Batch:	**Roll No.:**
Experiment / assignment / Tutorial	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: To demonstrate Matplot library in python by studying its key features and its functions

AIM: Write a program to demonstrate matplotlib visualisation functions using datasets

OUTCOME: Student will be able to:

CO1: Formulate problem statement and develop the logic (algorithm/flowchart) for its solution.

CO5: Illustrate the use of Python packages.

AIM: Write a program to explore the Matplotlib library

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)

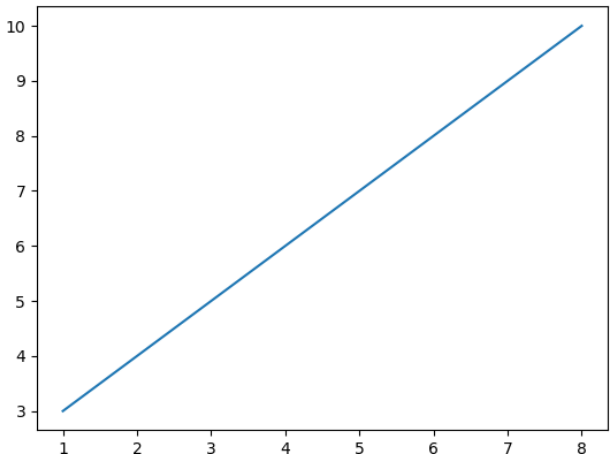
- x, y: These parameter are the horizontal and vertical coordinates of the data points. x values are optional.
- fmt: This parameter is an optional parameter and it contains the string value.
- data: This parameter is an optional parameter and it is an object with labelled data.

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):	**Outp ut**

```
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()
```



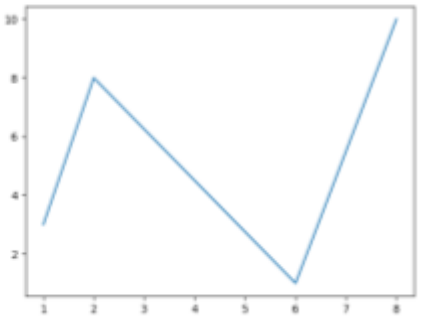
1) Multiple Points

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

Program:

```
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()
```

Outpu t:--



2) Matplotlib Line

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

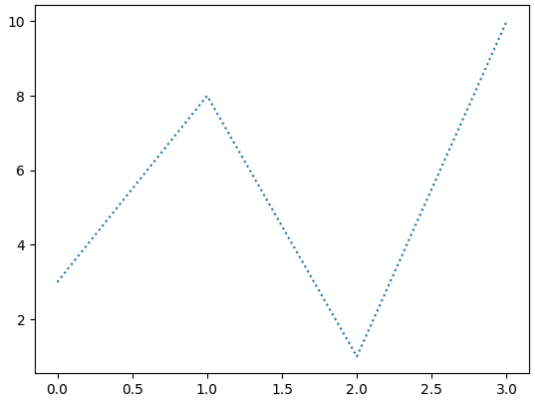
Following are the linestyles available in ***matplotlib*:**

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.

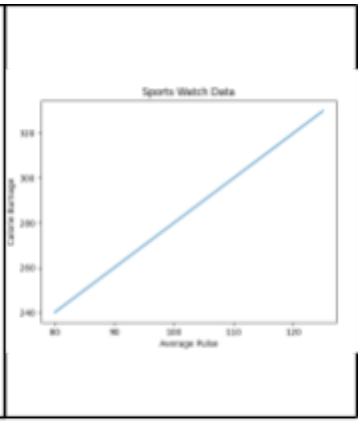
Syntax: matplotlib.pyplot.xlabel(xlabel, fontdict=None, labelpad=None, **kwargs)

b. Create a Title for a Plot

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

Program:--	**Output:--**
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()
```

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:

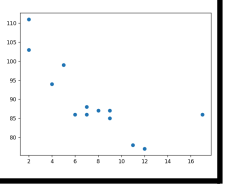
Syntax:-- matplotlib.pyplot.scatter(x_axis_data, y_axis_data, s=None, c=None, marker=None, cmap=None, vmin=None, vmax=None, alpha=None, linewidths=None, edgecolors=None)

- **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])
```

Output:--



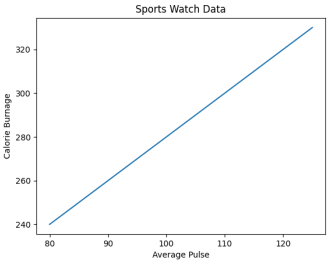
```
plt.scatter(x, y)
plt.show()
```

Add Grid Lines to a Plot

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

```
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()
```

Output: ?



