Experiment No. 10
Program to plot graph using matplotlib library
Name : Vaidehi D. Gadag
Branch/Div.: Comps-1 (C47)
Date of Performance: 03/04/2024
Date of Submission: 10/04/2024

Experiment No. 10

Title: Program to plot graph using matplotlib library

Aim: To explore the basics Matplotlib for data visualization.

Objective: To understand how to use graphs and charts for data analysis.

Theory:

Matplotlib is a low level graph plotting library in python that serves as a visualization utility. Matplotlib is open source and we can use it freely.

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias.

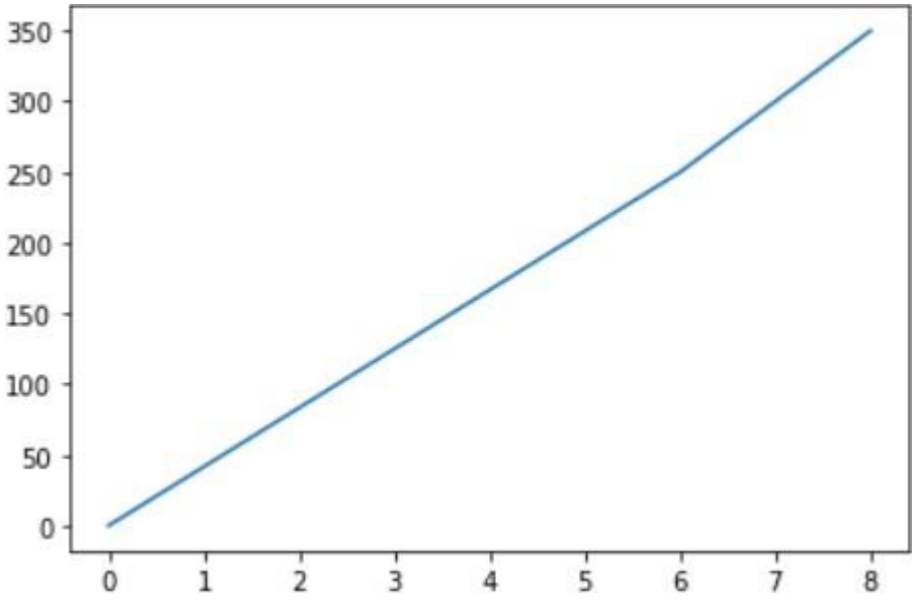
- The plot() function is used to draw points (markers) in a diagram.
- By default, the plot() function draws a line from point to point.
- The function takes parameters for specifying points in the diagram.
- Parameter 1 is an array containing the points on the x-axis.
- Parameter 2 is an array containing the points on the y-axis.Eg: (0,0), (6,250), (8,350)
  import matplotlib.pyplot as plt

import numpy as np

x = np.array([0,6,8])

