


Name of Student: Sunny Satish Halkatti		
Roll Number: 17 (B)		Lab Assignment Number: 2.3
Title of Lab Assignment: Implementation of Python Libraries - Matplotlib		
DOP: 01/05/2023		DOS: 08/05/2023
CO: CO2	PO: PO3,PO5,PO6, PO7,PO11,PO12	Signature: 

sunny17-practical-2-3

May 8, 2023

Aim: Implementation of Python Libraries - Matplotlib

Matplotlib is a data visualization library in Python that is used to create static, animated, and interactive visualizations in Python. It is a powerful tool for creating high-quality 2D graphics and plots, and it can be used to generate a wide range of visualizations, including line plots, scatter plots, bar plots, histograms, and many more.

Matplotlib is widely used in the scientific community for visualizing data, and it is also used in data science and machine learning applications for exploring and analyzing data. The library is built on top of NumPy, another popular Python library for scientific computing, and it provides a simple and intuitive interface for creating complex visualizations with just a few lines of code.

0.0.1 Functions used in Matplotlib

`plt.plot()`: This function is used to create line plots in Matplotlib. It takes two or more arrays of data as input and plots them as lines on a graph. The function also provides a wide range of customization options, including line styles, colors, markers, and more.

`plt.scatter()`: This function is used to create scatter plots in Matplotlib. It takes two arrays of data as input and plots them as individual points on a graph. The function also provides customization options for point size, color, marker style, and more.

`plt.bar()`: This function is used to create bar plots in Matplotlib. It takes one or more arrays of data as input and plots them as bars on a graph. The function also provides customization options for bar width, color, orientation, and more.

`plt.hist()`: This function is used to create histograms in Matplotlib. It takes one array of data as input and plots the distribution of the data as a series of bins on a graph. The function also provides customization options for bin size, color, and more.

`plt.pie()`: This function is used to create pie charts in Matplotlib. It takes one array of data as input and plots the data as slices of a pie on a graph. The function also provides customization options for slice colors, labels, and more.

`plt.imshow()`: This function is used to create image plots in Matplotlib. It takes an array of data as input and plots the data as an image on a graph. The function also provides customization options for color maps, interpolation, and more.

`plt.subplot()`: This function is used to create subplots in Matplotlib. It allows you to create multiple plots on the same figure and arrange them in a grid-like layout. The function also provides customization options for subplot spacing, titles, and more.

`plt.legend()`: This function is used to add a legend to a plot in Matplotlib. It allows you to label the different elements in a plot, such as lines or bars, and provide a key for interpreting the data. The function also provides customization options for legend location, font size, and more.

`plt.xlabel()` and `plt.ylabel()`: These functions are used to set the x and y-axis labels of a plot in Matplotlib. They allow you to provide a clear and concise description of the data being plotted and help readers understand the meaning of the axes.

`plt.title()`: This function is used to set the title of a plot in Matplotlib. It allows you to provide a brief summary of the data being plotted and can help readers quickly understand the main message of the plot.

`plt.xlim()` and `plt.ylim()`: These functions are used to set the limits of the x and y-axes of a plot in Matplotlib. They allow you to control the range of data being displayed on the plot and can be used to zoom in or out on specific parts of the data.

`plt.grid()`: This function is used to add a grid to a plot in Matplotlib. It can help readers better understand the data being plotted by providing a reference for the values on the x and y-axes.

`plt.subplots_adjust()`: This function is used to adjust the spacing between subplots in Matplotlib. It allows you to fine-tune the layout of your plots and ensure that they are clear and easy to read.

`plt.savefig()`: This function is used to save a plot as an image file in Matplotlib. It allows you to share your results with others and include your plots in reports, presentations, and other documents.

```
[ ]: import matplotlib.pyplot as plt
```

```
[ ]: %matplotlib inline
```

0.0.2 Example

```
[ ]: import numpy as np
x = np.linspace(0,5,11)
y = x**2
```

```
[ ]: x
```

```
[ ]: array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ])
```

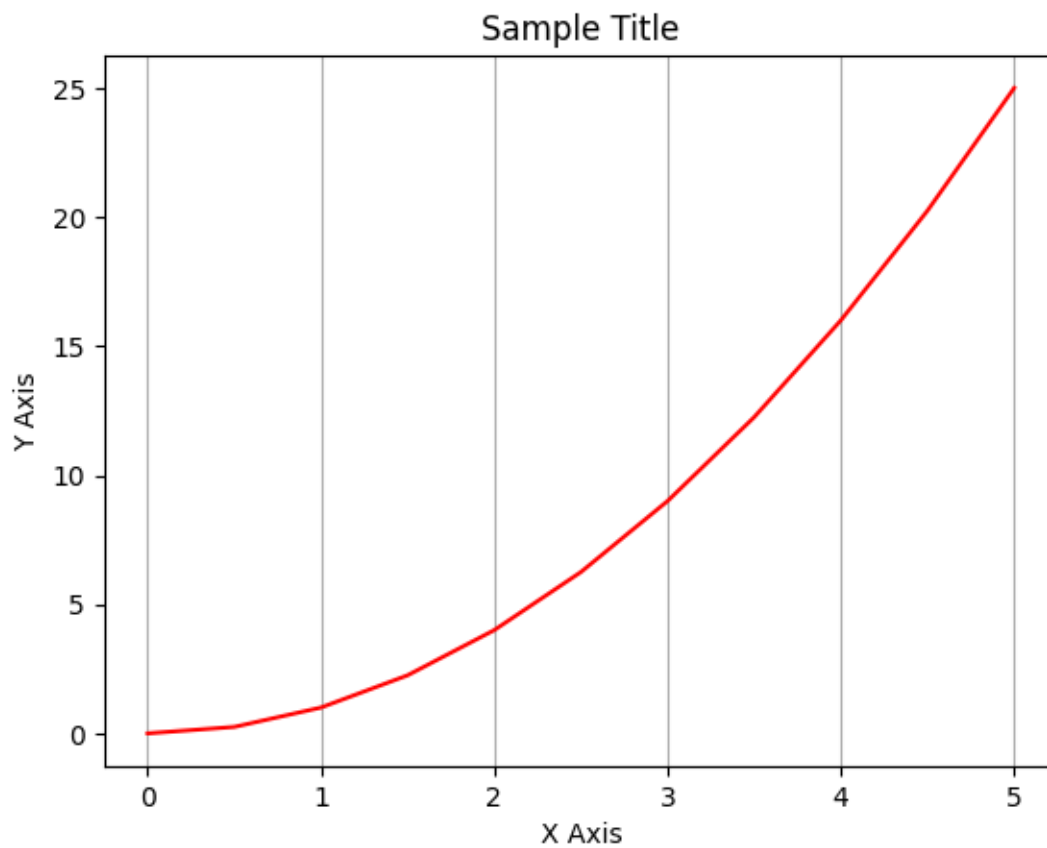
```
[ ]: y
```

```
[ ]: array([ 0. ,  0.25,  1. ,  2.25,  4. ,  6.25,  9. , 12.25, 16. ,
        20.25, 25. ])
```

0.0.3 Basic Matplotlib command

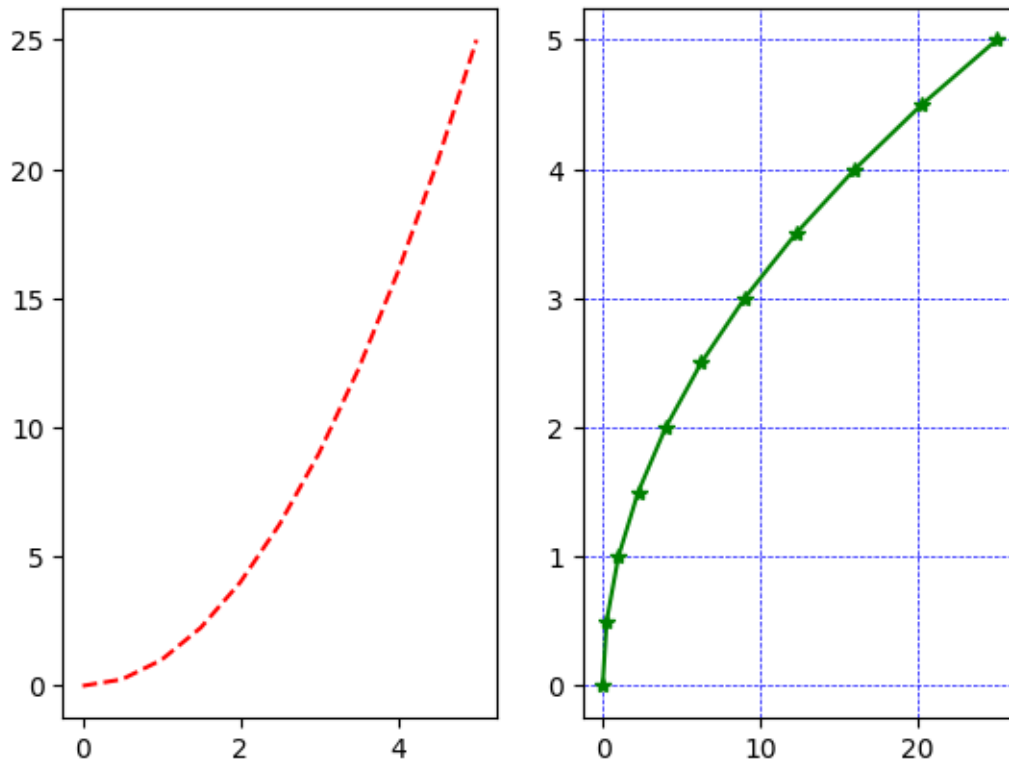
```
[ ]: plt.plot(x,y,'r') #r = red
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.title("Sample Title")
```

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



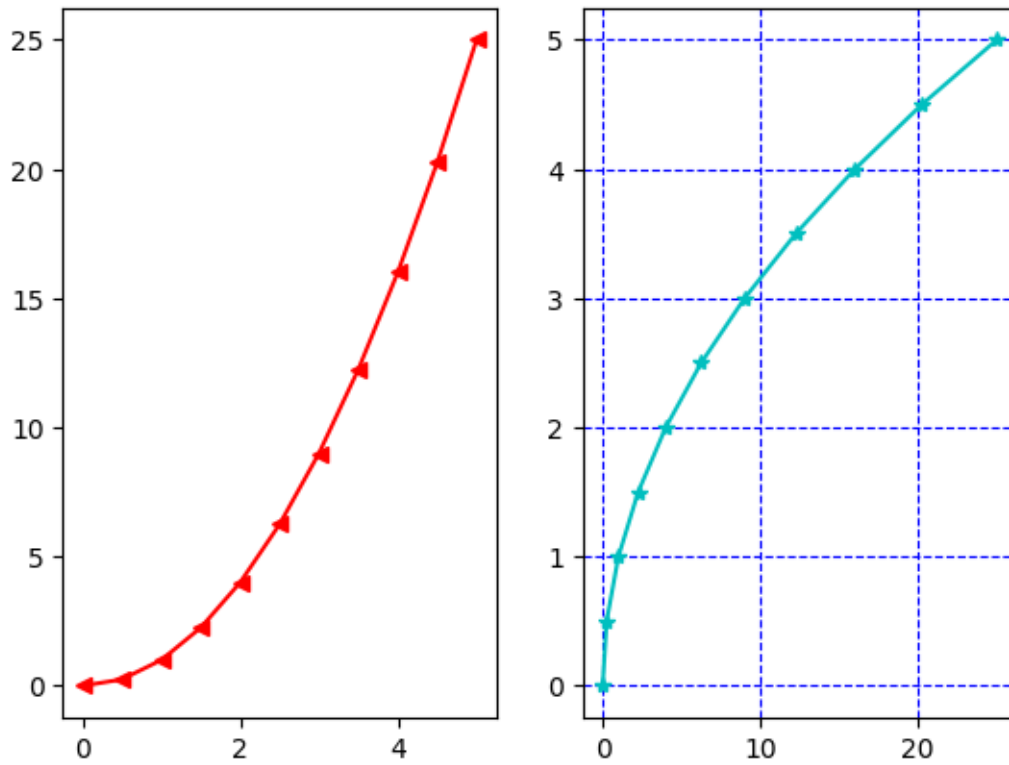
```
[ ]: plt.subplot(1,2,1)
plt.plot(x,y, 'r--')
plt.subplot(1,2,2)
plt.grid(color='blue',linestyle = '--',linewidth = 0.5)
plt.plot(y,x, 'g*-')
```

```
[ ]: [<matplotlib.lines.Line2D at 0x7f69284113c0>]
```



