Data Analytics and Machine Learning Foundation

Data Analytics Tools : Matplotlib

Agenda

- Introduction to Matplotlib
- Installing Matplotlib
- Matplotlib Figure Components
- Getting Started with Pyplot
- Review Useful Numpy Functionalities
- Matplotlib.Pyplot Functions
- Axis Function
- Figure Funtions
- Image Functions
- Examples- Plot, Bar, Scatter, Hist, Pie, 3D
- Subplots in Matplotlib

Introduction to Matplotlib

"We humans are typical visual creatures: we understand things better when we see things visualized"

- To make necessary statistical inferences, it becomes necessary to visualize your data and Matplotlib is one such solution for the Python users.
- matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts,
- shell, web application servers and other graphical user interface toolkits.
- It is a very powerful plotting library useful for those working with Python and NumPy.

Installing Matplotlib

- To install Matplotlib on your local machine, open Python command prompt and type following commands:

  python -m pip install -U pip python -m pip install -U matplotlib

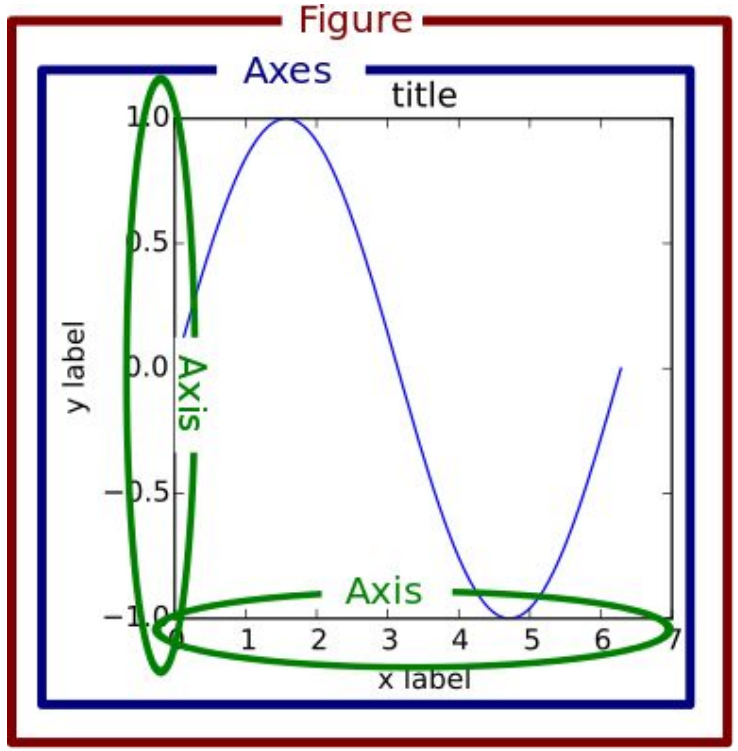
- For Anaconda users, it comes with the distribution package itself.

- For Jupyter Notebooks, use:

  ! conda install matplotlib OR
  ! pip install matplotlib

Matplotlib Figure Components

- A Matplotlib figure can be categorized into several parts as below:
  - **Figure**
  - **Axes**
  - **Axis**
  - **Artist**



Getting Started with pyplot

- Pyplot is a module of Matplotlib which provides simple functions to add plot elements like lines, images, text, etc. to the current axes in the current figure.
- The most used module of Matplotib is Pyplot which provides an interface like MATLAB but instead, it uses Python and it is open source.
- To make a simple plot use:

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

- Here we import Matplotlib's Pyplot module and Numpy library as most of the data that we will be working with will be in the form of arrays only.

Review Useful Numpy Functionalities

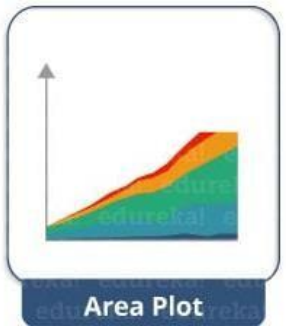
- numpy.arange([start,]stop, [step,]dtype=None)
- numpy.random.randint(low, high=None, size=None, dtype=int)
- numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0)
- numpy.sin([in_array])
- numpy.cos([in_array])
- numpy.pi

Review Useful Numpy Functionalities (Continue)

- np.linspace(10, 99, 50)
- np.sin([0, 2*np.pi/3, np.pi/3, np.pi])
- np.cos([0, 2*np.pi/3, np.pi/3, np.pi])
- np.arrange(30)
- np.arange(20, 50, 5)
- np.random.randint(1, 30, 2)