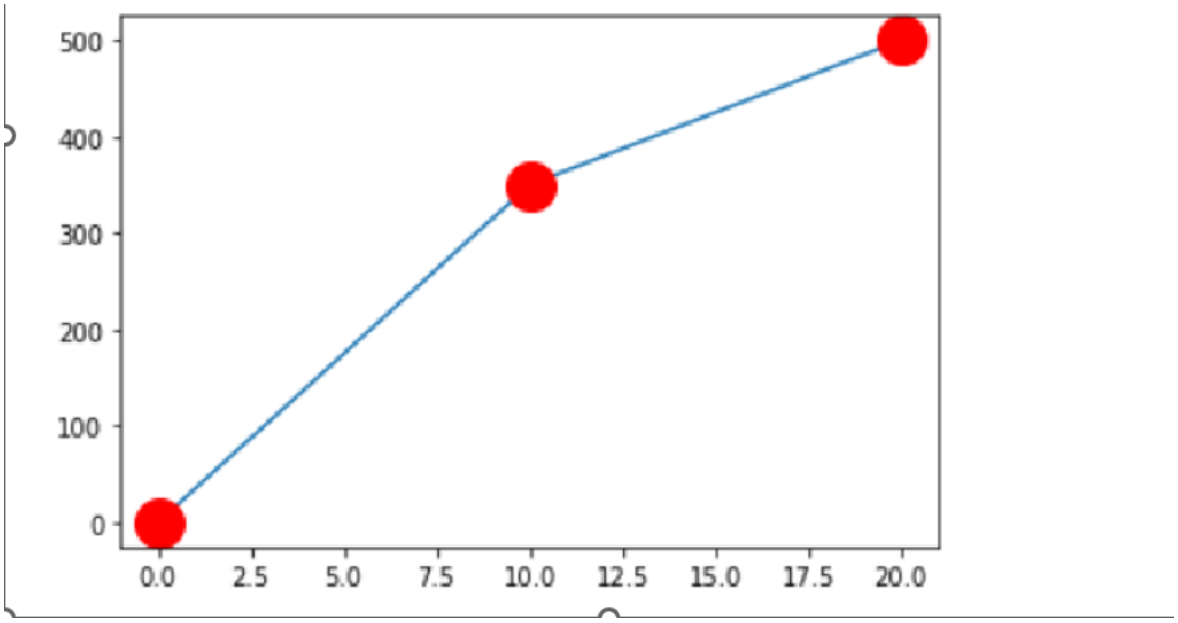
y = np.array([0,250,350])

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



- The keyword argument marker is to emphasize each point with a specified marker.
- The keyword argument markersize or the shorter version, ms is to set the size of the markers
- The keyword argument markeredgecolor or the shorter mec is to set the color of the edge of the markers
- The keyword argument markerfacecolor or the shorter mfc is to set the color inside the edge of the markers
-

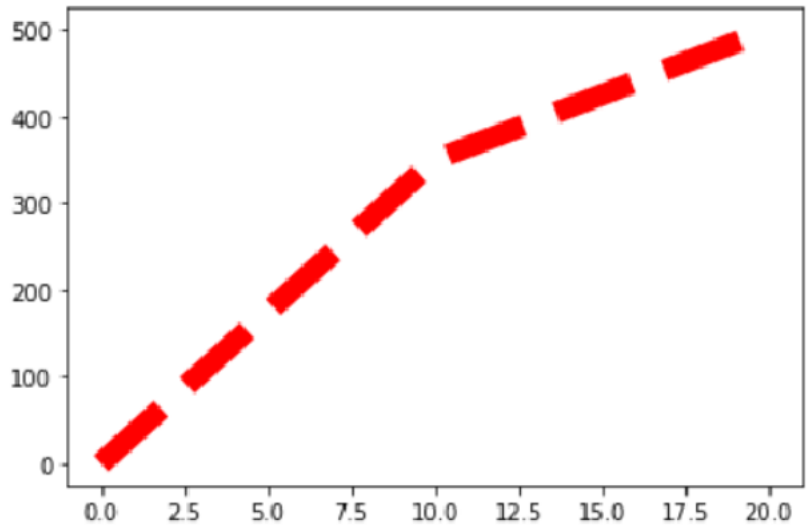
```
import matplotlib.pyplot as plt import numpy as np

x=np.array([0,10,20])

y=np.array([0,350,500]) plt.plot(x,y,marker='o',ms=20,mec='r',mfc='r') plt.show()
```



- The keyword argument linestyle, or shorter ls, to change the style of the plotted line.
- The line style can be written in a shorter syntax:
    - linestyle can be written as ls.
    - dotted can be written as :.
    - dashed can be written as --.
- the keyword argument color or the shorter c to set the color of the line