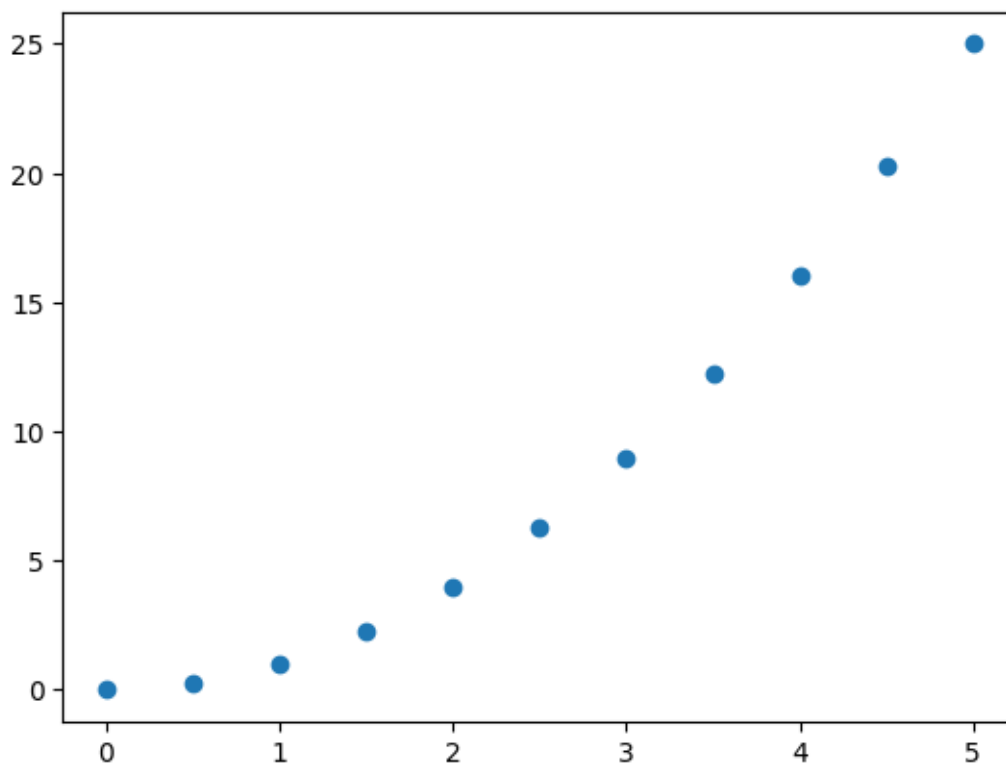
```
[ ]: plt.subplot(1,2,1)
plt.plot(x,y,'r<-')
plt.subplot(1,2,2)
plt.grid(color='blue',linestyle = '--',linewidth = 0.8)
plt.plot(y,x,'c*-')
```

```
[ ]: [<matplotlib.lines.Line2D at 0x7f69284bf3d0>]
```



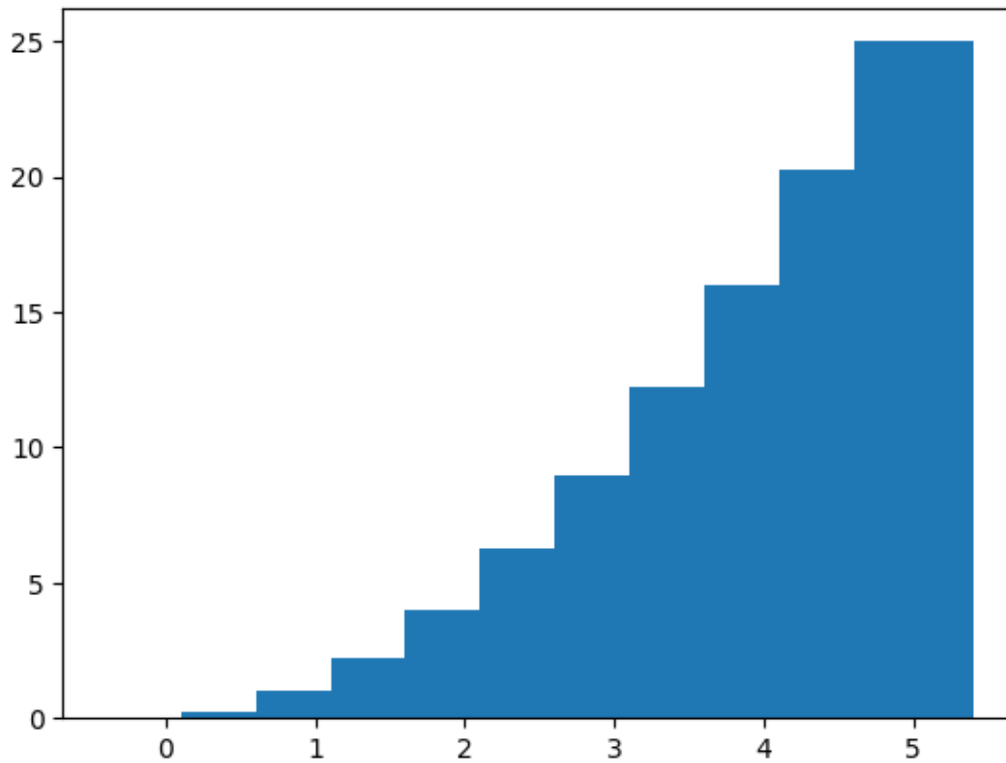
```
[ ]: plt.scatter(x,y)
```

```
[ ]: <matplotlib.collections.PathCollection at 0x7f6928367ac0>
```

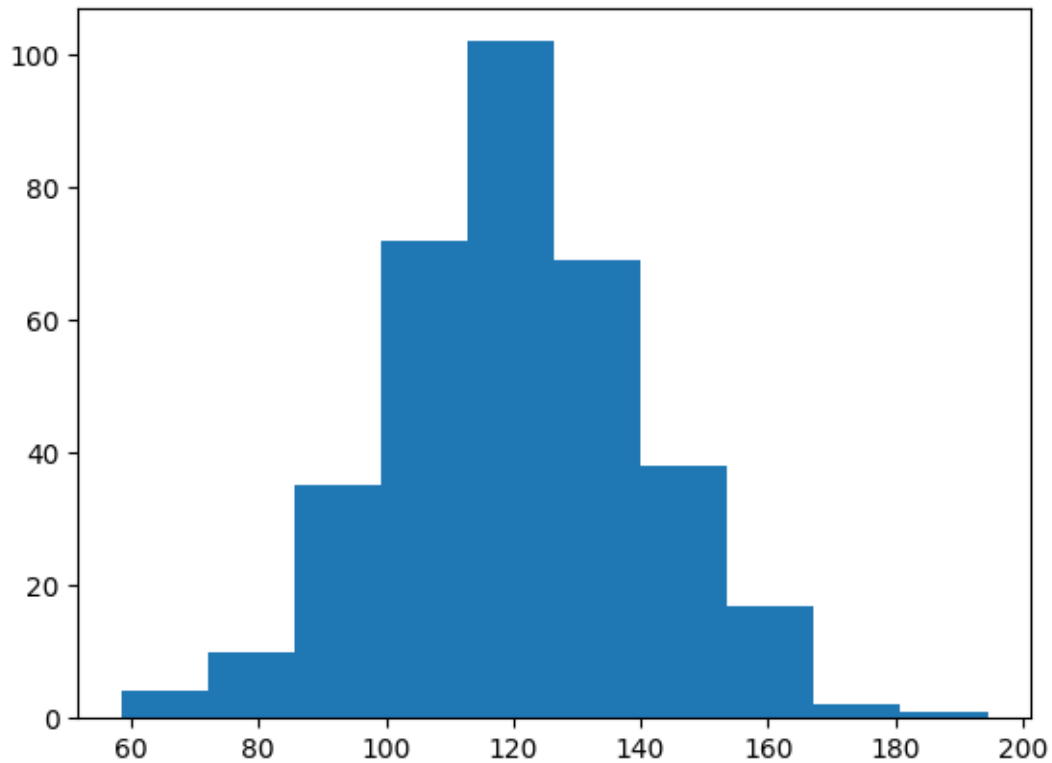


```
[ ]: plt.bar(x,y)
```

```
[ ]: <BarContainer object of 11 artists>
```



```
[ ]: x = np.random.normal(120,20,350)
      #np.random.normal(loc,scale and size) loc sets the mean distribution,
      #scale sets the standard deviation and size gives the shape of numpy array
      plt.hist(x)
      plt.show()
```

```
[ ]: x
```

```
[ ]: array([ 87.18930114, 109.56461149,  91.64188393, 141.0903821 ,
          116.92068142,  88.28149878, 117.71028951,  86.06506703,
          117.62740248, 135.3388476 , 110.94828943, 118.88789419,
          144.85193834, 108.05934462, 139.8518624 , 132.60239319,
          121.13402183, 131.91364531, 107.701472 , 136.3869506 ,
          103.66452356, 138.19975445, 127.40424279, 113.2174437 ,
          107.7618862 ,  70.95860425, 148.41072548, 158.57287595,
          117.55665417, 129.17232095, 113.92530505, 123.05439602,
          160.33213741, 128.39610648, 129.21905515, 130.44338731,
          126.42244665,  99.96104033, 128.32940489, 121.48364623,
          153.55180112, 153.04805623, 132.85638901,  98.75426698,
          128.08143717, 108.9498287 , 114.81824805, 110.97469717,
          128.54457694, 122.66715013, 122.50670259,  85.04336524,
          110.25319646, 124.66081686,  83.47338329, 122.73831516,
          108.38731716, 137.68863149,  82.52149042, 162.73028798,
           83.26777138, 134.460058 , 132.01595099, 122.34808114,
          128.10284842, 121.40197649, 101.4542247 , 122.80620708,
          164.65405167, 141.09177543, 104.75423055, 145.98795479,
          120.03766851,  93.93277156, 130.342223 , 117.78714294,
          113.48083331, 118.90130694, 132.10442917,  73.57483991,
```

