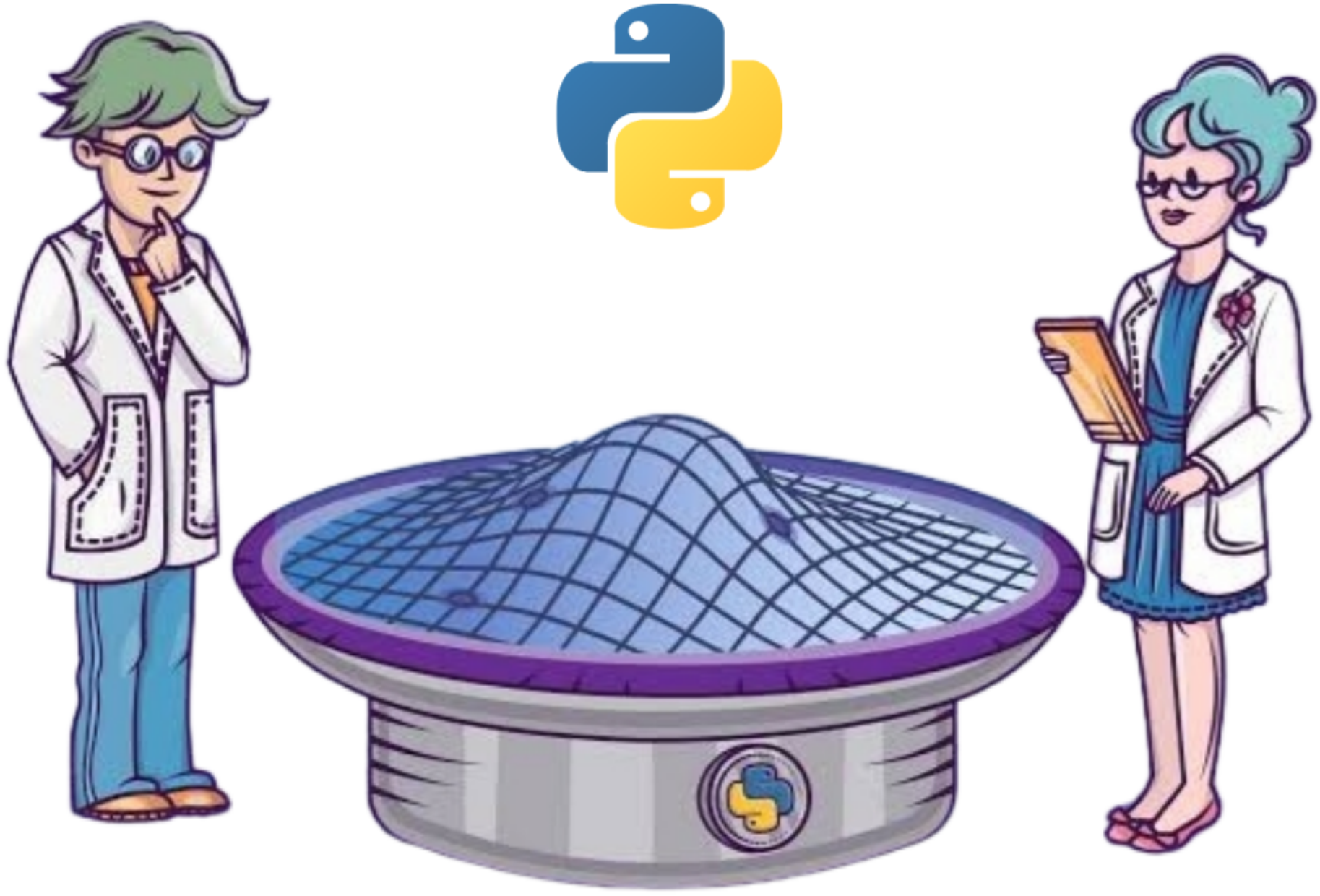
matplotlib



BEGINNER'S CODE GUIDE

matplotlib

Python library
for detailed graphs

pyplot

- is a Matplotlib module.
- Creates figure, labels, and plot area

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

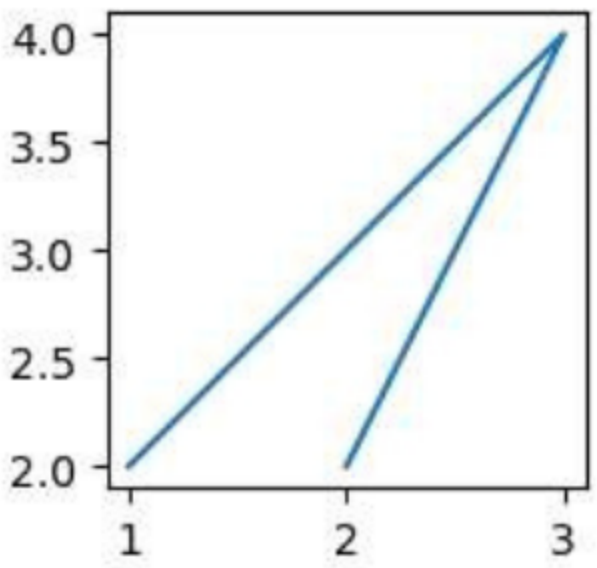
plot()

Draws points
with default connecting lines

syntax: plot(x aixs, y axis)

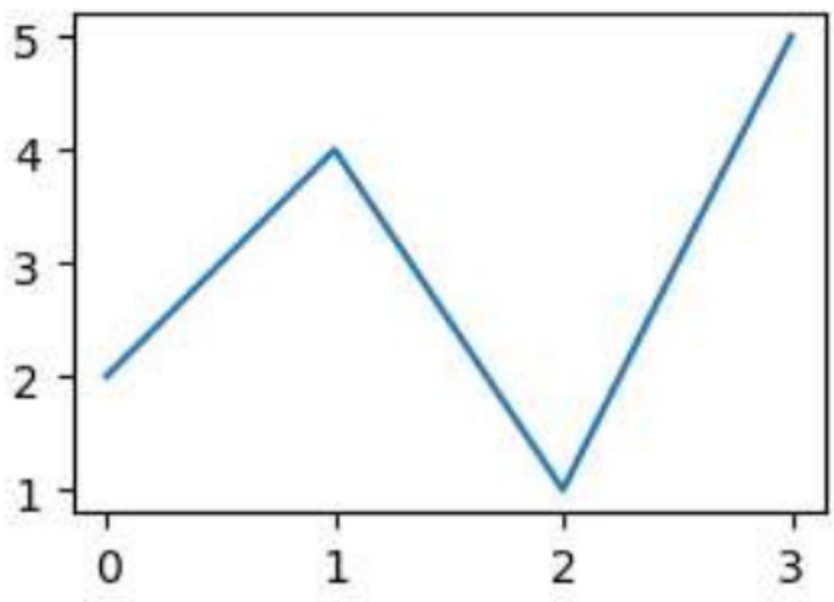
e.g. draw line in diagram

```
# Basic Line graph using plot()
plt.plot([1,3,2],[2,4,2])

# show plot & hide default title
plt.show()
```



Default X points values 0,1,2,3 ...

```
import numpy as np

ypoints = np.array([2, 4, 1, 5])

#Plotting without x-points
plt.figure()

plt.plot(ypoints)   # y axis

plt.show()
```



Format Strings

marker | line | color

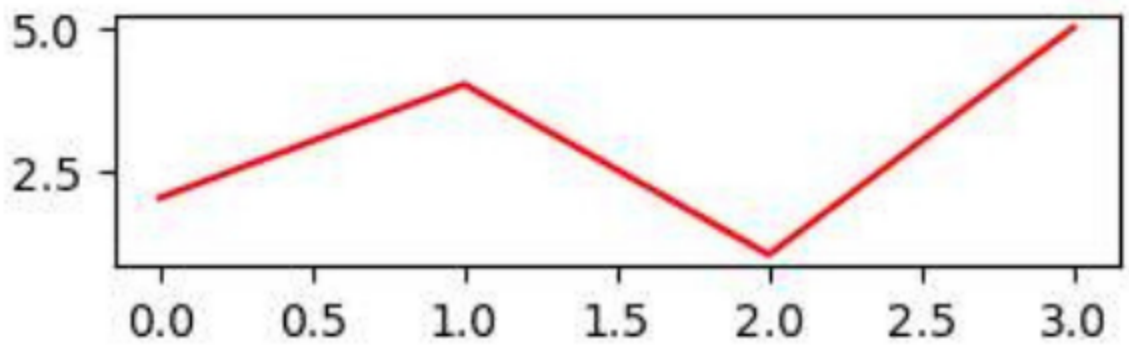
optional 3rd argument in plot() for 'marker,'line type' & 'color'

Line Color Change

'color' or 'c' or use shortcut string

- ways to specify colors:-
  - Named Colors: 'red', 'blue', 'green'
  - Short Color Codes: 'r' for red, 'b' for blue, 'g' for green
  - RGB & Hexadecimal Color Codes

```
plt.plot(ypoints,'r')


plt.show()
```

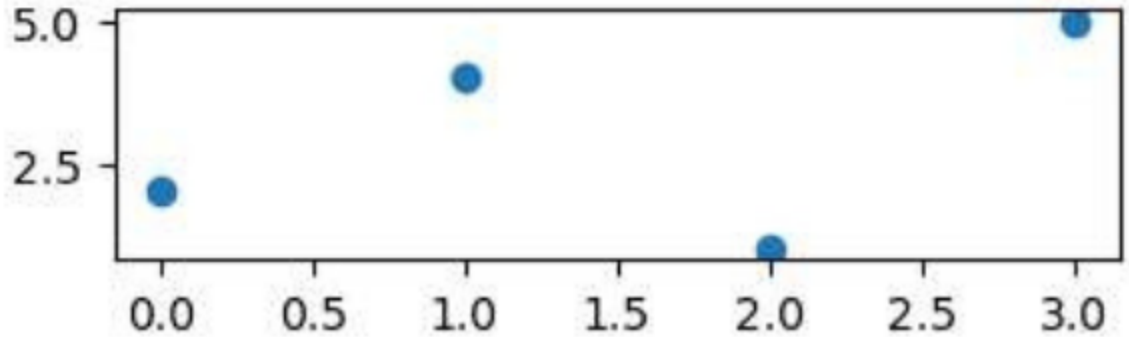


Change Line Style

'linestyle' or 'ls' or use shortcut string

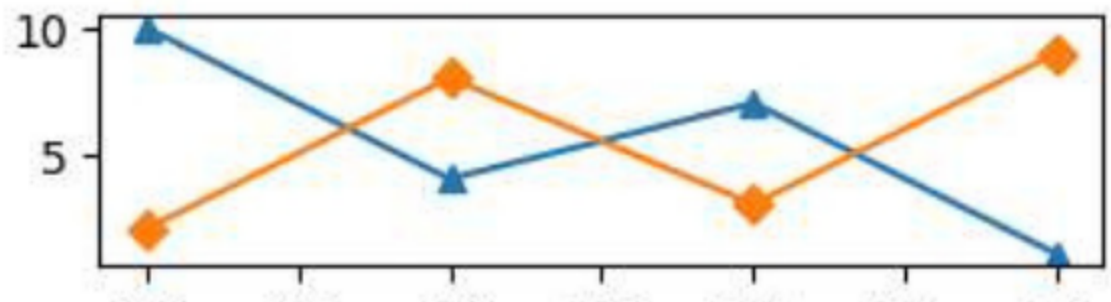
e.g: '--', '-.', '.', ',', 'o', 'v', '^', '<', '>', '+', '|', '_', '*', ':'

```
plt.plot(ypoints,'o')


plt.show()
```