Common matplotlib.pyplot Functions

- plot() – plot lines or markers to axes
- scatter() – makes scatter plot for X vs Y
- bar() – makes a bar graph
- barh() – makes a horizontal bar graph
- hist() – plot a histogram
- hist2d() – make a 2D histogram plot
- boxplot() – make a box and w
- pie() – plot a pie graph



Common matplotlib.pyplot Functions (Continue)

- polar() – make a polar plot
- stackplot() – draw a stacked area plot
- quiver() – plot a 2D field of arrows
- step() – make a step plot
- stem() – create a stem plot

Common Axis Functions

- axes() – add axes to figure
- text() – add text to axes
- title() – set title of current axes
- xlabel() & ylabel() – set x-axis or y-axis label for current axes
- xlim() & ylim() – set x or y limits of current axes
- xscale() & yscale() – set the scaling for x-axis or y-axis
- xticks() & yticks() – set x or y limits of current tick locations and labels

Common Figure Functions

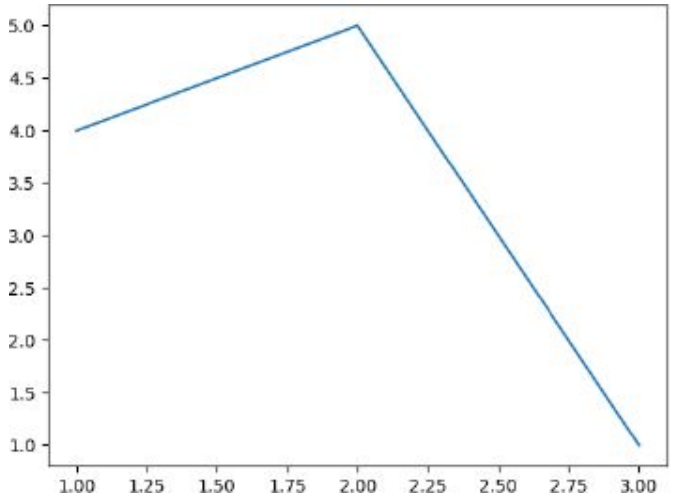
- figure() – creates new figure
- figtext() – adds text to figure
- show() – displays a figure
- savefig() – save current figure
- close() – close a figure window

Common Image Functions

- imread() – Read an image from a file into an array
- imsave() – Save an array as in image file
- imshow() – Display an image on the axes

Example Plot

```
from matplotlib import pyplot as plt
#Plotting to our canvas
plt.plot([1,2,3],[4,5,1])
#Showing what we plotted
plt.show()
```

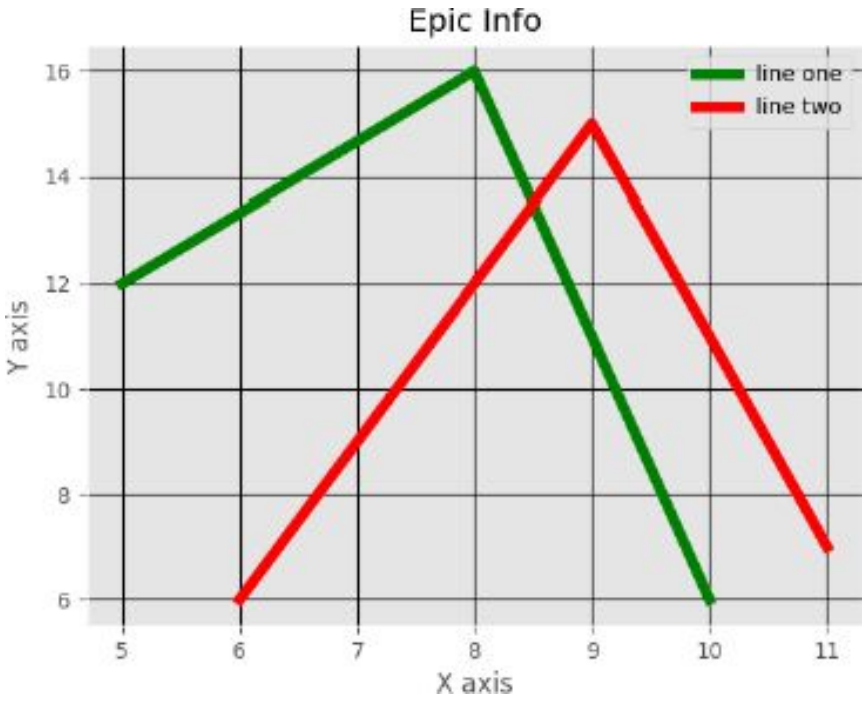


Example Plot

```
from matplotlib import pyplot as plt
x = [5,2,7]
y = [2,16,4]
plt.plot(x,y)
plt.title('Info')  plt.ylabel('Y
axis')  plt.xlabel('X axis')
plt.show()
```

