

Name: Shreyans Tatiya
Batch: C5_3 **Roll No.:** 16010123325

Experiment / assignment / Tutorial

Grade: AA / AB / BB / BC / CC / CD / DD

Signature of the Staff In-charge with date

TITLE: Matplotlib library in Python

AIM: Write a program to explore the Matplotlib library

Expected OUTCOME of Experiment: To demonstrate Matplot library in python

Resource Needed: Python IDE

Theory:

What is Matplotlib?

1. Matplotlib

Matplotlib is a data visualization library and 2-D plotting library of Python. It was initially released in 2003 and it is the most popular and widely-used plotting library in the Python community. It comes with an interactive environment across multiple platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, etc. It can be used to embed plots into applications using various GUI toolkits like Tkinter, GTK+, wxPython, Qt, etc. So you can use Matplotlib to create plots, bar charts, pie charts, histograms, scatterplots, error charts, power spectra, stemplots, and whatever other visualization charts you want! The Pyplot module also provides a MATLAB-like interface that is just as versatile and useful as MATLAB while being free and open source.

2. Plotly

Plotly is a free open-source graphing library that can be used to form data visualizations. Plotly (plotly.py) is built on top of the Plotly JavaScript library (plotly.js) and can be used to create web-based data visualizations that can be displayed in Jupyter notebooks or web applications using Dash or saved as individual HTML files. Plotly provides more than 40 unique chart types like scatter plots, histograms, line charts, bar charts, pie charts, error bars, box plots, multiple axes, sparklines, dendrograms, 3-D charts, etc. Plotly also provides contour plots, which are not that common in other data visualization libraries. In addition to all this, Plotly can be used offline with no internet connection.

Plotting x and y points

The `plot()` function is used to draw points (markers) in a diagram.

By default, the `plot()` function draws a line from point to point.

Syntax:

`matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwargs)`

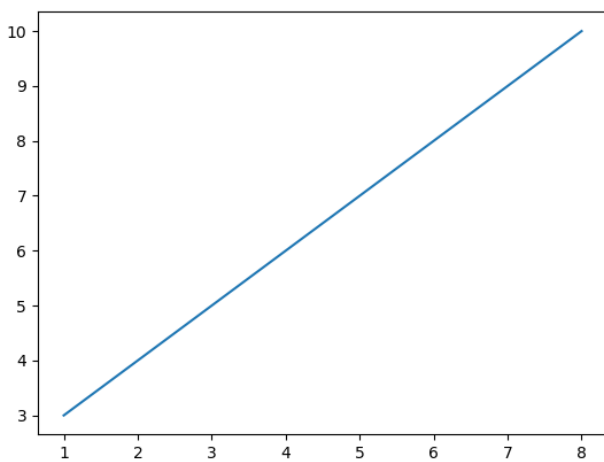
- `x, y`: These parameter are the horizontal and vertical coordinates of the data points. `x` values are optional.
- `fmt`: This parameter is an optional parameter and it contains the string value.
- `data`: This parameter is an optional parameter and it is an object with labelled data.

Returns:

This returns the following:

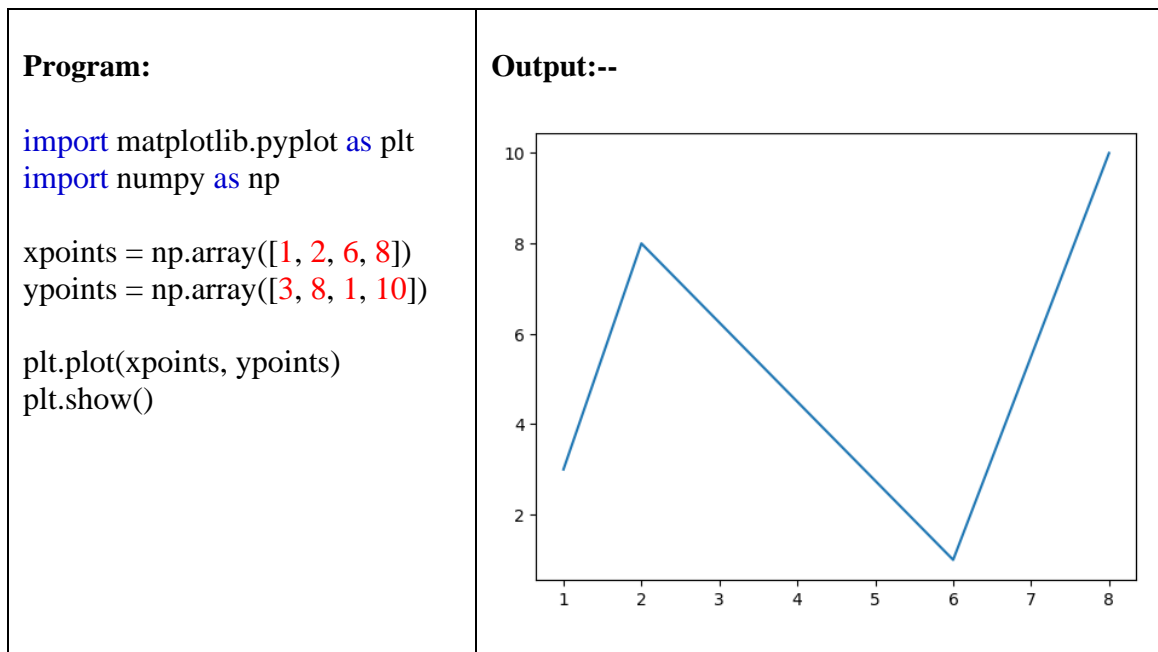
lines : This returns the list of Line2D objects representing the plotted data.

Example:-

Draw a line in a diagram from position (1, 3) to position (8, 10):	Output
<pre>import matplotlib.pyplot as plt import numpy as np xpoints = np.array([1, 8]) ypoints = np.array([3, 10]) plt.plot(xpoints, ypoints) plt.show()</pre>	

1) Multiple Points

You can plot as many points as you like, just make sure you have the same number of points in both axis.



2) Matplotlib Line

Linestyle:--- You can use the keyword argument **linestyle**, or shorter **ls**, to change the style of the plotted line:

Following are the linestyles available in *matplotlib*:

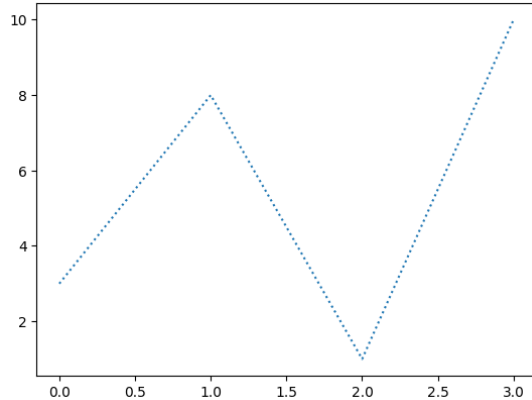
Using *linestyle* Argument:

- Solid
- Dashed
- Dotted
- Dashdot
- None

Syntax: plt.plot(xdata, ydata, linestyle='dotted')	
Program	Output:

Use a dotted line:

```
import matplotlib.pyplot as plt
import numpy as np
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, linestyle = 'dotted')
plt.show()
```



3)Matplotlib Labels and Title

a.Create Labels for a Plot

With Pyplot, you can use the `xlabel()` and `ylabel()` functions to set a label for the x- and y-axis.

The `xlabel()` function in pyplot module of matplotlib library is used to set the label for the x-axis.

Syntax: `matplotlib.pyplot.xlabel(xlabel, fontdict=None, labelpad=None, **kwargs)`

b. Create a Title for a Plot

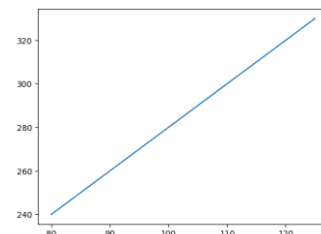
With Pyplot, you can use the `title()` function to set a title for the plot.

Program:--

```
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])
```

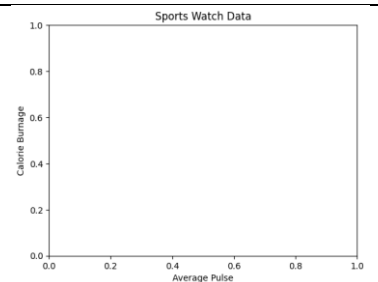
Output:--



```
plt.plot(x, y)
```

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

```
plt.show()
```



4) Matplotlib Scatter

Creating Scatter Plots

With Pyplot, you can use the `scatter()` function to draw a scatter plot.

