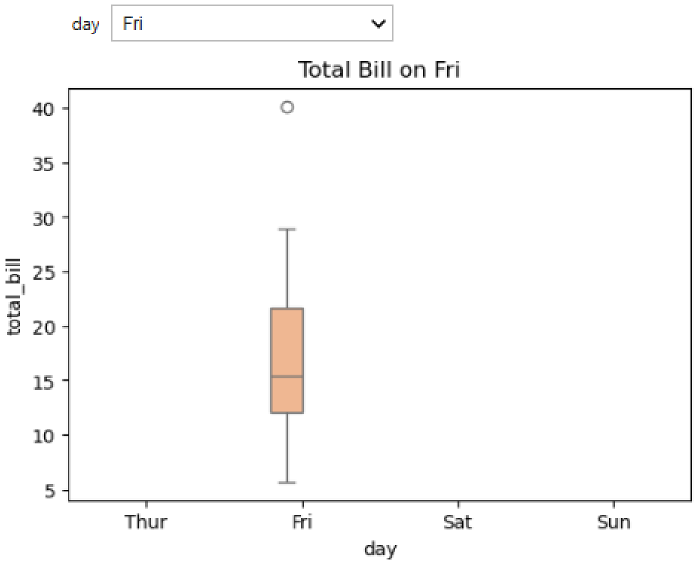


Python Library: Data Visualization

Dynamic Plot: ipywidgets



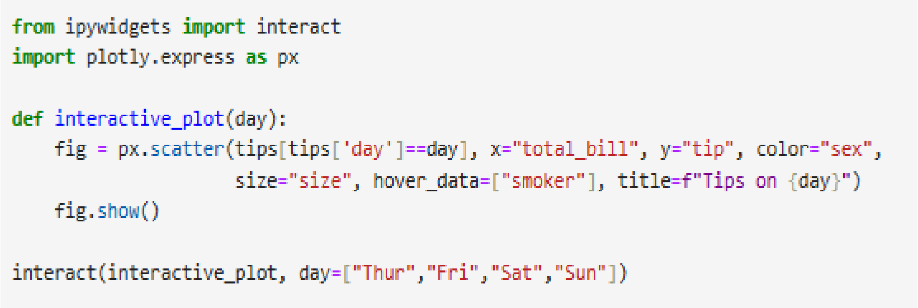
```
from ipywidgets import interact
import matplotlib.pyplot as plt
import seaborn as sns

def plot_day(day):
    plt.figure(figsize=(6,4))
    sns.boxplot(x="day", y="total_bill", hue="day",data=tips[tips['day']==day], palette="pastel")
    plt.title(f"Total Bill on {day}")
    plt.show()

interact(plot_day, day=["Thur","Fri","Sat","Sun"])
```



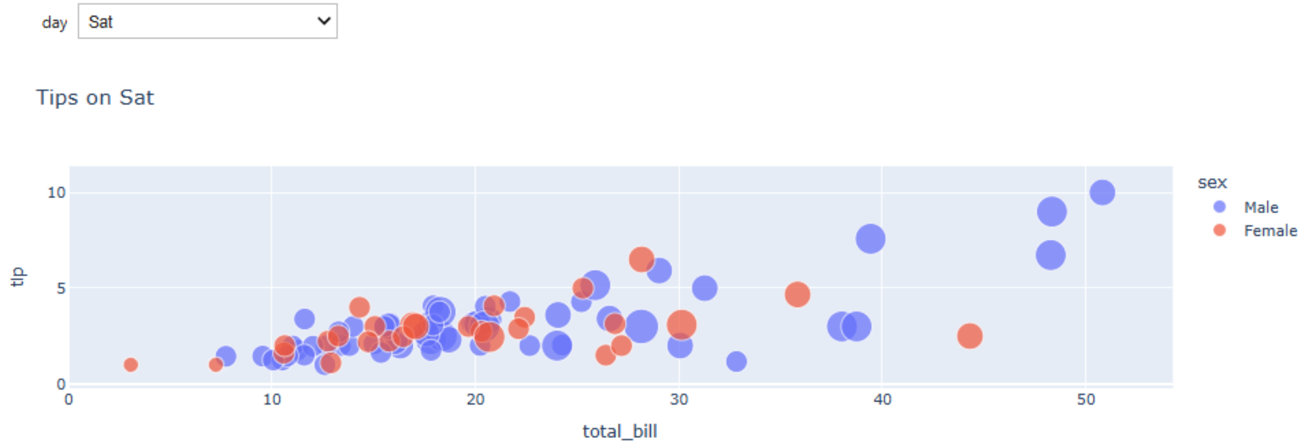
day Fri

Python Library: Data Visualization

Combining Plotly + ipywidgets



```
from ipywidgets import interact
import plotly.express as px

def interactive_plot(day):
    fig = px.scatter(tips[tips['day']==day], x="total_bill", y="tip", color="sex",
                     size="size", hover_data=["smoker"], title=f"Tips on {day}")
    fig.show()

interact(interactive_plot, day=["Thur","Fri","Sat","Sun"])
```



Dhirubhai Ambani University
Formerly known as
DA-IICT