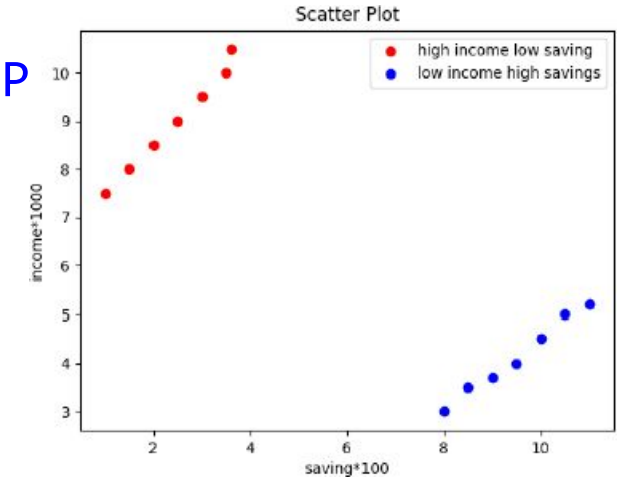
Example ggplot

```
from matplotlib import pyplot as plt  from
matplotlib import style  style.use('ggplot')
x = [5,8,10]
y = [12,16,6]
x2 = [6,9,11]
y2 = [6,15,7]
plt.plot(x,y,'g',label='line one', linewidth=5)
plt.plot(x2,y2,'c',label='line
two',linewidth=5)  plt.title('Epic Info')
plt.ylabel('Y  axis')     plt.xlabel('X  axis')
plt.legend()
plt.grid(True,color='k')  plt.show()
```



Example Scatter Plot

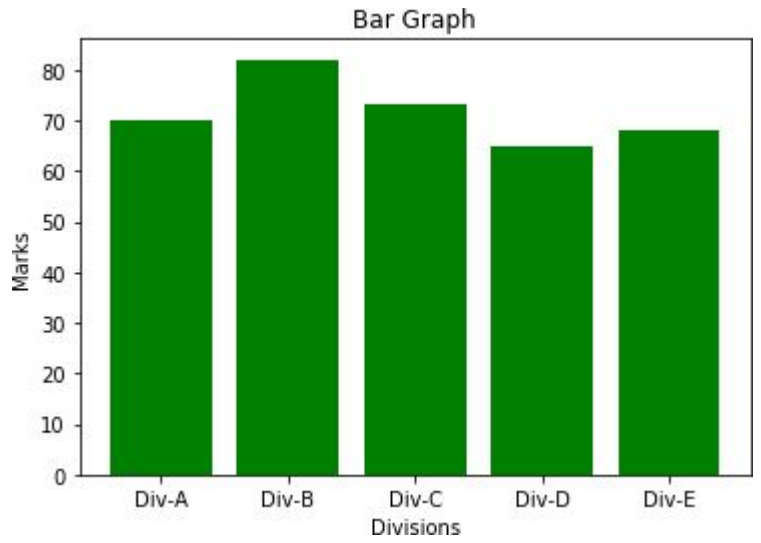
```
import matplotlib.pyplot as plt
x = [1,1.5,2,2.5,3,3.5,3.6]
y = [7.5,8,8.5,9,9.5,10,10.5]
x1=[8,8.5,9,9.5,10,10.5,11]
y1=[3,3.5,3.7,4,4.5,5,5.2]
plt.scatter(x,y, label='high income low saving',color='r')
plt.scatter(x1,y1,label='low income high savings',color='b')
plt.xlabel('saving*100')
plt.ylabel('income*1000')        plt.title('Scatter    P
plt.legend()
plt.show()
```



Example Bar Graph

```
divisions = ["Div-A", "Div-B", "Div-C", "Div-D", "Div-E"]
division_average_marks = [70, 82, 73, 65, 68]

plt.bar(divisions, division_average_marks, color='green')
plt.title("Bar Graph")
plt.xlabel("Divisions")
plt.ylabel("Marks")
plt.show()
```