The `scatter()` function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

Syntax:-- `matplotlib.pyplot.scatter(x_axis_data, y_axis_data, s=None, c=None, marker=None, cmap=None, vmin=None, vmax=None, alpha=None, linewidths=None, edgecolors=None)`

- **x_axis_data**- An array containing x-axis data
- **y_axis_data**- An array containing y-axis data
- **s**- marker size (can be scalar or array of size equal to size of x or y)
- **c**- color of sequence of colors for markers
- **marker**- marker style
- **cmap**- cmap name
- **linewidths**- width of marker border
- **edgecolor**- marker border color
- **alpha**- blending value, between 0 (transparent) and 1 (opaque)

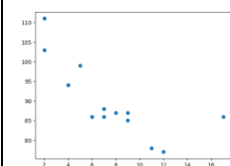
Except `x_axis_data` and `y_axis_data` all other parameters are optional and their default value is None. Below are the scatter plot examples with various parameters.

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

```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
```

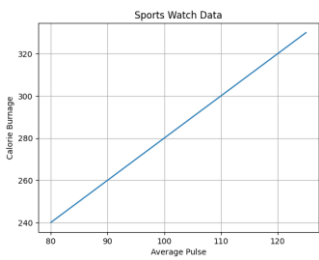
```
plt.scatter(x, y)
plt.show()
```

Output:--



Add Grid Lines to a Plot

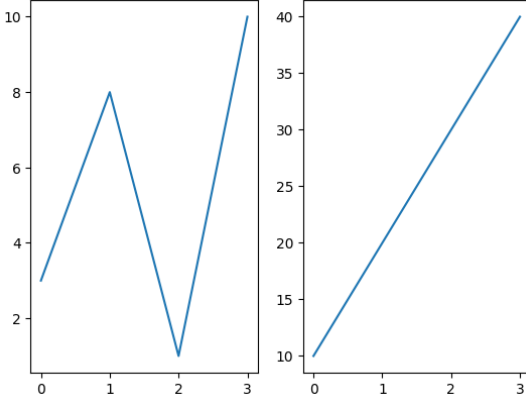
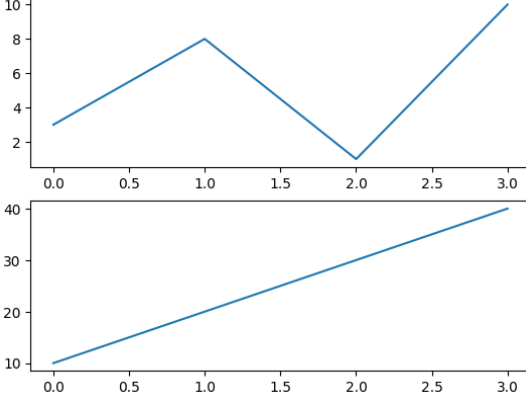
With Pyplot, you can use the `grid()` function to add grid lines to the plot.

<pre>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.title("Sports Watch Data") plt.xlabel("Average Pulse") plt.ylabel("Calorie Burnage") plt.plot(x, y) plt.grid() plt.show()</pre>	<p>Output:</p> 
--	---

5) Display Multiple Plots

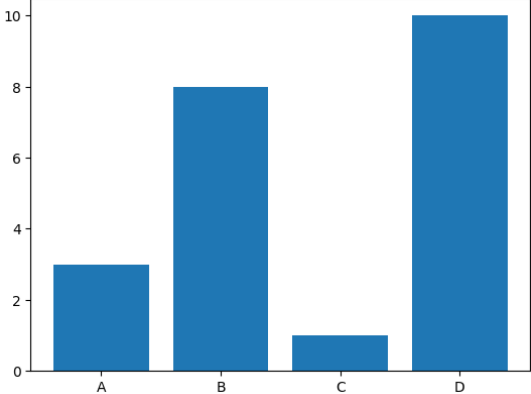
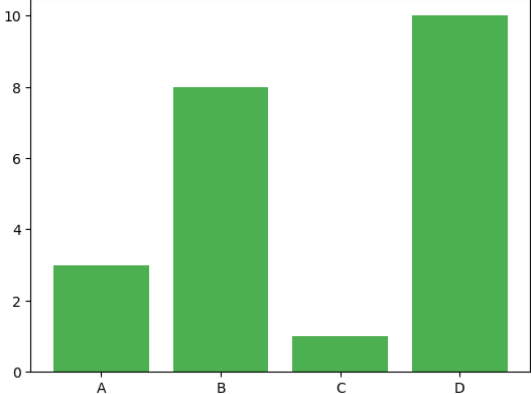
With the `subplot()` function you can draw multiple plots in one figure.

