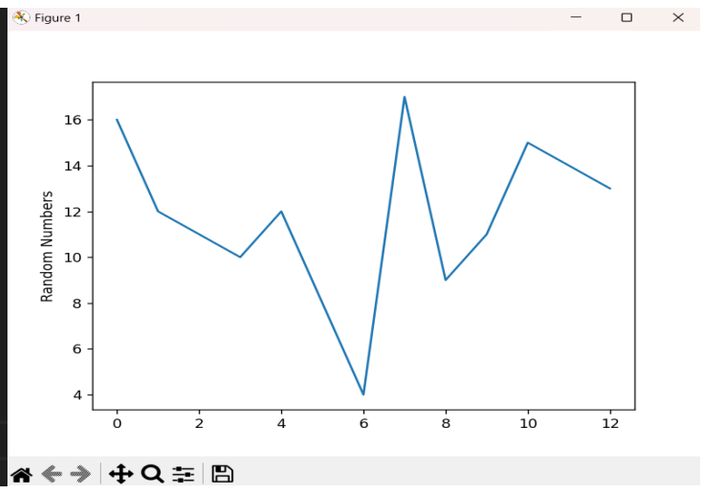
Matplotlib Library

- Data visualization is an important and extremely versatile component of the Python environment.
- Python Programming language basically has two libraries to create charts and graphs, including matplotlib and seaborn
- The matplotlib has many sub-packages which can be accessed by matplotlib.axes, matplotlib.backends, matplotlib.compat, matplotlib.delaunay, matplotlib.projections, matplotlib.pyplot, matplotlib.sphinxext, matplotlib.style, matplotlib.testing, matplotlib.tests, matplotlib.widgets etc.
- The matplotlib.pyplot is a collection of command style functions that make matplotlib work like Matlab software. Matlab is an abbreviation for “matrix laboratory”. It operates on whole matrices and arrays.
- In matplotlib.pyplot, various states are preserved across function calls, hence it is easy to keep track of intermediate factors, such as the nature of figure, plotting area; the plotting functions are used for x- and y- axis.
- **Matplotlib.__version__ :** return the version of the matplotlib

Charts Using plot() Function:

- The plot() is a versatile command and can take an arbitrary number of arguments means we can plot figures corresponding to one axis, for two axes, considering single and multiple data, etc.
- We can plot figures of different shapes and colors.
- For every x and y pairs of arguments, there is a optional third argument, which is the format string that indicates the color and line type of the plot.
- The letters and symbols of the format string are from MATLAB and are used for concatenating a color string with a line style string.
- The default format string is **“b-”**, which is a **solid blue line.**

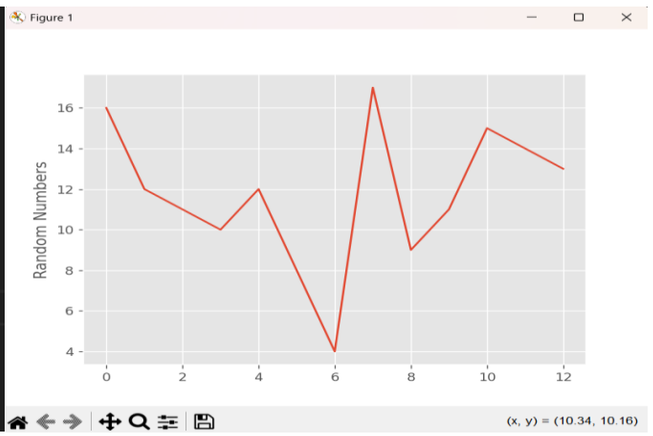
```
#Program for drawing a line chart.
import matplotlib.pyplot as plt

#creating a line chart of single list
plt.plot([16,12,11,10,12,8,4,17,9,11,15,14,13])

#Adding label on y-axis.
plt.ylabel("Random Numbers")

#Displaying a line chart with the random numbers
plt.show()
```



Created by Omesh Sehrawat
Linked link - https://www.linkedin.com/in/omesh-sehrawat-836b40186/
GitHub link - https://github.com/omeshsehrawat/Python_Libraries/tree/main

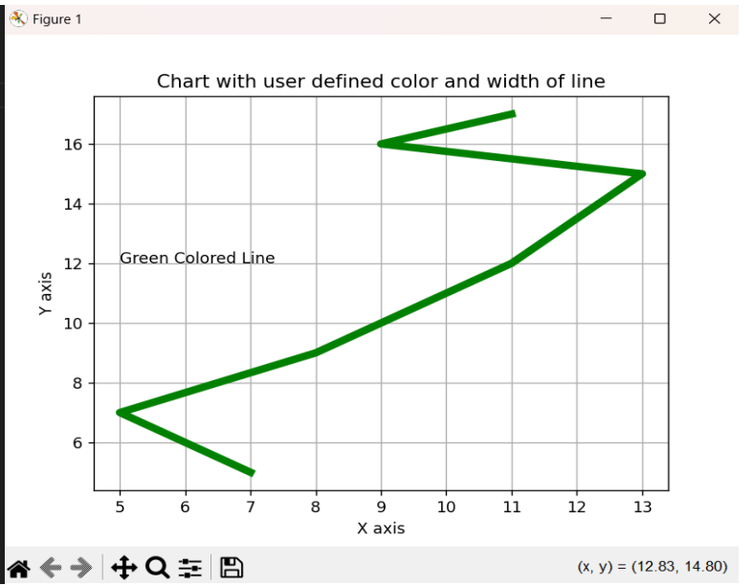
- **“ggplot”** style is imported from matplotlib.style. It changes background color with gray background with axis and red solid line covering all the points.

```
#Program for creating a chart considering gglot from style
import matplotlib.pyplot as plt
import matplotlib.style
matplotlib.style.use('ggplot')

#Creating a line chart with gglot
plt.plot([16,12,11,10,12,8,4,17,9,11,15,14,13])
plt.ylabel('Random Numbers')

#Displaying a chart
plt.show()
```



- **title()** function is used to add title of the chart
- **xlabel()** function is used to add label on x-axis
- **grid()** function is used to add grid to the chart
- **text()** function is add text on the plot at a locations by specifying the location
- We can also control properties of line drawn on the chart by changing its attributes like **linewidth, color, dash style,** etc.

```
#Program for creating line chart with grids and special effects
import matplotlib.pyplot as plt

plt.plot([7,5,8,11,13,9,11],[5,7,9,12,15,16,17], color = 'g', linewidth = 5.0)

#Displaying title of the chart
plt.title('Chart with user defined color and width of line')

#Label on x-axis
plt.xlabel('X axis')

#Label on y-axis
plt.ylabel('Y axis')

#Displaying a grid
plt.grid(True)

#Displayng text on the plot area
plt.text(5, 12, 'Green Colored Line')

#Displaying the chart
plt.show()
```



- We can add limit of axis using **axis()** command.