120.07182217, 97.96136541, 89.21498297, 110.83381412,
 150.27168146, 114.53812442, 116.59427423, 127.48503429,
 90.3463805 , 111.39521508, 120.9949404 , 97.17525231,
 128.82812754, 154.60368097, 130.70016026, 160.68969948,
 93.83263832, 102.43592577, 128.10594918, 113.33894443,
 93.67539638, 110.03359809, 113.15686279, 130.85569889,
 133.88758705, 107.66316234, 139.54086132, 132.91627543,
 113.37443662, 115.89427604, 102.37434785, 111.62228348,
 96.77877352, 90.850881 , 119.72312728, 120.4366145 ,
 126.09477351, 129.71654405, 157.43671565, 100.00795218,
 106.44548391, 120.21665858, 118.24117887, 130.50914538,
 122.92363092, 134.84641236, 106.81925528, 107.18615996,
 114.30109967, 125.77149954, 194.44991915, 141.03813733,
 90.68444174, 118.20784609, 140.63405268, 110.95314556,
 67.38343415, 94.72822469, 120.1705617 , 102.57118788,
 127.30605683, 126.59666614, 130.471871 , 120.84773721,
 123.13710473, 126.98081486, 132.58864988, 151.9698762 ,
 150.12116148, 110.44402104, 118.03913022, 133.61577897,
 76.16130709, 121.99850457, 119.90397205, 131.11498536,
 149.88884282, 131.7803472 , 124.39808929, 153.52486604,
 93.21224681, 103.56398887, 154.14129767, 143.47854831,
 128.53969889, 82.99833164, 122.75610671, 122.61794473,
 128.71685455, 101.71663983, 141.7820295 , 153.8197448 ,
 97.00228669, 143.61374477, 119.53913697, 105.97983892,
 101.38361736, 141.12452736, 123.99730147, 112.77972648,
 142.34941535, 116.44094104, 124.82961276, 93.81972942,
 112.34056087, 112.99463861, 143.74513271, 97.58164179,
 124.25699023, 117.90164555, 94.35279802, 125.85811125,
 176.19465968, 104.76583348, 117.85147733, 96.06819628,
 113.20772119, 130.96488972, 114.21967374, 118.81894529,
 120.47731997, 114.45508758, 121.39307408, 122.66244928,
 127.22630229, 130.78217715, 171.49075904, 58.43832601,
 134.64183471, 118.60420646, 84.40061459, 105.84767035,
 110.24060984, 106.09668074, 128.67281369, 125.3065772 ,
 99.80002537, 149.46749039, 152.46595034, 159.95649586,
 155.79892102, 95.16390506, 111.12629901, 111.46396163,
 120.01006723, 164.59474438, 162.2639604 , 129.3839597 ,
 106.26305068, 103.21958879, 113.97649705, 146.06623777,
 107.04151263, 81.33047984, 120.35412806, 129.82010593,
 123.40136996, 115.07706775, 99.32608898, 138.10317826,
 149.73868331, 84.44987194, 150.28786739, 151.22464162,
 122.63830529, 99.57943801, 105.00559641, 109.99689291,
 118.69591599, 116.00123401, 130.89940095, 103.35044467,
 117.46959693, 103.64527982, 106.51325993, 102.97417317,
 142.7323006 , 154.58029373, 95.05536638, 155.05310489,
 115.98174787, 150.25077984, 97.63130458, 97.16259949,
 112.68690363, 117.75114417, 122.04428633, 58.39477562,

```

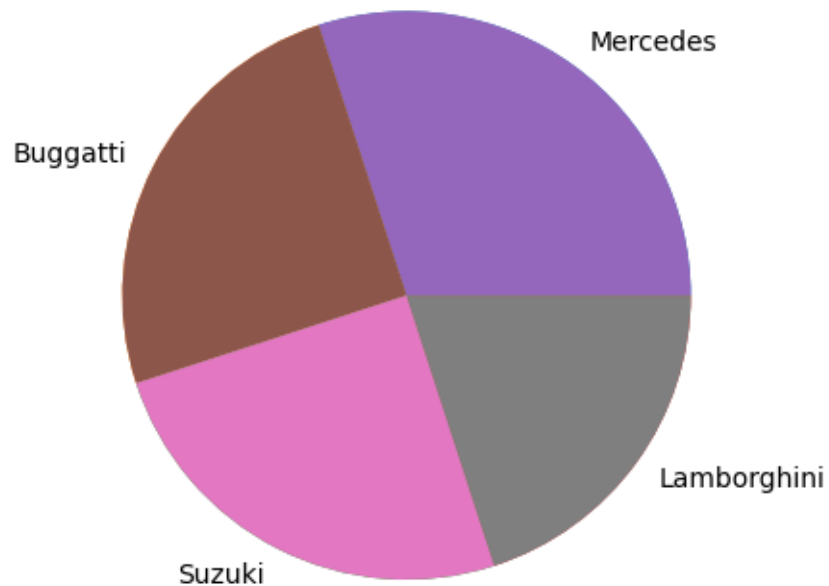
96.51388371, 135.4794465 , 121.17747873, 109.98928307,
136.3567971 , 133.62879688, 117.38549192, 105.46137918,
113.42724728, 113.90215196, 118.06687658, 132.10331236,
149.67027251, 141.76928324, 121.01513472, 107.43456974,
117.62992673, 96.01275798, 103.10215049, 101.21795697,
90.91692672, 136.92488482, 129.59434282, 102.28608213,
148.36039233, 113.47735697, 163.57608995, 108.68173574,
107.44865677, 140.03155567, 125.7075302 , 90.66268316,
107.55572749, 128.04902317, 108.80372431, 105.59268358,
105.38062136, 118.56523896, 121.70538399, 100.51960174,
141.09617967, 113.85199796, 122.35622737, 116.3287168 ,
124.12990755, 137.7269233 , 122.56038792, 123.80963198,
108.22816045, 92.98138114, 134.28772332, 127.20000342,
131.89607357, 93.08824917, 108.82485191, 150.89487855,
138.26371743, 113.78991481, 117.28436868, 97.06443742,
120.8934627 , 129.44785076, 143.23223805, 90.31306473,
115.12548976, 110.61771398, 113.34181718, 124.77963985,
100.69375512, 130.3065502 , 94.05283061, 130.55180014,
141.36997949, 108.12026181, 126.19037716, 134.9381119 ,
112.60802962, 160.80869532, 129.23094212, 141.81337327,
143.27559718, 108.9872597 ] )

```

```

[ ]: y = np.array([30,25,25,20])
plt.pie(y)
mylabels = ['Mercedes','Buggatti','Suzuki','Lamborghini']
plt.pie(y,labels = mylabels)
plt.show()

```



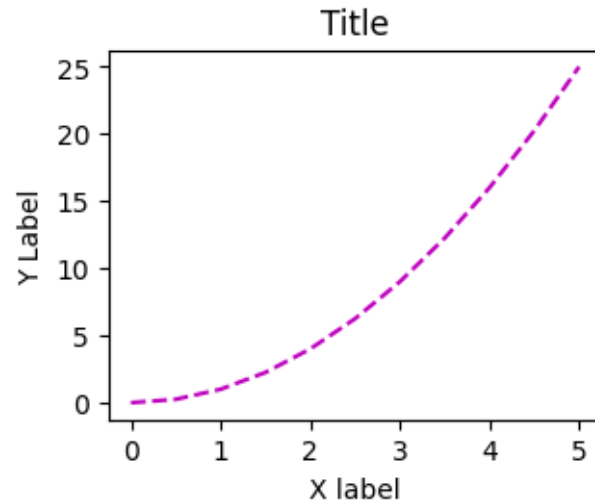
0.0.4 Introduction to the Object Oriented Method

```
[ ]: x = np.linspace(0,5,11)
     y= x**2
```

```
[ ]: #Create Figure (empty canvas)
fig = plt.figure()
#Addd set of axes to figure
axes = fig.add_axes([0.2,0.2,0.4,0.4]) #left, bottom, width, height(range 0 to 1)

#Plot on that set of axes
axes.plot(x,y,'m--')
axes.set_xlabel('X label')
axes.set_ylabel('Y Label')
axes.set_title('Title')
```

```
[ ]: Text(0.5, 1.0, 'Title')
```



0.0.5 Inset Graphs

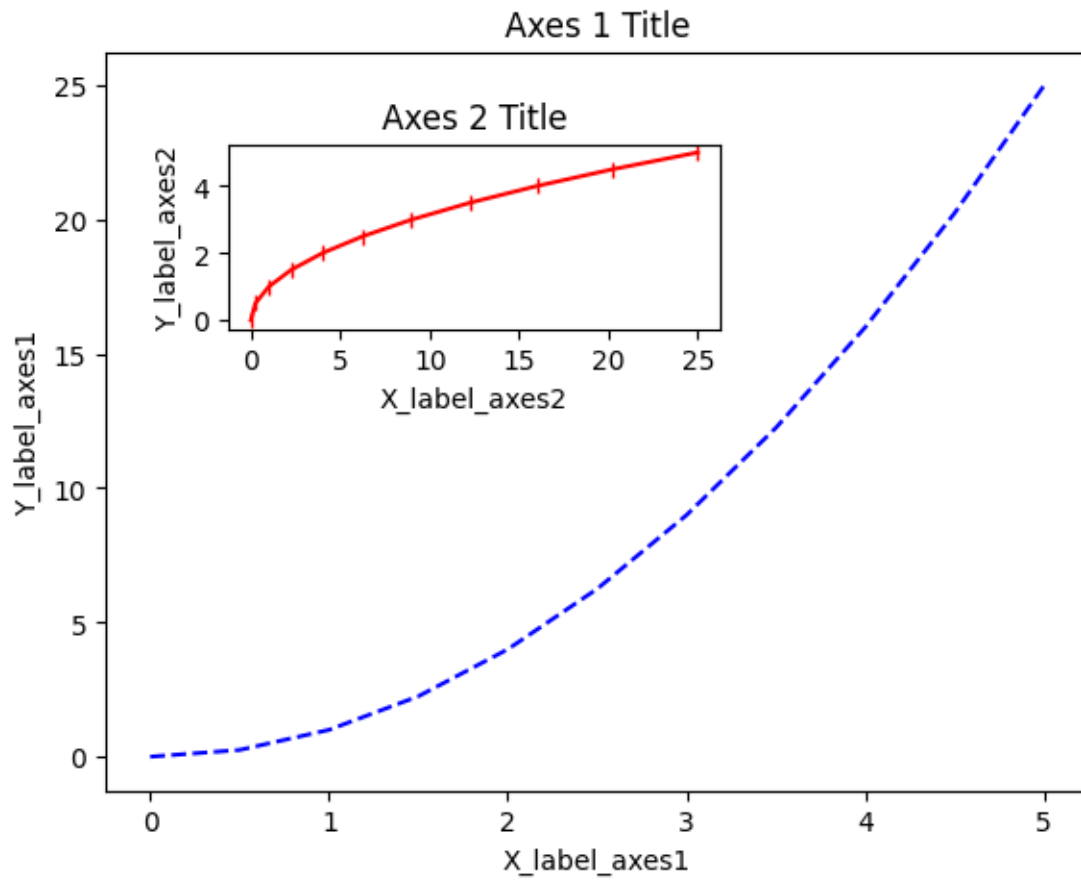
```
[ ]: #Create blank canvas
fig = plt.figure()

axes1 = fig.add_axes([0.1,0.1,0.8,0.8]) #main axes
axes2 = fig.add_axes([0.2,0.6,0.4,0.2]) #inset axes

#Larger Figure Axes 1
axes1.plot(x,y,'b--')
axes1.set_xlabel('X_label_axes1')
axes1.set_ylabel('Y_label_axes1')
axes1.set_title('Axes 1 Title')

#Insert Figure Axes 2
axes2.plot(y,x,'r|-')
axes2.set_xlabel('X_label_axes2')
axes2.set_ylabel('Y_label_axes2')
axes2.set_title('Axes 2 Title')
```

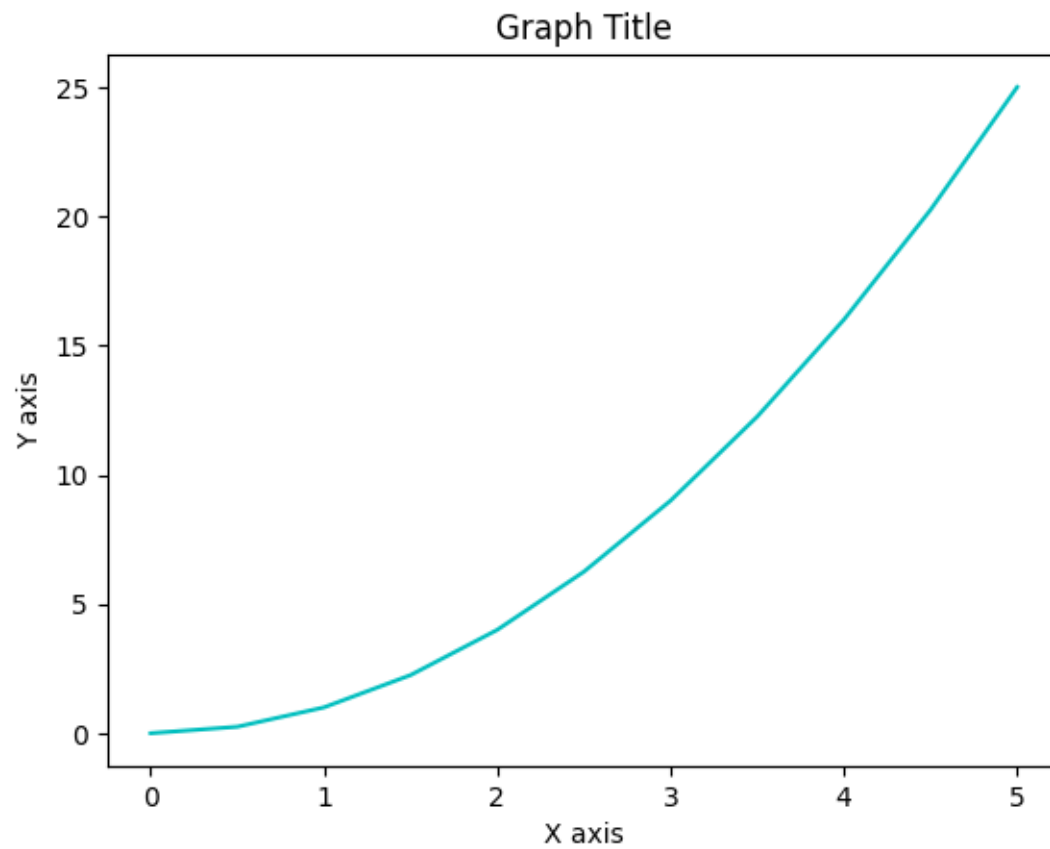
```
[ ]: Text(0.5, 1.0, 'Axes 2 Title')
```



