Matplotlib – Part 1

Part 1

> The first and last lines of code mostly will be

import matplotlib.pyplot as plt ---> first line

plt.show() ----> code to display plots

> There are two methods to plot in matplotlib - one is functional method and other is object oriented method, we will discuss functional first but will shift to object oriented one since its the better way

## Functional way -

import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(0,5,11)

y = x \*\* 2

# Functional way

plt.plot(x,y) -------> code to draw a plot

plt.xlabel("Numbers") -------> code to specify x label

plt.ylabel("Squares") -------> code to specify y label

plt.title("Numbers & Squares plotted") -------> code to specify title

plt.show()

> Creating sub-plots on the same canvas

import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(0,5,11)

y = x \*\* 2

# Functional way

plt.subplot(1,2,1) #----> (number of rows, number of columns, plot on which we want to work)

plt.plot(x,y,"r") #----> "r" specifies red colour line

plt.subplot(1,2,2)

plt.plot(y,x,"b")

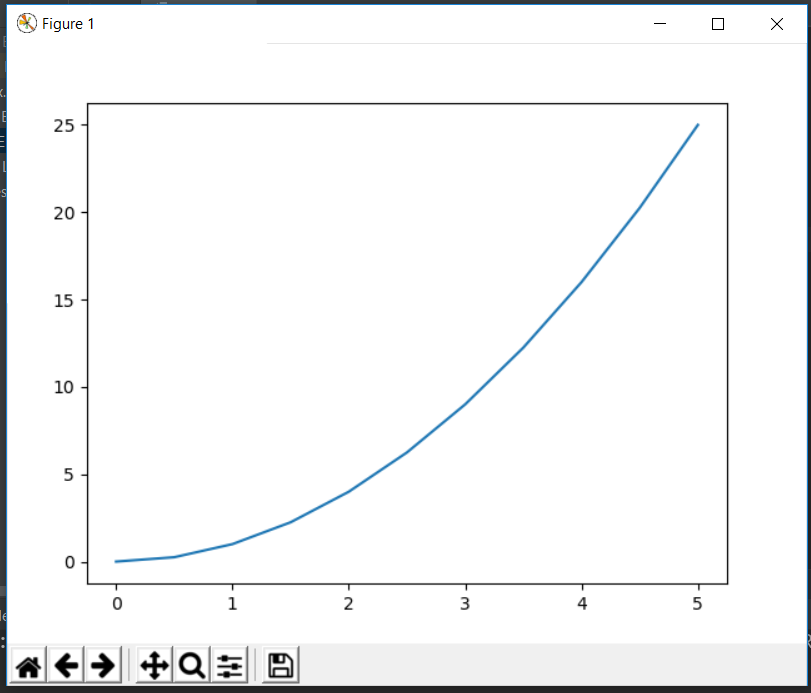
plt.show()

## Object Oriented way –

### Basic plot

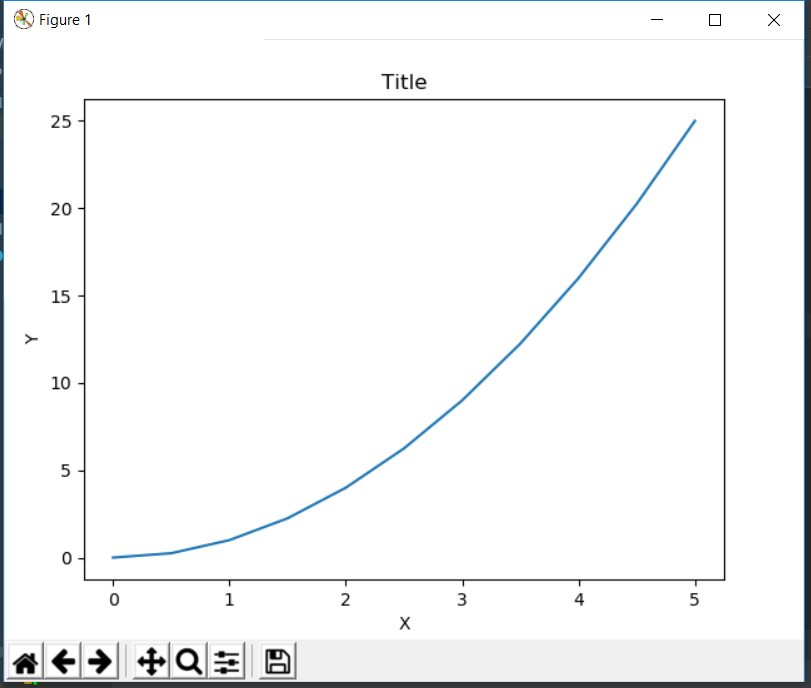
We instantiate here figure object and add axes to it. The add\_axes method always take a list -dimensions [left, bottom, width, height] . All quantities are in fractions of figure width and height.