5) Display Multiple Plots

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

```
subplot(nrows, ncols, index, **kwargs)
```

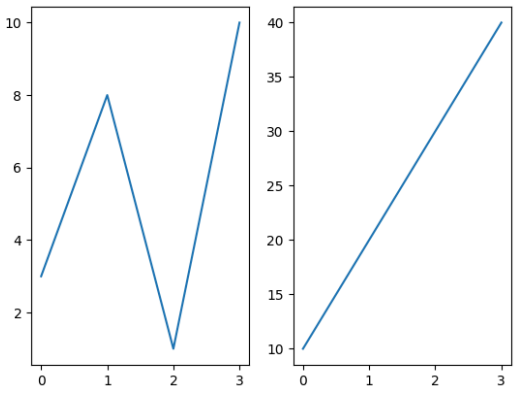
The layout is organized in rows and columns, which are represented by the *first* and *second* argument.

The third argument represents the index of the current plot.

Program:-

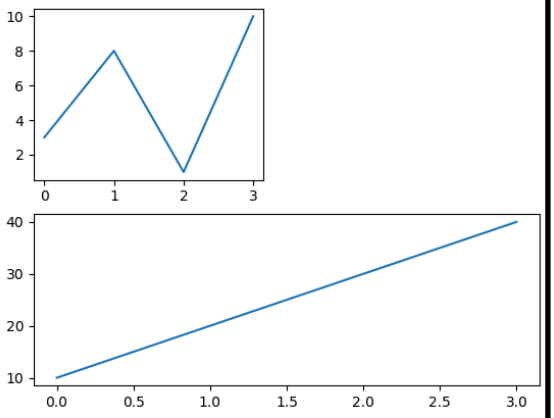
```
import matplotlib.pyplot as plt
import numpy as np

#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()
```

Output:-- ?




```
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()
```

Output:--

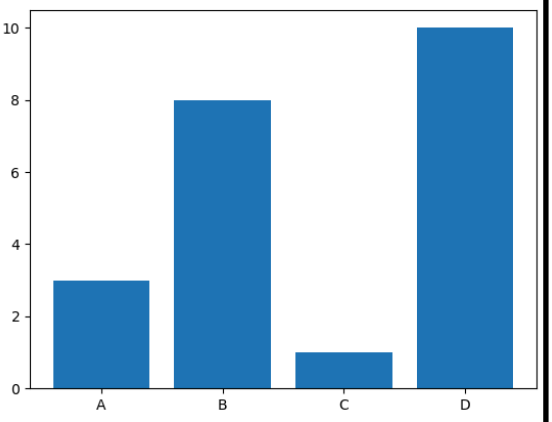


6) Creating Bars

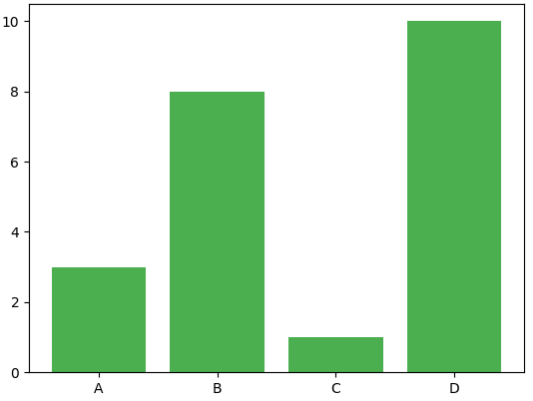
With Pyplot, you can use the bar() function to draw bar graphs.

```
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()
```

Output:--



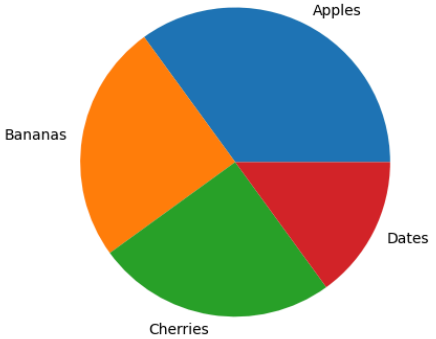

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x, y, color =
"#4CAF50") plt.show()
```



7) Creating Pie Chart with Labels:

```
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()
```

Output: ?



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 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:

· Swarn chart · Pair chart · Pair grid · Facet Grid · Scatter plot	· Regression plot · Count plot · Bar plot · Violin plot · Heat map

Ans.

1. Swarm Chart

A **swarm plot** displays distribution with non-overlapping points.

Utility:

Used for visualizing the spread of data points in a category.

2. Pair Chart

A **pair plot** creates pairwise scatterplots for all numerical columns.

Utility:

Ideal for exploring relationships in datasets with multiple variables.

3. Pair Grid

A **pair grid** gives more customization compared to pairplot.

Utility:

You can tailor the diagonal, upper, and lower plots.

4. Facet Grid

A **facet grid** creates subplots for subsets of the dataset based on column values.

Utility:

Best for analyzing trends within specific subsets of data.

Here's a Python implementation and explanation of the utility of the mentioned charts using appropriate datasets and seaborn and matplotlib libraries:

5. Scatter Plot

A **scatter plot** displays the relationship between two numerical variables.

Utility:

Used for identifying correlations or clustering.

6. Regression Plot

A **regression plot** overlays a regression line on a scatter plot.

Utility:

Used to visualize trends and relationships.

7. Count Plot

A **count plot** shows the count of observations for each category.

Utility:

Best for understanding categorical distributions.

8. Bar Plot

A **bar plot** aggregates and displays mean values (or other statistics).

Utility:

Best for showing comparisons across categories.

9. Violin Plot

A **violin plot** combines boxplots and KDE to show distributions.

Utility:

Used for visualizing spread and density in a dataset.

10. Heat Map

A **heat map** visualizes data as a color-coded matrix.

Utility:

Used for correlation matrices or tabular data visualization.

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

ANS.

Seaborn is a powerful Python data visualization library built on top of **Matplotlib**. It provides high-level functions for creating aesthetically pleasing and informative statistical graphics. Seaborn simplifies the process of visualizing and analyzing data with built-in themes and color palettes, while also integrating well with pandas DataFrames.

Different categories of plot in Seaborn are as follows:

1. **Relational**: Scatter and line plots.
2. **Categorical**: Box, bar, swarm, and violin plots.
3. **Distribution**: Histograms, KDE, and rug plots.
4. **Matrix**: Heatmaps and cluster maps.
5. **Regression**: Trendline visualization.
6. **Multi-Plot Grids**: FacetGrid, PairGrid, and pair plots.

Books/ Journals/ Websites referred:

1. Matplotlib Plotting (w3schools.com) – 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

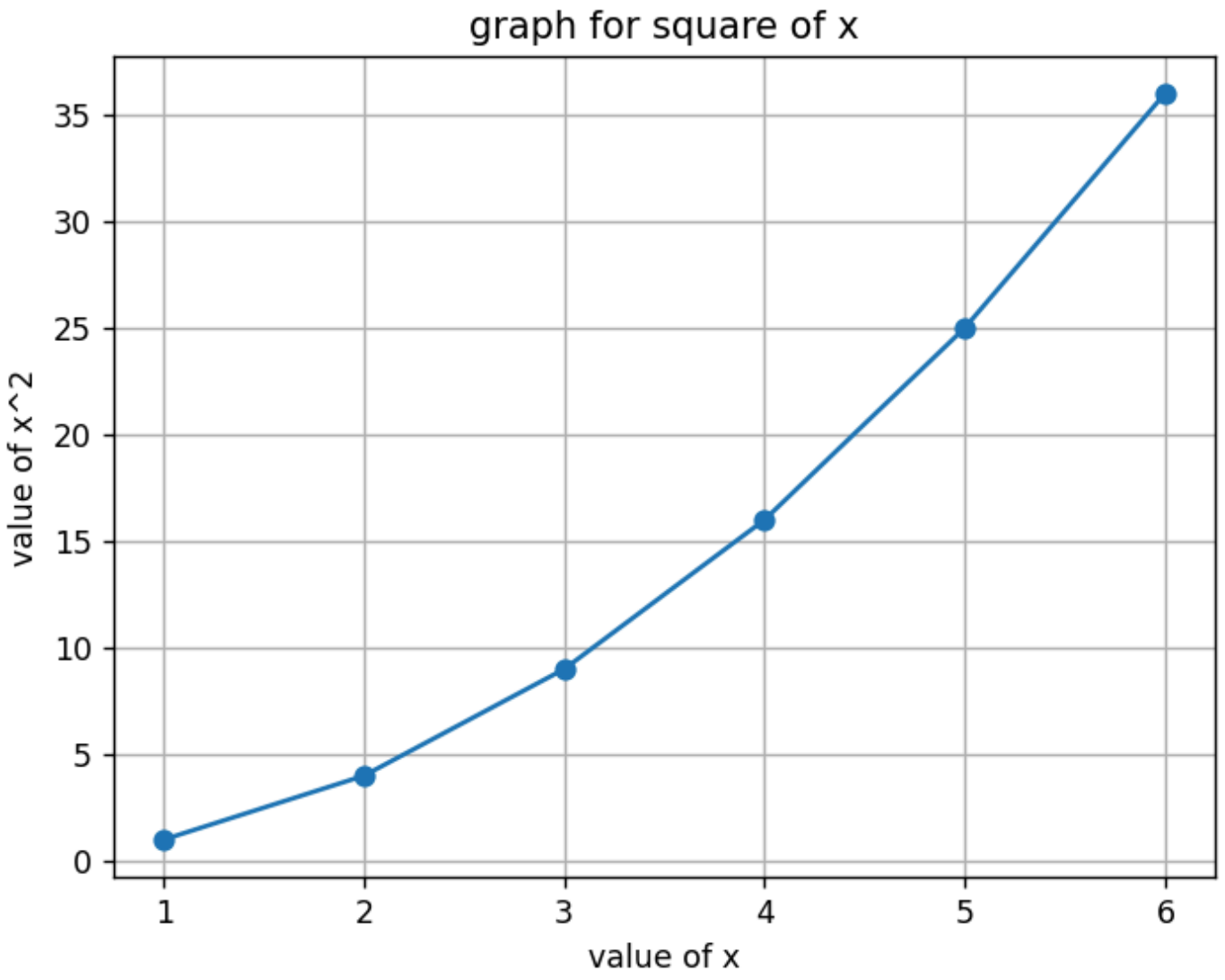
---

Implementation details:

1.