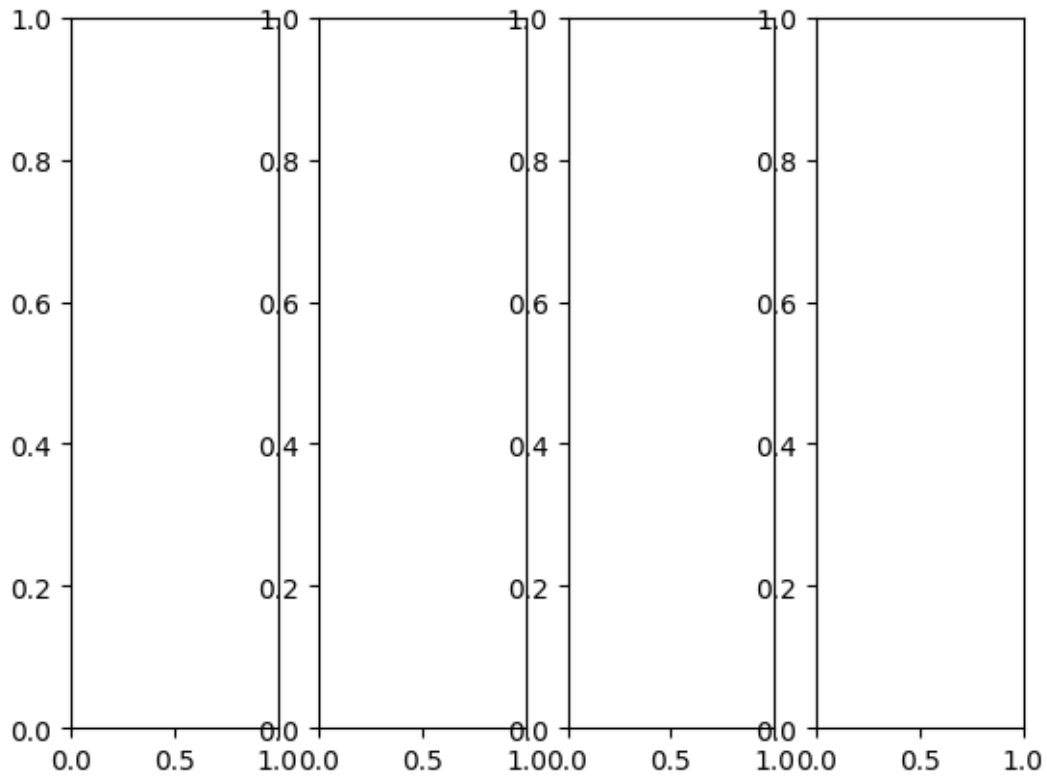
0.0.6 Subplots

```
[ ]: fig, axes = plt.subplots()
    axes.plot(x, y, 'c')
    axes.set_xlabel('X axis')
    axes.set_ylabel('Y axis')
    axes.set_title('Graph Title')
```

```
[ ]: Text(0.5, 1.0, 'Graph Title')
```



```
[ ]: #Empty canvas of 1 by 2 subplots  
fig, axes = plt.subplots(nrows=1, ncols=4)
```



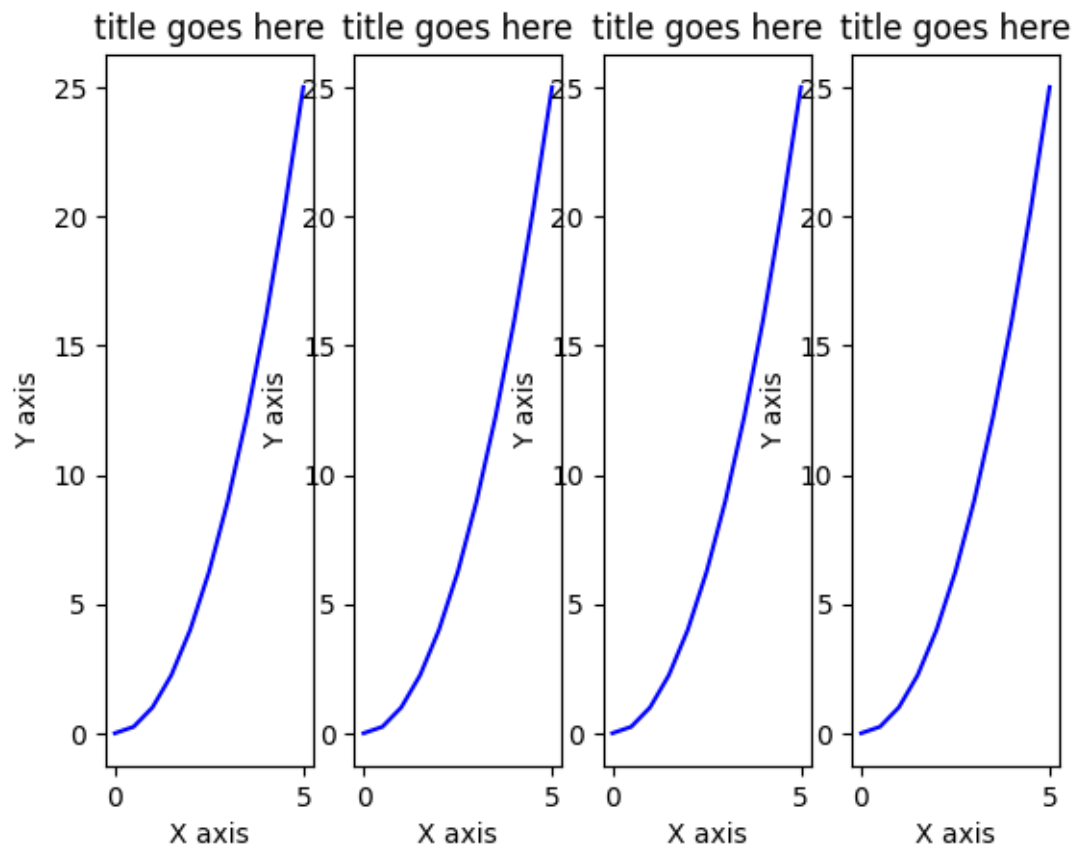
```
[ ]: #Axes is an array of axes to plot on
axes
```

```
[ ]: array([<Axes: >, <Axes: >, <Axes: >, <Axes: >], dtype=object)
```

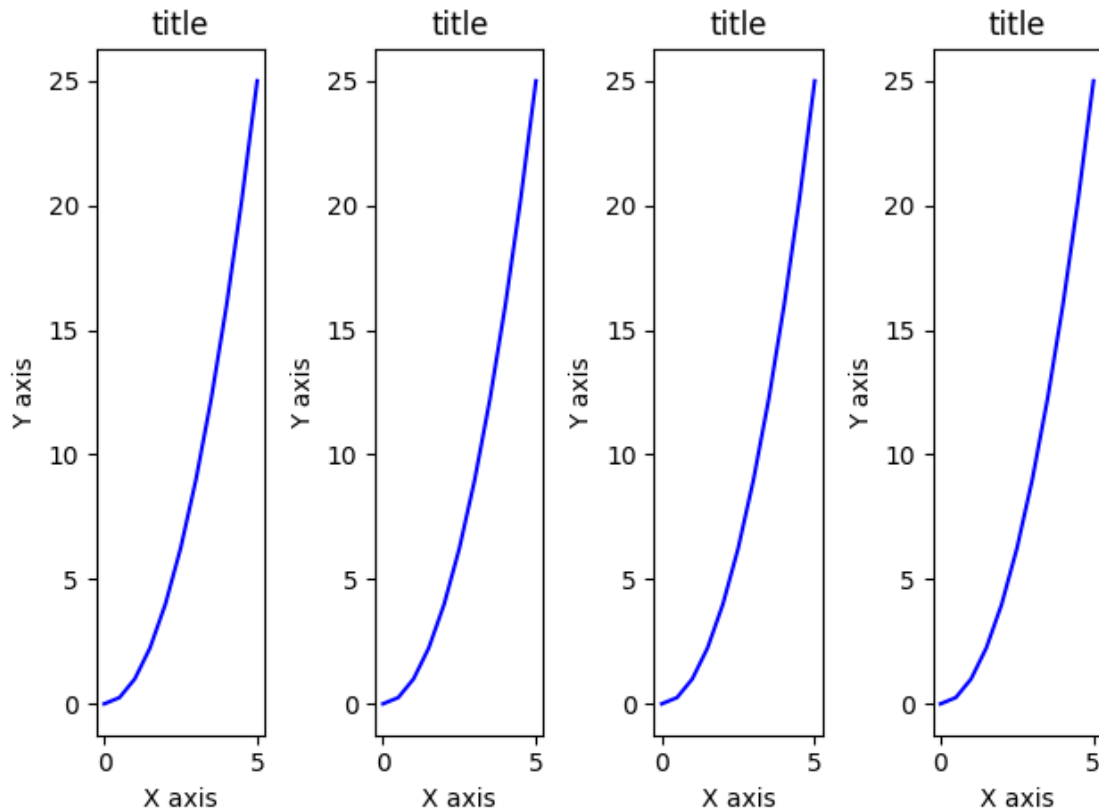
We can iterate through this array

```
[ ]: for ax in axes:
      ax.plot(x,y,'b')
      ax.set_xlabel('X axis')
      ax.set_ylabel('Y axis')
      ax.set_title('title goes here')
fig
```

```
[ ]:
```

```
[ ]: fig, axes = plt.subplots(nrows=1, ncols=4)
for ax in axes:
    ax.plot(x, y, 'b')
    ax.set_xlabel('X axis')
    ax.set_ylabel('Y axis')
    ax.set_title('title')
fig.tight_layout()
```



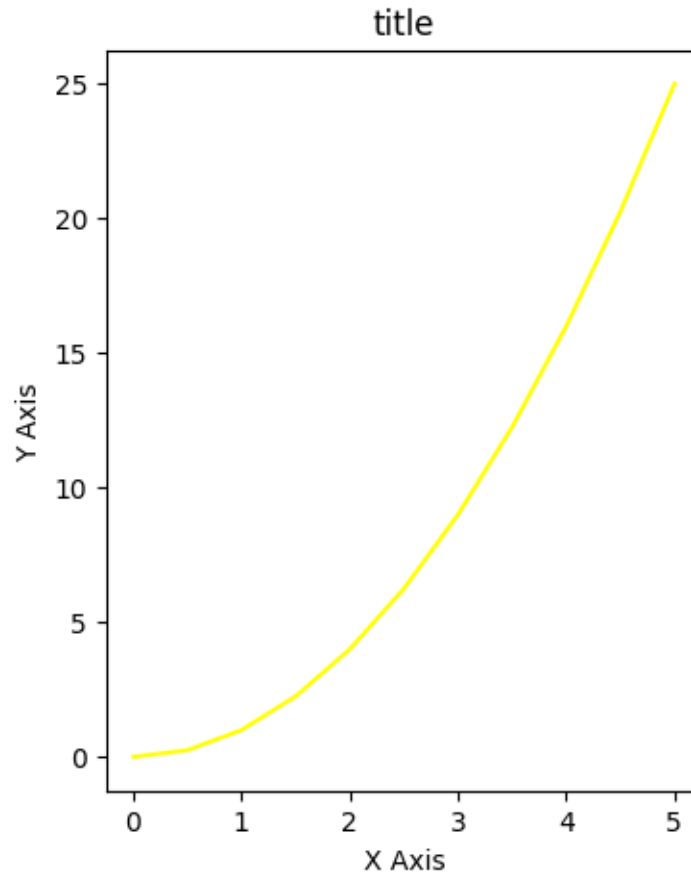
```
[ ]: x = np.linspace(0,5,11)
     y = x ** 2
```

```
[ ]: fig = plt.figure(figsize=(4,4),dpi=50)
```

<Figure size 200x200 with 0 Axes>

```
[ ]: fig,axes = plt.subplots(figsize=(4,5),dpi=100)
     axes.plot(x,y,'yellow')
     axes.set_xlabel('X Axis')
     axes.set_ylabel('Y Axis')
     axes.set_title('title')
```

```
[ ]: Text(0.5, 1.0, 'title')
```



```
[ ]: import os
import matplotlib.pyplot as plt
script_dir= os. path.dirname('graph')
results_dir= os. path.join(script_dir, 'Results/')
sample_file_name = "sample"
if not os. path.isdir (results_dir):
    os.makedirs(results_dir)
```

```
[ ]: fig.savefig("graph2.png", dpi=200)
```