import matplotlib.pyplot as plt  
import numpy as np  
x = np.linspace(0,5,11)  
y = x \*\* 2  
  
fig = plt.figure()  
axes = fig.add\_axes([0.1,0.1,0.8,0.8])  
axes.plot(x,y)  
plt.show()



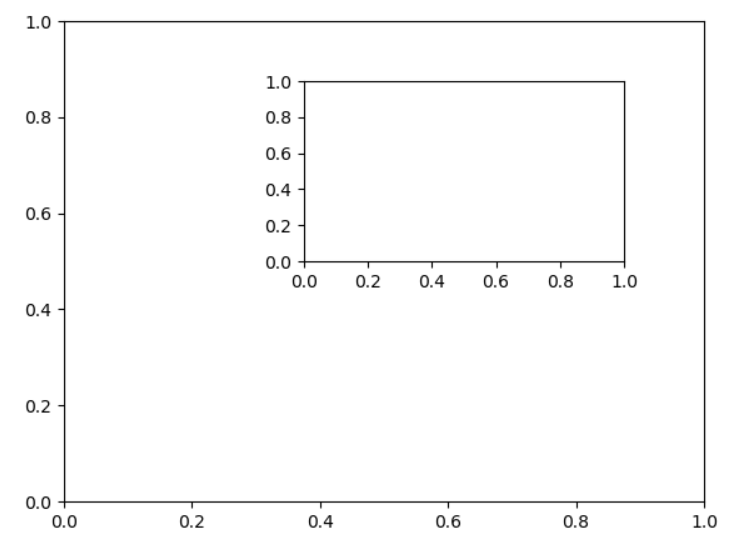
### Specifying labels and title

import matplotlib.pyplot as plt  
import numpy as np  
x = np.linspace(0,5,11)  
y = x \*\* 2  
  
fig = plt.figure()  
axes = fig.add\_axes([0.1,0.1,0.8,0.8])  
axes.plot(x,y)  
axes.set\_xlabel("X")  
axes.set\_ylabel("Y")  
axes.set\_title("Title")  
plt.show()

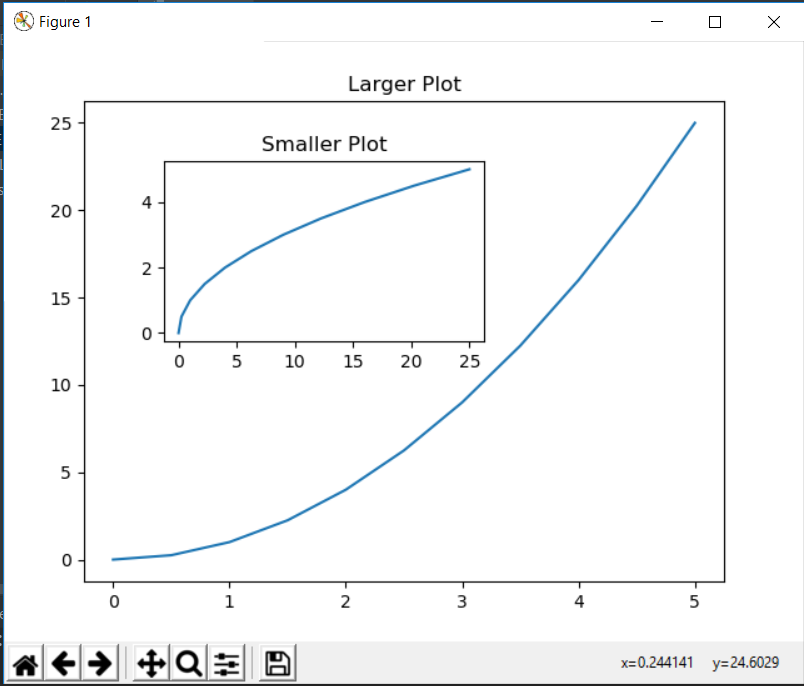


### Creating multi-plots and understanding add\_axes() method

import matplotlib.pyplot as plt  
import numpy as np  
x = np.linspace(0,5,11)  
y = x \*\* 2  
  