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

x=np.array([0,10,20])

y=np.array([0,350,500]) plt.plot(x,y,color='red',ls='--',lw=10)
```

```
Out[16]: [<matplotlib.lines.Line2D at 0x24587eee970>]
```



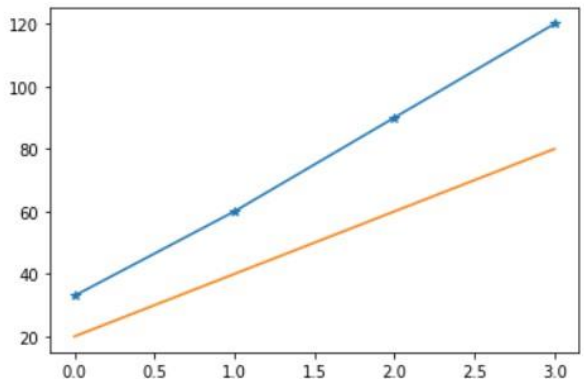
Many plotting can be done by adding more plt.plot() functions

import matplotlib.pyplot as plt import numpy as np

y1=np.array([33,60,90,120])

y2=np.array([20,40,60,80]) plt.plot(y1,marker='*') plt.plot(y2)

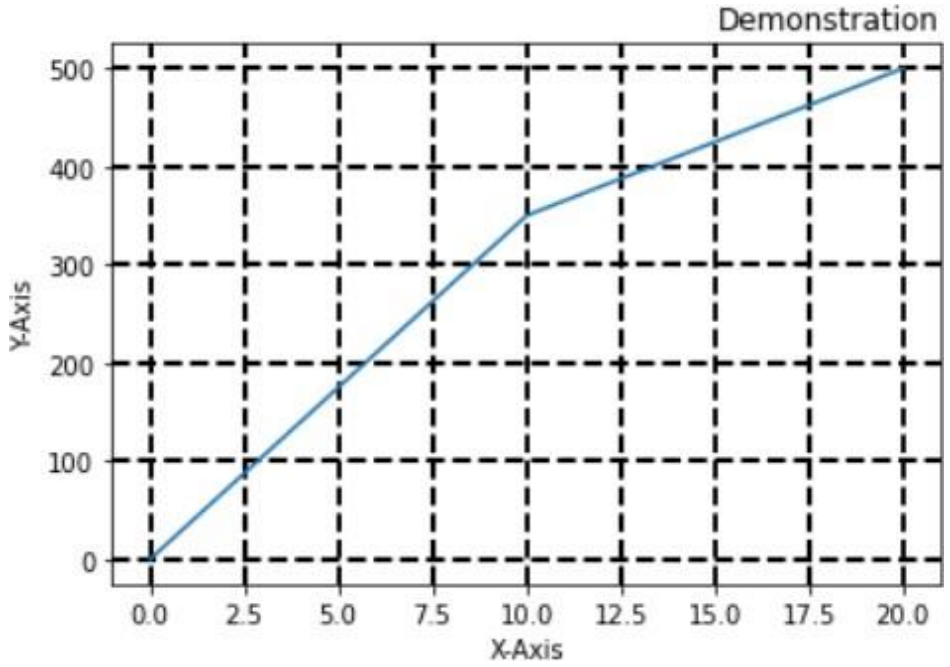
```
Out[22]: [<matplotlib.lines.Line2D at 0x2458853efa0>]
```



- With Pyplot, you can use the xlabel() and ylabel() functions to set a label for the x- and y-axis.
- With Pyplot, you can use the title() function to set a title for the plot.
- You can use the loc parameter in title() to position the title.
- Legal values are: 'left', 'right', and 'center'. Default value is 'center'.
- With Pyplot, you can use the grid() function to add grid lines to the plot.

- You can use the axis parameter in the grid() function to specify which grid lines to display.
- Legal values are: 'x', 'y', and 'both'. Default value is 'both'.

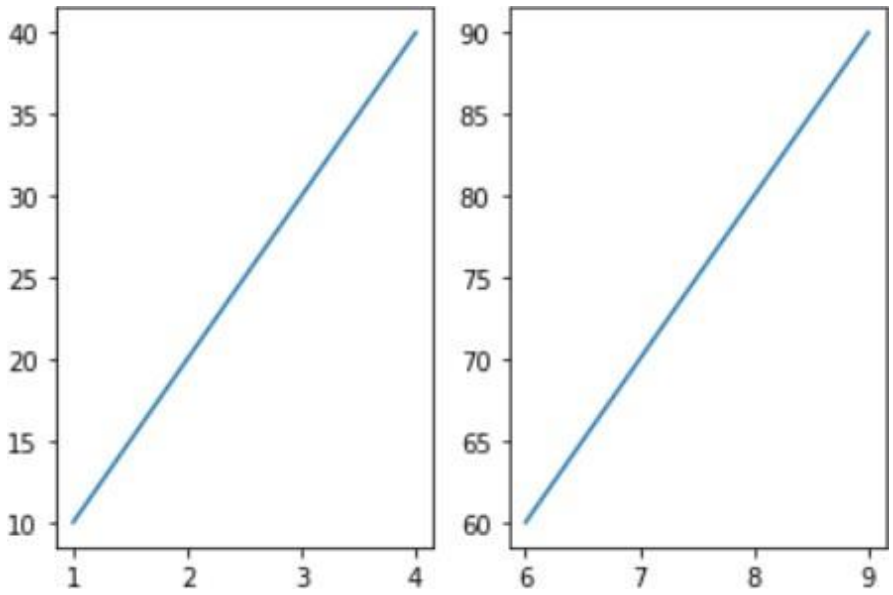
```
import
matplotlib.pyplot as
pltimport numpy as
np

