



Vidyavardhini's College of Engineering & Technology
Department of Computer Engineering

Experiment No. 10
Program to plot graph using matplotlib library
Date of Performance: 03/04/2024
Date of Submission: 10/04/2024



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

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)

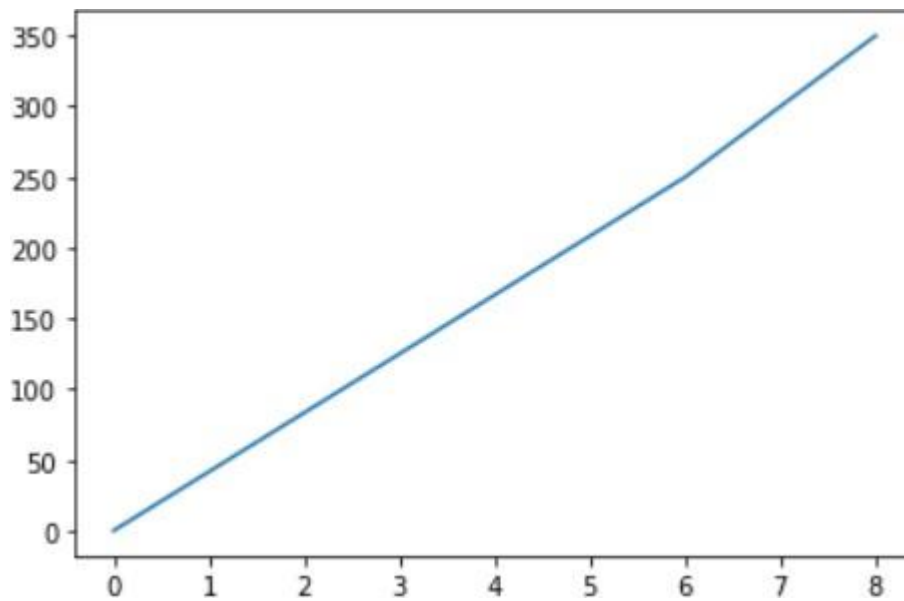
```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

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

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

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

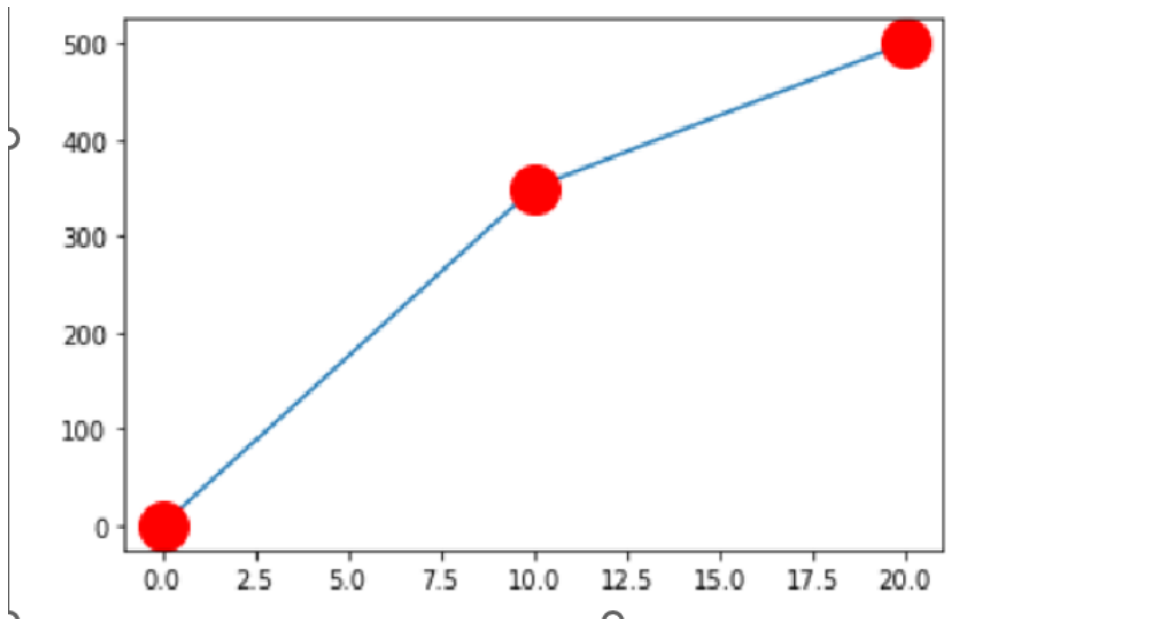


- The keyword argument marker is to emphasize each point with a specified marker.
- The keyword argument markersize or the shorter version, ms is to set the size of the markers
- The keyword argument markeredgecolor or the shorter mec is to set the color of the edge of the markers
- The keyword argument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

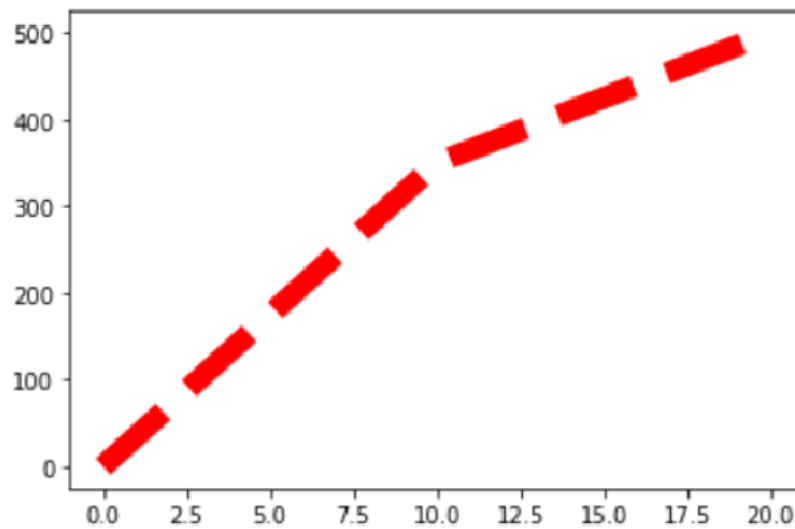
```
import matplotlib.pyplot as plt import numpy as np
```