```
import matplotlib.pyplot as plt
import numpy as np
x=np.array([1,2,3,4,5,6])
y=np.array([1,4,9,16,25,36])

plt.plot(x,y,marker='o')
```

```
plt.xlabel('value of x')
plt.ylabel('value of x^2')
plt.title('graph for square of x')
plt.grid()
plt.show()
```

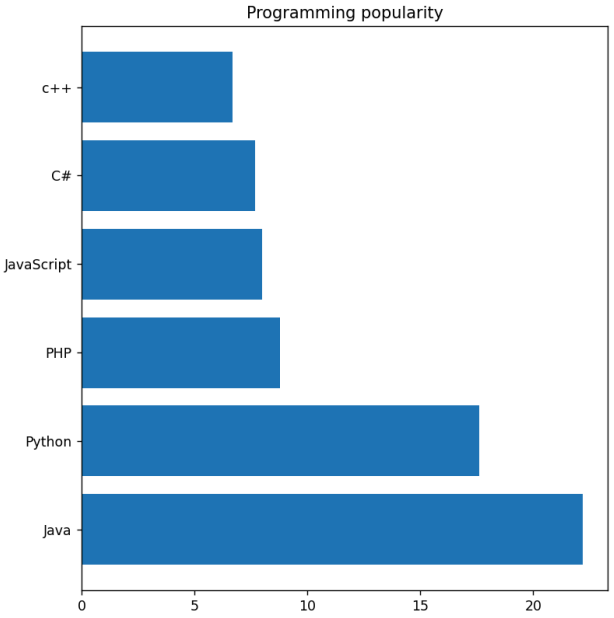


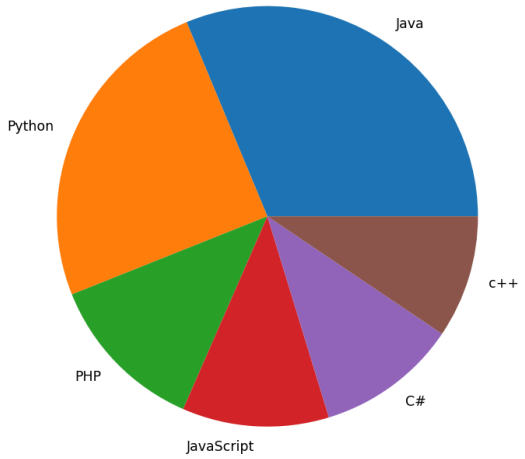
2.