Multiple Lines Plot x & y axis seperately

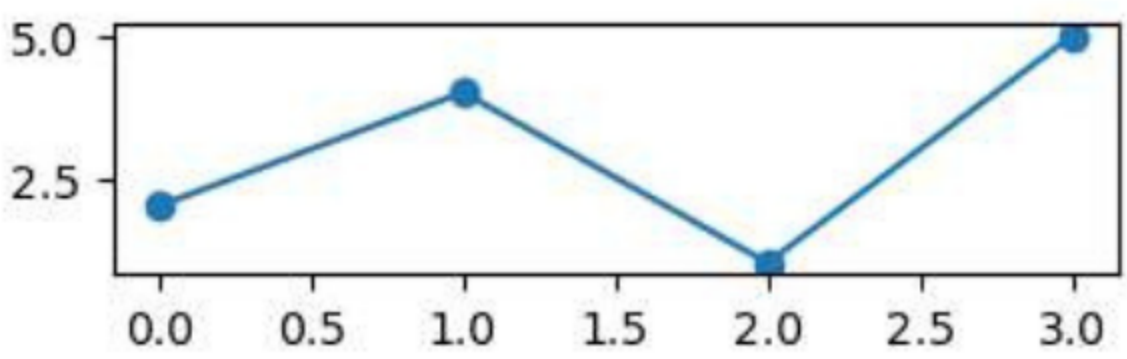
```
x = [10,4,7,1]
y = [2,8,3,9]

plt.plot(x, marker = '^')
plt.plot(y, marker = 'D')

plt.show()
```



Marker mark the data points on a line plot

```
styles available: 'o','s','D','d',
'x','X','P','p','H','h','v','1','2','3','4'
'^','<','>','*','.', ',' ,'|','_','+'
```

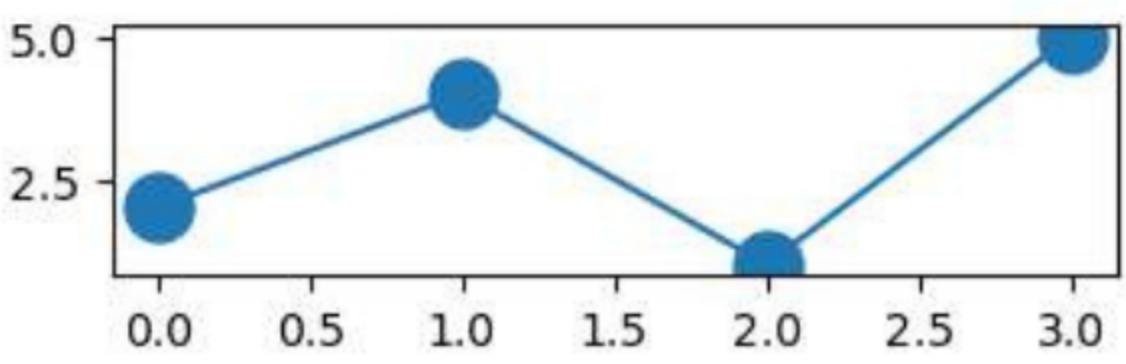
```
plt.plot(ypoints, marker = 'o')

plt.show()
```



Marker Size

argument 'markersize' or the shorter version, 'ms'

```
plt.plot(ypoints, marker = 'o',
         ms = 15)

plt.show()
```

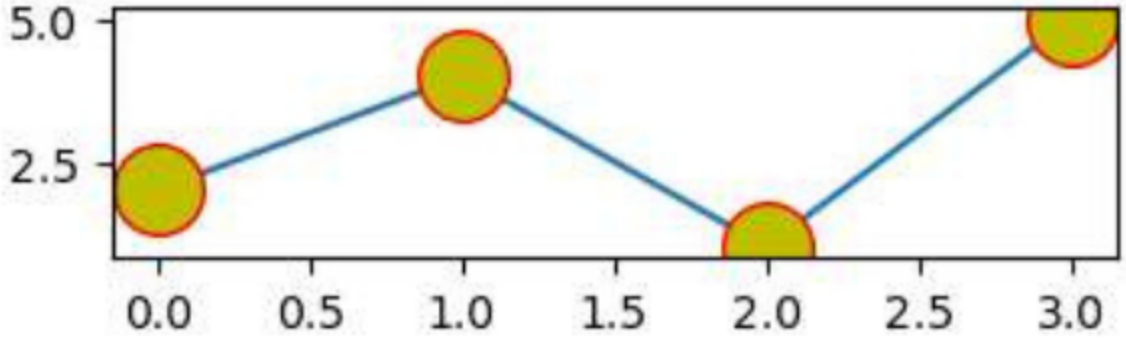


Marker Color

'markeredgecolor' or 'mec' to set the color of the edge

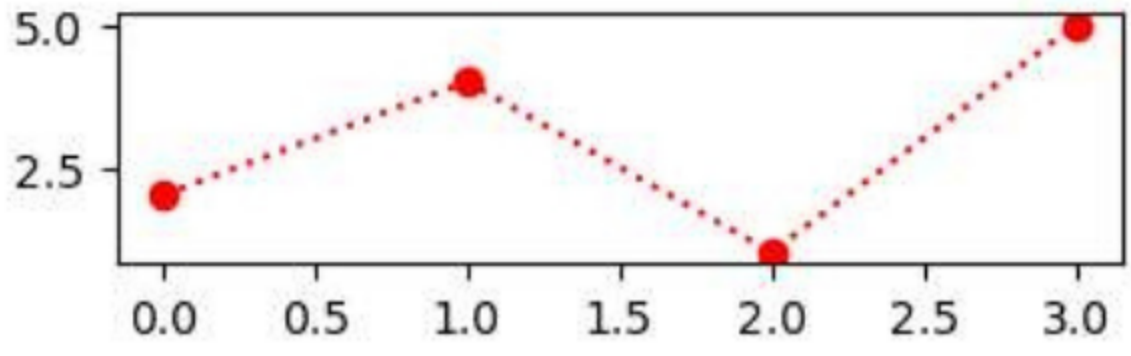
'markerfacecolor' or 'mfc' to set the color inside the edge

```
plt.plot(ypoints, marker = 'o',
   ms = 20, mec = 'r', mfc = 'y')

plt.show()
```



Marker + Line + Color

```
#marker='o',line=':',color red 'r'
plt.plot(ypoints, 'o:r')

plt.show()
```