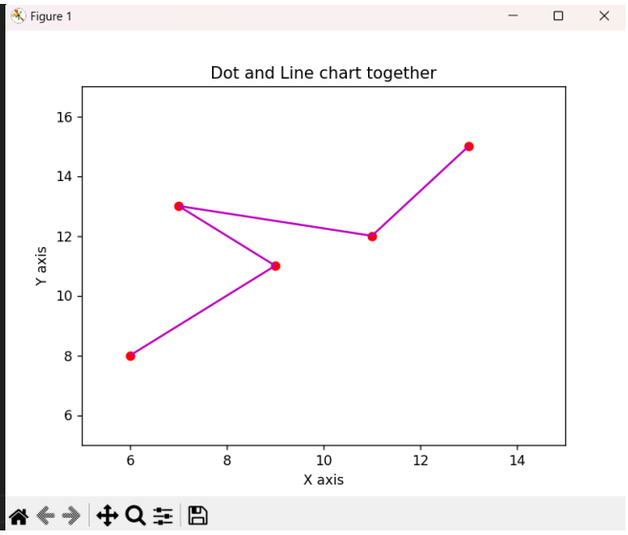
Created by Omesh Sehrawat
Linked link - https://www.linkedin.com/in/omesh-sehrawat-836b40186/
GitHub link - https://github.com/omeshsehrawat/Python_Libraries/tree/main

Using, **axis([xmin, xmax, ymin, ymax]) –** “xmin” and “xmax” specify the minimum and maximum coordinate of x-axis respectively. Similarly, “**ymin”** and “**ymax”** specify the minimum and maximum coordinate of y-axis respectively.

- `plt.plot([6,9,7,11,13], [8,11,13,12,15], 'ro', [6,9,7,11,13], [8,11,13,12,15], 'm')`

‘ro’ plots the read dots and ‘m’ plots magenta-colored line

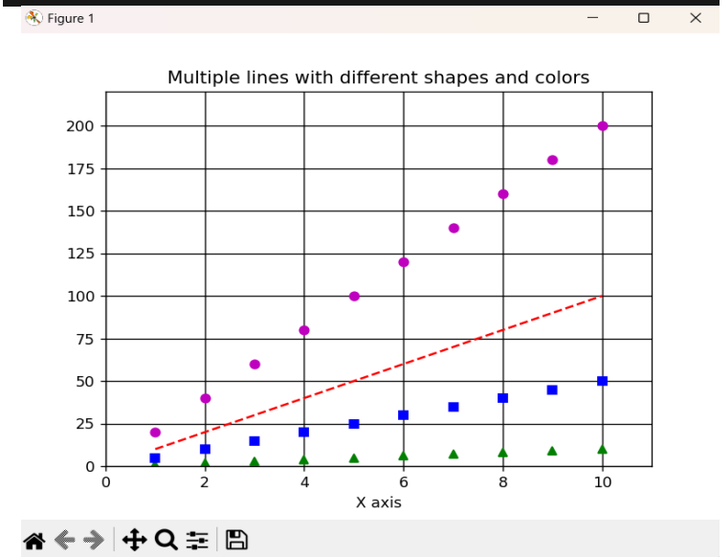
```
#Program to draw dot and lines on the same chart with axes limit
import matplotlib.pyplot as plt
plt.plot([6,9,7,11,13], [8,11,13,12,15], 'ro', [6,9,7,11,13], [8,11,13,12,15], 'm')

#Adding label to the x-axis
plt.xlabel('X axis')

#Adding label to the y-axis
plt.ylabel('Y axis')

#Adding title to the chart
plt.title("Dot and Line chart together")

#Setting the limit of both the axes
plt.axis([5,15,5,17])

#Displaying the chart
plt.show()
```



- A dashed line is drawn using **“—“**, a square is drawn using **“s”**, a triangle using **“^”** symbol, and a circle is drawn using **“o”**.
- Blue dashes will be drawn using **“b—”,** red squares will be drawn using **“rs”**, green circle will be drawn usng **“go”**, and magenta triangle will be drawn using **“m^”.plt.plot([6,9,7,11,13], [8,11,13,12,15], 'ro', [6,9,7,11,13], [8,11,13,12,15], 'm')**
- Unlike the **axis()** command that was used to set the limits of both the axis in one single command, the commands **xlim()** and **ylim()** help to set limits of x- and y-axes, respectively.

```python
#Program to display multiple lines with different shapes and colors
import matplotlib.pyplot as plt

#Creating four lists
a = list(range(1,11))
b = list(range(5,55,5))
c = list(range(10,110,10))
d = list(range(20,210,20))
print("First list:", a)
print("Second list:", b)
print("Third list:", c)
print("Fourth list:", d)

#Creating a chart with different colors and effects for different data
plt.plot(a, a, 'g^', a, b, 'bs', a, c, 'r--', a, d, 'mo')

#Set limits of x-axis
plt.xlim(0, 11)

#Set limits of y-axis
plt.ylim(0, 220)

plt.xlabel('X axis')
plt.title('Multiple lines with different shapes and colors')

#Displaying grid in chart
plt.grid(True, color='k')

#Displaying the chart
plt.show()
```

```
First list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Second list: [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
Third list: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
Fourth list: [20, 40, 60, 80, 100, 120, 140, 160, 180, 200]
```