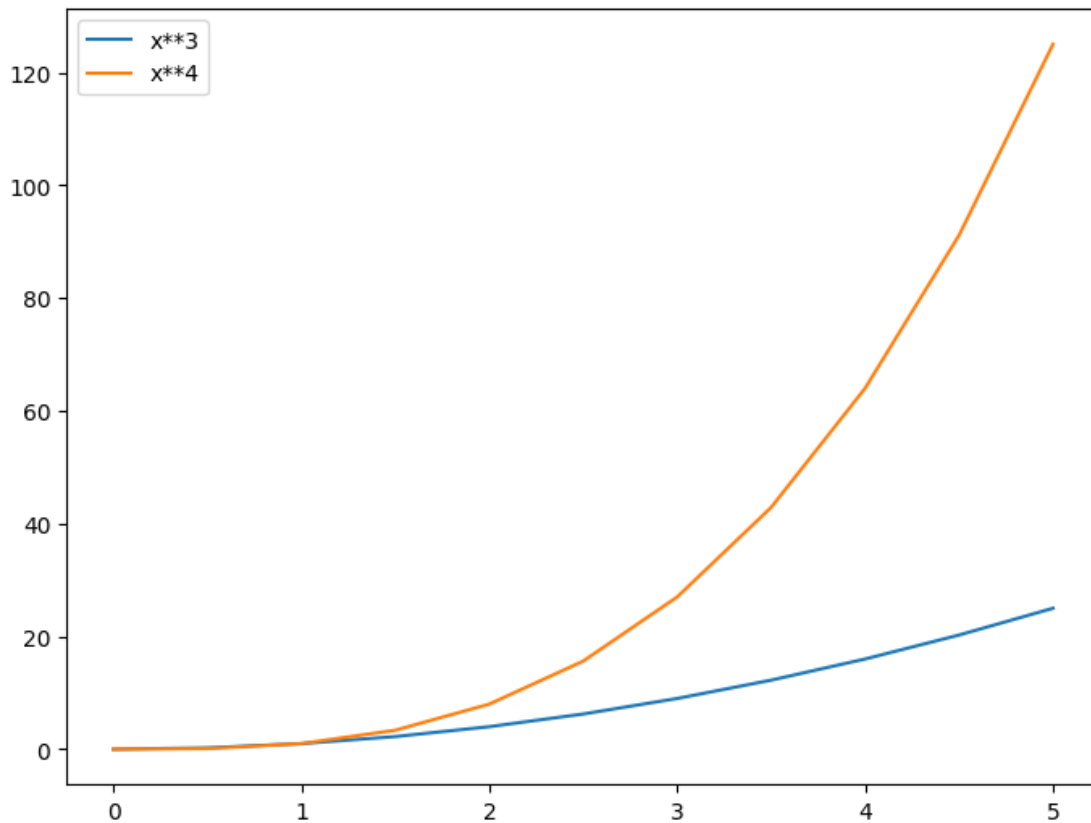
0.0.7 Legends

You can use the label="label text" keyword argument when plots or other objects are added to the figure, and then using the leg method without arguments to add the legend to the figure:

```
[ ]: fig = plt.figure()
ax = fig.add_axes ([0,0,1,1])
ax.plot(x, x**2, label="x**3")
ax.plot(x, x**3, label="x**4")
```

```
ax. legend()
```

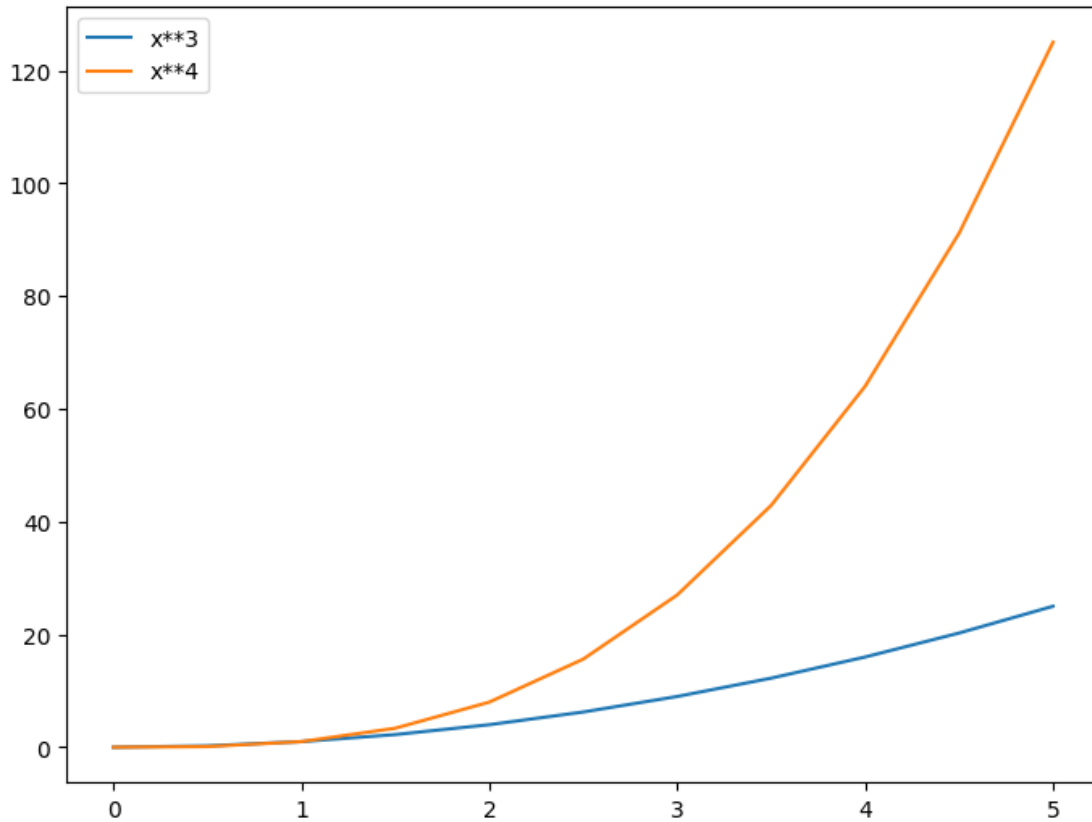
```
[ ]: <matplotlib.legend.Legend at 0x7f6908e338e0>
```



The legend function takes an optional keyword argument `loc` that can be used to specify where in the figure the legend is to be drawn. The allowed values of `loc` are numerical codes for the various places the legend can be drawn.

```
[ ]: # Lots of options....
ax. legend (loc=1) # upper right corner
ax. legend (loc=2) # upper left corner
ax. legend (loc=3) # lower left corner
ax. legend (loc=4) # lower right corner
# .. many more options are available
# Most common to choose
ax. legend (loc=0) # let matplotlib decide the optimal location
fig
```

```
[ ]:
```



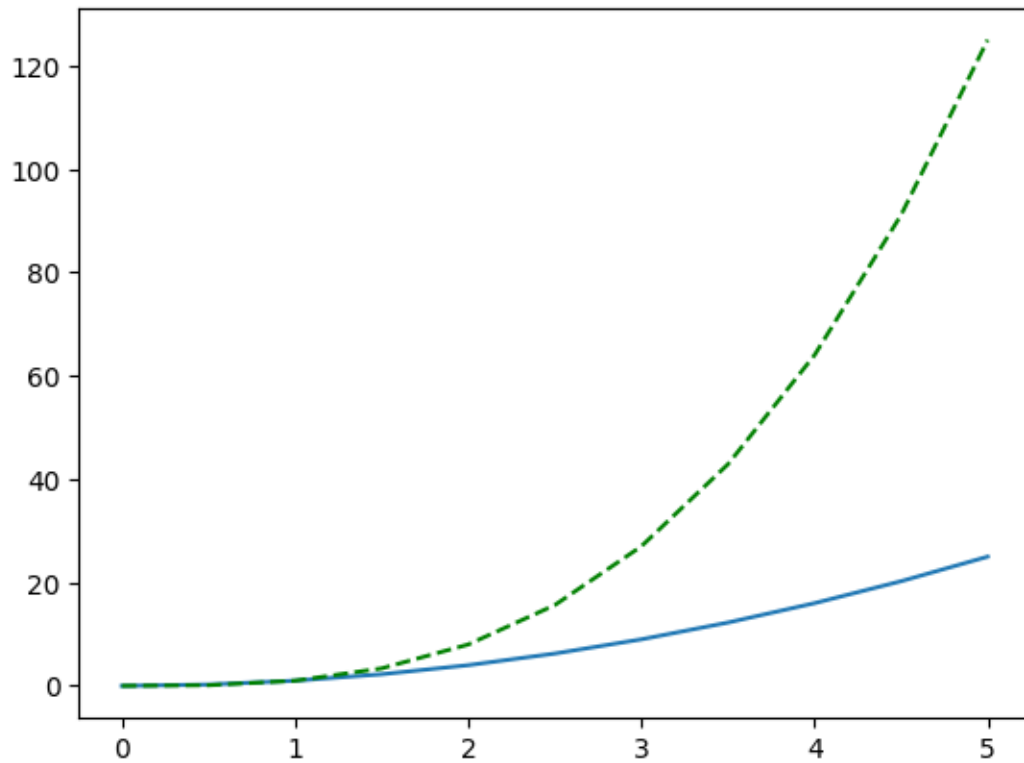
0.0.8 Setting colors, linewidths, linetypes

Matplotlib gives you a lot of options for customizing colors, linewidths, and linetypes.

With matplotlib, we can define the colors of lines and other graphical elements in a number of ways. for example, 'b.-' means a b line with dots:

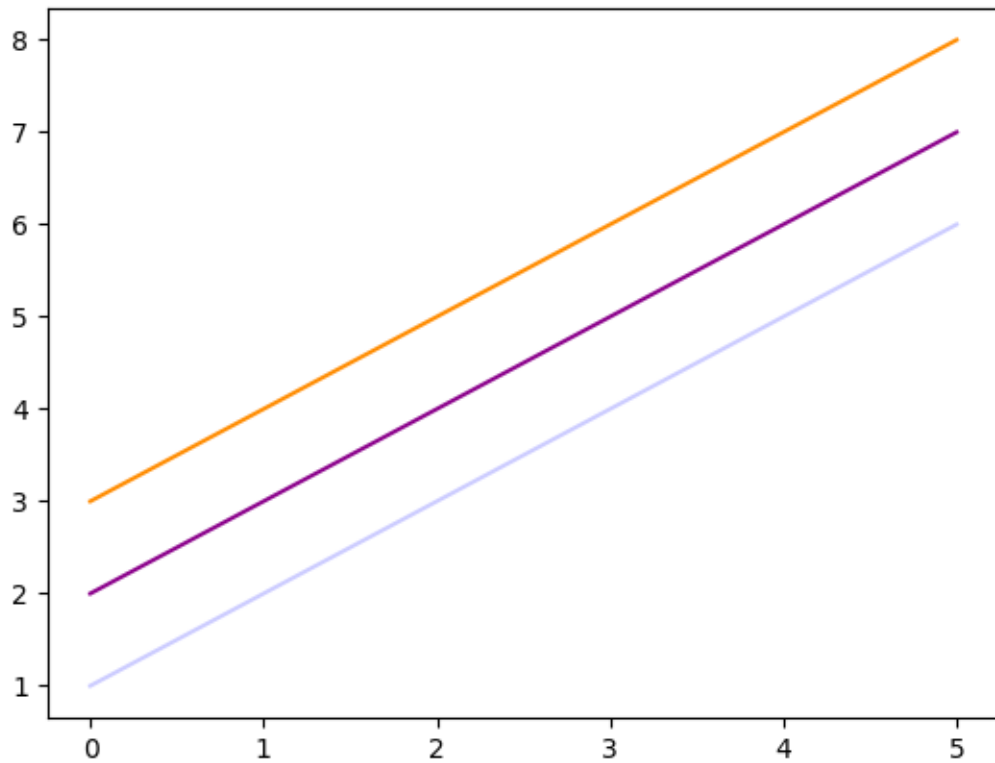
```
[ ]: # MATLAB style line color and style
fig, ax = plt.subplots()
ax.plot(x, x**2, 'b.-') # blue line with dots
ax.plot(x, x**3, 'g--') # green dashed line
```

```
[ ]: [<matplotlib.lines.Line2D at 0x7f6908dd9150>]
```