Example Bar Graph (Stacked Horizontally)

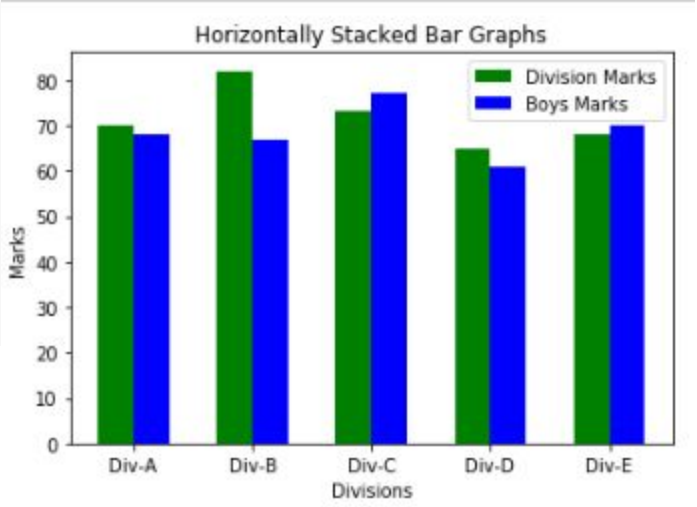
```
divisions = ["Div-A", "Div-B", "Div-C", "Div-D", "Div-E"]
division_average_marks = [70, 82, 73, 65, 68]
boys_average_marks = [68, 67, 77, 61, 70]

index = np.arange(5)
width = 0.30

plt.bar(index, division_average_marks, width, color='green',label='Division Marks')
plt.bar(index+width, boys_average_marks, width, color='blue',label='Boys Marks')
plt.title("Horizontally Stacked Bar Graphs")

plt.ylabel("Marks")
plt.xlabel("Divisions")
plt.xticks(index+ width/2, divisions)

plt.legend(loc='best')
plt.show()
```



Example Bar Graph (Stacked Vertically)

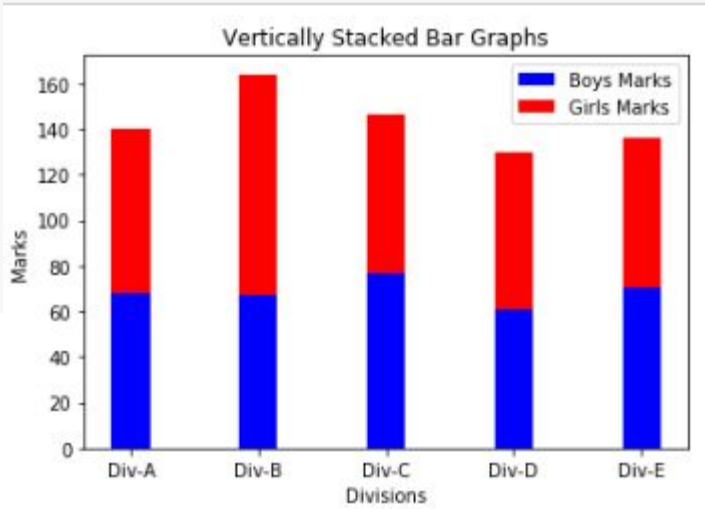
```
divisions = ["Div-A", "Div-B", "Div-C", "Div-D", "Div-E"]
boys_average_marks = [68, 67, 77, 61, 70]
girls_average_marks = [72, 97, 69, 69, 66]

index = np.arange(5)
width = 0.30

plt.bar(index, boys_average_marks, width, color="blue", label="Boys Marks")
plt.bar(index, girls_average_marks, width, color="red", label="Girls Marks", bottom=boys_average_marks)

plt.title("Vertically Stacked Bar Graphs")
plt.xlabel("Divisions")
plt.ylabel("Marks")
plt.xticks(index, divisions)

plt.legend(loc='best')
plt.show()
```