<pre>subplot(nrows, ncols, index, **kwargs)</pre> <p>The layout is organized in rows and columns, which are represented by the <i>first</i> and <i>second</i> argument.</p> <p>The third argument represents the index of the current plot.</p>

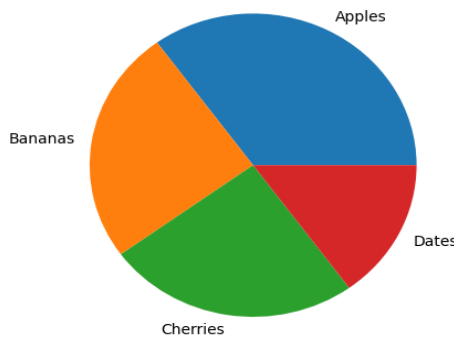
<p>Program:-</p> <pre>import matplotlib.pyplot as plt import numpy as np #plot 1: x = np.array([0, 1, 2, 3]) y = np.array([3, 8, 1, 10]) plt.subplot(1, 2, 1) plt.plot(x,y) #plot 2: x = np.array([0, 1, 2, 3]) y = np.array([10, 20, 30, 40]) plt.subplot(1, 2, 2) plt.plot(x,y) plt.show()</pre>	<p>Output:--</p> 
<pre>import matplotlib.pyplot as plt import numpy as np #plot 1: x = np.array([0, 1, 2, 3]) y = np.array([3, 8, 1, 10]) plt.subplot(2, 1, 1) plt.plot(x,y) #plot 2: x = np.array([0, 1, 2, 3]) y = np.array([10, 20, 30, 40]) plt.subplot(2, 1, 2) plt.plot(x,y) plt.show()</pre>	<p>Output:--</p> 

6) Creating Bars

With Pyplot, you can use the `bar()` function to draw bar graphs.

<pre>import matplotlib.pyplot as plt import numpy as np x = np.array(["A", "B", "C", "D"]) y = np.array([3, 8, 1, 10]) plt.bar(x,y) plt.show()</pre>	<p>Output:--</p> 
<pre>import matplotlib.pyplot as plt import numpy as np x = np.array(["A", "B", "C", "D"]) y = np.array([3, 8, 1, 10]) plt.bar(x, y, color = "#4CAF50") plt.show()</pre>	

7) Creating Pie Chart with Labels:

<pre>import matplotlib.pyplot as plt import numpy as np y = np.array([35, 25, 25, 15]) mylabels = ["Apples", "Bananas", "Cherries", "Dates"] plt.pie(y, labels = mylabels) plt.show()</pre>	<p>Output:</p> 
--	--

Problem Definition:

Note:-- All plot should be labelled on X-axis and Y-axis with Grid for each program.

1. Write a Python program to draw a line using given axis values with suitable label in the x axis, y axis and a title.

2. a) Write a Python programming to display a bar chart of the popularity of programming Languages. Also draw Pie chart for **popularity** Data values.

Sample data:

Programming languages: Java, Python, PHP, JavaScript, C#, C++

Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7

b) Write a Python program to display a horizontal bar chart of the popularity of programming Languages. **Hint: use the `barh()` function**

3) Prepare a dataset using list as **Weight** and **height** parameters for your batch students and draw a scatter plot with appropriate label and title.

Post Lab Questions:--

1) Considering datasets of your choice, create and explain the utility of following charts:

1) Swarn chart	6) Regression plot
2) Pair chart	7) Count plot
3) Pair grid	8) Bar plot
4) Facet Grid	9) Violin plot
5) Scatter plot	10) Heat map

2) What is the Seaborn library? What are Different categories of plot in Seaborn.

Books/ Journals/ Websites referred:

1. [Matplotlib Plotting \(w3schools.com\)](https://www.w3schools.com/matplotlib/) – Reference website.
2. Reema Thareja, Python Programming: Using Problem Solving Approach, Oxford University Press, First Edition 2017, India
3. Sheetal Taneja and Naveen Kumar, Python Programming: A modular Approach, Pearson India, Second Edition 2018, India

Implementation details:

1.