x=np.array([0,10,20])
y=np.array([0,
350,500])
plt.plot(x,y)
plt.xlabel("X-
Axis")
plt.ylabel("Y-
Axis")
plt.title('Demonstration',loc='right')
plt.grid(color='black',linestyle='--
',linewidth=2)
```



SubPlots:
With the subplots() function you can draw multiple plots in one figure.
The subplots() function takes three arguments that describes the layout of the figure.
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.

```
x=np.array([1,2,3,4])
y=np.array([10,20,30,40])

plt.subpl
ot(1,2,1)
plt.plot(x
,y)


x=np.array([6,7,8,9])
y=np.array([60,70,80,90])

plt.subpl
ot(1,2,2)
plt.plot(x
,y)
```

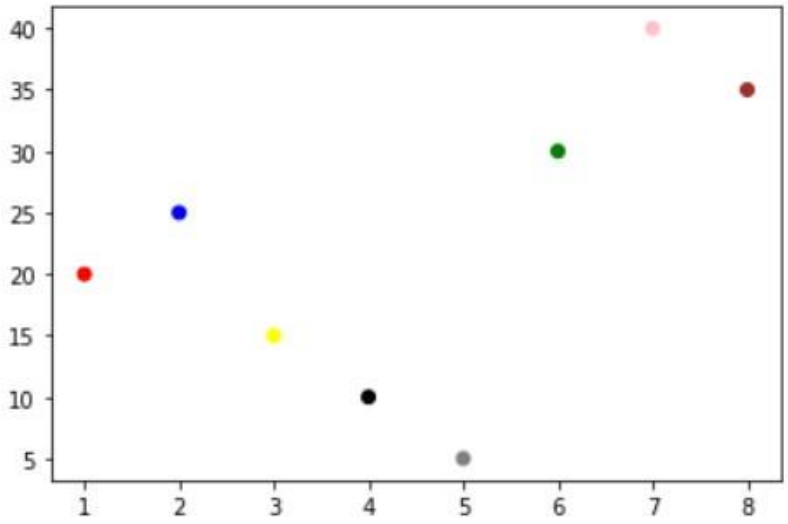


Scatter Plots:

- 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 samelength, one for the values of the x-axis, and one for values on the y-axis.
- You can set your own color for each scatter plot with the color or the c argument.

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

x=np.array([1,2,3,4,5,6,7,8])
y=np.array([20,25,15,10,5,30,40,35])
c=np.array(['red','blue','yellow','black','grey','green','pin
k','brown'])plt.scatter(x,y,color=c)
```

```
Out[53]: <matplotlib.collections.PathCollection at 0x24588442040>
```