```
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>]
```



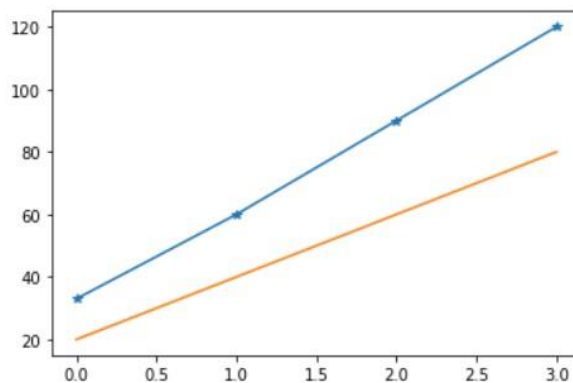
Many plotting can be done by adding more plt.plot() functions

```
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>]
```



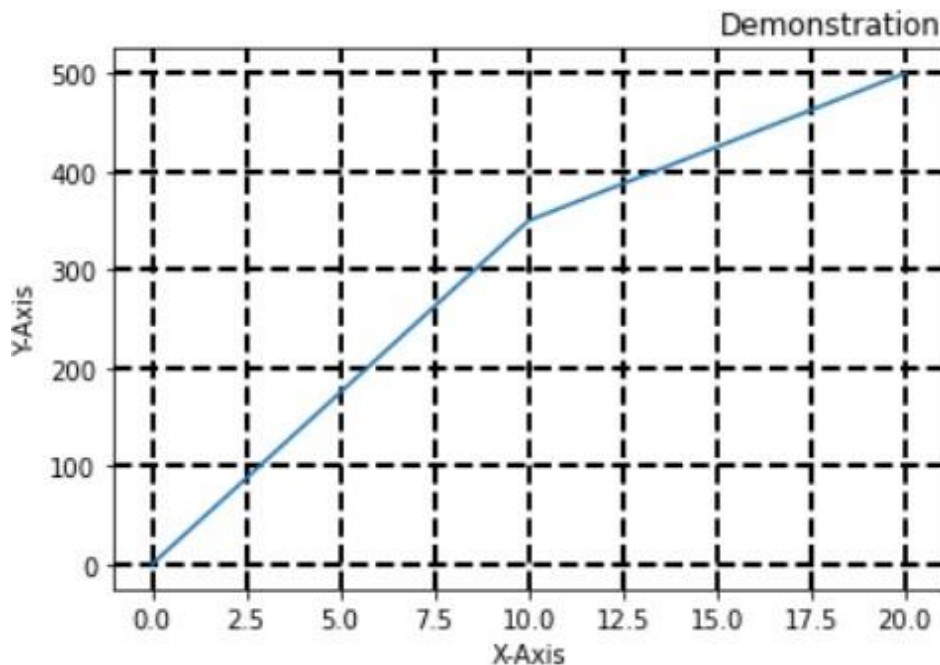
- With Pyplot, you can use the xlabel() and ylabel() functions to set a label for the x- and y-axis.
- With Pyplot, you can use the title() function to set a title for the plot.
- You can use the loc parameter in title() to position the title.
- Legal values are: 'left', 'right', and 'center'. Default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



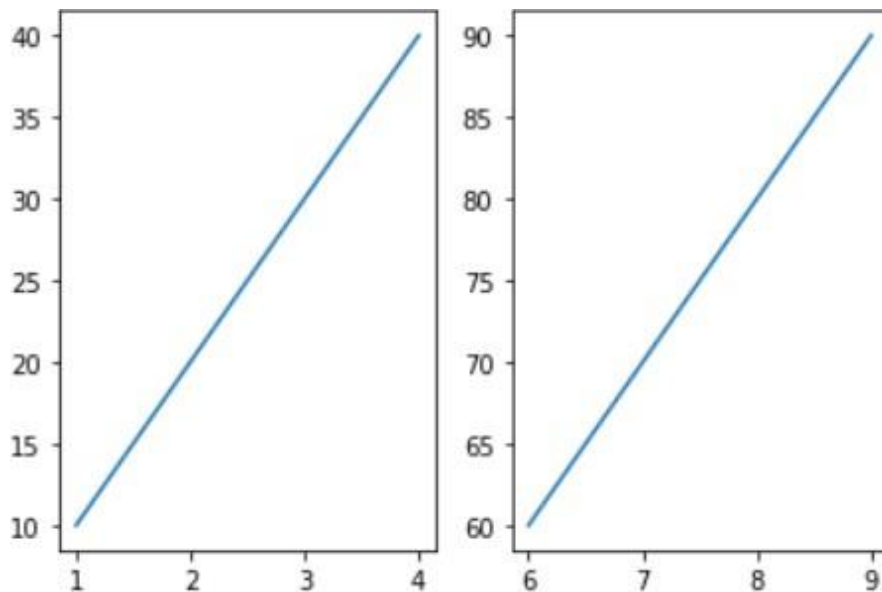
second argument. The third argument represents the index of the current plot.