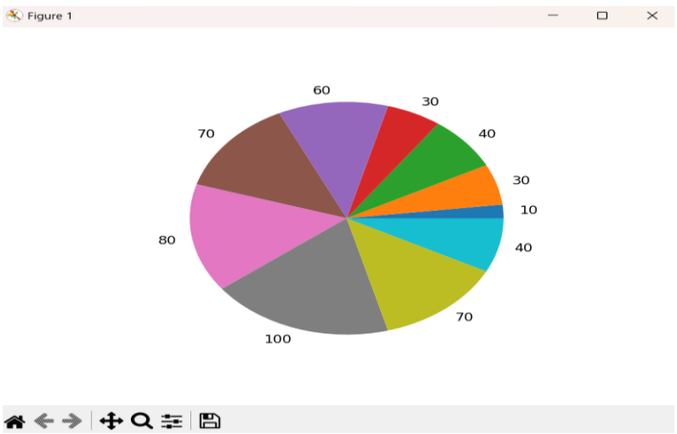


Different Charts:

➢ Pie Chart:

- Pie charts visualize absolute and relative frequencies.
- A pie chart is a circle partitioned into segments where each of the segments represents a category which depends upon the relative frequency and is determined by the angle.
- A pie chart is drawn using **pie()** function considering single list as an argument.

```python
#Program for creating a pie chart
import matplotlib.pyplot as plt
list1 = [10,30,40,30,60,70,80,100,70,40]

#Creating and displaying a pie chart
plt.pie(list1, labels=list1)
plt.show()
```

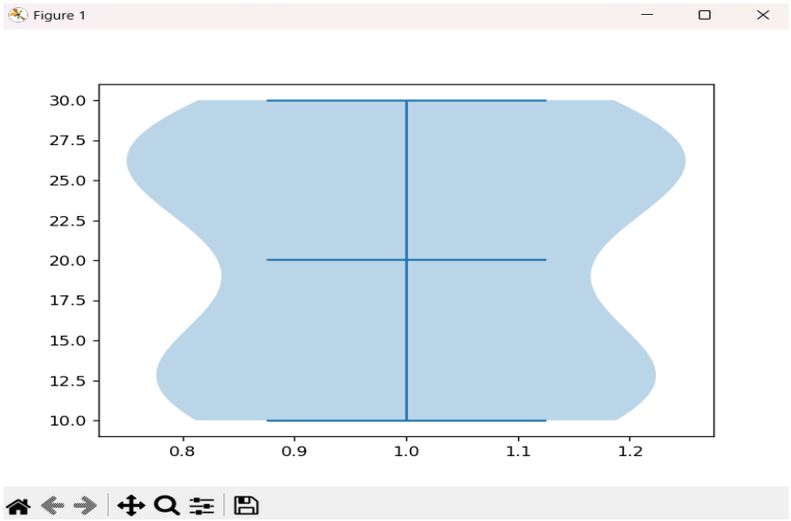


Created by Omesh Sehrawat
Linked link - https://www.linkedin.com/in/omesh-sehrawat-836b40186/
GitHub link - https://github.com/omeshsehrawat/Python_Libraries/tree/main

➢ Violin Plot:

- Violin plot is a combination of box and kernel density plot.
- It is drawn using **violinplot()** from matplotlib.pyplot.
- **Violinplot(data_set, showmeans=True/Flase, showextrema=True/False) –** showmeans represent the mean of the dataset and showextrema display the extreme points of dataset.

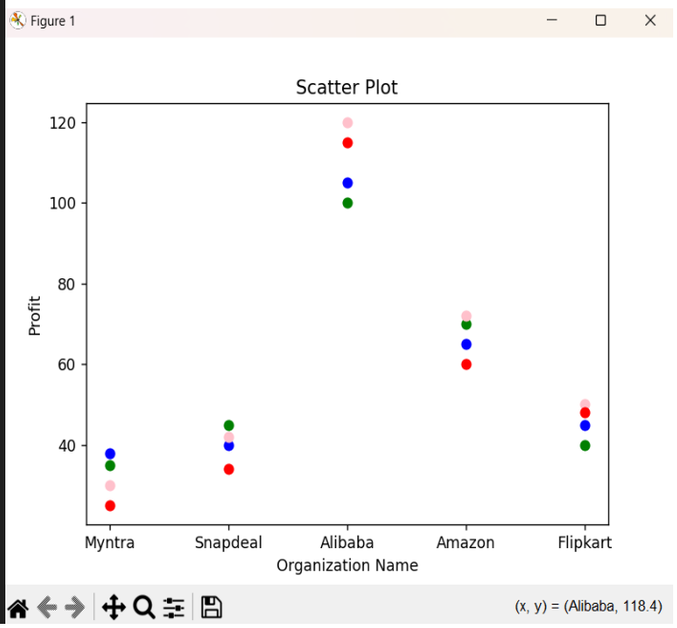
```
#Program to draw a violin plot
import matplotlib.pyplot as plt
plt.violinplot([10,11,12,18,23,27,29,30,26,28,24,23,14,12,16,12,12,27,27], showmeans=True, showextrema=True)

#Displaying the chart
plt.show()
```



➢ Scatter Plot:

- The scatter plot is created using the **scatter()** function and is helpful is displaying bivariate data.
- Scatter plots show many points plotted in the Cartesian plane.
- Each point represents the values of two variables.
- One variable is chosen in the horizontal axis and another in the vertical axis.
- The **“color”** is the default argument in the function which represents the color of the dot.