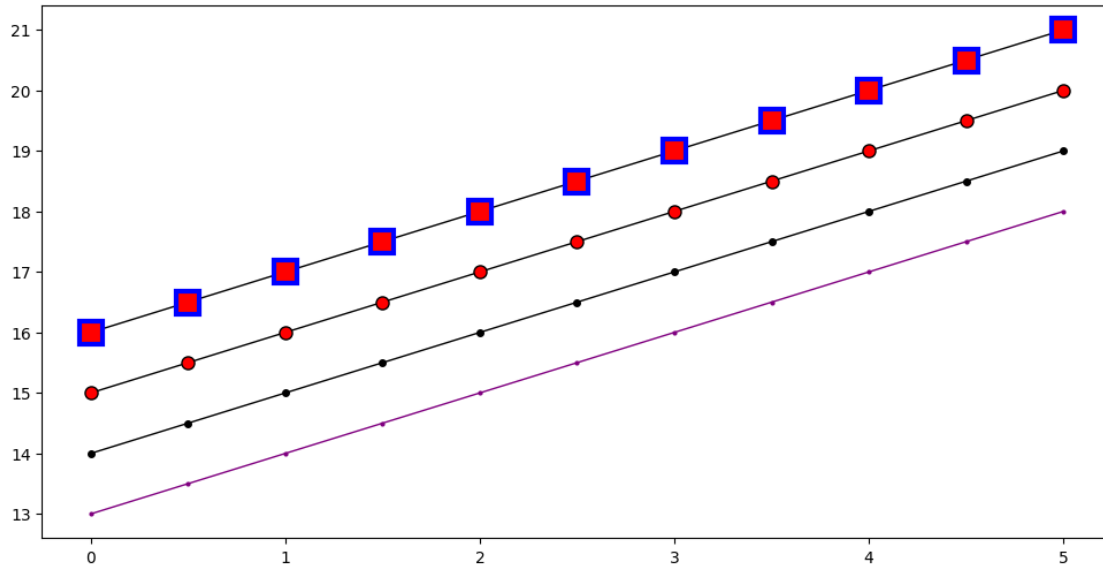


```
[ ]: fig, ax = plt.subplots ()  
ax.plot(x,x+1, color="blue", alpha=0.2) # half-transparent  
ax.plot(x, x+2, color="#8B008B") # RGB hex code  
ax.plot(x, x+3, color="#FF8C00") # RGB hex code
```

```
[ ]: [<matplotlib.lines.Line2D at 0x7f6908ce2ce0>]
```



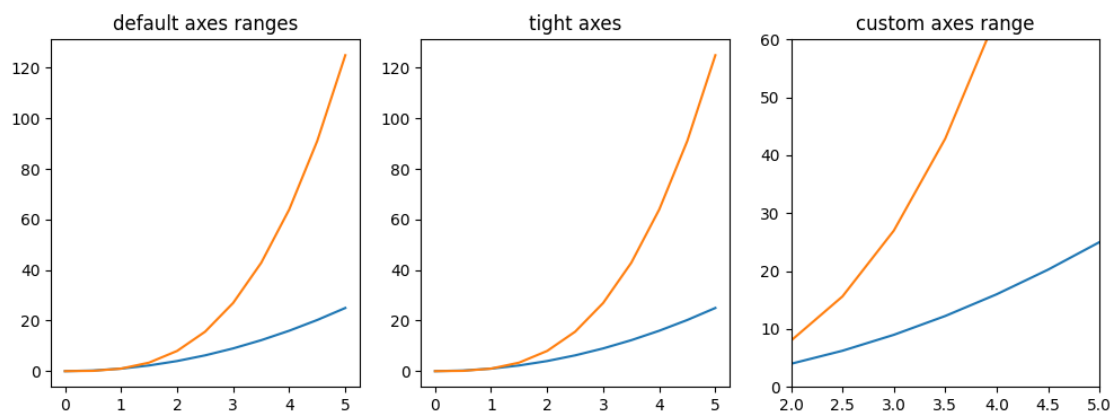
```
[ ]: fig, ax = plt.subplots (figsize=(12,6))
ax.plot(x, x+13, color="purple", lw=1, ls='-', marker='o', markersize=2)
ax.plot(x, x+14, color="k", lw=1, ls='-', marker='o', markersize=4)
ax.plot(x, x+15, color="k", lw=1, ls='-', marker='o', markersize=8,
        ↪markerfacecolor="red")
ax.plot(x, x+16, color="k", lw=1, ls='-', marker='s', markersize=15,
        markerfacecolor="red", markeredgewidth=3, markeredgecolor="blue");
```



0.0.9 Plot range

We can configure the ranges of the axes using the `set_ylim` and `set_xlim` methods in the axis object, or axis ('tight') for automatically getting "tightly fitted" axes ranges:

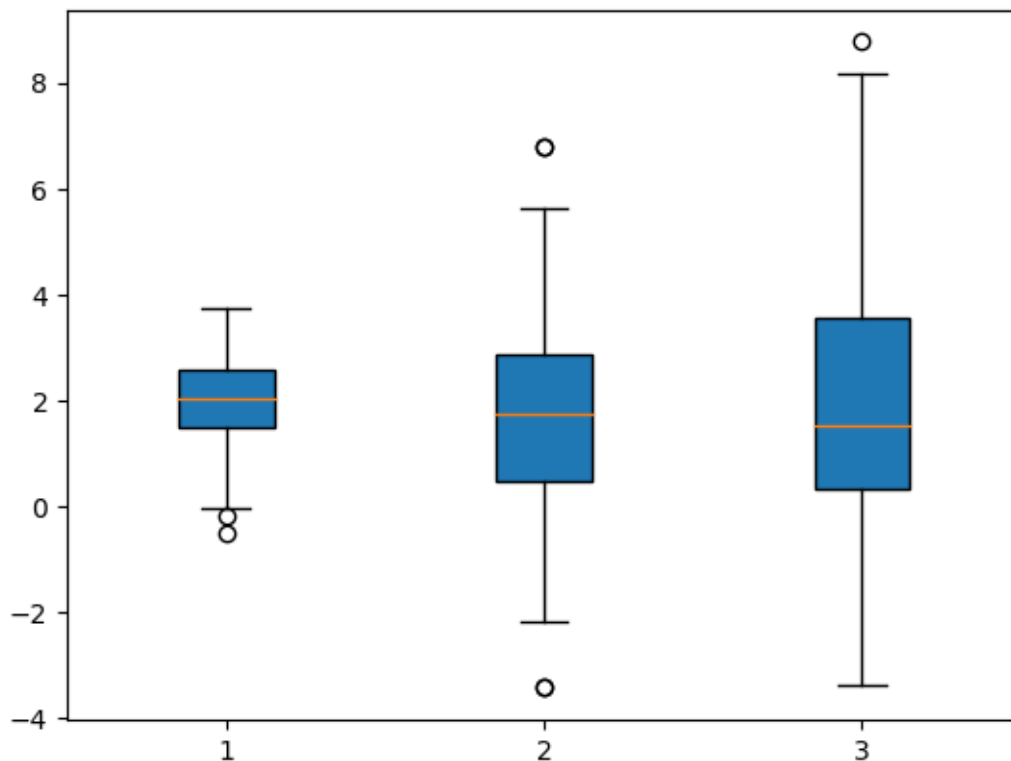
```
[ ]: fig, axes = plt.subplots (1, 3, figsize=(12, 4))
axes[0].plot(x, x**2, x, x**3)
axes[0].set_title("default axes ranges")
axes[1].plot(x, x**2, x, x**3)
axes [1].axis ('tight')
axes [1].set_title("tight axes")
axes [2].plot(x, x**2, x, x**3)
axes [2].set_ylim ([0, 60])
axes [2].set_xlim ([2, 5])
axes [2].set_title("custom axes range");
```



0.0.10 BoxPlot

It is used to identify outliers in datasets.

```
[ ]: data = [np.random.normal (2, std, 100) for std in range(1, 4)]  
# rectangular box plot Loading...  
plt.boxplot (data, vert=True, patch_artist=True);
```



```
[ ]: data
```

```
[ ]: [array([ 2.4688972 ,  2.64271637,  2.34678387,  1.10566766,  3.32272882,  
            1.86984859,  3.06096032,  0.32193277,  0.21504969,  0.87103553,  
            2.981296   ,  1.69209761,  2.81875693,  1.77718815,  0.98320841,  
            3.41183926, -0.05270584,  2.09801635,  2.10719721,  1.99350758,  
            2.0215349 ,  3.10751727,  1.30355415,  2.34576298,  2.78160898,  
            0.70780499,  0.9494209 ,  2.34444204,  1.72370695,  2.91401589,  
            1.72417277,  0.20864128,  2.10976972,  1.78418866,  3.07483991,  
            1.98786209,  3.17055953,  2.92967936,  1.22136738,  1.85330268,  
            2.06724093,  2.3466724 ,  1.91873379,  1.8700031 ,  1.58218483,  
            1.98829574,  1.7283403 ,  0.43623893,  3.27607871,  2.21988676,
```

```

2.10297543, 1.54114218, 1.53618113, 2.02093285, 1.37260643,
3.496911 , 2.27865172, 1.70907607, 0.87589769, 1.32478009,
2.51139605, 1.18417928, 3.39223308, 2.65644265, 2.19621242,
1.86144114, 3.2577997 , 3.33610515, 0.99771269, 2.49842148,
2.41268861, -0.18014344, 1.98656358, 3.5399378 , 1.01289121,
2.18524494, 1.62694601, 3.73034614, 2.16647355, 1.52086316,
1.43745832, 3.15490436, 3.52309514, 2.79554632, 2.11227894,
2.25081598, 2.07721443, 0.46117829, 2.42005466, 2.41127871,
-0.51821898, 1.38624186, 2.72619709, 0.78993363, 1.95162412,
1.03307345, 1.53163182, 3.16499449, 2.39270485, 2.53013662]),
array([ 0.3903033 , 2.57446922, -0.49738419, 0.36263862, 2.65157955,
0.38156308, 1.45855099, 2.52485026, 5.16763571, 4.13752903,
4.44639647, 1.90184528, 2.1880741 , -1.30796251, 1.62969854,
2.66774743, 3.84562765, 5.61860669, 3.30868871, 3.21273318,
3.0233193 , 0.47800352, 2.20432848, 3.03982891, -0.67765269,
4.4929251 , 0.48181024, 2.16055526, -0.29587076, -3.43623411,
2.1295323 , 6.78972044, 0.19292378, 1.08244564, -1.56718293,
3.09124961, -0.10063548, 2.83925724, 1.73604408, 2.44467894,
1.93217398, -3.42046713, 0.60743418, -0.59090558, 0.56373224,
3.06710801, -1.0524088 , 1.61452343, 0.18683303, 1.58440998,
2.39934958, -1.79064361, 3.55731521, 3.51592536, 0.97618489,
6.77194505, 4.05735548, 2.62835014, -0.85910632, 3.10522358,
1.98430912, -0.36183708, 1.46365888, -0.24131236, 3.40073837,
4.65325039, -0.51026395, 1.90751362, 1.59928076, 2.24947054,
4.01028213, -0.45713994, 1.30220561, 1.99024811, 2.07455082,
4.19312585, 2.14941079, 2.93442901, 0.3534931 , 1.50720067,
0.60494435, 1.16801606, 1.43103011, 0.73801513, 1.11714791,
2.41164559, 2.02115201, 0.68118476, 1.54965405, 1.5395391 ,
1.74063266, 0.88378262, 2.12513839, 4.24831244, -1.53538089,
5.38412056, -2.1940829 , 2.29292947, 0.81320489, 0.31557001]),
array([ 1.37346791, 6.59228164, 5.78322515, 0.56523977, 6.70288588,
1.45372051, 2.93922702, -3.14656785, -1.06934628, 1.26991876,
1.07831895, 0.54950445, 3.34101744, -1.56447336, -2.12803791,
-0.73298892, -2.16075668, 0.72937021, -2.17595483, -0.61470768,
1.00187384, -1.31710086, 1.81302568, 3.55477123, 1.81289395,
4.83145866, -1.63515902, 3.51861767, 0.70842632, 3.7472162 ,
0.48409825, -0.09202564, -0.15809677, 5.22805158, 2.93351515,
4.69482362, 0.4530187 , 1.52790933, 3.26419127, -2.21141913,
2.10935356, 3.49181341, 5.97651988, 0.62605489, 3.62024715,
1.52687658, 5.04300773, -2.11921963, 5.00080758, 1.31912287,
-1.37163413, 0.75922431, 2.41978878, -1.89674352, 1.87390929,
0.8725434 , 3.02009089, 7.2667198 , -2.32097494, 0.17677746,
3.2591528 , 0.54224756, 5.266219 , 0.37398149, 2.29668395,
3.25325214, 0.75676208, 8.76984572, 1.53030707, 3.79646513,
-0.61646727, 3.67281316, 5.70297729, 3.16435076, -0.08169143,
4.41583166, 0.83176755, 2.64198772, 1.64800832, 0.21991246,
8.15521493, 3.20378753, 1.53788238, -1.63558466, 7.32126004,