fig = plt.figure()  
axes1 = fig.add\_axes([0.1,0.1,0.8,0.8])  
axes2 = fig.add\_axes([0.4,0.5,0.4,0.3])  
plt.show()

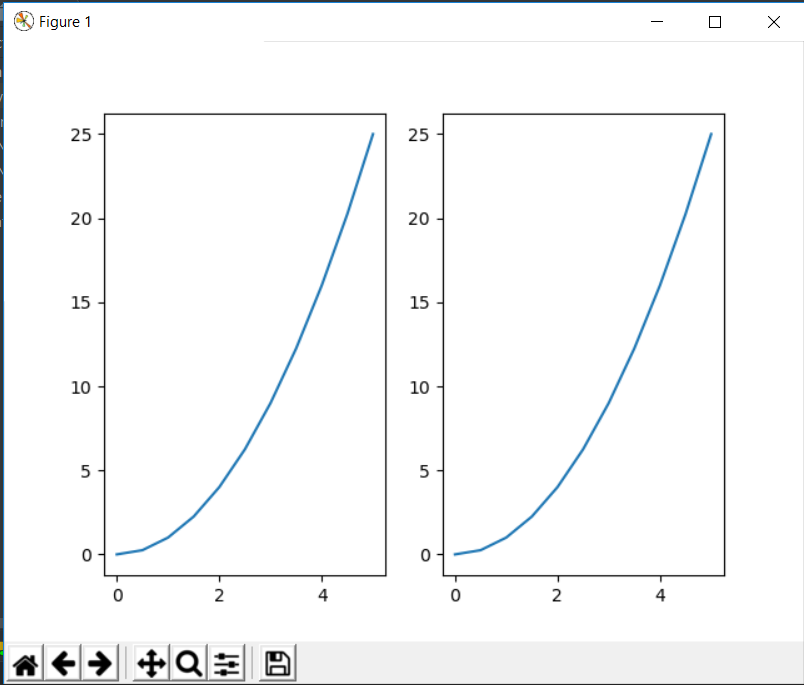


import matplotlib.pyplot as plt  
import numpy as np  
x = np.linspace(0,5,11)  
y = x \*\* 2  
  
fig = plt.figure()  
axes1 = fig.add\_axes([0.1,0.1,0.8,0.8])  
axes2 = fig.add\_axes([0.2,0.5,0.4,0.3])  
  
axes1.plot(x,y)  
axes1.set\_title("Larger Plot")  
axes2.plot(y,x)  
axes2.set\_title("Smaller Plot")  
plt.show()



### Creating subplots using object oriented way

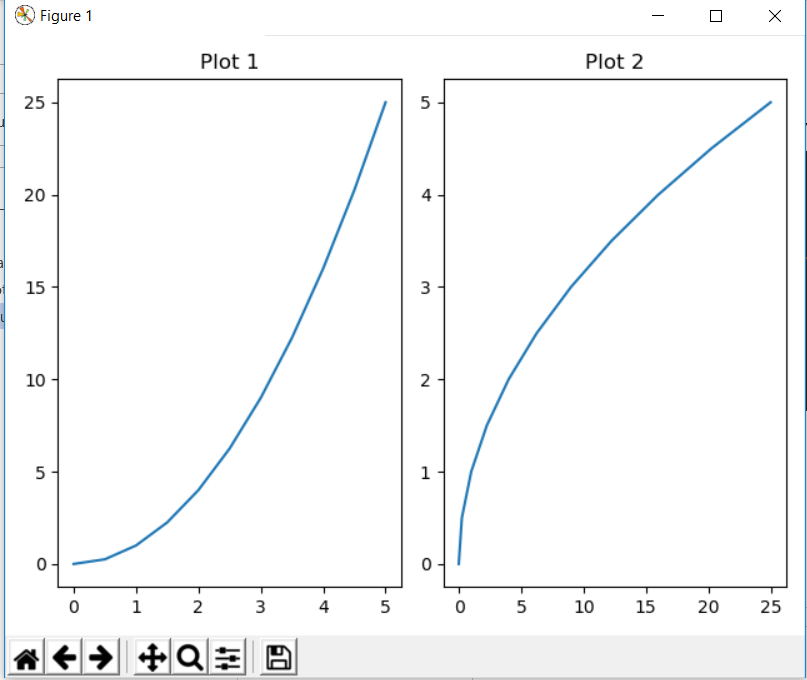
import matplotlib.pyplot as plt  
import numpy as np  
x = np.linspace(0,5,11)  
y = x \*\* 2  
  