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

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

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

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



Scatter Plots:

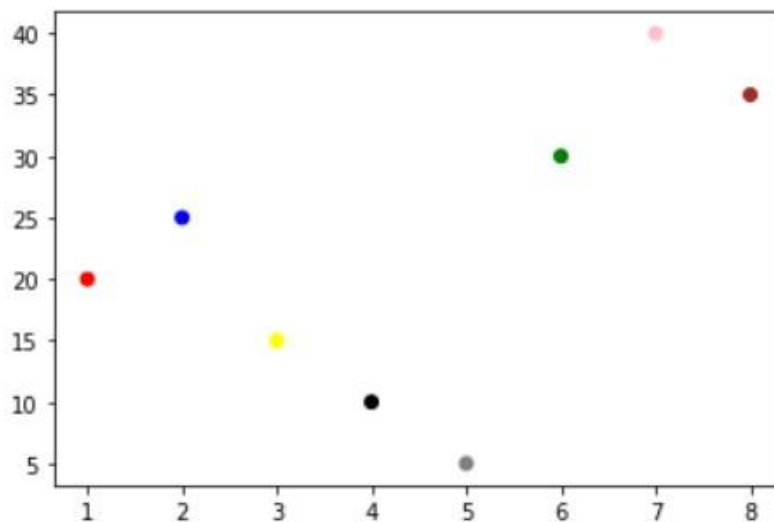
- With Pyplot, you can use the `scatter()` function to draw a scatter plot.
- The `scatter()` function plots one dot for each observation. It needs two arrays of the samelength, one for the values of the x-axis, and one for values on the y-axis.
- You can set your own color for each scatter plot with 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','pink',
k','brown'])plt.scatter(x,y,color=c)
```