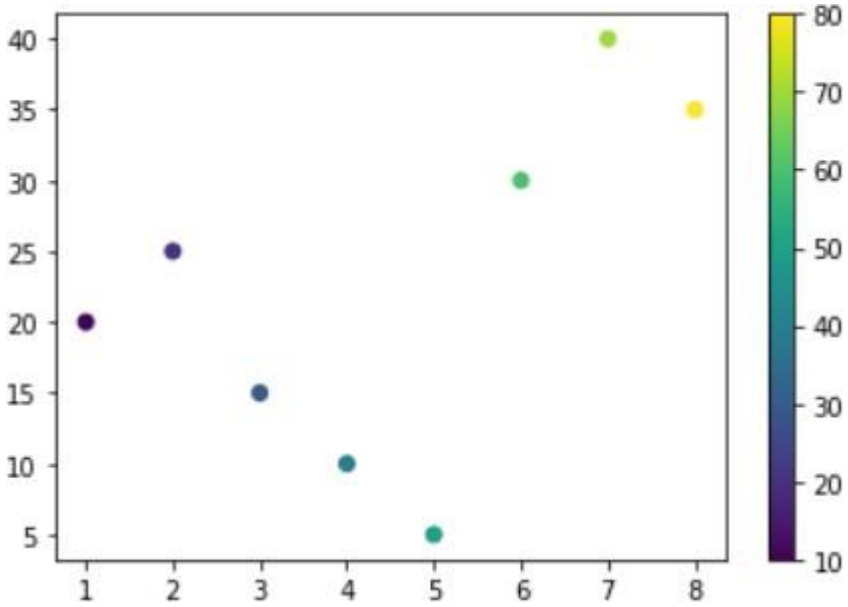


ColorMaps

The Matplotlib module has a number of available colormaps.
A colormap is like a list of colors, where each color has a value that ranges from 0 to 100.

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



x=np.array([1,2,3,4,5,6,7,8])
y=np.array([20,25,15,10,5,30,40,35])
col=np.array([10,20,30,40,50,60,70,80])
```

```
plt.scatter(x,y,c=col,cma
p='viridis')plt.colorbar()
plt.show()
```



Bar Graph

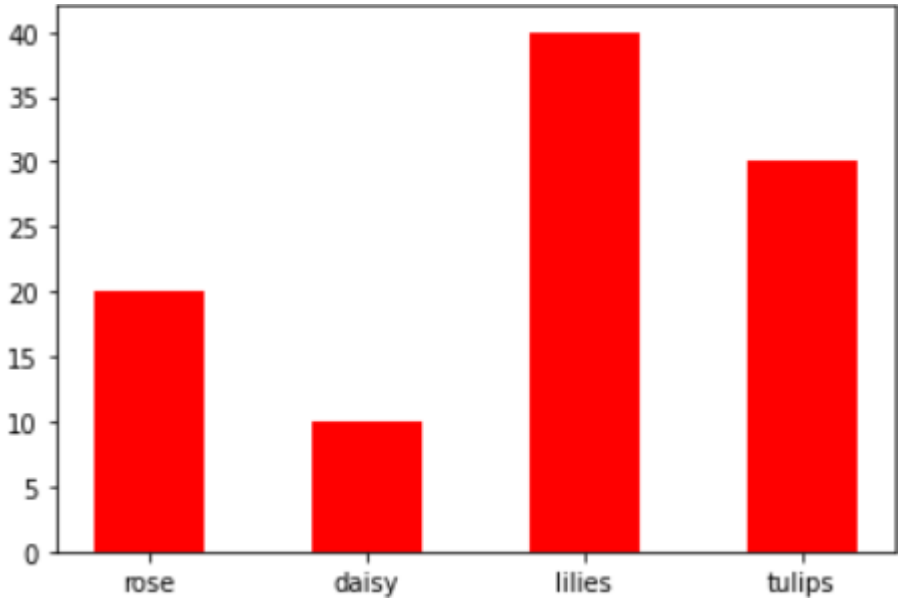
- With Pyplot, you can use the bar() function to draw bar graphs.
- The bar() function takes arguments that describes the layout of the bars.
- The categories and their values represented by the first and second argument as arrays.
- If you want the bars to be displayed horizontally instead of vertically, use the barh() function.
- The bar() and barh() takes the keyword argument color to set the color of the bars.

The bar() takes the keyword argument width to set the width of the bars.

- The barh() takes the keyword argument height to set the height of the bars.