```

```
3.71336005, 3.67721528, -0.58007504, 1.13882477, 2.66847734,
1.05375206, -3.38075189, 3.57264073, 4.48456545, 1.76270506,
-1.01134714, 7.0421811 , 3.2879894 , 1.00483333, 2.41362958]]]
```

1 Matplotlib Exercises

**** * NOTE: ALL THE COMMANDS FOR PLOTTING A FIGURE SHOULD ALL GO IN THE SAME CELL. SEPARATING THEM OUT INTO MULTIPLE CELLS MAY CAUSE NOTHING TO SHOW UP. * ****

2 Exercise 0

1. Take a dataset of your choice and plot graphs using matplotlib(convert numeric data to nominal where ever required)

Campus Recruitment Dataset: * This data set consists of Placement data of students in a XYZ campus. It includes secondary and higher secondary school percentage and specialization. It also includes degree specialization, type and Work experience and salary offers to the placed students. Link: <https://www.kaggle.com/datasets/benroshan/factors-affecting-campus-placement>

```
[ ]: import pandas as pd
import io
import matplotlib.pyplot as plt
%matplotlib inline
```

```
[ ]: from google.colab import files
uploaded = files.upload()
```

<IPython.core.display.HTML object>

Saving Placement_Data_Full_Class.csv to Placement_Data_Full_Class.csv

```
[ ]: data = pd.read_csv(io.BytesIO(uploaded['Placement_Data_Full_Class.csv']))
data.head()
```

```
[ ]: 
```

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	\
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	
1	2	M	79.33	Central	78.33	Others	Science	77.48	
2	3	M	65.00	Central	68.00	Central	Arts	64.00	
3	4	M	56.00	Central	52.00	Central	Science	52.00	
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	

	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0

3	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0

How much dependency between MBA percentage and Salary with specialisation

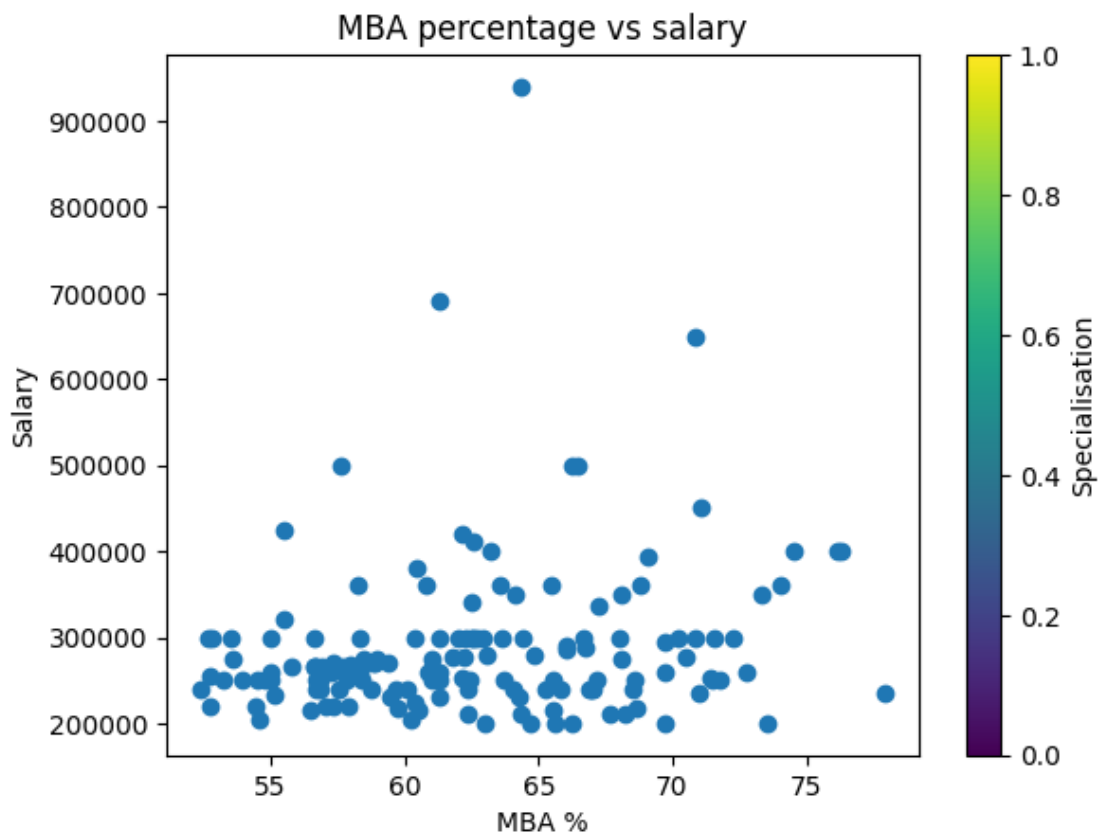
```
[ ]: import matplotlib.pyplot as plt

# Create scatter plot
plt.scatter(data['mba_p'], data['salary'])

# Set title and axis labels
plt.title('MBA percentage vs salary')
plt.xlabel('MBA %')
plt.ylabel('Salary')

# Add colorbar
colorbar = plt.colorbar()
colorbar.set_label('Specialisation')

# Display the plot
plt.show()
```



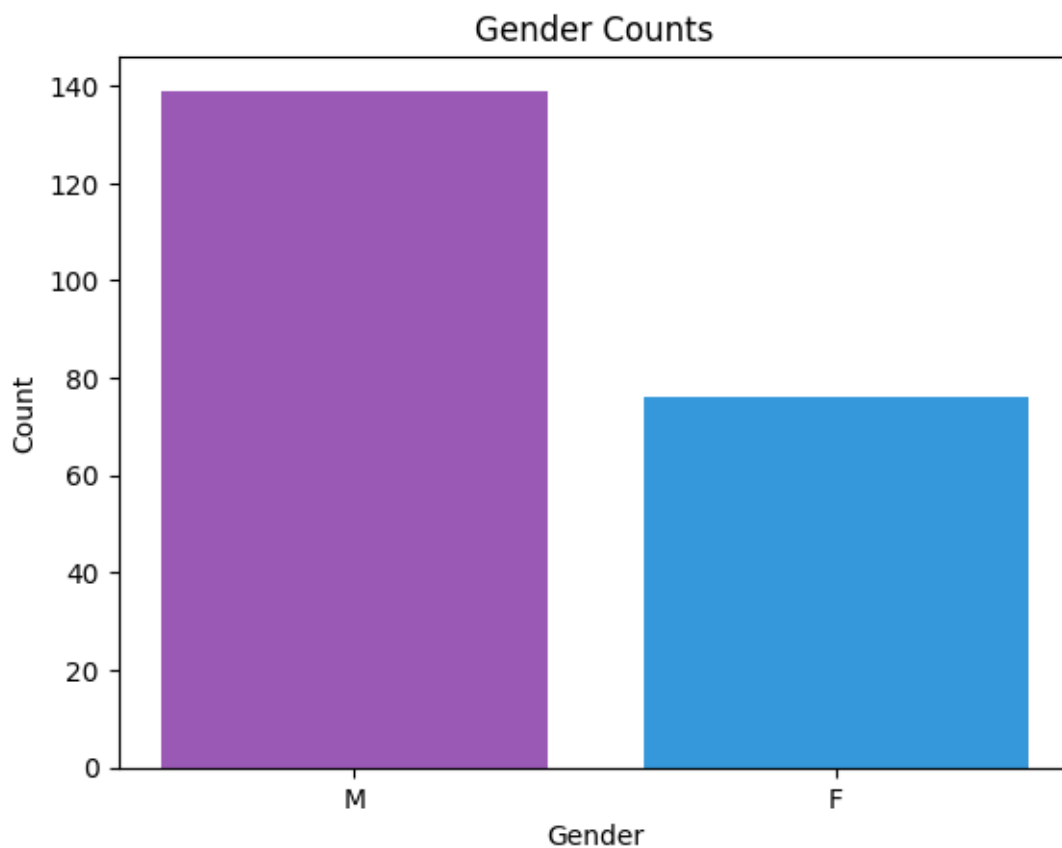
```
[ ]: fig, ax = plt.subplots()

colors = ['#9b59b6', '#3498db'] # specify colors for the bars

# create the bar plot
ax.bar(data['gender'].unique(), data['gender'].value_counts(), color=colors)

# set the title and axis labels
ax.set_title('Gender Counts')
ax.set_xlabel('Gender')
ax.set_ylabel('Count')

plt.show()
```

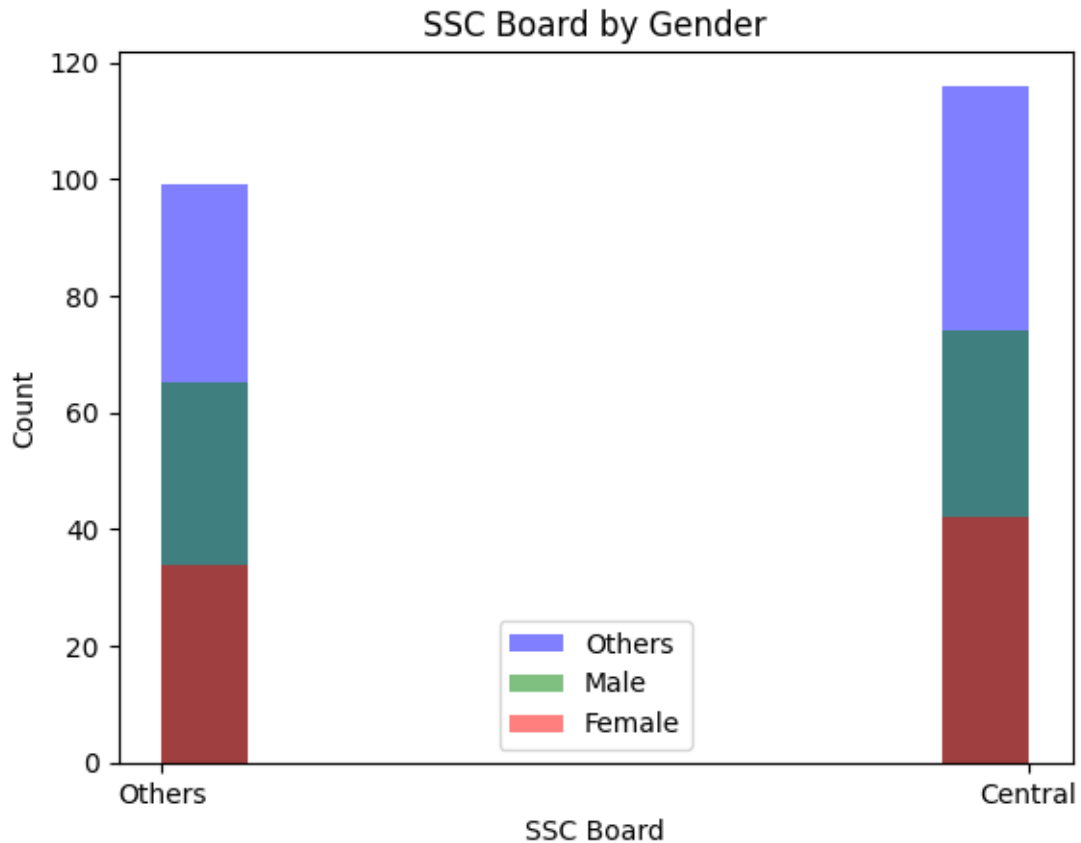


There are more males than females in this dataset. This can cause an imbalance.

```
[ ]: plt.hist(data['ssc_b'], color='b', alpha=0.5, label='Others')
plt.hist(data[data['gender'] == 'M']['ssc_b'], color='g', alpha=0.5, label='Male')
```

```
plt.hist(data[data['gender'] == 'F']['ssc_b'], color='r', alpha=0.5,
        label='Female')

plt.xlabel('SSC Board')
plt.ylabel('Count')
plt.title('SSC Board by Gender')
plt.legend()
plt.show()
```