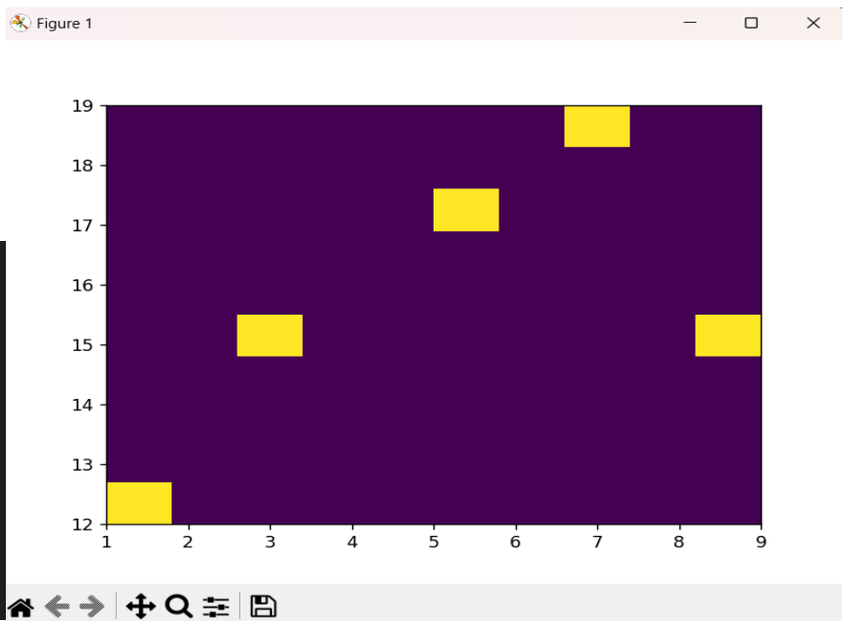
```python
#Program to draw a scatter plot
import matplotlib.pyplot as plt
ecommerce = ['Myntra', 'Snapdeal', 'Alibaba', 'Amazon', 'Flipkart']
Q1_Profit = [35,45,100,70,40]
Q2_Profit = [38,40,105,65,45]
Q3_Profit = [30,42,120,72,50]
Q4_Profit = [25,34,115,60,48]

#Creating a scatter plot
plt.scatter(ecommerce, Q1_Profit, color='green')
plt.scatter(ecommerce, Q2_Profit, color='blue')
plt.scatter(ecommerce, Q3_Profit, color='pink')
plt.scatter(ecommerce, Q4_Profit, color='red')

#Adding title to the chart and labels to the axis
plt.xlabel('Organization Name')
plt.ylabel('Profit')
plt.title('Scatter Plot')

#Displaying a scatter plot
plt.show()
```



➢ Histogram:

- Histogram is based on the idea to categorize data into different groups and plot the bars for each category with height.
- A histogram represents the frequencies of values of a variable gathered into ranges.
- Each bar in a histogram represents the height of the number of values present in that range.

```python
#Program to display a histogram
import matplotlib.pyplot as plt

#Creating a histogram using hist2d() function
list1 = list(range(1,10,2))
list2 = [12,15,17,19,15]
plt.hist2d(list1, list2)

#Displaying the histogram
plt.show()
```



➢ Creating Multiple Charts on One Image:

Created by Omesh Sehrawat
Linked link - https://www.linkedin.com/in/omesh-sehrawat-836b40186/
GitHub link - https://github.com/omeshsehrawat/Python_Libraries/tree/main

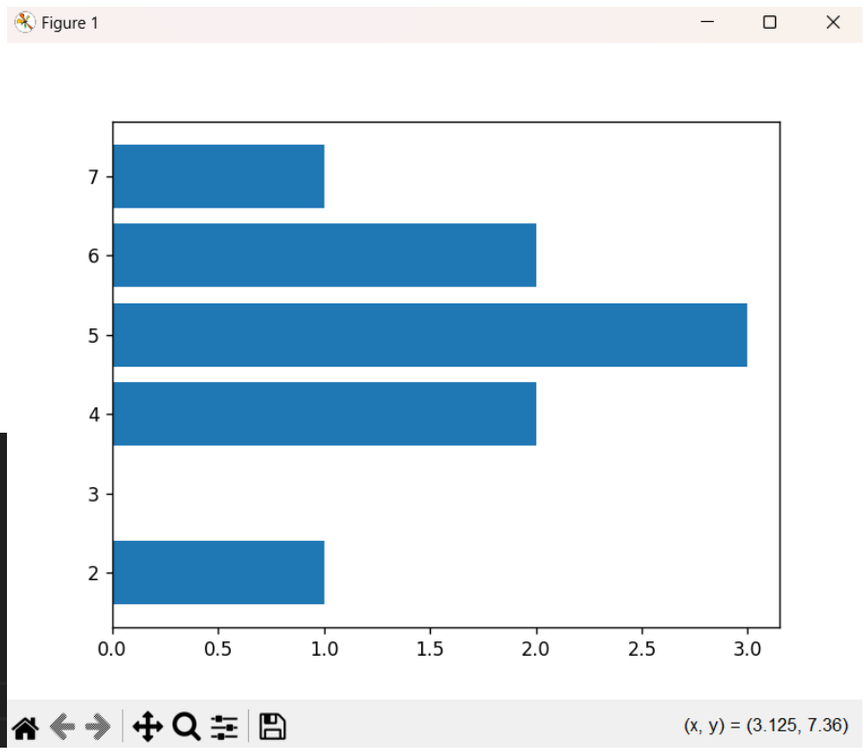
- We can have multiple figures on one chart in the form of m x n array of charts using the function **subplot().**
- **Syntax: subplot(first_arg, second_arg, third_arg)**
  first_arg – denotes no. of rows, **second_arg –** denotes no. of columns, **third_arg –** denotes the cell no.
  eg.- (4,3,2)- second cell means second column of first row.
  (4,3,10) – tenth cell means second column of third row.

- **figure()** function represents the number of figure starting from 1, it has **figsize** as the argument which depicts the dimensions of the figure.
- When we draw multiple charts on one chart, we give a super title to the main image using **suptitle()** function.