```
import
matplotlib.pyplot as
pltimport numpy as
np
x=np.array(['rose','daisy','lilies','tulip
s']) y=np.array([20,10,40,30])
```

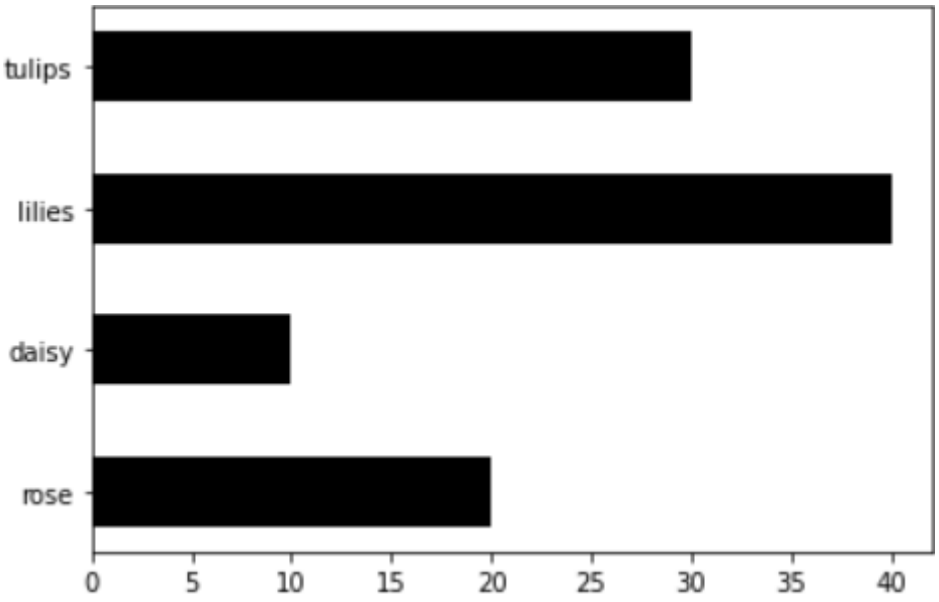
plt.bar(x,y,color='red',width=0.5)



import
matplotlib.pyplot as
pltimport numpy as
np

x=np.array(['rose','daisy','lilies','tulips']) y=np.array([20,10,40,30])

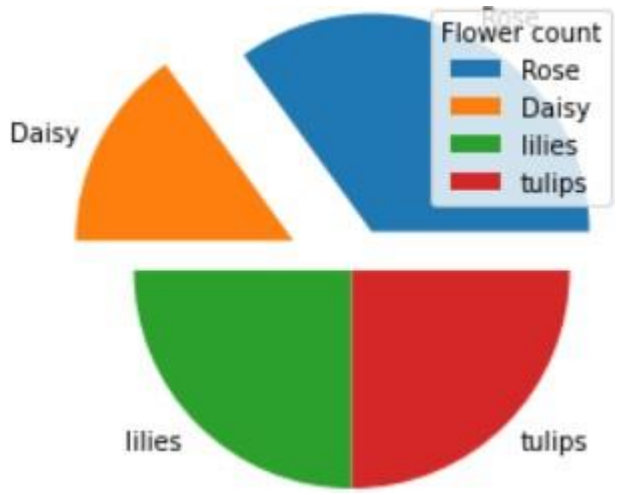
plt.barh(x,y,color='black',height=0.5)



Pie Charts

- With Pyplot, you can use the pie() function to draw pie charts.
- The pie chart draws one piece (called a wedge) for each value in the array .
- By default the plotting of the first wedge starts from the x-axis and move counterclockwise.
- Add labels to the pie chart with the label parameter.
- The label parameter must be an array with one label for each wedge.
- The default start angle is at the x-axis, but you can change the start angle by specifying astartangle parameter.
- The startangle parameter is defined with an angle in degrees, default angle is 0.
- The explode parameter allows you to do that.
- The explode parameter, if specified, and not None, must be an array with one value for eachwedge.
- Each value represents how far from the center each wedge is displayed