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



ColorMaps

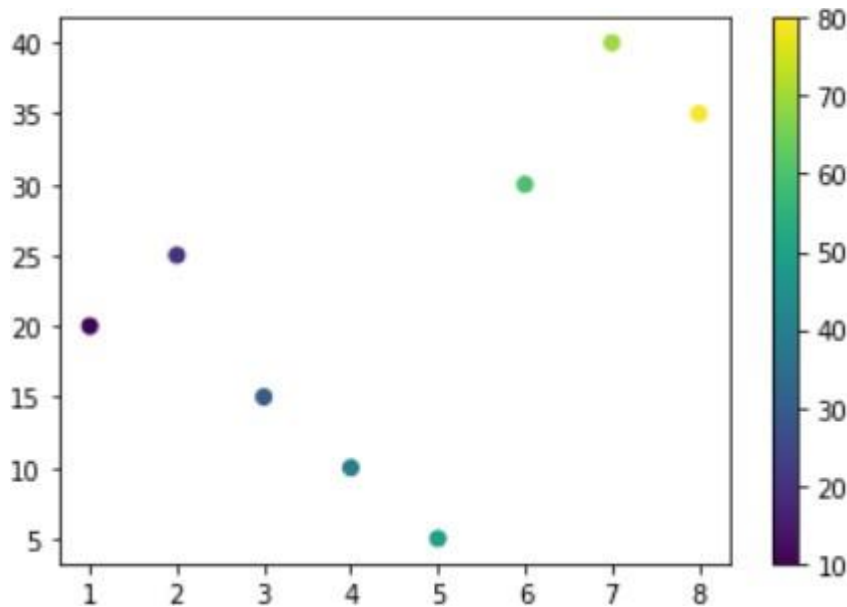
The Matplotlib module has a number of available colormaps. A colormap is like a list of colors, where each color has a value that ranges from 0 to 100.

```
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,cmap=
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.

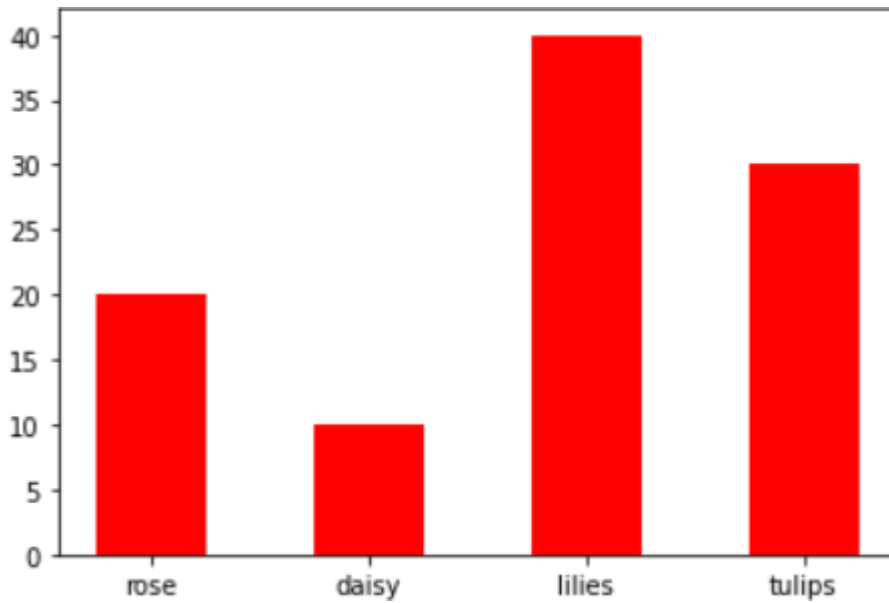
The `bar()` takes the keyword argument `width` to set the width of the bars.

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

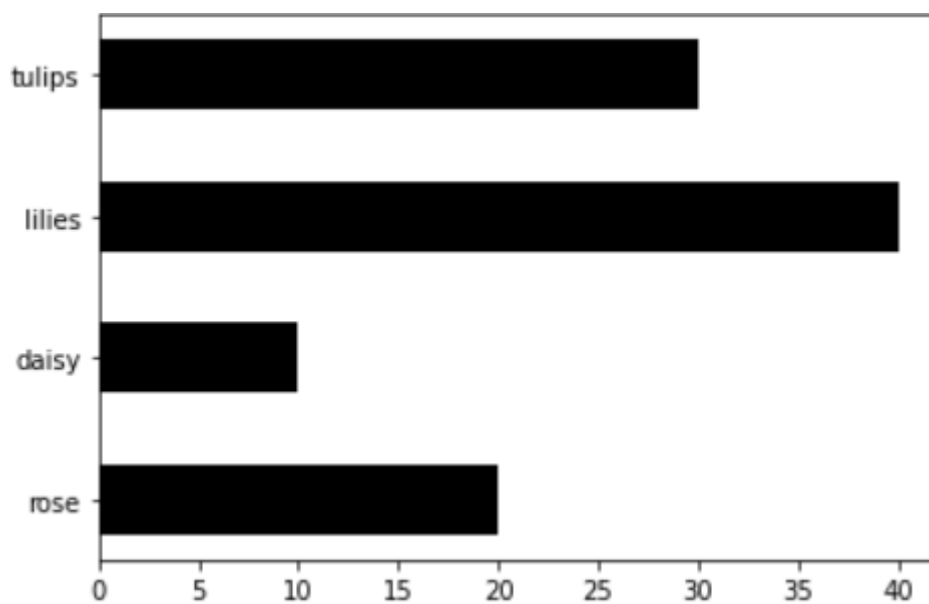
```
import
matplotlib.pyplot as
pltimport numpy as
np
x=np.array(['rose','daisy','lilies','tulip
s']) y=np.array([20,10,40,30])
```