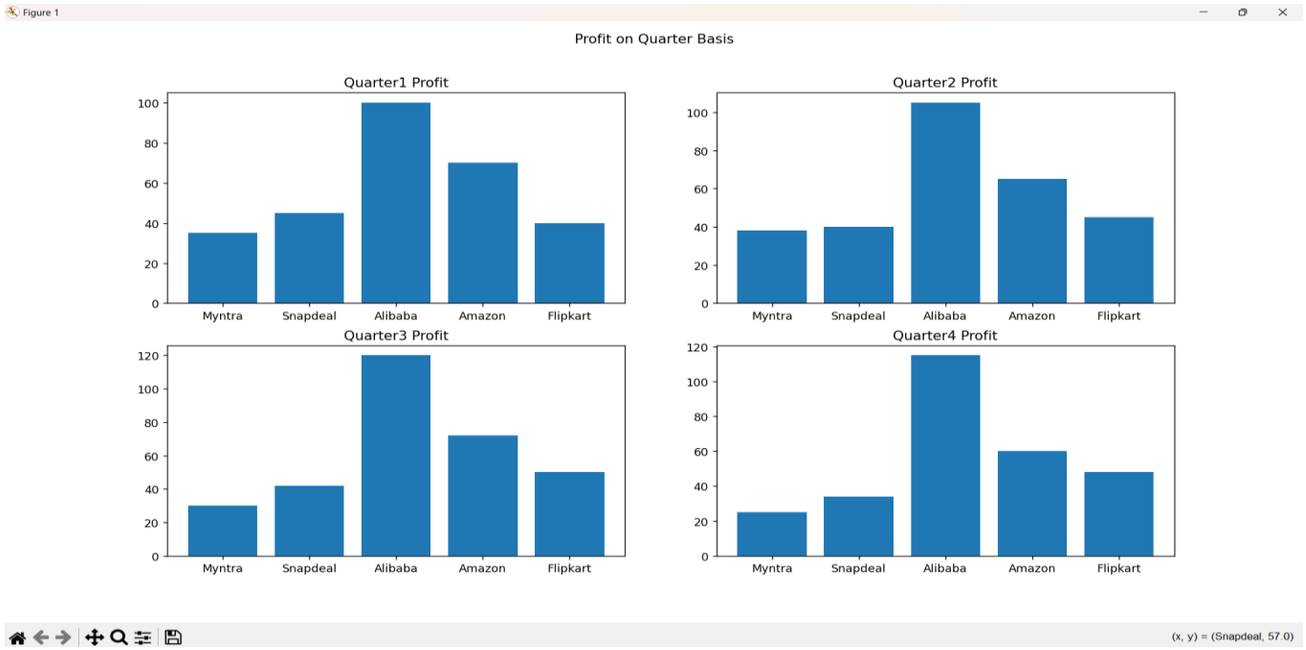
➢ Bar Chart:

- A bar chart visualizes the relative or absolute frequencies of observed values of variable.
- It consists of one bar for each category. A bar chart represents data in rectangular bars with length of the bar proportional to the value of the variable.
- Python uses the function **bar()** to create bar charts.
- The height of each bar is determined by either the absolute frequency or the raltive frequency of the respective category and is shown on the y-axis.
- The horizontal bar chart can be created by using the **barh()** function.

```
import matplotlib.pyplot as plt

#Creating a horizontal barplot
plt.barh([2,4,5,6,7], width=[1,2,3,2,1])

#Displaying the chart
plt.show()
```

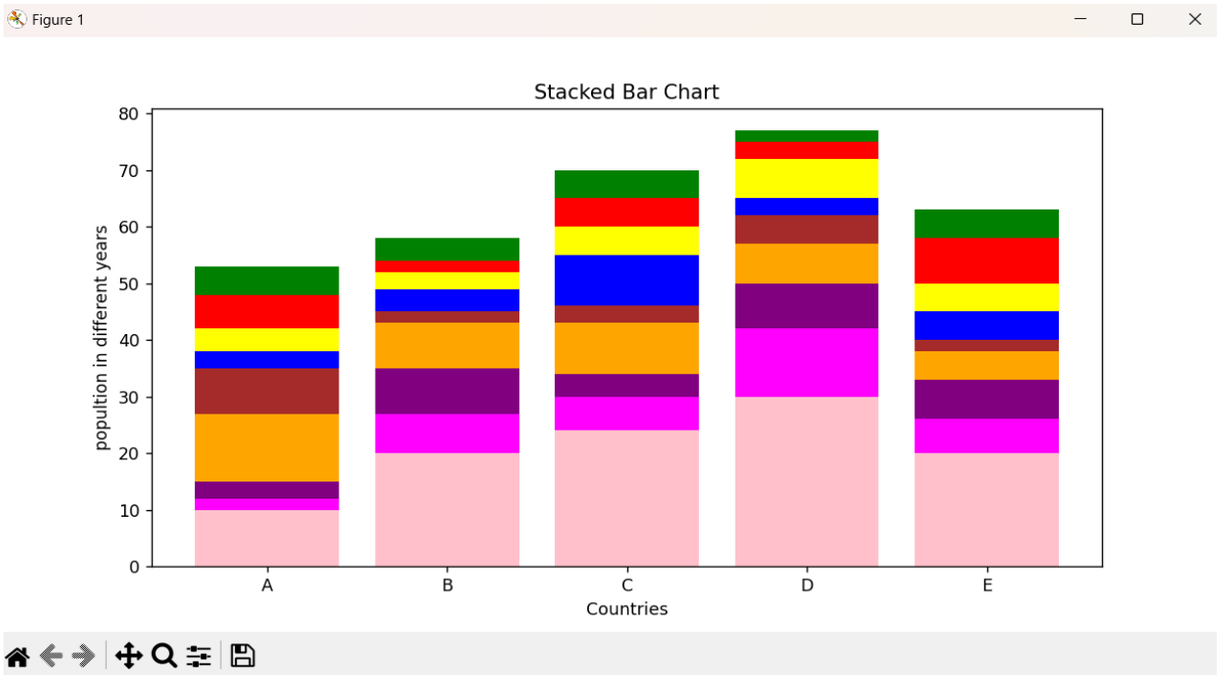


```python
#Program for drawing multiple bar charts on one image
import matplotlib.pyplot as plt
ecommerce = ['Myntra', 'Snapdeal', 'Alibaba', 'Amazon', 'Flipkart']
Q1_Profit = [35,45,100,70,40]
Q2_Profit = [38,40,105,65,45]
Q3_Profit = [30,42,120,72,50]
Q4_Profit = [25,34,115,60,48]

#Creating different bar charts on one image using subplot() function
plt.figure(1, figsize=(10, 10))

#Creating bar chart in first cell of figure having 2 rows, 3 columns
plt.subplot(221)
plt.bar(ecommerce, Q1_Profit)
plt.title("Quarter1 Profit")

#Creating a bar chart in second cell
plt.subplot(2,2,2)
plt.bar(ecommerce, Q2_Profit)
plt.title('Quarter2 Profit')

#Creating a bar chrat in third cell
plt.subplot(223)
plt.bar(ecommerce, Q3_Profit)
plt.title('Quarter3 Profit')

#Creating a bar chart in fourth cell
plt.subplot(2,2,4)
plt.bar(ecommerce, Q4_Profit)
plt.title('Quarter4 Profit')

#Adding a Main title for the figure
plt.suptitle('Profit on Quarter Basis')

#Displaying the chart
plt.show()
```



- A **stacked bar** chart can be drawn for multiple data in which each bars can be given different colors using “color” argument in the function.