```
import
matplotlib.pyplot as
pltimport numpy as
np
y=np.array([35,15,25,
25])
l=np.array(['Rose','Daisy','lilies','tulips'])e=np.array([0.2,0.3,0,0])


plt.pie(y,labels=l,explo
de=e)
plt.legend(title="Flowe
r count")plt.show()
```



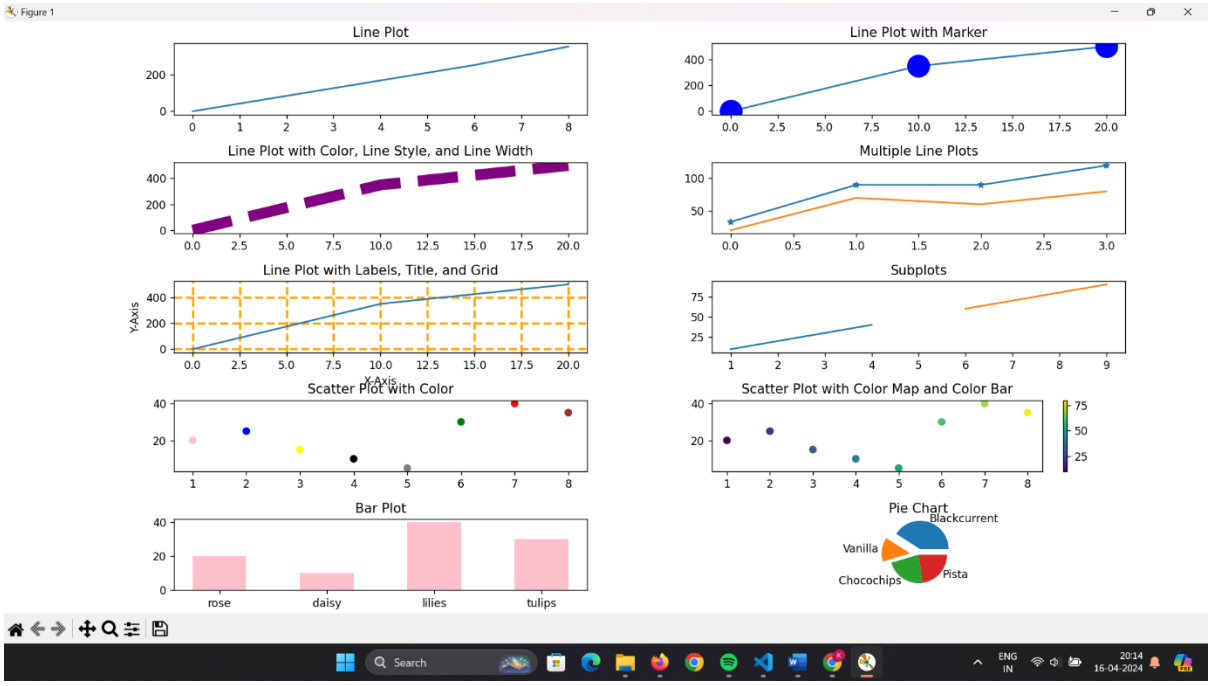
Code:

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

#creating the subplots for the graphs to not overlap each other
fig, axs = plt.subplots(5, 2, figsize=(5, 10))

# Line Plot
x1 = np.array([0, 6, 8])
y1 = np.array([0, 250, 350])
axs[0, 0].plot(x1, y1)
axs[0, 0].set_title('Line Plot')

# Line Plot with Marker and Customization
x2 = np.array([0, 10, 20])
y2 = np.array([0, 350, 500])
axs[0, 1].plot(x2, y2, marker='o', ms=20, mec='blue', mfc='b')
axs[0, 1].set_title('Line Plot with Marker')

# Line Plot with Color, Line Style, and Line Width
x3 = np.array([0, 10, 20])
y3 = np.array([0, 350, 500])
axs[1, 0].plot(x3, y3, color='purple', ls='--', lw=10)
axs[1, 0].set_title('Line Plot with Color, Line Style, and Line Width')

# Multiple Line Plots
y4_1 = np.array([33, 90, 90, 120])
y4_2 = np.array([20, 70, 60, 80])
axs[1, 1].plot(y4_1, marker='*')
axs[1, 1].plot(y4_2)
```

```
axs[1, 1].set_title('Multiple Line Plots')

# Line Plot with Labels, Title, and Grid
x5 = np.array([0, 10, 20])
y5 = np.array([0, 350, 500])
axs[2, 0].plot(x5, y5)
axs[2, 0].set_xlabel("X-Axis")
axs[2, 0].set_ylabel("Y-Axis")
axs[2, 0].set_title('Line Plot with Labels, Title, and Grid')
axs[2, 0].grid(color='orange', linestyle='--', linewidth=2)

# Subplots
x6_1 = np.array([1, 2, 3, 4])
y6_1 = np.array([10, 20, 30, 40])
axs[2, 1].plot(x6_1, y6_1)
x6_2 = np.array([6, 7, 8, 9])
y6_2 = np.array([60, 70, 80, 90])
axs[2, 1].plot(x6_2, y6_2)
axs[2, 1].set_title('Subplots')

# Scatter Plot with Color
x7 = np.array([1, 2, 3, 4, 5, 6, 7, 8])
y7 = np.array([20, 25, 15, 10, 5, 30, 40, 35])