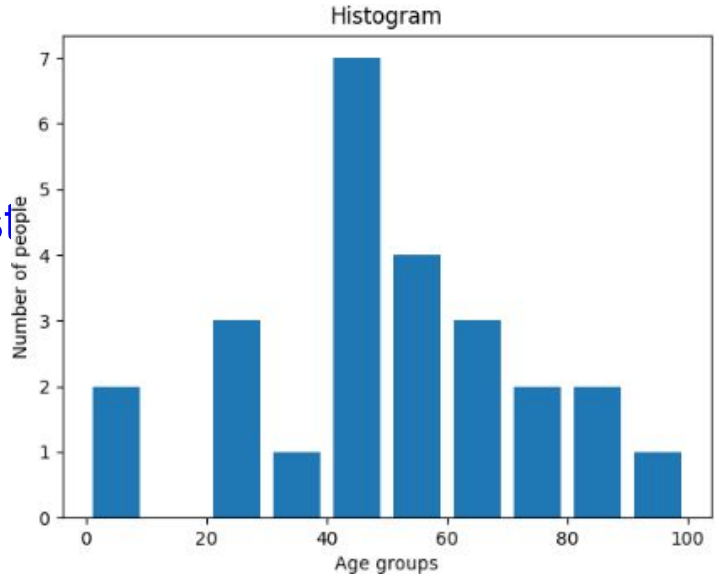


Example Histogram

```
import matplotlib.pyplot as plt

population_age =
[22,55,62,45,21,22,34,42,42,4,2,102,95,85,55,110,120,70,65,55,111,115,
80,75,65,54,44,43,42,48]

bins = [0,10,20,30,40,50,60,70,80,90,100]
plt.hist(population_age, bins, histtype='bar', rwidth=0.8)
plt.xlabel('age groups')  plt.ylabel('Number of people')  plt.title('Hist
plt.show()
```



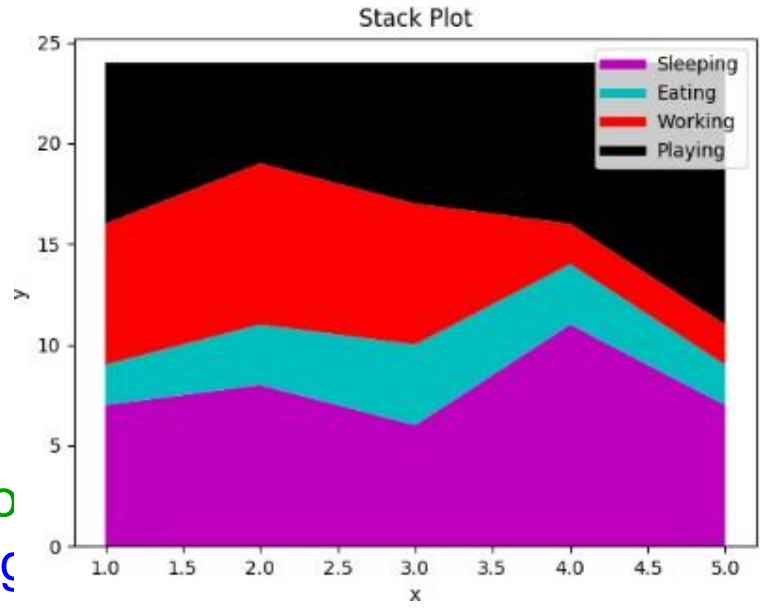
Example Stackplot

```
import matplotlib.pyplot as plt

days = [1,2,3,4,5]
 sleeping =[7,8,6,11,7]

 eating = [2,3,4,3,2]
 working =[7,8,7,2,2]

 playing = [8,5,7,8,13]
 plt.plot([],[],color='m', label='Sleeping', linewidth=5)
 plt.plot([],[],color='c', label='Eating', linewidth=5)  plt.plot([],[],colo
 label='Working', linewidth=5)  plt.plot([],[],color='k', label='Playing
 linewidth=5)
 plt.stackplot(days, sleeping,eating,working,playing, colors=['m','c','r','k'])
 plt.show()
```



Example Pie-Chart

```
import matplotlib.pyplot as plt  days = [1,2,3,4,5]

sleeping =[7,8,6,11,7]

eating = [2,3,4,3,2]
working =[7,8,7,2,2]

playing = [8,5,7,8,13]

slices = [7,2,2,13]

activities = ['sleeping','eating','working','playing']

cols = ['c','m','r','b']

plt.pie(slices, labels=activities, colors=cols, startangle=90, shadow= True, explode=(0,0.1,0,0), autopct='%1.1f%%')
plt.title('Pie Plot')
plt.show()
```

3D Plotting

- To use this functionality, we first import the module mplot3d as follows:

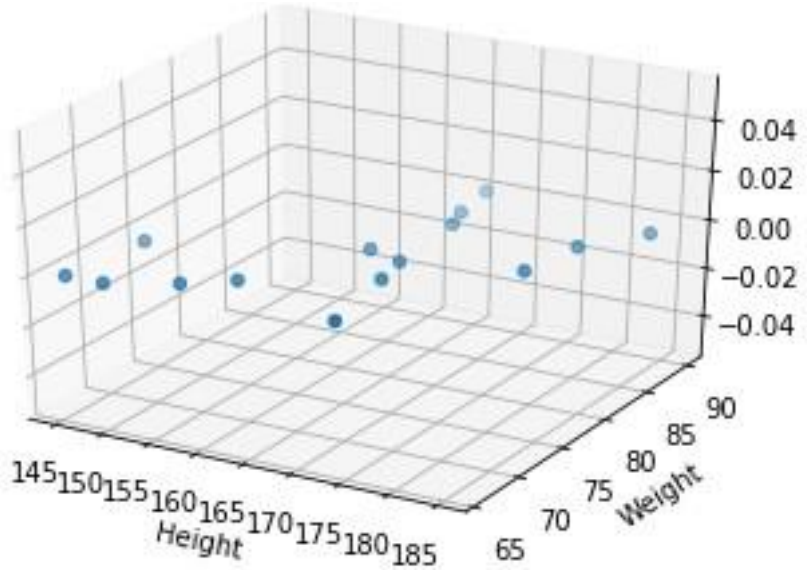
```
from mpl_toolkits import mplot3d
```

- Once the module is imported, a three-dimensional axes is created by passing the keyword **projection='3d'** to the **axes()** method of Pyplot module. Once the object instance is created, we pass our arguments to 3D method.

Example 3D Plotting

```
height = np.array([167,170,149,165,155,180,166,146,
                   159,185,145,168,172,181,169])
weight = np.array([86,74,66,78,68,79,90,73,
                   70,88,66,84,67,84,77])
plt.xlim(140,200)
plt.ylim(60,100)
```

```
ax = plt.axes(projection='3d')
ax.scatter3D(height,weight)
ax.set_xlabel("Height")
ax.set_ylabel("Weight")
plt.show()
```



Working With Multiple Plots

- To create multiple axes plots inside a single figure object we can divide a figure into multiple subplots

```
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(4, 4))
```

- To extract individual axes:

```
ax1, ax2, ax3, ax4 = ax.flatten() # flatten a 2d NumPy array to 1d
```

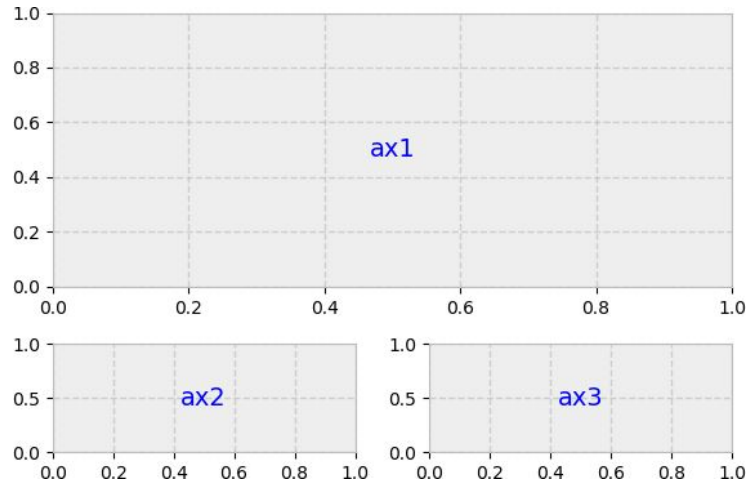
- With this, axes can be plotted and treated further individually
- Alternatively;
- ax[row, col] can be used to represent any subplot axes, OR

```
fig, (ax1, ax2, ax3, ax4) = plt.subplots(nrows=2, ncols=2, figsize=(4, 4))
```

Working With Multiple Plots

- Subplots can be altered to occupy custom grid space by defining grid occupancy as follows:

```
gridsize = (3, 2)
fig = plt.figure(figsize=(12, 8))
ax1 = plt.subplot2grid(gridsize, (0, 0), colspan=2, rowspan=2)
ax2 = plt.subplot2grid(gridsize, (2, 0))
ax3 = plt.subplot2grid(gridsize, (2, 1))
```



Summary

- We can create various visual plots and graphs using Matplotlib library of python
- We can work on axes and grids to get custom plot areas
- We can work on with images analysis
- We can plot visuals in 3D as well


thank you