```
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)
```



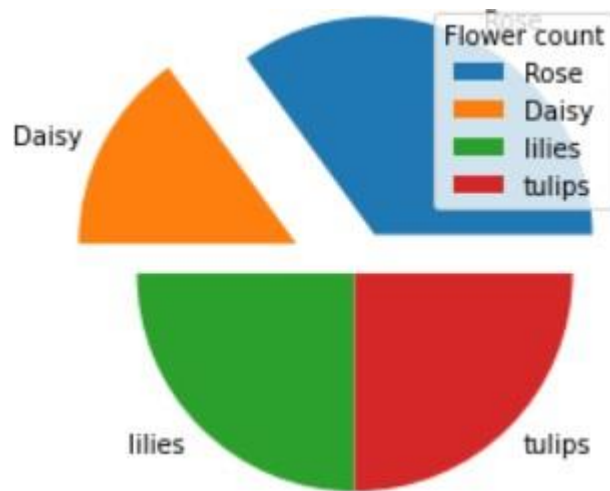


Pie Charts

- 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 each wedge.
- 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
```

#1st code

```
#x=np.array([0,6,8])
#y=np.array([0,250,350])
```

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

#2nd code

```
#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()
```

#3rd code

```
#x=np.array([0,10,20])
#y=np.array([0,350,500])

#plt.plot(x,y,color='red',ls='--',lw=10)
#plt.show()
```

#4th code



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

```
#y2=np.array([20,40,60,80])
```

```
#plt.plot(y1,marker='*')
```

```
#plt.plot(y2)
```

```
#plt.show()
```