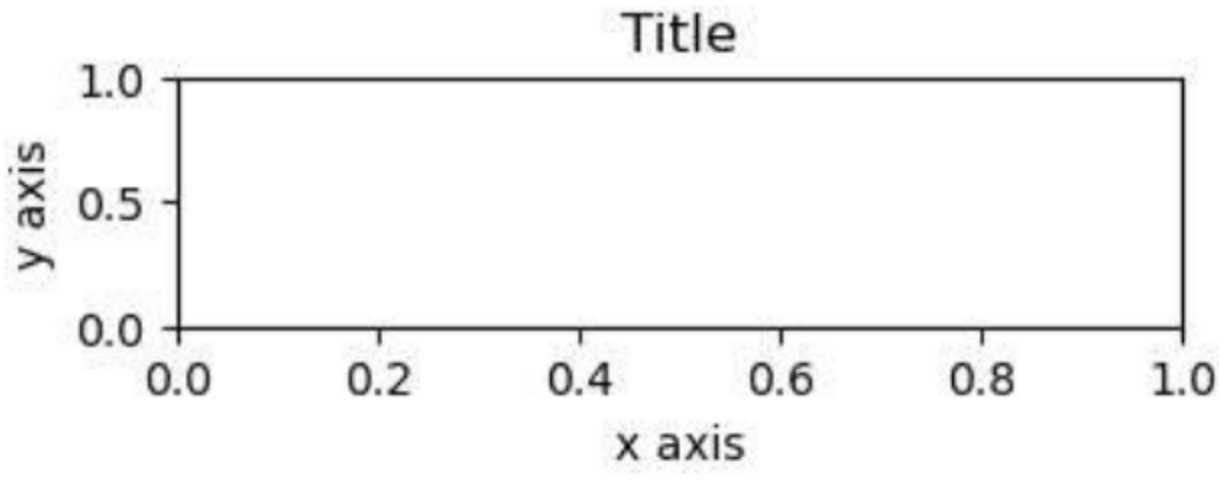


Labels and Title

xlabel() and ylabel() functions to set label for x- & y-axis

title() function to set a title

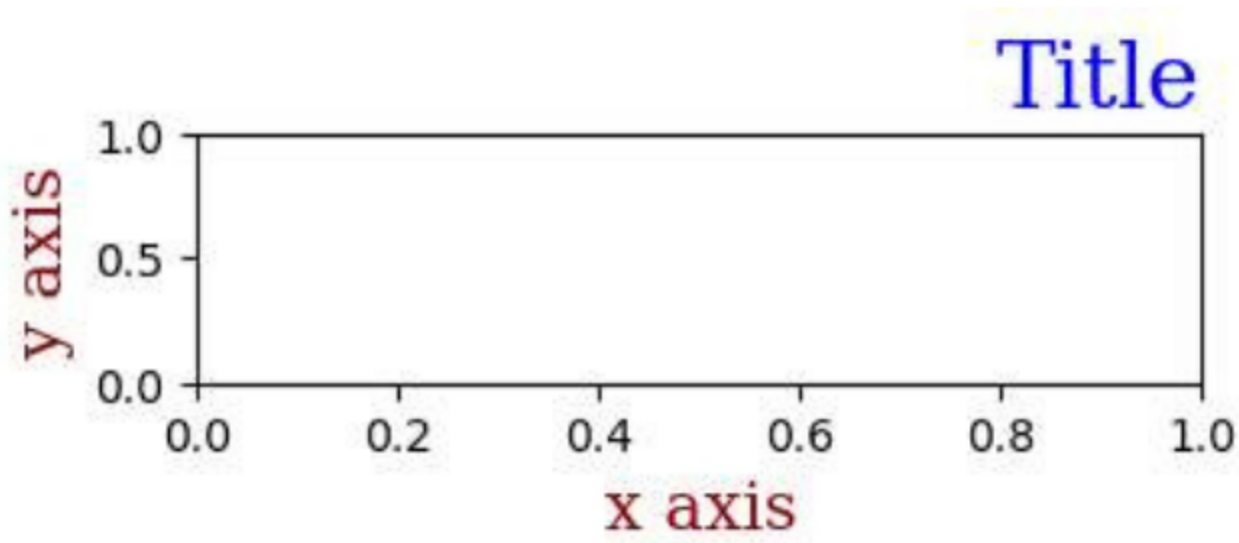
```
plt.title("Title")
plt.xlabel("x axis")
plt.ylabel("y axis")

plt.show()
```



Font of Title and Labels

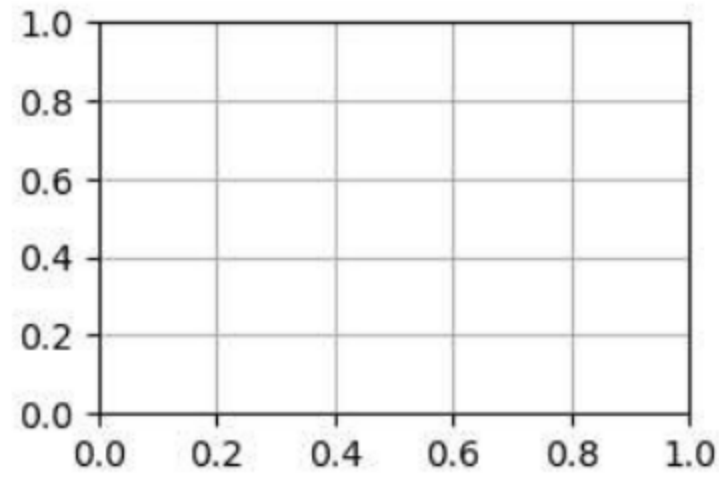
'fontdict' parameter

```
font1 = {'family':'serif',
    'color':'blue','size':20}
font2 = {'family':'serif',
    'color':'darkred','size':15}

#'loc' to position the title
plt.title("Title",
 fontdict = font1, loc = 'right')
plt.xlabel("x axis",
           fontdict = font2)
plt.ylabel("y axis",
           fontdict = font2)

plt.show()
```



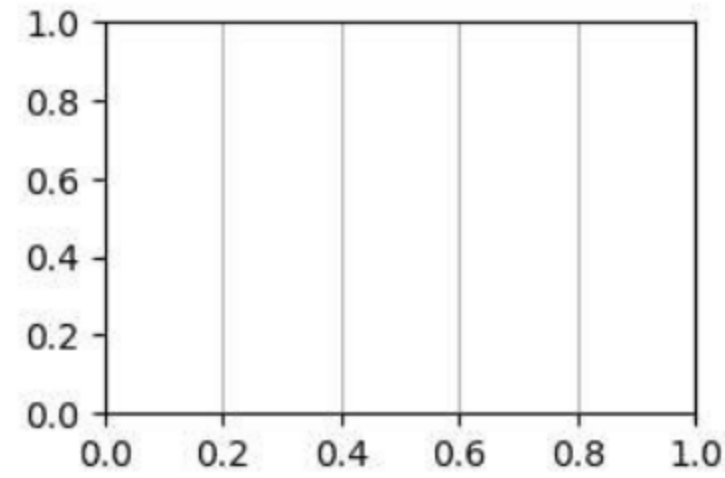
Grid Lines

```
plt.grid()
```



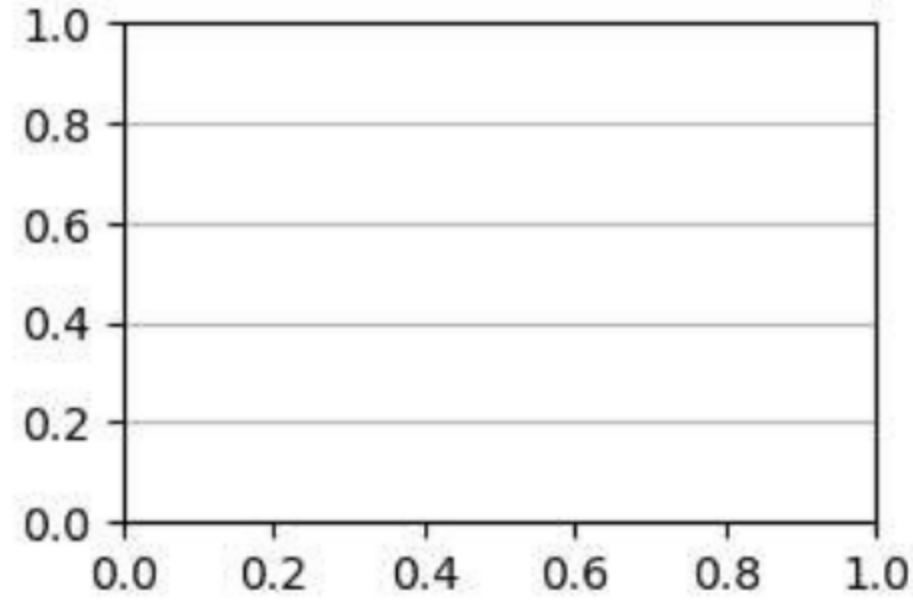
grid lines for the x-axis

```
plt.grid(axis='x')
```



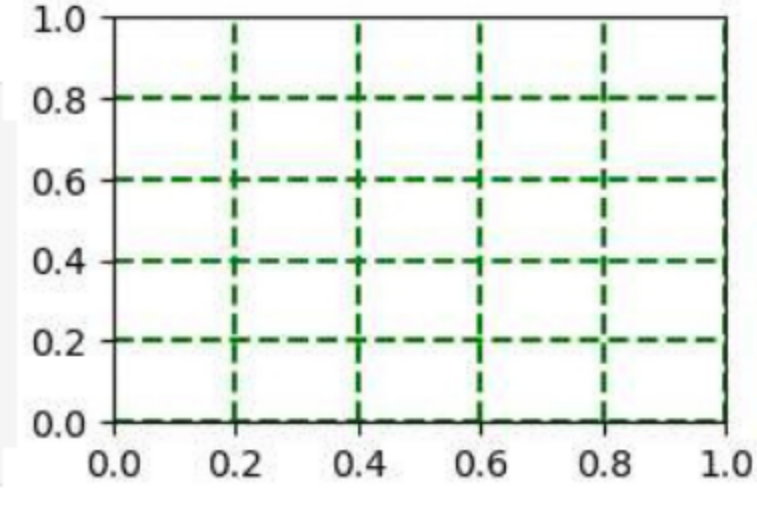
grid lines for the y-axis

```
plt.grid(axis='y')
```



Line Properties for the Grid

```
plt.grid(color = 'green',
         linestyle = '--',
         linewidth = 1.5)
```



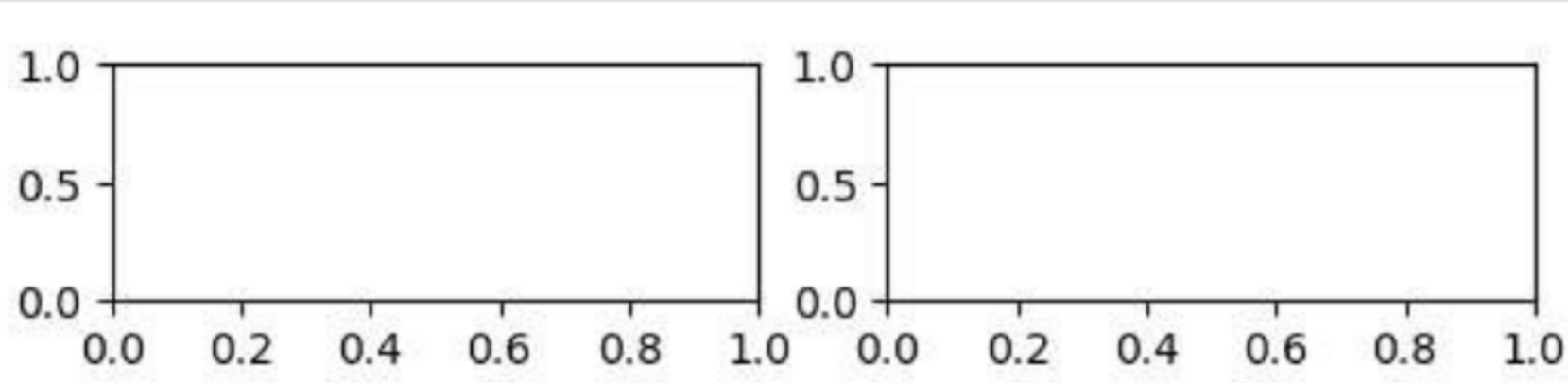
Multiple Plots

Mthod 1: Subplot

draw multiple plots in one figure
Syntax: subplot(rows, columns, index)