```
import matplotlib.pyplot as plt
import numpy as np
prolang=np.array(['Java','Python','PHP','JavaScript','C#','C++'])
pop=np.array([22.2, 17.6, 8.8, 8, 7.7, 6.7])

plt.subplot(1,2,1)
plt.barh(prolang,pop)
plt.title("Programming popularity")

plt.subplot(1,2,2)
```

```
plt.pie(pop,labels=prolang)

plt.show()
```

OUTPUT:





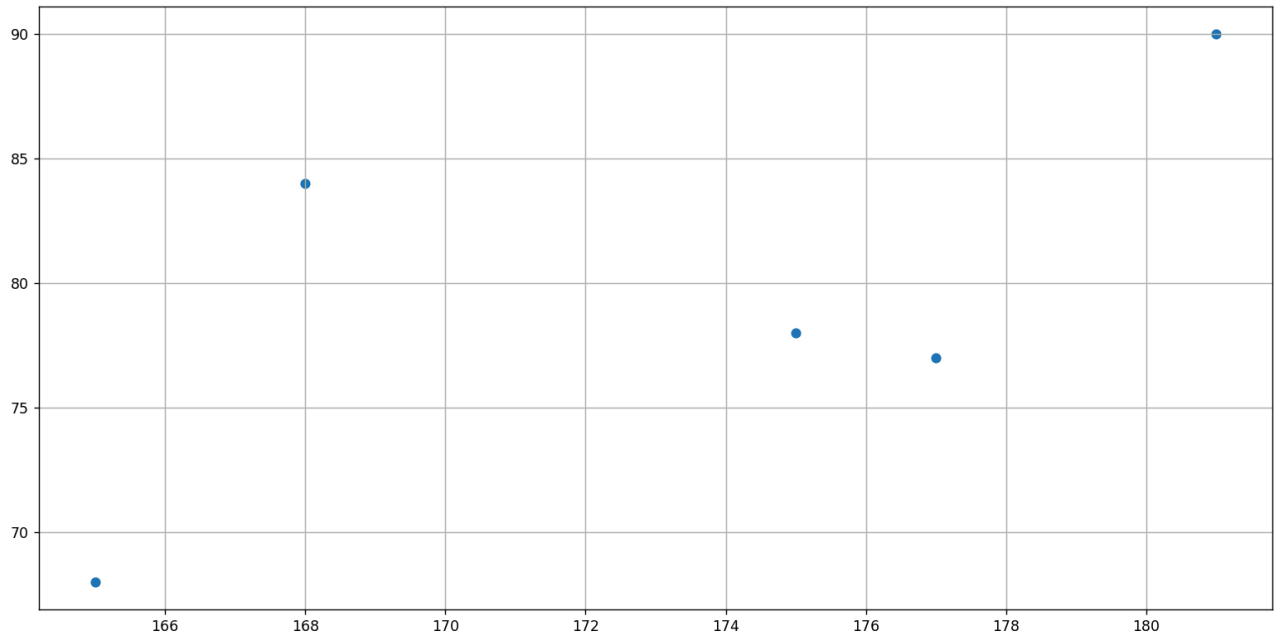
3.

```
import matplotlib.pyplot as plt
import numpy as np
nm=[]
h=[]
w=[]
n=int(input("enter number of students in your batch: "))
for i in range(n):
    print(f"enter data of student number {i+1}")
    height=int(input("enter their height in cm: "))
    weight=int(input("enter their height in kg: "))
    print("\n")

    h.append(height)
    w.append(weight)

harray=np.array(h)
warray=np.array(w)

plt.scatter(harray,warray)
```

```
plt.grid()
plt.show()
```

OUTPUT:

```
enter number of students in your batch: 5
enter data of student number 1
enter their height in cm: 165
enter their height in kg: 68


enter data of student number 2
enter their height in cm: 175
enter their height in kg: 78


enter data of student number 3
enter their height in cm: 168
enter their height in kg: 84


enter data of student number 4
enter their height in cm: 177
enter their height in kg: 77


enter data of student number 5
enter their height in cm: 181
enter their height in kg: 90
```



Conclusion:

Date: **Signature of faculty in-charge**