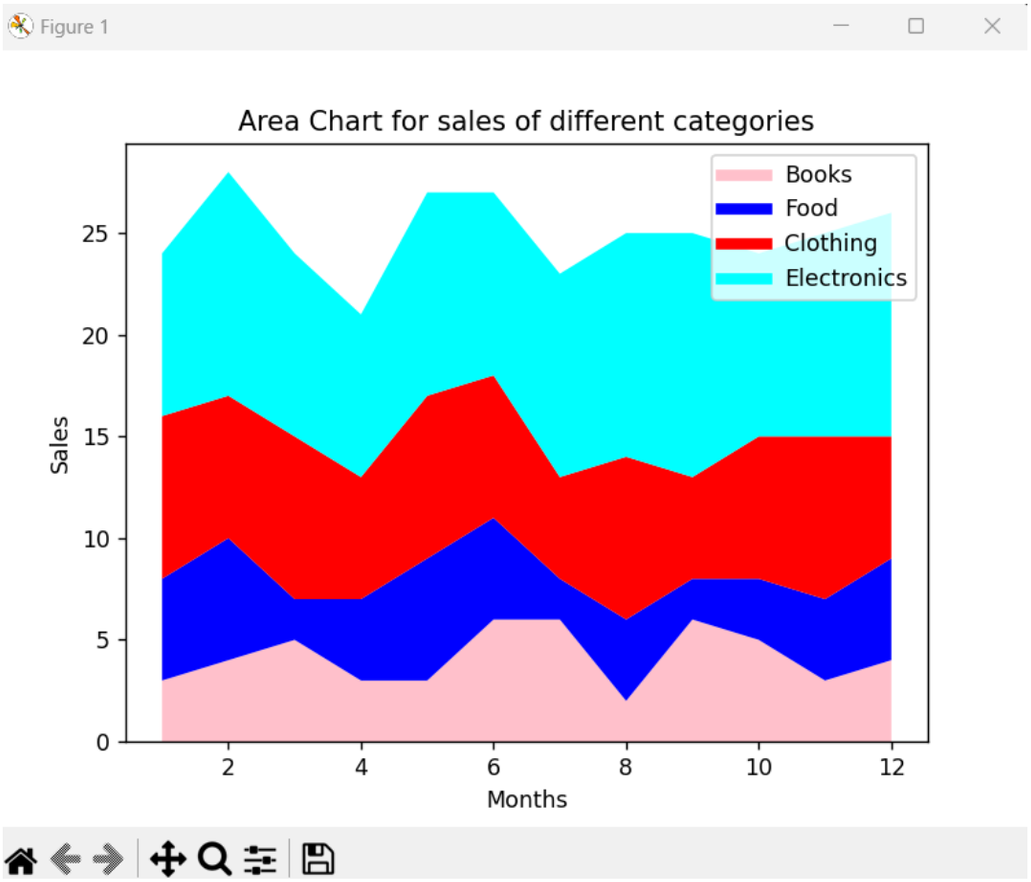
```
#Program to create a stacked bar chart
import matplotlib.pyplot as plt
plt.figure(1, figsize=(10,5))
countries = ['A', 'B', 'C', 'D', 'E']
Population_1930 = [10,20,24,30,20]
Population_1940 = [12,27,30,42,26]
Population_1950 = [15,35,34,50,33]
Population_1960 = [27,43,43,57,38]
Population_1970 = [35,45,46,62,40]
Population_1980 = [38,49,55,65,45]
Population_1990 = [42,52,60,72,50]
Population_2000 = [48,54,65,75,58]
Population_2010 = [53,58,70,77,63]

#Creating a stacked bar chart
plt.bar(countries, Population_2010, color='green')
plt.bar(countries, Population_2000, color='red')
plt.bar(countries, Population_1990, color='yellow')
plt.bar(countries, Population_1980, color='blue')
plt.bar(countries, Population_1970, color='brown')
plt.bar(countries, Population_1960, color='orange')
plt.bar(countries, Population_1950, color='purple')
plt.bar(countries, Population_1940, color='magenta')
plt.bar(countries, Population_1930, color='pink')

#Adding labels and title to the chart
plt.xlabel('Countries')
plt.ylabel('popultion in different years')
plt.title('Stacked Bar Chart')

#Displaying a bar chart
plt.show()
```



➢ Area Plot:

- Area plots are pretty much similar to the line plot. They are also known as **stack plots**.
- These plots can be used to track changes over time for two or more related groups that make up one whole category.

```
#Program to create an area plot
import matplotlib.pyplot as plt
month_sales = [1,2,3,4,5,6,7,8,9,10,11,12]
Electronics = [8,11,9,8,10,9,10,11,12,9,10,11]
Clothing = [8,7,8,6,8,7,5,8,5,7,8,6]
Food = [5,6,2,4,6,5,2,4,2,3,4,5]
Books = [3,4,5,3,3,6,6,2,6,5,3,4]

plt.plot([], [], color='pink', label='Books', linewidth=5)
plt.plot([], [], color='blue', label='Food', linewidth=5)
plt.plot([], [], color='red', label='Clothing',  linewidth=5)
plt.plot([], [], color='cyan', label='Electronics', linewidth=5)

plt.stackplot(month_sales, Books, Food, Clothing, Electronics, colors=['pink', 'blue', 'red', 'cyan'])
plt.xlabel('Months')
plt.ylabel('Sales')
plt.title('Area Chart for sales of different categories')
plt.legend()

#Displaying the chart
plt.show()
```