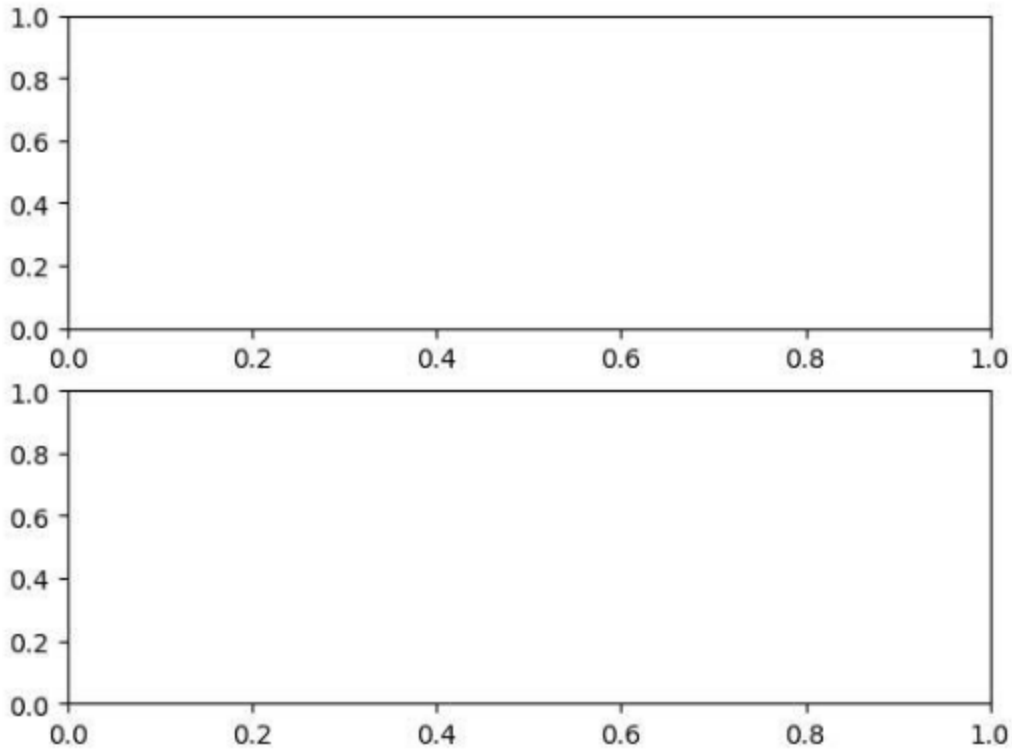
Draw Two Plots : 1 row 2 column

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

plt.subplot(1, 2, 2)  # 2nd plot

plt.show()
```



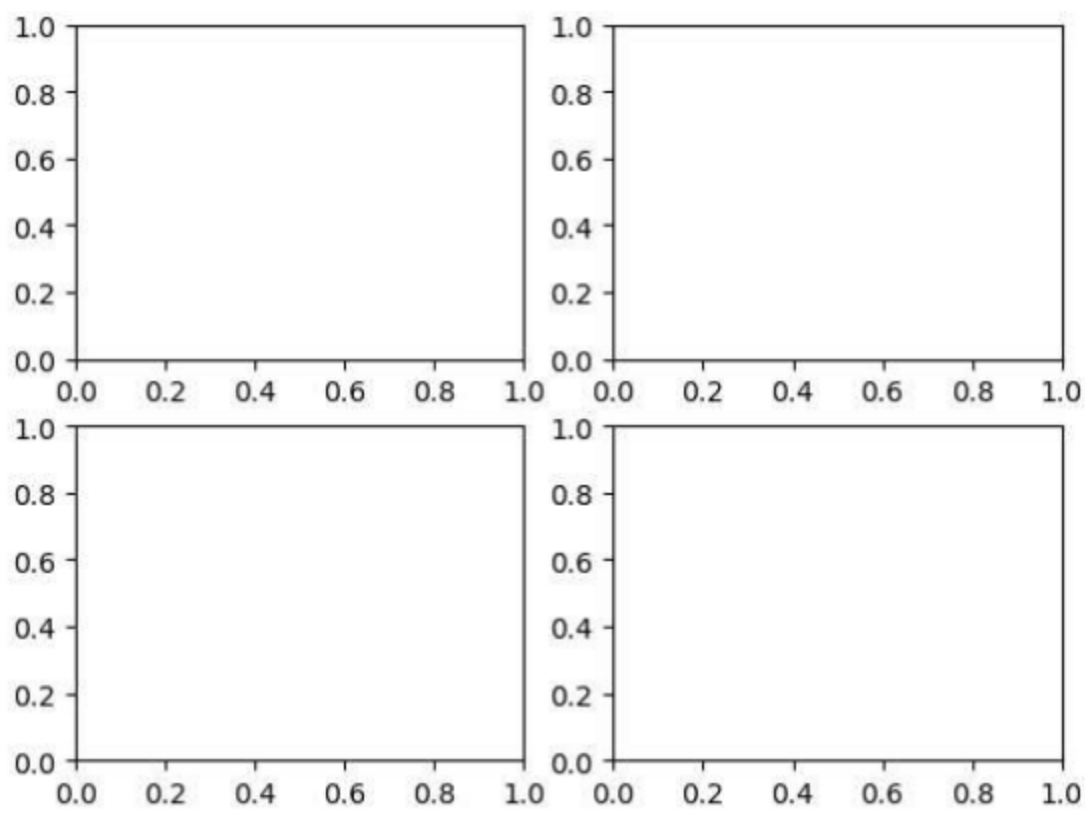
2 row and 1 column

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

plt.subplot(2, 1, 2)  # 2nd plot

plt.show()
```



2 row and 2 column subplot

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

plt.subplot(2, 2, 2) # Subplot 2

plt.subplot(2, 2, 3) # Subplot 3

plt.subplot(2, 2, 4) # Subplot 4
```



Method 2: add_subplot

Syntax:- add_subplot(rows, cols, index)

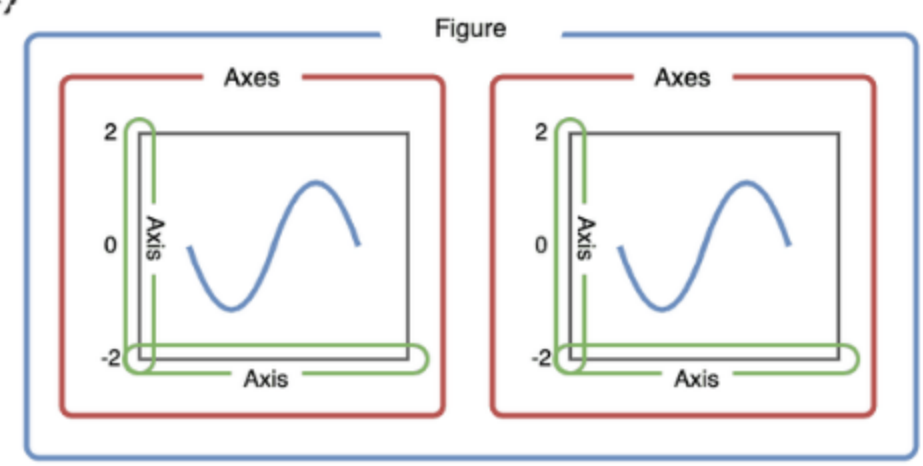
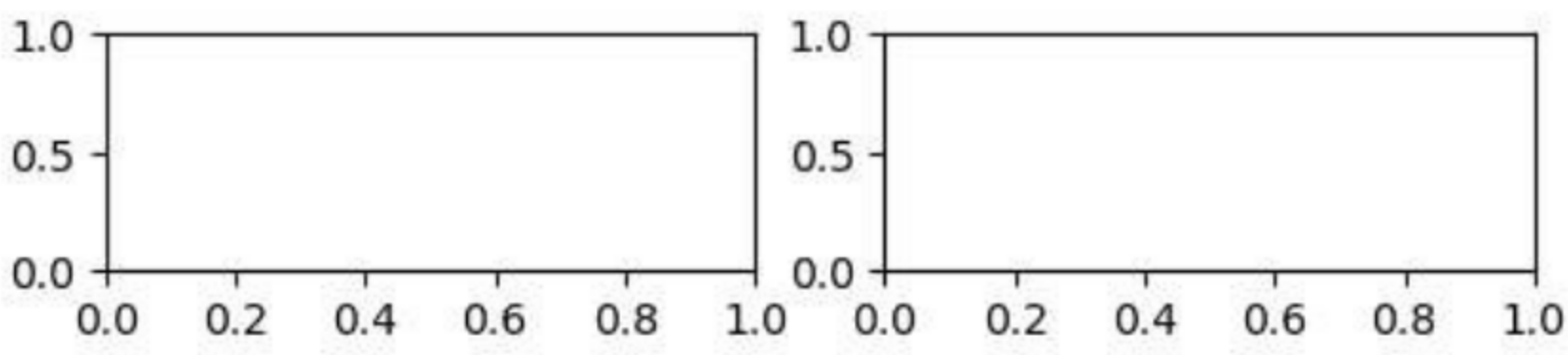


Figure and Axes

- Figure represents entire area where your data visualizations will be displayed
  - like a container that holds all the individual plots or charts
- Axes is like an individual plot or chart within the figure
  - e.g. line charts, bar graphs, or scatter plots.

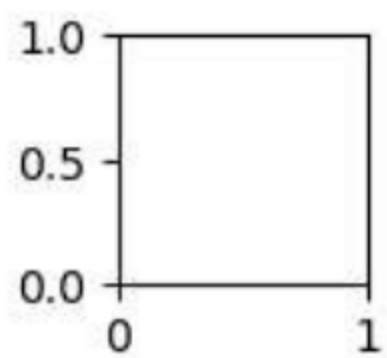
```
# Add subplots to the figure
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)
```



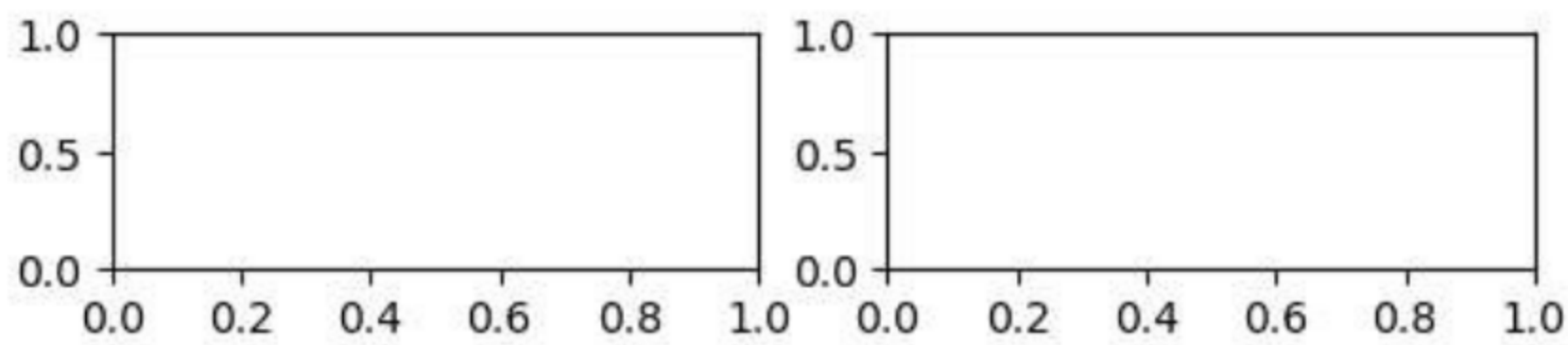
Method 3 : subplots()