fig,axes = plt.subplots(nrows=1,ncols=2)  
for current\_ax in axes:  
 current\_ax.plot(x,y)  
plt.show()



While doing tuple unpacking on the line – fig,axes = …., we can see that axes is an array of axes and we can iterate over it

We can also access this array by indexing as below

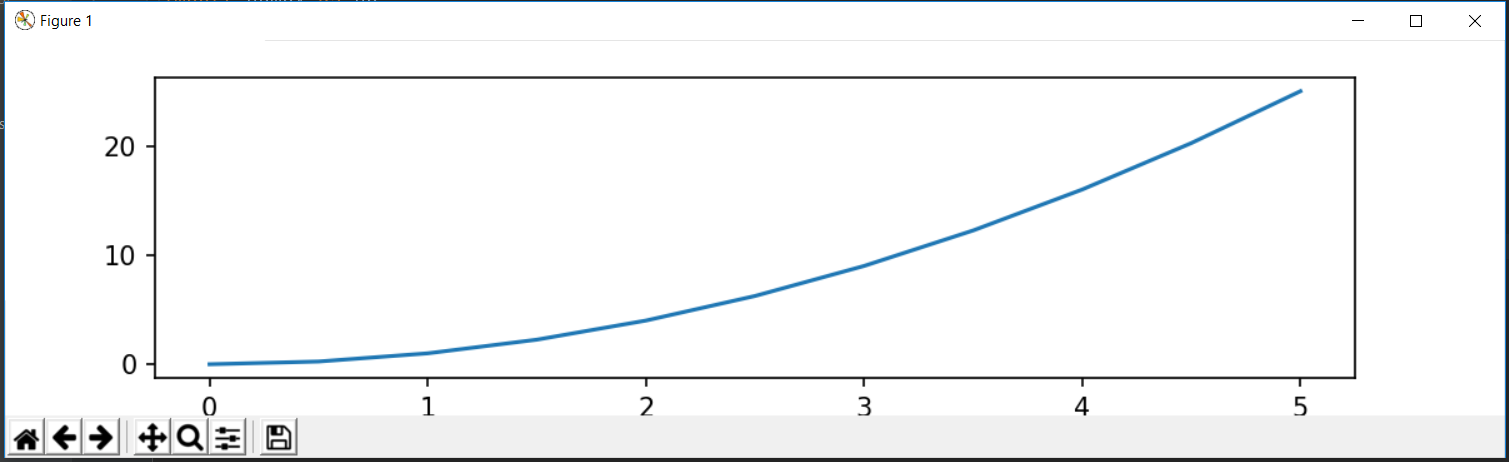
import matplotlib.pyplot as plt  
import numpy as np  
x = np.linspace(0,5,11)  
y = x \*\* 2  
  
fig,axes = plt.subplots(nrows=1,ncols=2)  
axes[0].plot(x,y)  
axes[0].set\_title("Plot 1")  
  
axes[1].plot(y,x)  
axes[1].set\_title("Plot 2")  
  
plt.tight\_layout() # this line take care of overlapping plots  
plt.show()