#5th code

```
# 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)
```

```
# plt.show()
```

#6th code

```
# x=np.array([1,2,3,4])
```

```
# y=np.array([10,20,30,40])
```

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

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

```
# x=np.array([6,7,8,9])
```

```
# y=np.array([60,70,80,90])
```

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

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

```
# plt.show()
```

#7th code

```
# 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','pink','brown'])
```

```
# plt.scatter(x,y,color=c)
```

```
# plt.show()
```



#8th code

```
# 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,cmap='viridis')
# plt.colorbar()
# plt.show()
```

#9th code

```
# x=np.array(['rose','daisy','lilies','tulips'])
# y=np.array([20,10,40,30])
# plt.bar(x,y,color='red',width=0.5)
# plt.show()
```

#10th code

```
# x=np.array(['rose','daisy','lilies','tulips'])
# y=np.array([20,10,40,30])
# plt.barh(x,y,color='black',height=0.5)
# plt.show()
```

#11th code

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

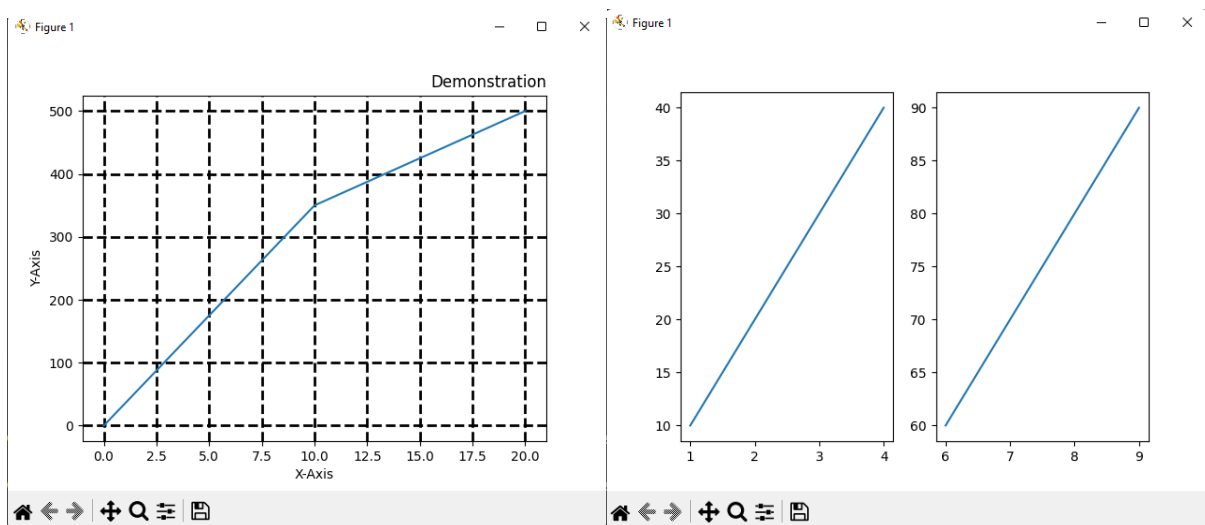
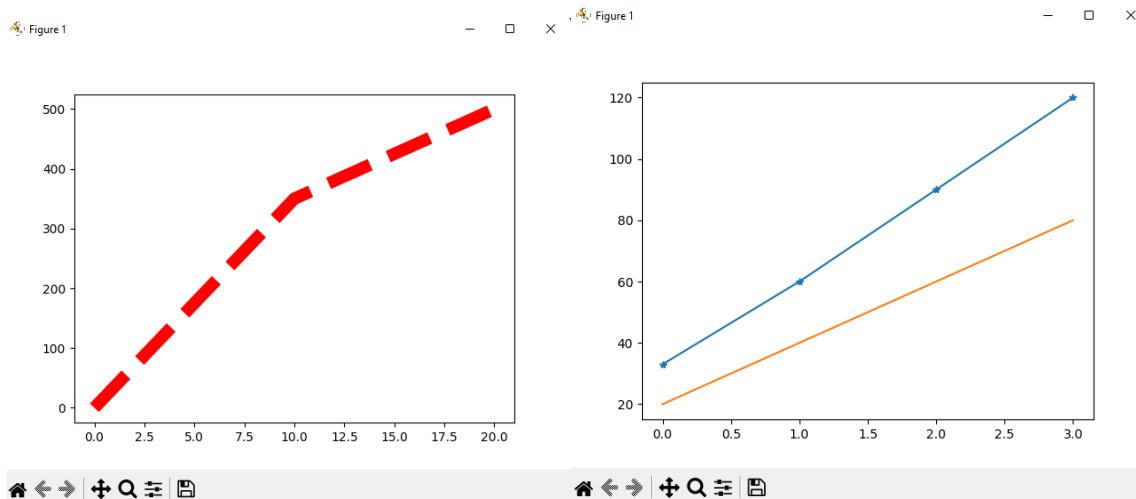
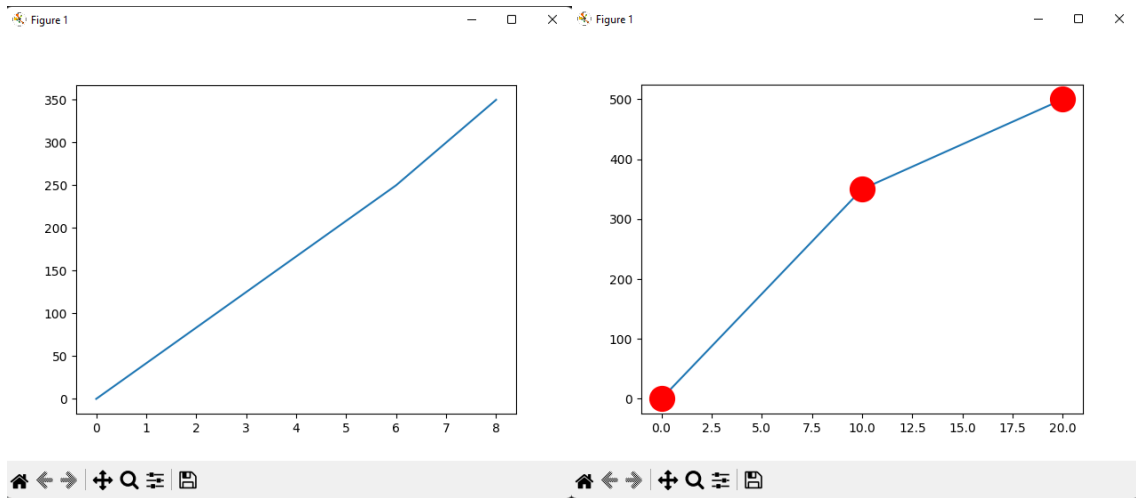
plt.pie(y,labels=l,explode=e)
plt.legend(title="Flower count")
plt.show()
```



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

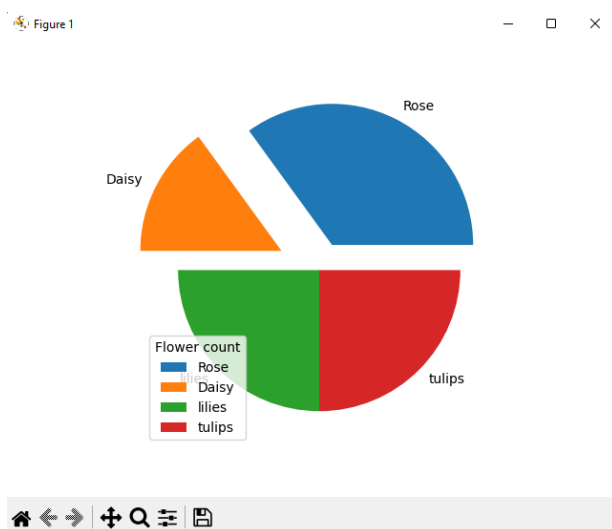