```
plt.subplots()

# it return 2 values in tuple.
# (<Figure with 1 Axes>, <Axes:>)
```

```
(<Figure size 100x100 with 1 Axes>, <Axes: >)
```



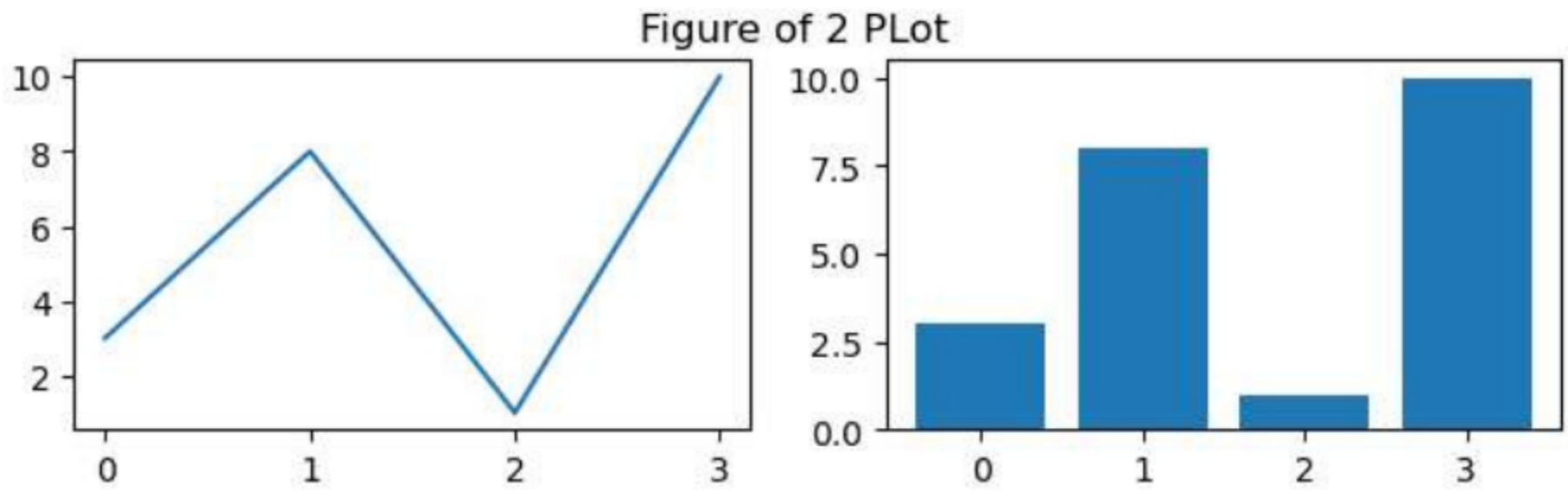

```
# Unpack a tuple into fig and ax.

fig, ax = plt.subplots(1,2)

# 1 row 2 column
```



Create Multiple Axes With Example

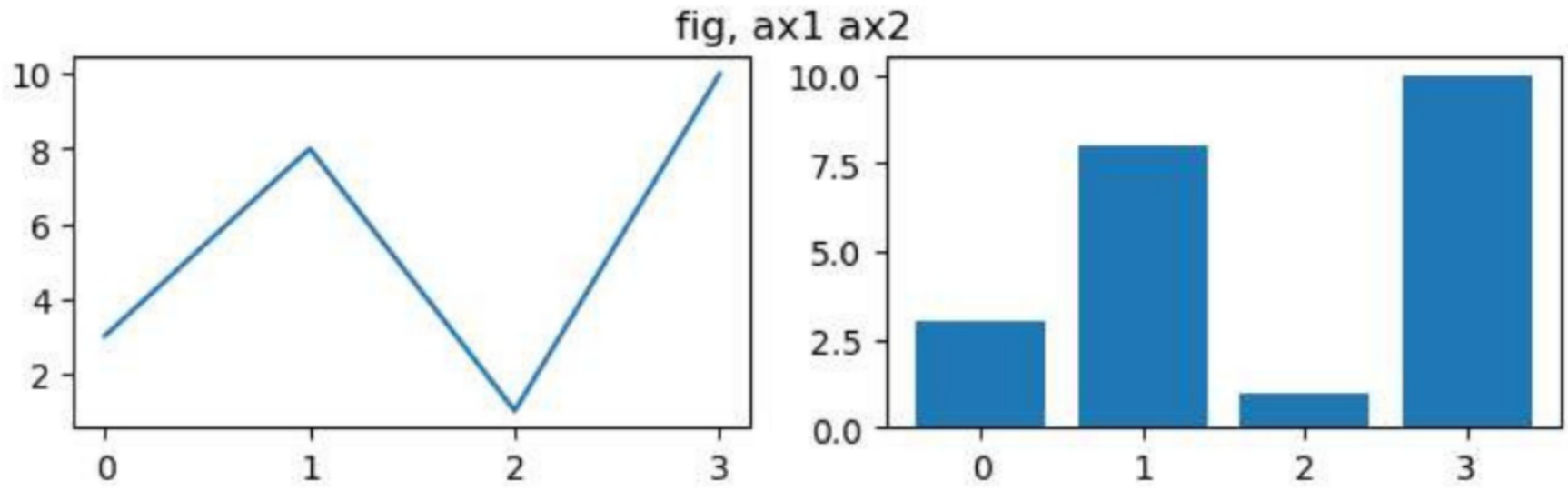
```
# create data
x = [0, 1, 2, 3]
y = [3, 8, 1, 10]
```

```
fig, ax = plt.subplots(1,2)

ax[0].plot(x,y)
ax[1].bar(x,y)

fig.suptitle('Figure of 2 PLot')
plt.show()
```



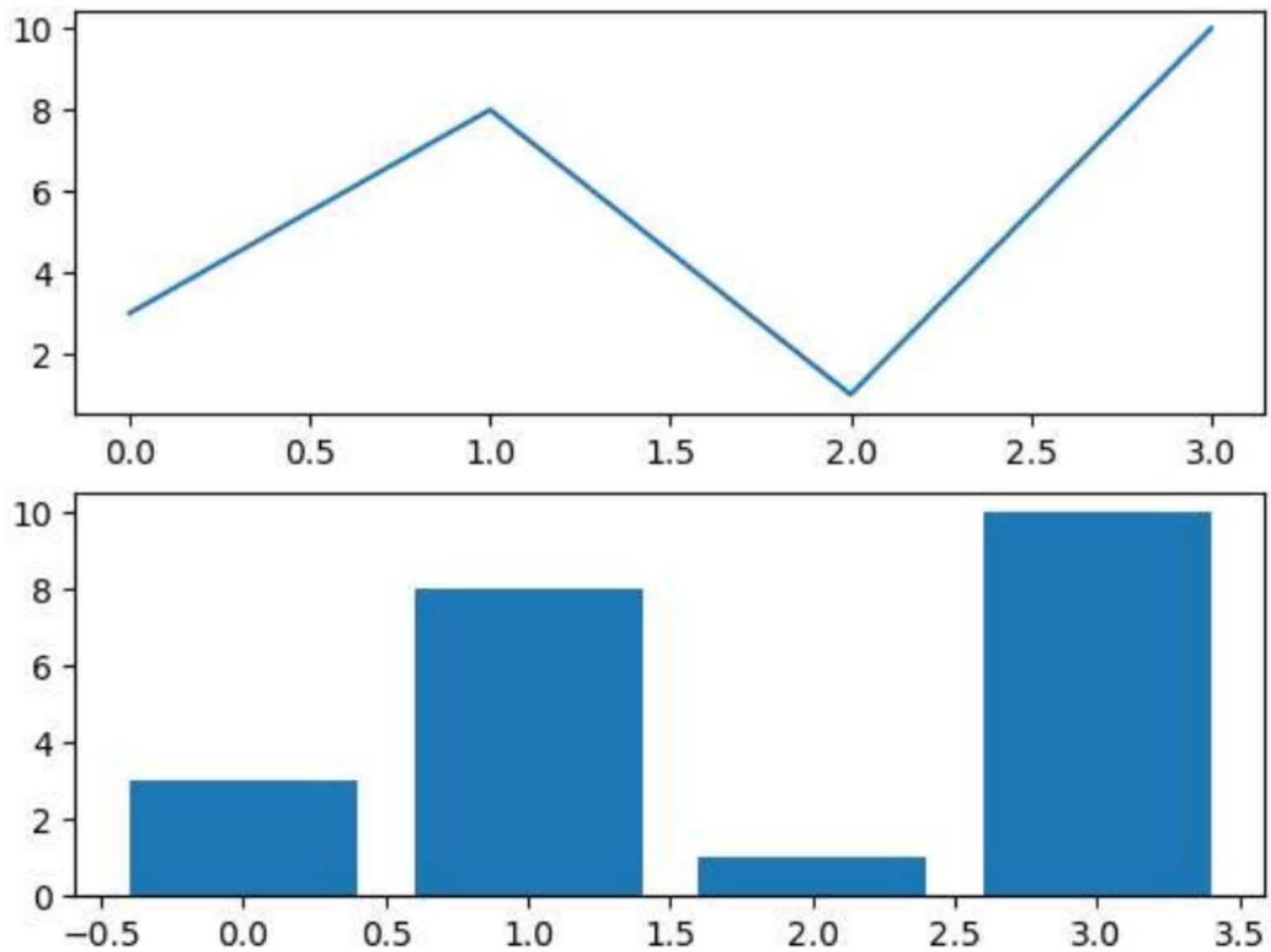
Alternate code