## Figure Size, Aspect Ratio and DPI

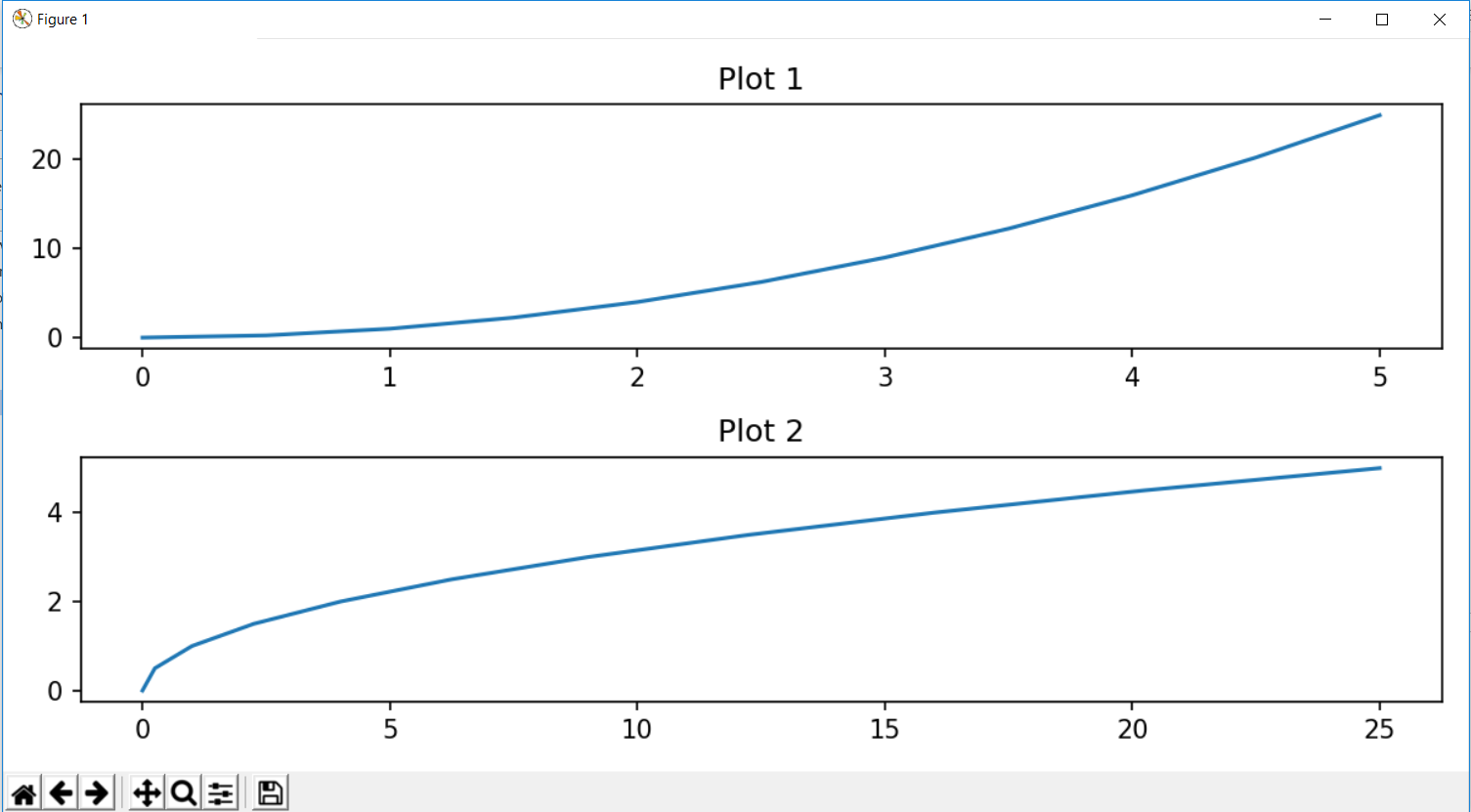
Specifying figure size and dpi-

import matplotlib.pyplot as plt  
import numpy as np  
x = np.linspace(0,5,11)  
y = x \*\* 2  
  
fig = plt.figure(figsize=(8,2),dpi = 150)  
axes = fig.add\_axes([0.1,0.1,0.8,0.8])  
axes.plot(x,y)  
  
plt.tight\_layout()  
plt.show()



This can also be done with subplots-

import matplotlib.pyplot as plt  
import numpy as np  
x = np.linspace(0,5,11)  
y = x \*\* 2  
  
fig,axes = plt.subplots(figsize=(8,4),dpi = 150,nrows=2,ncols=1)  
axes[0].plot(x,y)  
axes[0].set\_title("Plot 1")  
  
axes[1].plot(y,x)  
axes[1].set\_title("Plot 2")  
  
  
plt.tight\_layout()  
plt.show()