c7 = np.array(['Pink', 'blue', 'yellow', 'black', 'grey', 'green', 'red', 'brown'])
scatter7 = axs[3, 0].scatter(x7, y7, color=c7)
axs[3, 0].set_title('Scatter Plot with Color')

# Scatter Plot with Color Map and Color Bar
x8 = np.array([1, 2, 3, 4, 5, 6, 7, 8])
y8 = np.array([20, 25, 15, 10, 5, 30, 40, 35])
col8 = np.array([10, 20, 30, 40, 50, 60, 70, 80])
scatter8 = axs[3, 1].scatter(x8, y8, c=col8, cmap='viridis')
axs[3, 1].set_title('Scatter Plot with Color Map and Color Bar')
plt.colorbar(scatter8, ax=axs[3, 1])

# Bar Plot
x9 = np.array(['rose', 'daisy', 'lilies', 'tulips'])
y9 = np.array([20, 10, 40, 30])
axs[4, 0].bar(x9, y9, color='pink', width=0.5)
axs[4, 0].set_title('Bar Plot')

# Pie Chart
y10 = np.array([45, 15, 25, 25])
l10 = np.array(['Blackcurrent', 'Vanilla', 'Chocochips', 'Pista'])
e10 = np.array([0.2, 0.3, 0, 0])
axs[4, 1].pie(y10, labels=l10, explode=e10)
```

```
# axs[4, 1].legend(title="Icecream", loc='lower left')
axs[4, 1].set_title('Pie Chart')

plt.tight_layout()

plt.show()
```

Output:



Conclusion:

In conclusion, the Matplotlib library in Python is a powerful tool for creating and visualizing data through graphs and diagrams. The plot() function, which is a key utility within the library's pyplot submodule, allows users to create a graph by specifying points in a diagram through two arrays - one for the x-axis and one for the y-axis. By default, the function draws a line from point to point, creating a visual representation of the data. Matplotlib is a valuable resource for anyone looking to visualize and analyse data using Python, with its flexibility, ease of use, and open-source availability making it a popular choice for data visualization tasks. With just a few lines of code, as demonstrated in the example above, users can quickly and easily create professional-quality graphs and diagrams.