```
fig, (ax1,ax2)= plt.subplots(1,2)

ax1.plot(x,y)
ax2.bar(x,y)

fig.suptitle('fig, ax1 ax2')
plt.show()
```



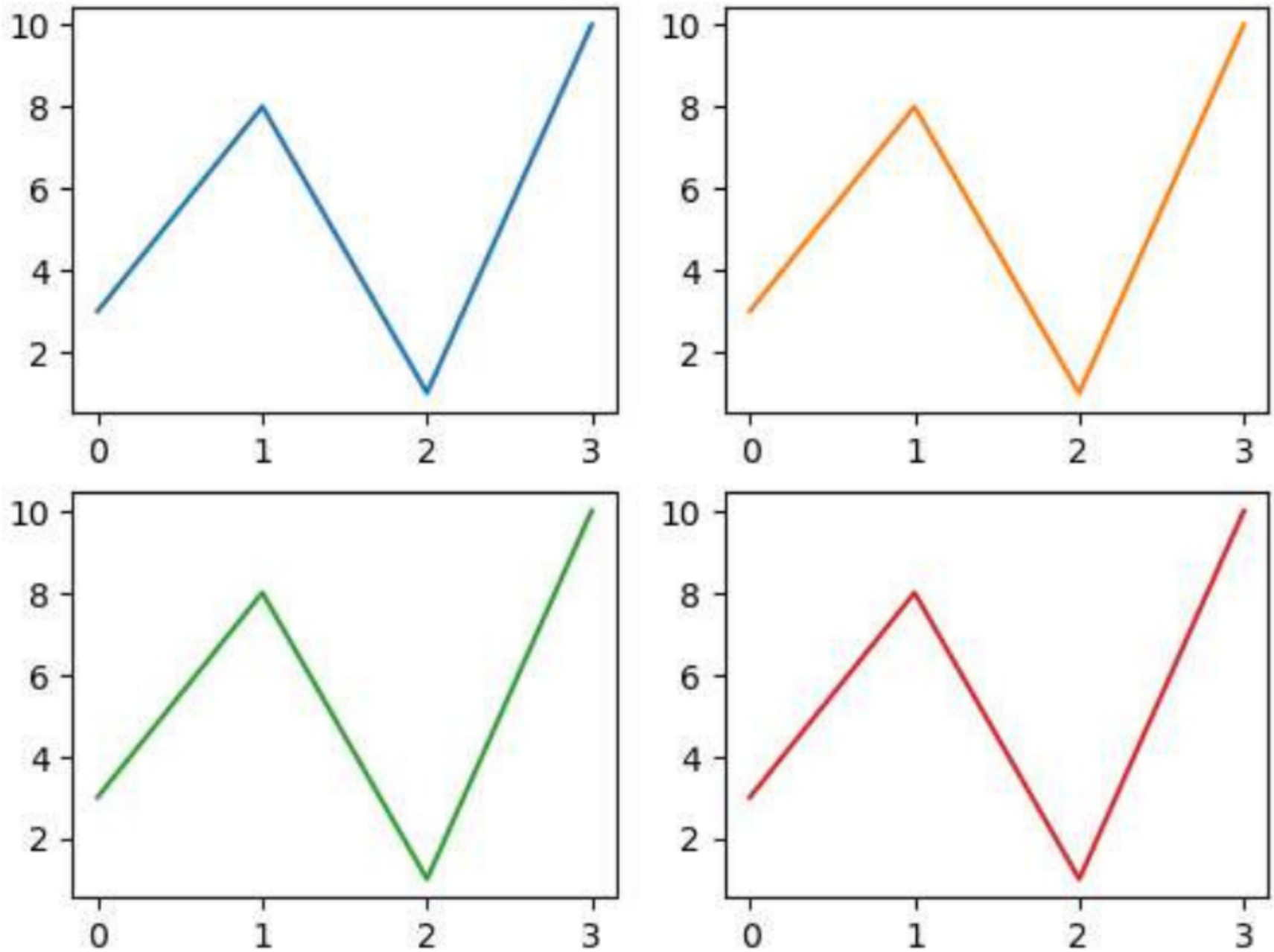
Vertically stacked subplots

```
fig, ax = plt.subplots(2)
# row=0,column=2

ax[0].plot(x,y)
ax[1].bar(x,y)
```



2 rows 2 columns

```
fig, ax = plt.subplots(2, 2)

ax[0, 0].plot(x, y)
ax[0, 1].plot(x, y, 'tab:orange')
ax[1, 0].plot(x, y, 'tab:green')
ax[1, 1].plot(x, y, 'tab:red')
```

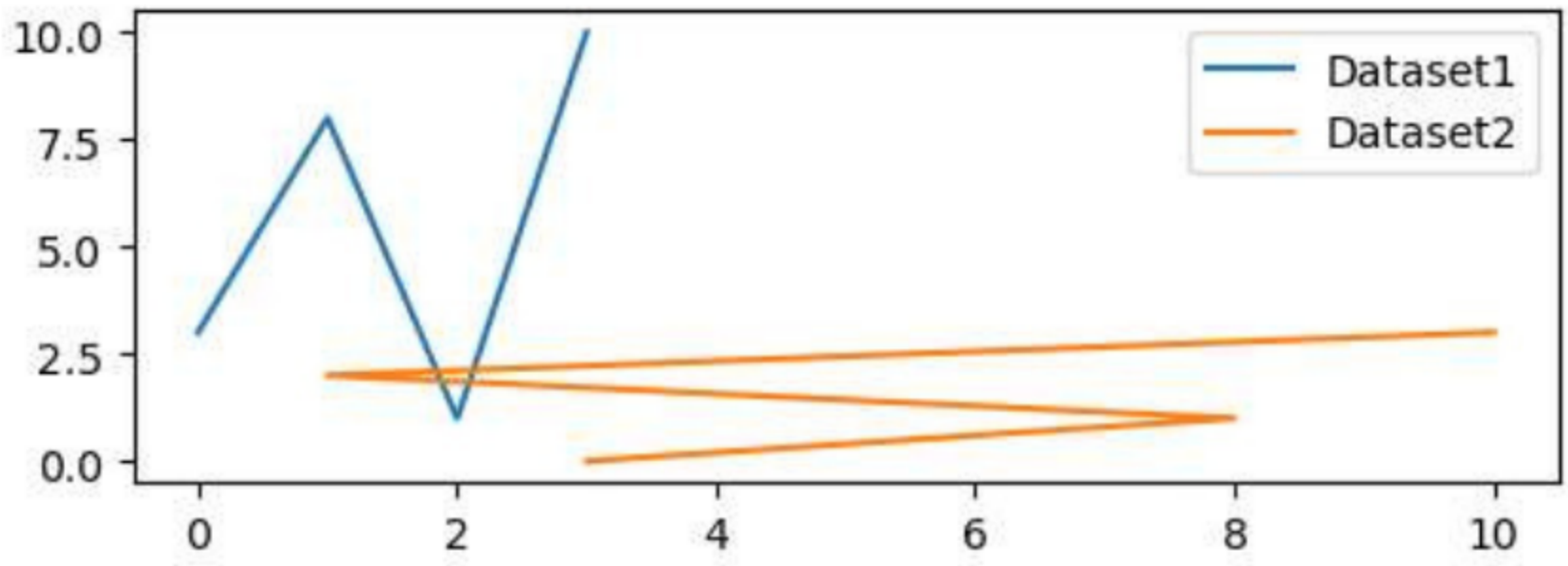


Alternate way to code

```
fig, ((ax1, ax2), (ax3, ax4))
= plt.subplots(2, 2)

ax1.plot(x, y)
ax2.plot(x, y, 'tab:orange')
ax3.plot(x, y, 'tab:green')
ax4.plot(x, y, 'tab:red')
```

legend()

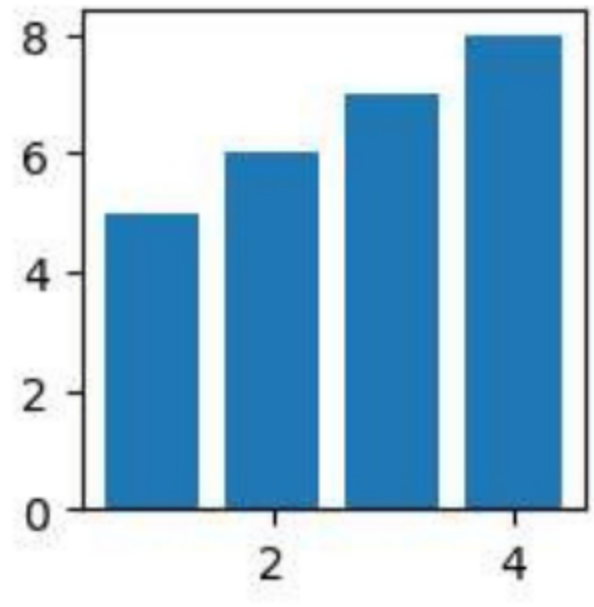
```
plt.plot(x,y)
plt.plot(y,x)

plt.legend(['Dataset1','Dataset2'])

plt.show()
```



More Different Types of Plots

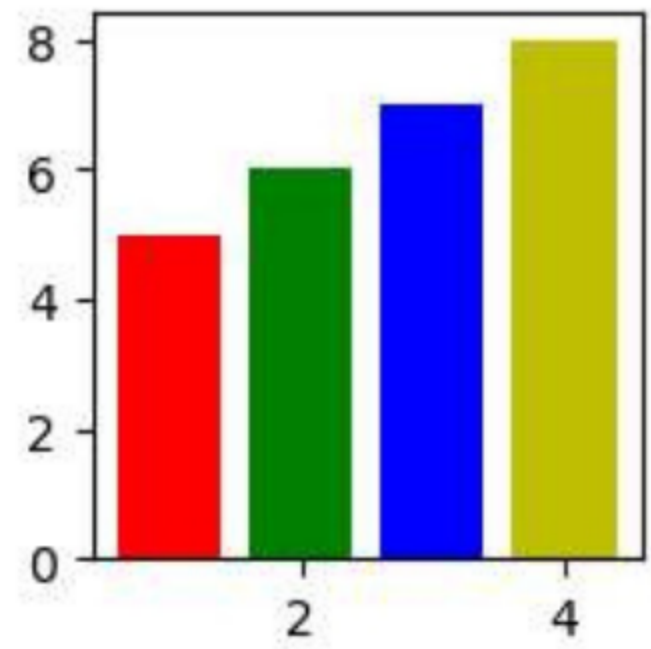
Bar Graph bar()