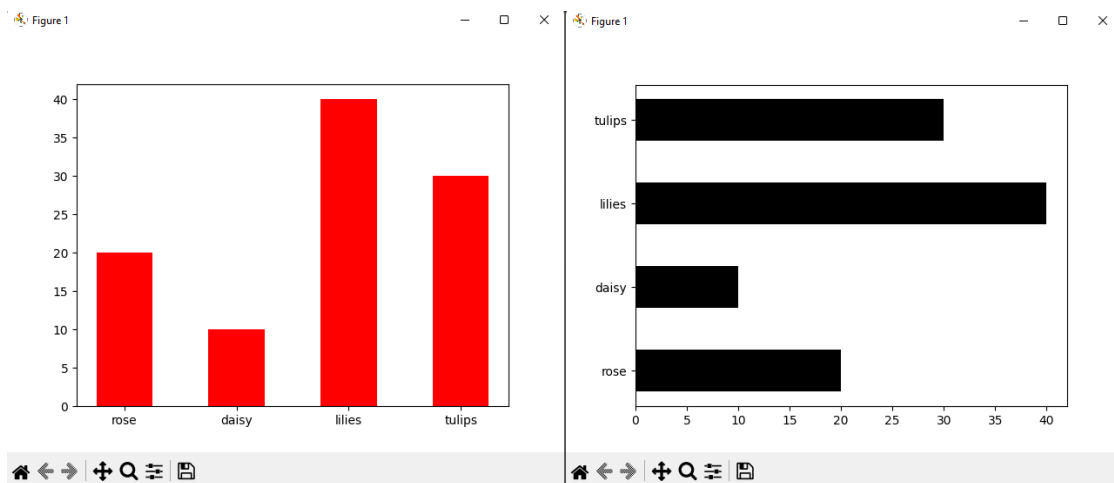
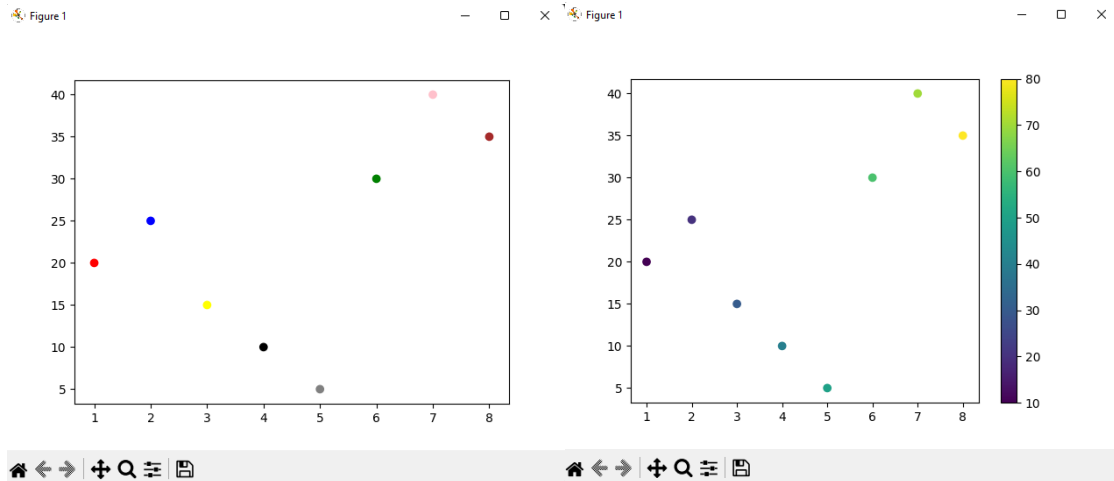
Output:





Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering





Conclusion:

After performing the experiment on data visualization using Matplotlib, it's evident that Matplotlib offers a versatile toolkit for creating a wide range of visualizations, including line plots, scatter plots, bar plots, and pie charts. By leveraging Matplotlib's functionalities, we can effectively explore and analyze data, gaining insights that aid decision-making processes. This experiment underscores the importance of data visualization in conveying information effectively and the utility of Matplotlib in achieving this goal.