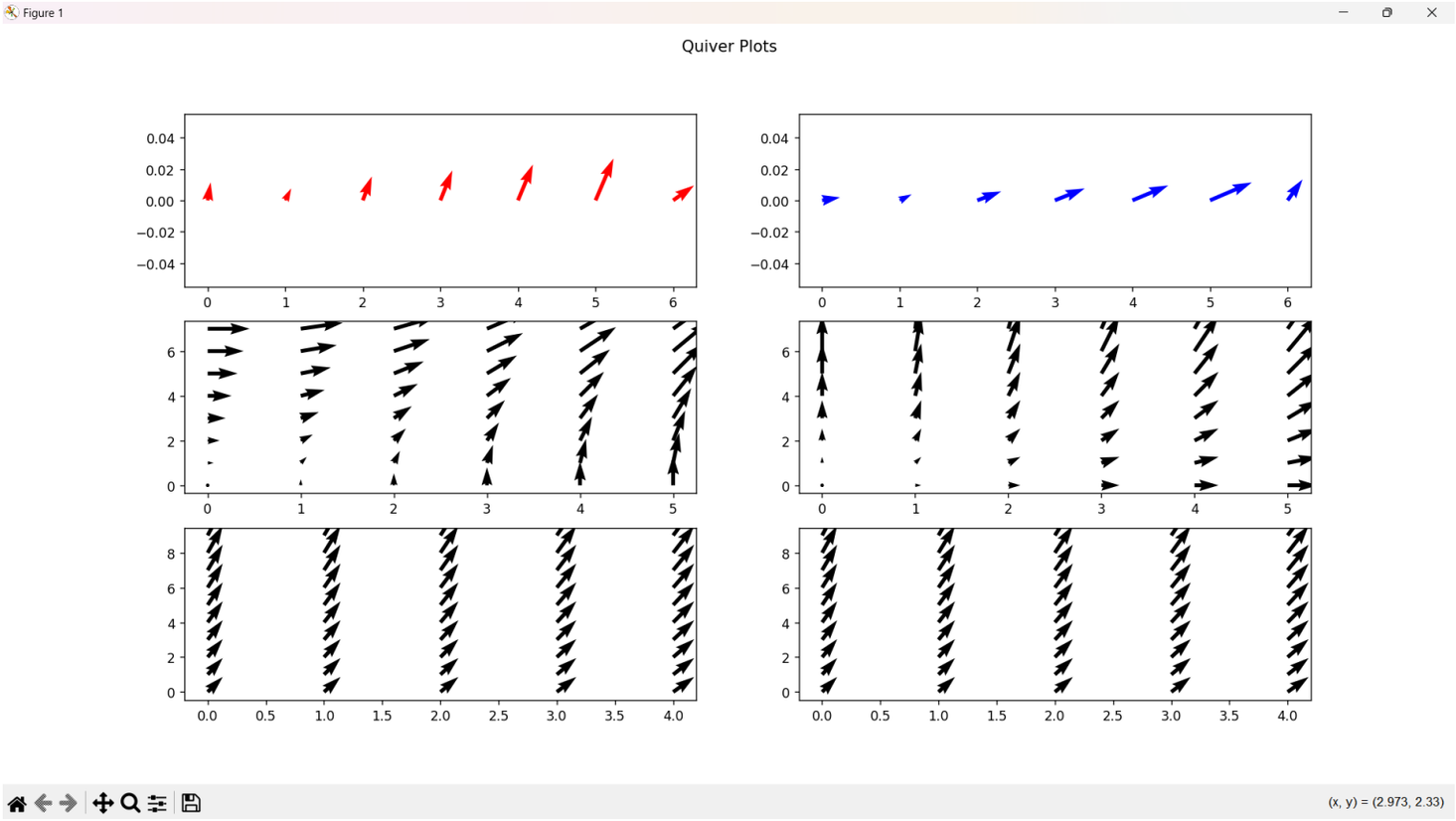


➢ Quiver Plot:

- The **quiver()** function takes two important arguments that represents 1-D or 2-D arrays or squences.
- The **quiver()** function plots a 2-D field of arrows representing the two datasets.
- Color is an optional argument which represents the color of the arrow.

```python
#Program to draw multiple quiver plots in single image
import matplotlib.pyplot as plt
import numpy as np
a = [10,20,30,40,50,60,70]
b = [60,40,80,100,120,140,50]
plt.figure(1, figsize=(15,15))

#Creating a quiver plot in first cell of image having 3 rows and 2 columns
plt.subplot(3,2,1)
plt.quiver(a, b, color='r')

#Creating a quiver plot in second cell
plt.subplot(3,2,2)
plt.quiver(b, a, color='b')

#Creating a quiver plot in third cell
plt.subplot(3,2,3)
x = 8
y = 6
M,N = np.mgrid[0:x, 0:y]
plt.quiver(M,N)

#Creating a quiver plot in fourth cell
plt.subplot(3,2,4)
plt.quiver(N,M)

#Creating a quiver plot in fifth cell
plt.subplot(3,2,5)
x = 20
y = 15
M,N = np.mgrid[10:x, 10:y]
plt.quiver(N, M)

#Creating a quiver plot in sixth cell
plt.subplot(3,2,6)
plt.quiver(N,M)
plt.suptitle('Quiver Plots')
#Displaying the image
plt.show()
```

Created by Omesh Sehrawat
Linked link - https://www.linkedin.com/in/omesh-sehrawat-836b40186/
GitHub link - https://github.com/omeshsehrawat/Python_Libraries/tree/main