```
x = [1,2,3,4]
y = [5,6,7,8]

plt.bar(x,y)
```



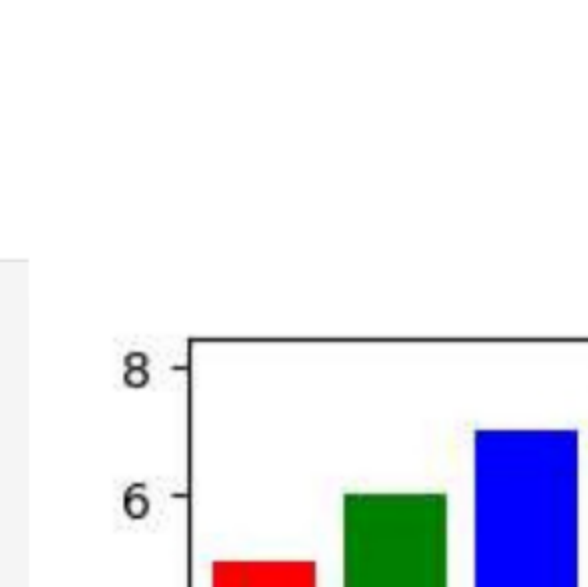
Change color of bar

```
# set colors of bar
c = ['r','g','b','y']

plt.bar(x,y, color=c)

plt.show()
```



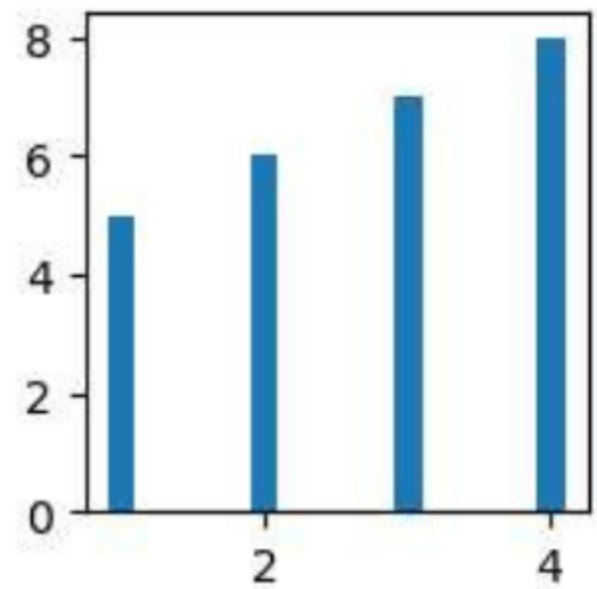
horizontal Bar graph barh()

```
plt.barh(x,y)

plt.show()
```

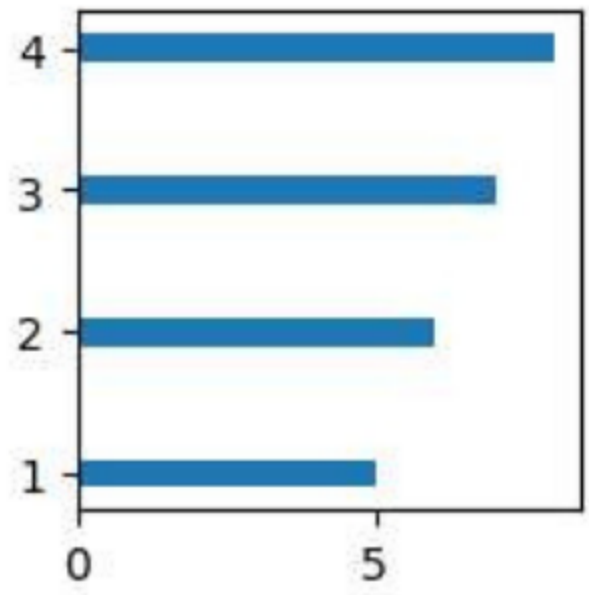


change width of bar

```
# width use in vertical bar
plt.bar(x,y, width = 0.2)
```

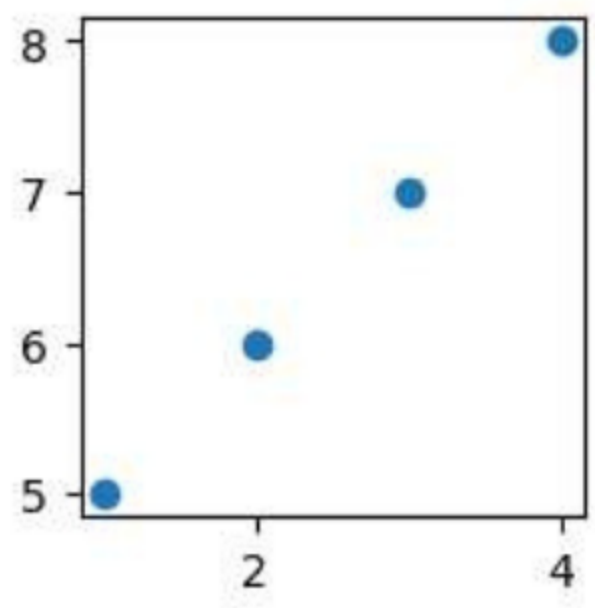


```
# height use in horizontal
plt.barh(x,y, height = 0.2)

plt.show()
```



Scatter Plot `scatter()`

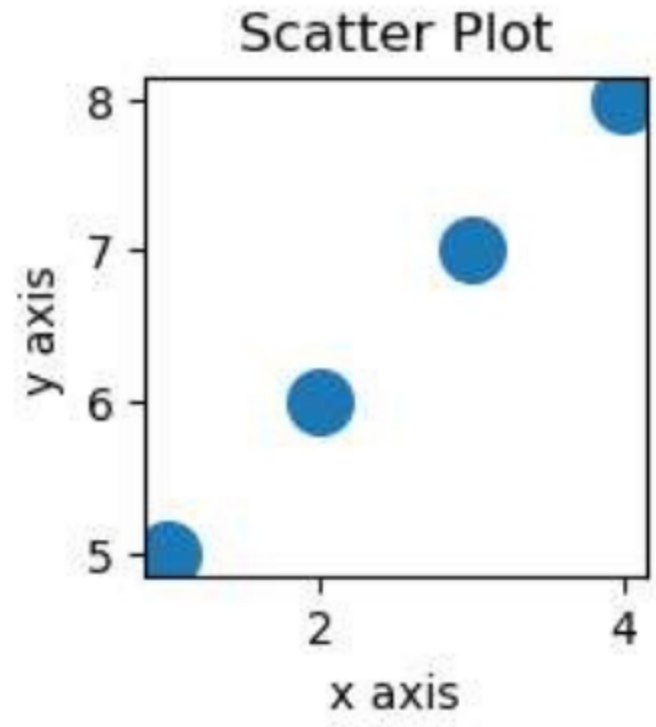
```
x = [1,2,3,4]
y = [5,6,7,8]

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



change size of scatter dot

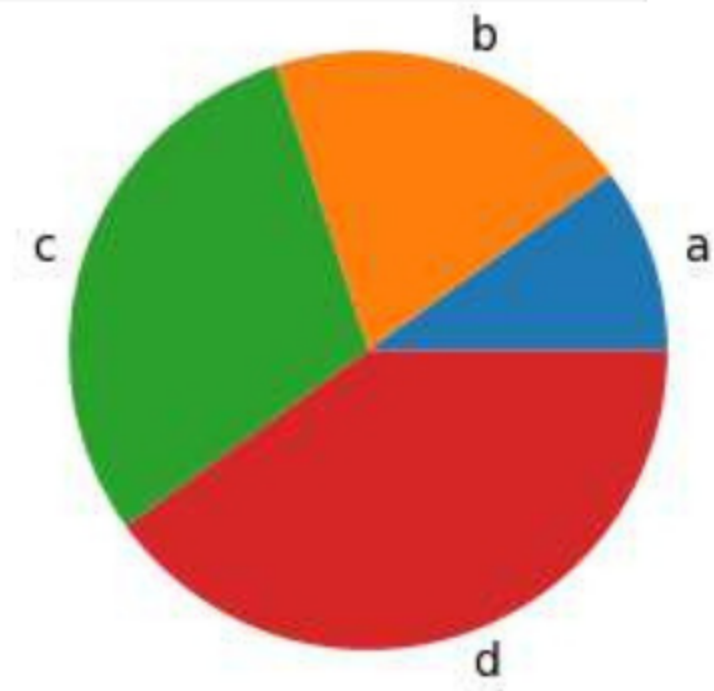
`s parameter in scatter()`

```
plt.scatter(x,y, s=200)

plt.show()
```



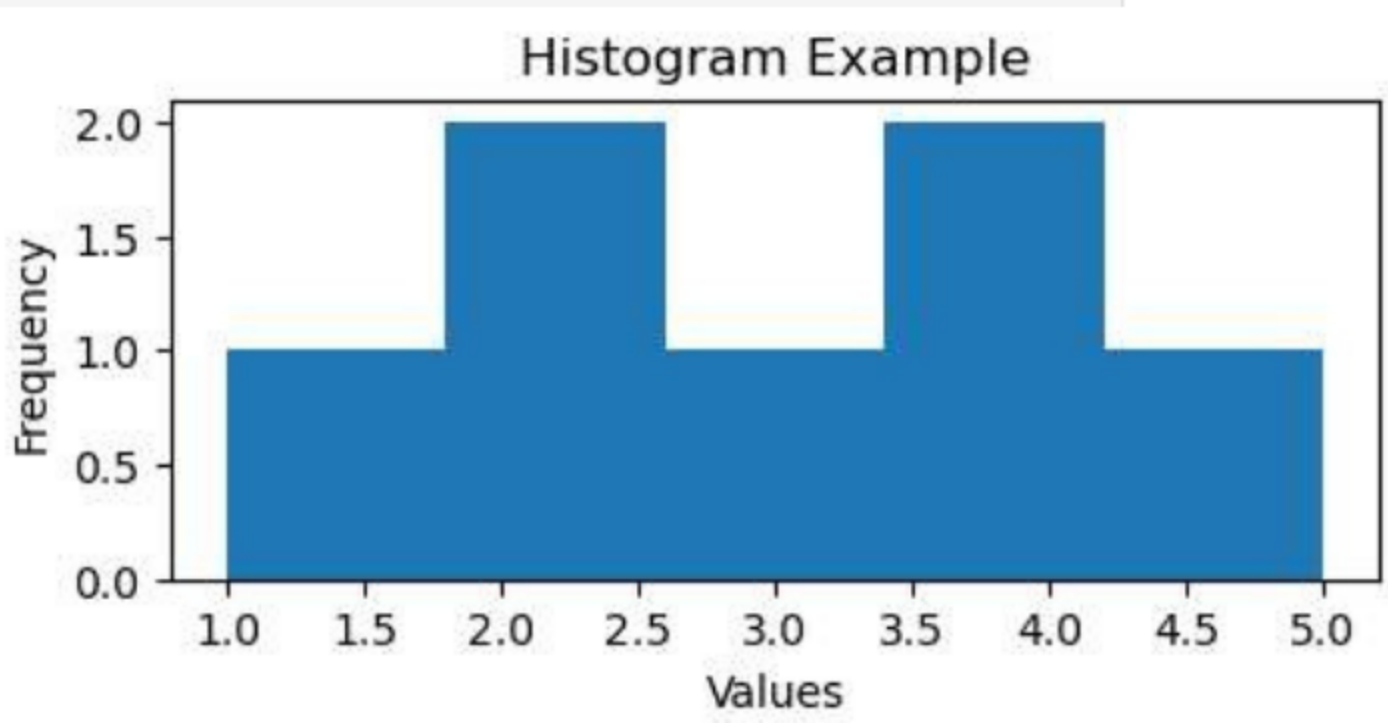
Pie Chart

```
x = [10,20,30,40] # Create Data
y=['a','b','c','d'] #Create Labels

plt.pie(x, labels=y)

plt.show()
```