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4]
y = [0, 2, 4, 6, 8]

fig, ax = plt.subplots()

ax.plot(x, y)

ax.set_title('Line Plot Example')
ax.set_xlabel('X Axis Label')
ax.set_ylabel('Y Axis Label')

plt.show()
```

2. a)

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

y = np.array([22.2, 17.6, 8.8, 8, 7.7, 6.7 ])
lang = ["Java", "Python", "PHP", "JavaScript", "C#", "C++" ]

plt.pie(y, labels = lang)
plt.show()
```

2. b)

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

y = np.array([22.2, 17.6, 8.8, 8, 7.7, 6.7])

lang = ["Java", "Python", "PHP", "JavaScript", "C#", "C++"]

fig, ax = plt.subplots()

ax.barh(lang, y)

ax.set_title('Popularity of Programming Languages')
ax.set_xlabel('Popularity')
ax.set_ylabel('Language')

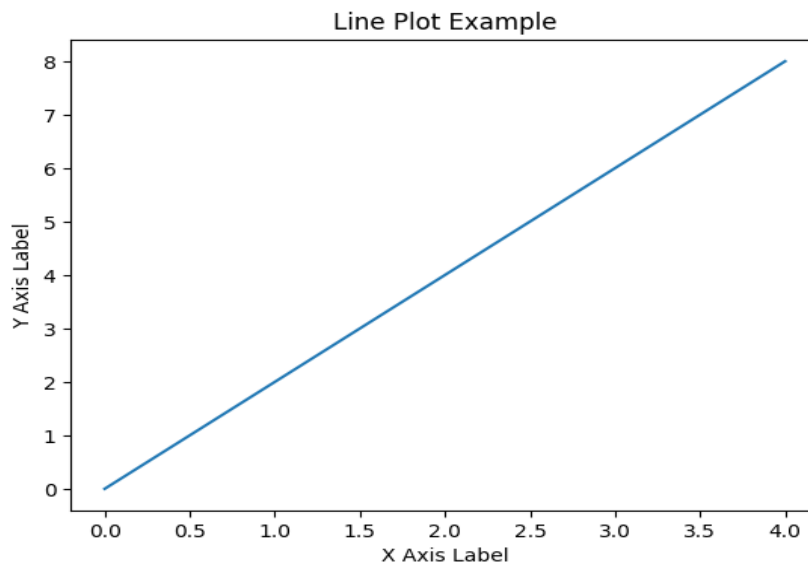
plt.show()
```

3)

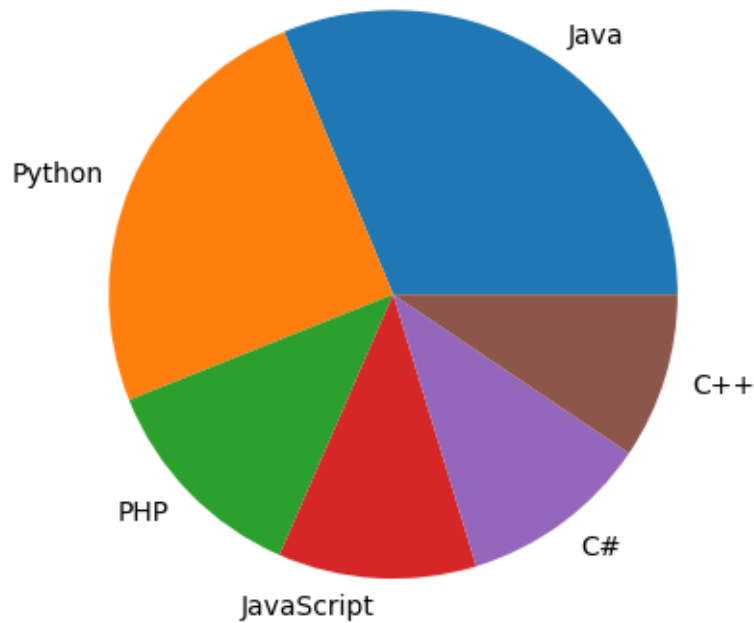
```
students = [{'Weight': 56, 'Height': 165},  
            {'Weight': 75, 'Height': 175},  
            {'Weight': 62, 'Height': 160}]  
  
weights = [student['Weight'] for student in students]  
heights = [student['Height'] for student in students]  
  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.scatterplot(x=weights, y=heights)  
  