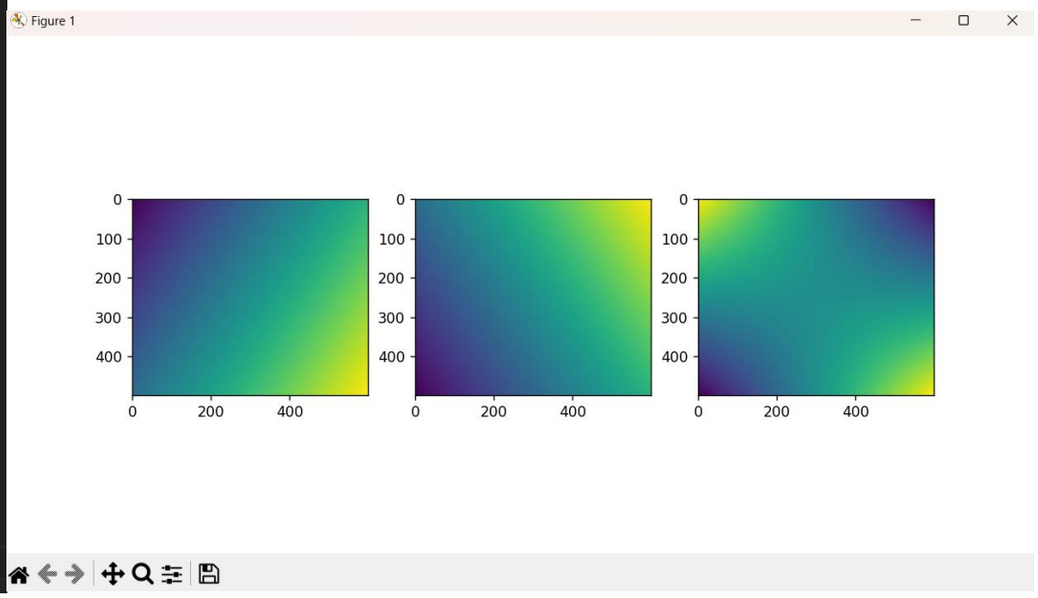


➢ Mesh Grid:

- The purpose of meshgrid is to create a rectangular grid out of an array of x values and an array of y values.
- A meshgrid is created using **meshgrid()** function.
- This chart is used for analysis of large data

Created by Omesh Sehrawat
Linked link - https://www.linkedin.com/in/omesh-sehrawat-836b40186/
GitHub link - https://github.com/omeshsehrawat/Python_Libraries/tree/main

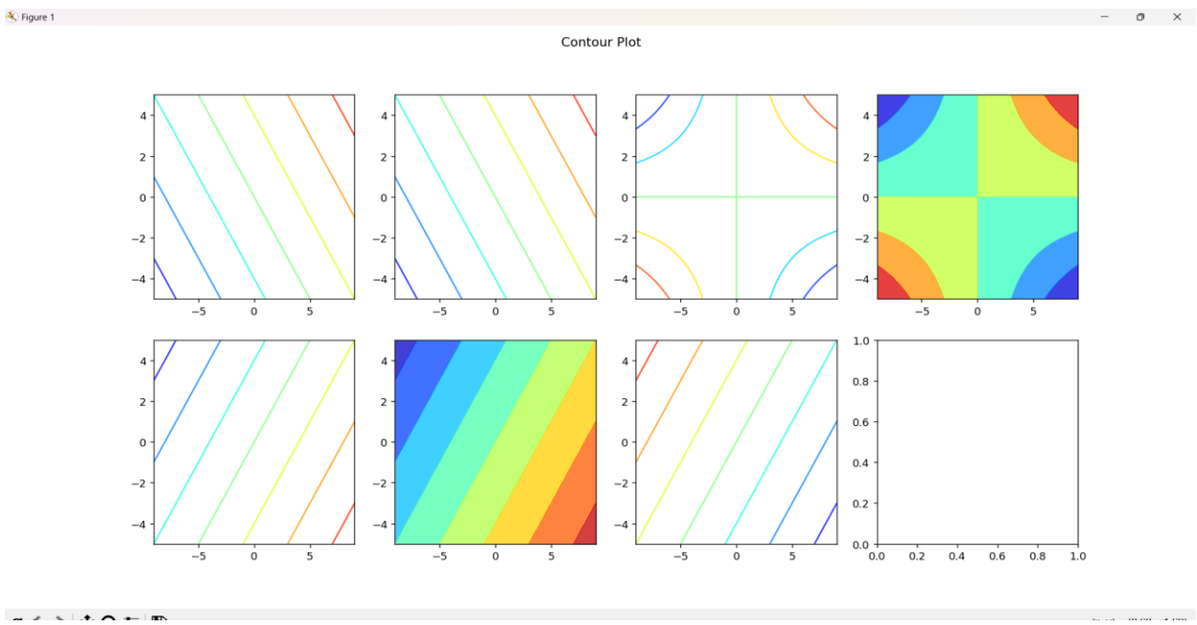
```python
#Program for creating a meshgrid
import matplotlib.pyplot as plt
import numpy

#Creating large dataset
val1 = numpy.linspace(-9, 9, 600)
val2 = numpy.linspace(-5, 5, 500)

#Creating a meshgrid chart
A,B = numpy.meshgrid(val1, val2)
plt.figure(1, figsize=(10,5))
plt.subplot(131)
plt.imshow(A+B)
plt.subplot(132)
plt.imshow(A-B)
plt.subplot(133)
plt.imshow(A*B)

plt.show()
```



➢ Contour Plot:

- A contour plot is a graphical technique for representing a three-dimensional surface and is created using **contour().**
- It is drawn by plotting constant **z** slices, called **contours,** on a two-dimensional format.
- Variables **"x"** and **"y"** impact the variable **"z".**

```python
#Program to create a contour plot
import matplotlib.pyplot as plt
import numpy as np

#Creating large datasets
val1 = np.linspace(-9, 9, 600)
val2 = np.linspace(-5, 5, 500)

#Creating a meshgrid chart
A,B = np.meshgrid(val1, val2)
plt.figure(1, figsize=(15, 10))
plt.subplot(2, 4, 1)
plt.contour(A, B, A+B, alpha=.75, cmap='jet')
plt.subplot(2,4,2)
plt.contour(A, B, A+B, alpha=.75, cmap='jet')
plt.subplot(2,4,3)
plt.contour(A, B, A*B, alpha=.75, cmap='jet')
plt.subplot(2,4,4)
plt.contourf(A, B, A*B, alpha=.75, cmap='jet')
plt.subplot(2,4,5)
plt.contour(A, B, A-B, alpha=.75, cmap='jet')
plt.subplot(2,4,6)
plt.contourf(A, B, A-B, alpha=.75, cmap='jet')
plt.subplot(2,4,7)
plt.contour(A, B, B-A, alpha=.75, cmap='jet')
plt.subplot(2,4,8)

#Displaying the contour plot
plt.suptitle('Contour Plot')
plt.show()
```



Created by Omesh Sehrawat
Linked link - https://www.linkedin.com/in/omesh-sehrawat-836b40186/
GitHub link - https://github.com/omeshsehrawat/Python_Libraries/tree/main