## Saving a figure

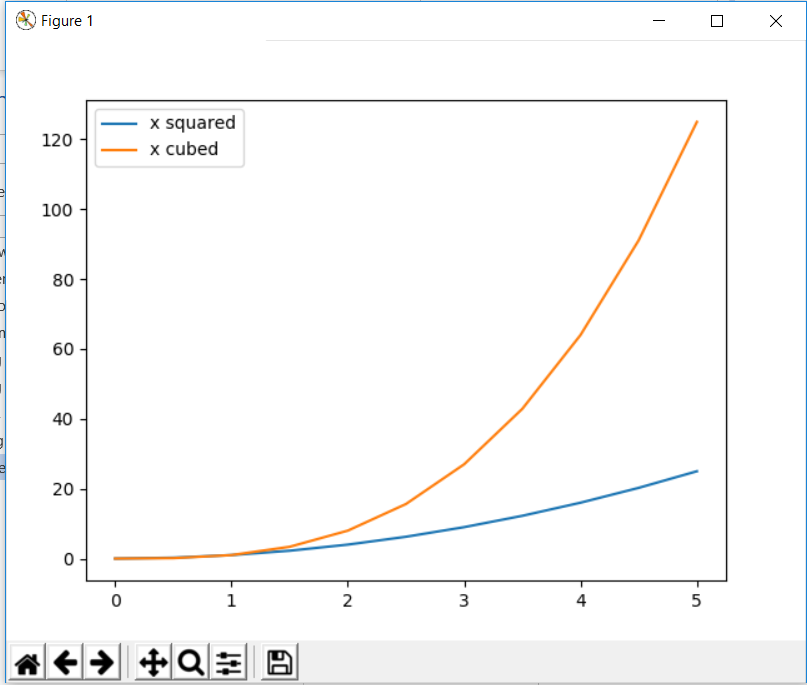
Matplotlib supports variety of file formats such as jpg,png,svg,pdf,etc.

import matplotlib.pyplot as plt  
import numpy as np  
x = np.linspace(0,5,11)  
y = x \*\* 2  
  
fig,axes = plt.subplots(figsize=(8,4),dpi = 150,nrows=2,ncols=1)  
axes[0].plot(x,y)  
axes[0].set\_title("Plot 1")  
  
axes[1].plot(y,x)  
axes[1].set\_title("Plot 2")  
fig.savefig("C:/Users/Vinayak/Desktop/IELTS Writings/test.pdf",dpi = 300)  
  
plt.tight\_layout()  
plt.show()

## Putting legends in the plot

Location code for legends can be seen from the documentation of matplotlib. Also, if we don’t specify label while defining plot, then axes.legend() line will give warning that legends are not present.

import matplotlib.pyplot as plt  
import numpy as np  
x = np.linspace(0,5,11)  
y = x \*\* 2  
  
fig = plt.figure()  
axes = fig.add\_axes([0.1,0.1,0.8,0.8])  
axes.plot(x,x\*\*2,label="x squared")  
axes.plot(x,x\*\*3,label="x cubed")  
  
axes.legend(loc=0)  
#plt.tight\_layout()  
plt.show()



## Setting appearance with Matplotlib

### For colours, we can specify the colour name as string or RGB hex code of the colour as string

import matplotlib.pyplot as plt  
import numpy as np  
x = np.linspace(0,5,11)  
y = x \*\* 2  
  
fig = plt.figure()  
axes = fig.add\_axes([0.1,0.1,0.8,0.8])  
axes.plot(x,x\*\*2,color="#FF00FF")  
  
#plt.tight\_layout()  
plt.show()

### For line width and line transparency

Linewidth parameter can also be written as lw and alpha is the parameter for transparency

import matplotlib.pyplot as plt  
import numpy as np  
x = np.linspace(0,5,11)  
y = x \*\* 2  
  
fig = plt.figure()  
axes = fig.add\_axes([0.1,0.1,0.8,0.8])  
axes.plot(x,x\*\*2,color="#FF00FF",linewidth=3,alpha=0.5)  
  
#plt.tight\_layout()  
plt.show()