plt.xlabel('Weight')  
plt.ylabel('Height')  
plt.title('Scatter plot of Weight and Height')  
  
plt.show()
```

Output:

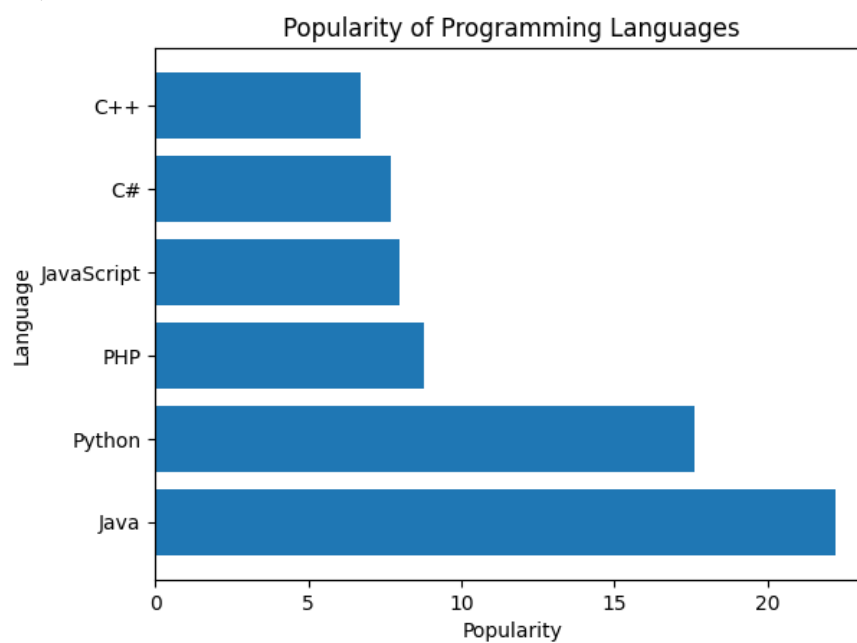
1.



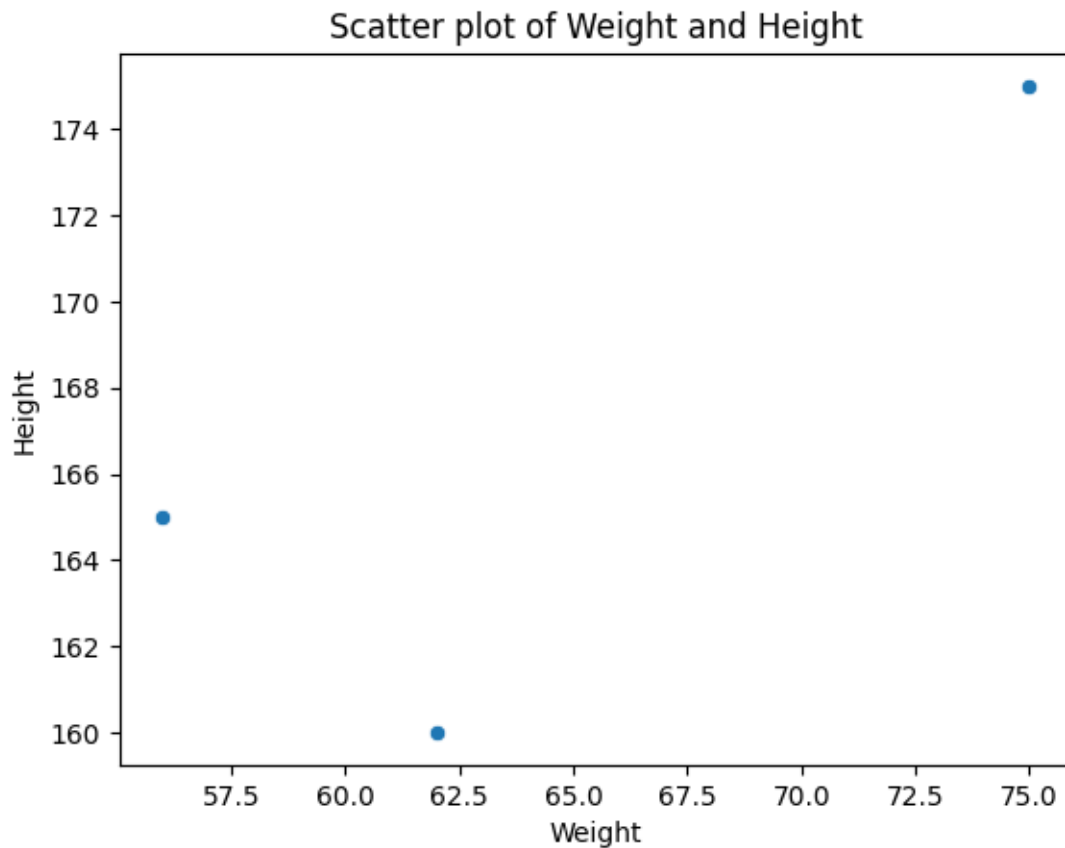
2.a)



2.b)



3)



Post Lab Questions:--

1)

Swarm Chart:

Utility:

Swarm charts are useful for visualizing the distribution of data points within categorical groups. They are similar to violin plots but offer a clearer representation of individual data points.

Example Dataset:

Student exam scores categorized by subject

Pair Chart:

Utility:

Pair charts effectively visualize the relationship between two variables by displaying scatter plots of each pair of variables in a matrix layout. This allows for quick identification of correlations and patterns between multiple variables.

Example Dataset:

Body mass index (BMI) vs. age for a group of individuals

Pair Grid:

Utility:

Pair grids extend the concept of pair charts by adding a third dimension, allowing for the visualization of relationships between three variables. They are particularly useful for exploring complex interactions between variables.

Example Dataset:

Sales figures for different product categories across different regions and time periods

Facet Grid:

Utility:

Facet grids provide a flexible way to visualize data subsets based on different categorical variables. They are useful for comparing distributions or trends across different groups.

Example Dataset:

Customer purchase behavior segmented by age group, gender, and location

Scatter Plot:

Utility:

Scatter plots are the most common type of chart for visualizing the relationship between two numerical variables. They allow for quick identification of patterns, trends, and outliers.

Example Dataset:

Relationship between temperature and rainfall for a given location

Regression Plot:

Utility:

Regression plots are used to visualize the relationship between a dependent variable and one or more independent variables. They provide a line of best fit that represents the overall trend in the data.

Example Dataset:

Predicting house prices based on factors like size, location, and amenities

Count Plot:

Utility:

Count plots are bar charts that show the frequency of different categories in a dataset. They are useful for summarizing categorical data and identifying the most common values.

Example Dataset:

Distribution of customer satisfaction ratings for a product or service