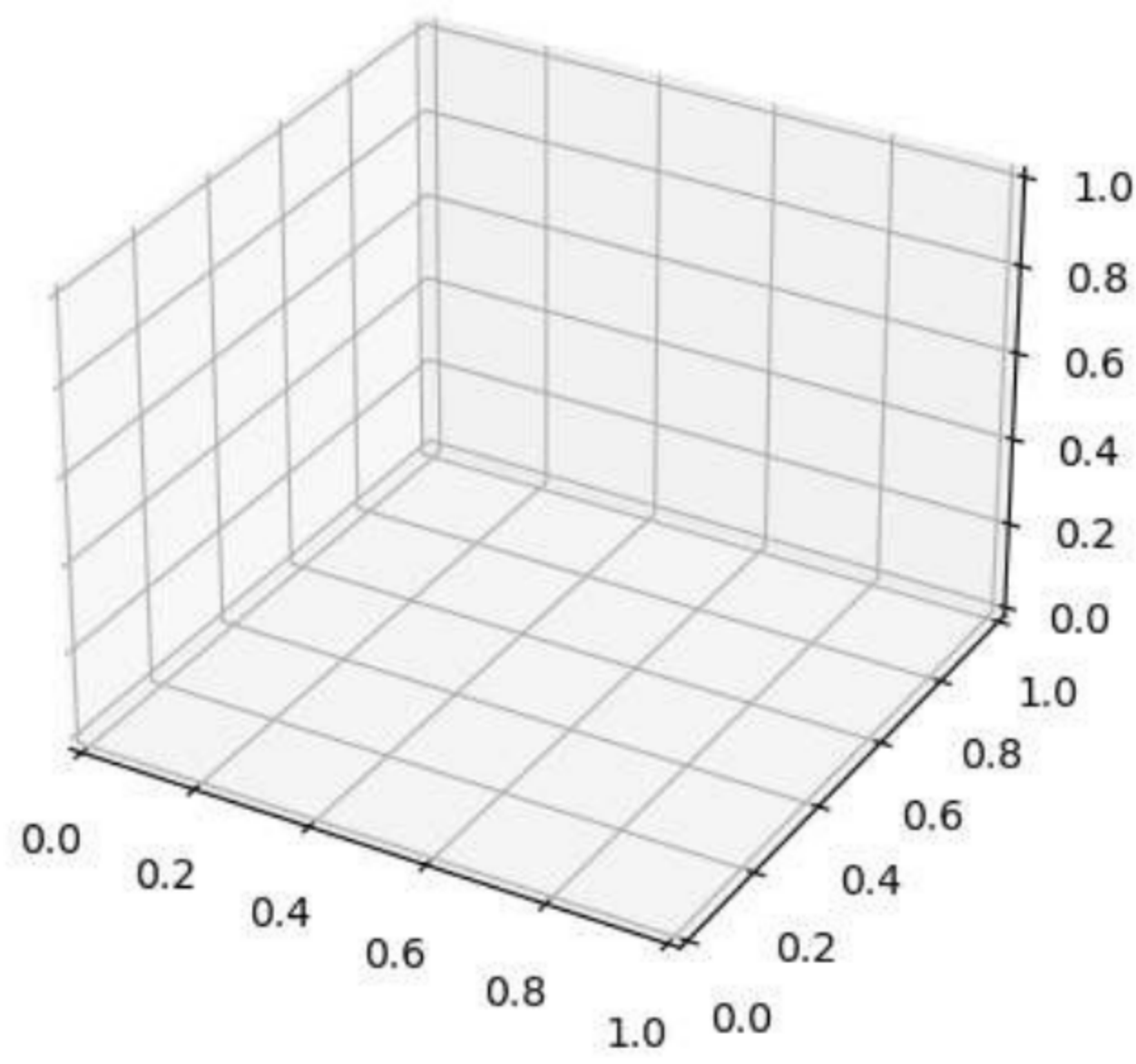
Histograms

```
plt.hist([1,2,2,3,4,4,5],bins=5)

# Labeling (optional)
plt.title('Histogram Example')
plt.xlabel('Values')
plt.ylabel('Frequency')

plt.show()
```



3D Plots

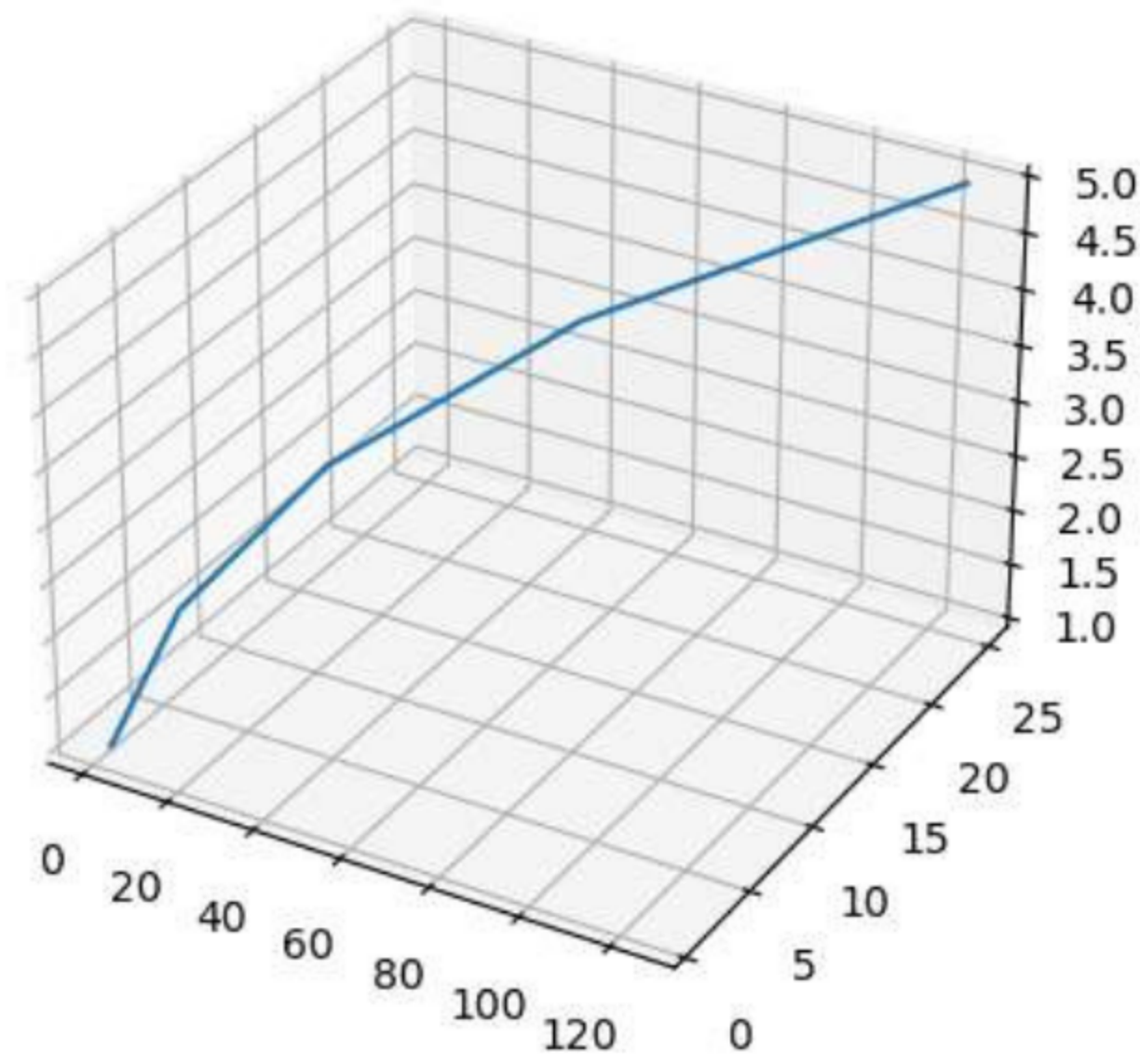
```
# Creating the figure object
fig = plt.figure()

# creates the 3d plot
ax = plt.axes(projection = '3d')
```



3D Line Plot

```
# creating data
x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]
z = [1, 8, 27, 64, 125]
```

```
fig = plt.figure()

# creates the 3d plot
ax = plt.axes(projection = '3d')
ax.plot3D(z, y, x)
```



Working with Images

image module from Python Imaging Library (PIL)

```
from PIL import Image
```

Access, Load & Display the Image

```
# Acess the image from folder
# r is syntax for path
fname = r'samplepic.JPG'

# Load the image in application
image = Image.open(fname)

# Display the image in graph
plt.imshow(image)

plt.show()
```

Add effects on image

```
# convert to Luminance grayscale
image=Image.open(fname).convert('L')

# colormap to blue
plt.imshow(image, cmap='Blues')
plt.show()
```

Thank you