Bar Plot:

Utility:

Bar plots are versatile charts used to compare categorical or numerical data. They are effective for comparing values between different groups or categories.

Example Dataset:

Comparison of sales figures for different product categories

Violin Plot:

Utility:

Violin plots combine box plots and density plots to represent the distribution of data within categories. They provide a more detailed view of the distribution compared to box plots alone.

Example Dataset:

Distribution of test scores for different student groups

Heat Map:

Utility:

Heat maps are used to visualize the intensity of values across two dimensions. They are particularly useful for exploring relationships and patterns in large datasets.

Example Dataset:

Stock market price fluctuations over time

2)

Seaborn is a data visualization library built on top of Matplotlib in Python. It provides a higher-level interface for creating statistical graphics that are aesthetically pleasing and informative. Seaborn is particularly well-suited for exploring and understanding complex datasets.

Seaborn offers a wide range of plot types, categorized into the following main groups:

Relational Plots: These plots show the relationship between two or more variables. Examples include scatter plots, line plots, and joint plots.

Distributional Plots: These plots show the distribution of a single variable. Examples include histograms, box plots, and violin plots.

Categorical Plots: These plots show the distribution of a categorical variable or the relationship between a categorical and a numerical variable. Examples include bar plots, count plots, and category plots.

Matrix Plots: These plots show the relationship between multiple variables in a matrix format. Examples include pair plots and heatmaps.

Time Series Plots: These plots show the behavior of a variable over time. Examples include line charts, time series heatmaps, and autocorrelation plots.

Seaborn's high-level interface makes it easy to create these plots with minimal code, and its consistent styling makes the plots aesthetically pleasing and easy to interpret. Additionally, Seaborn integrates seamlessly with Pandas DataFrames, making it a powerful tool for data exploration and visualization.

Conclusion:

Matplotlib is a powerful and versatile Python library for creating various types of charts and plots for data visualization.

Date: 20 -10-2023

Signature of faculty in-charge