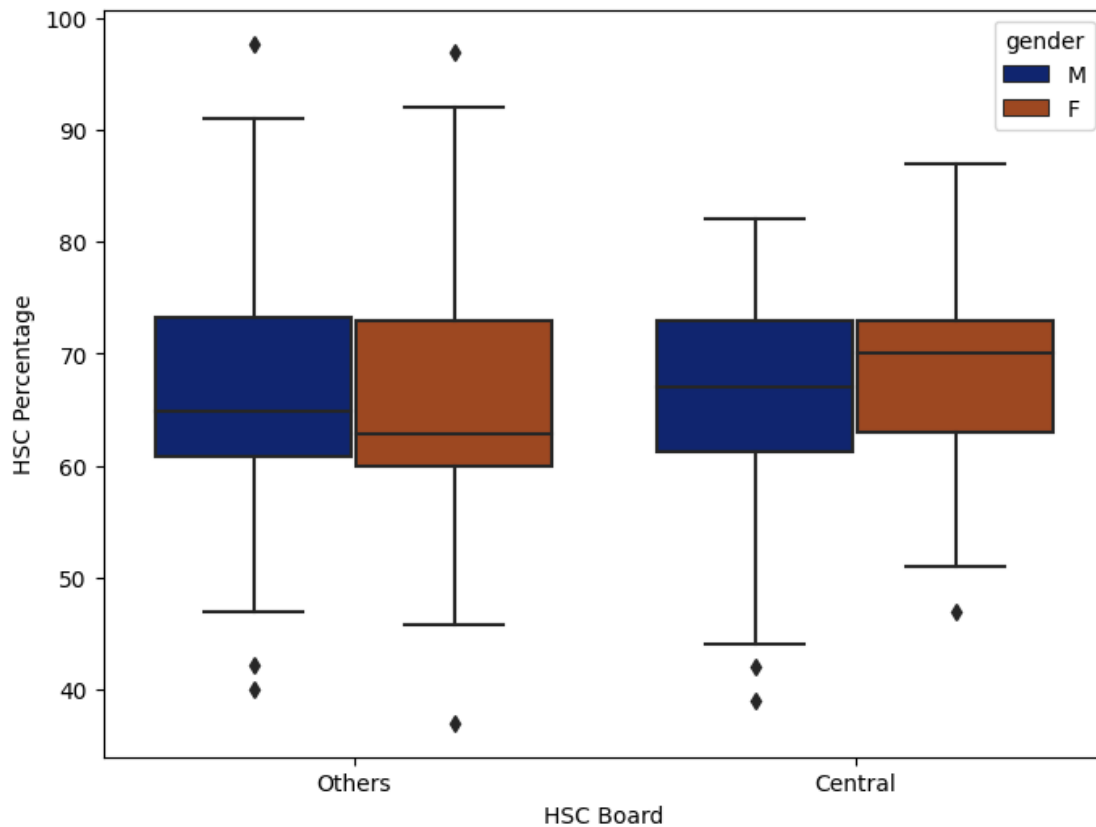
Females have more population in Central-based secondary school compared to Other secondary schools. The same trend is for male. Overall, the male population in secondary schools are more than females, irrespective of the type of school.

```
[ ]: import seaborn as sns
fig, ax = plt.subplots(figsize=(8,6))

sns.boxplot(x="hsc_b", y="hsc_p", hue="gender", palette="dark", data=data,
            ax=ax)

ax.set_xlabel("HSC Board")
```

```
ax.set_ylabel("HSC Percentage")
plt.show()
```



The boxplot clearly shows presence of outliers. High school percentage of male is better in Others compared to females. However, in Central, females score more than the male. If we compare the school types, both male and female students perform better in Central.

```
[ ]: import matplotlib.pyplot as plt

fig, ax = plt.subplots()
df = data.groupby(['degree_t', 'gender'], as_index=False)['degree_p'].mean()
width = 0.35
x = np.arange(len(df['degree_t'].unique()))

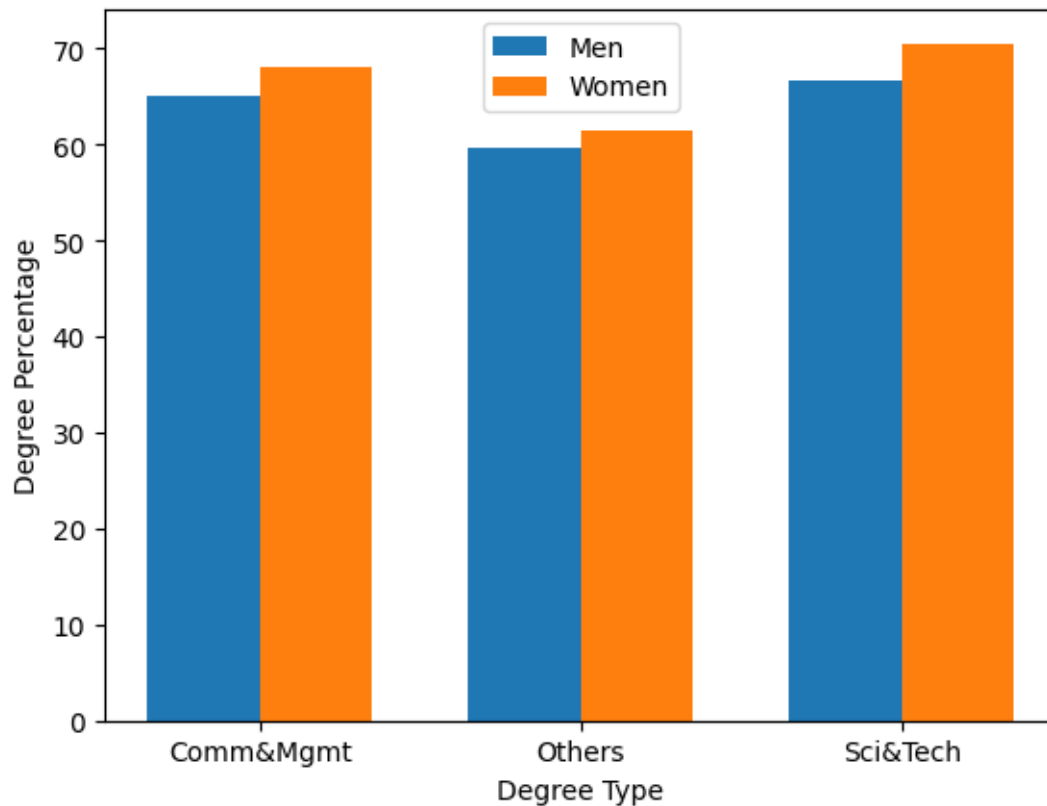
male_means = df[df['gender']=='M']['degree_p']
female_means = df[df['gender']=='F']['degree_p']

rects1 = ax.bar(x - width/2, male_means, width, label='Men')
rects2 = ax.bar(x + width/2, female_means, width, label='Women')
```

```

ax.set_ylabel('Degree Percentage')
ax.set_xlabel('Degree Type')
ax.set_xticks(x)
ax.set_xticklabels(df['degree_t'].unique())
ax.legend()
plt.show()

```

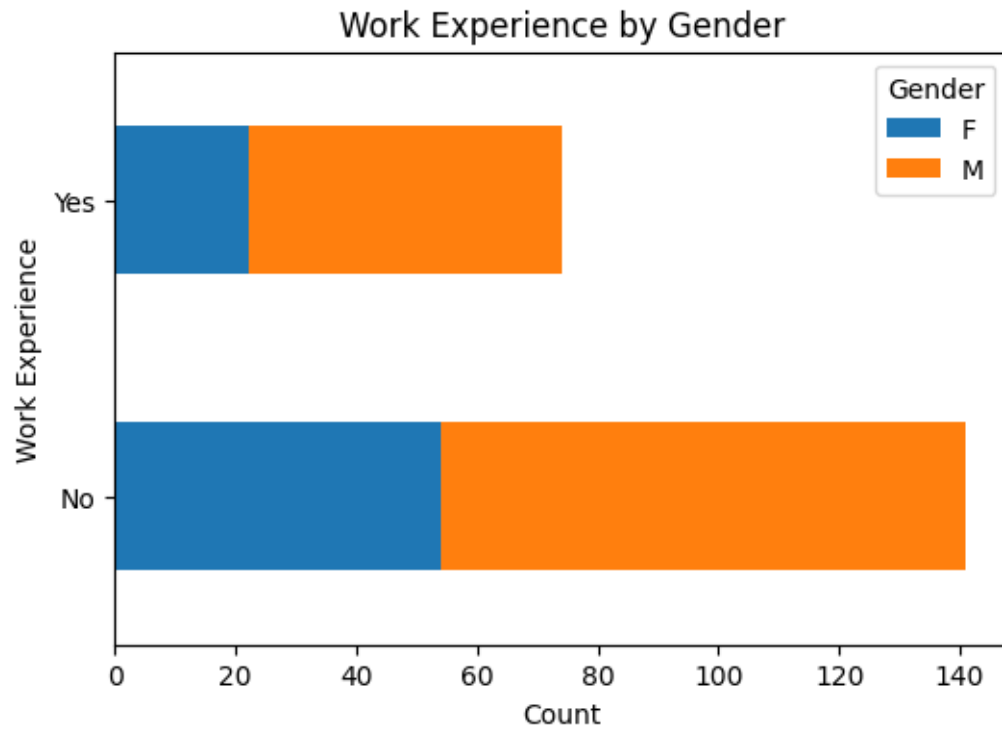


There is more population of female students in degree-level education compared to male students. More students tend to choose Sci&Tech compared to the Comm&Mgmt stream. The population of female in individual streams of the degrees is higher.

```

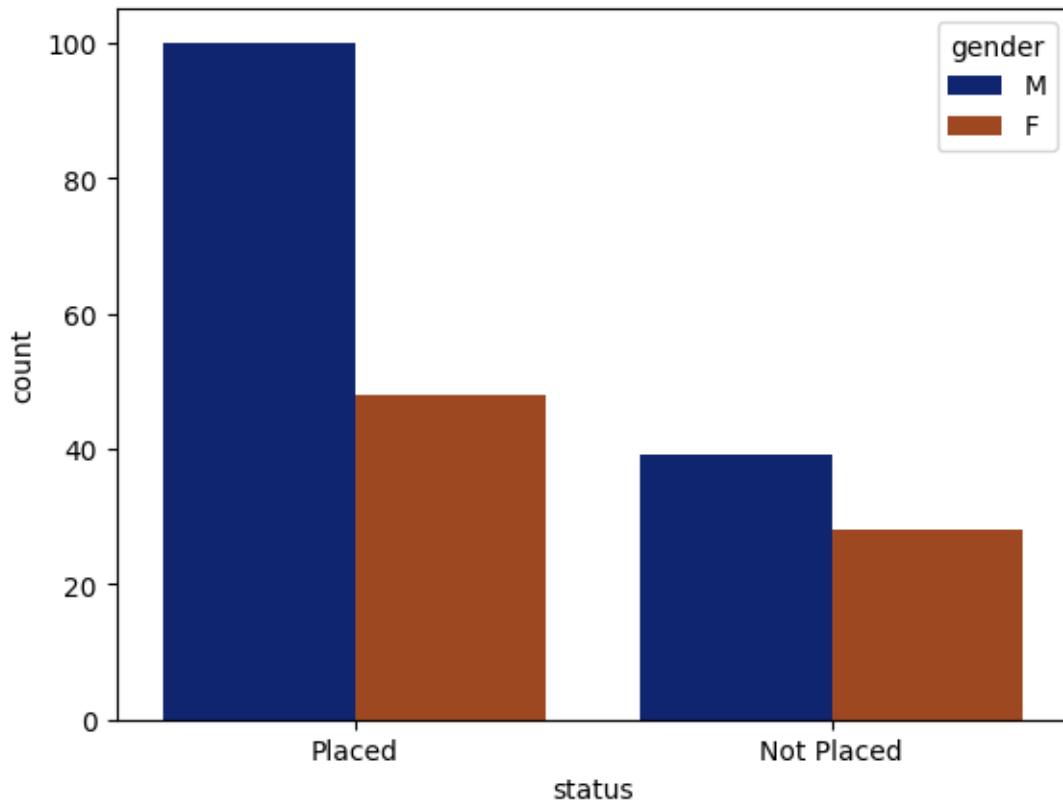
[ ]: fig, ax = plt.subplots(figsize=(6, 4))
data.groupby(['workex', 'gender'])['workex'].count().unstack().
    .plot(kind='barh', stacked=True, ax=ax)
ax.set_xlabel('Count')
ax.set_ylabel('Work Experience')
ax.set_title('Work Experience by Gender')
ax.legend(title='Gender')
plt.show()

```

Overall, most of the students don't tend to have work experience. More population of male have no experience compared to female.

```
[ ]: sns.countplot(x="status", hue="gender", palette="dark", data=data);
```



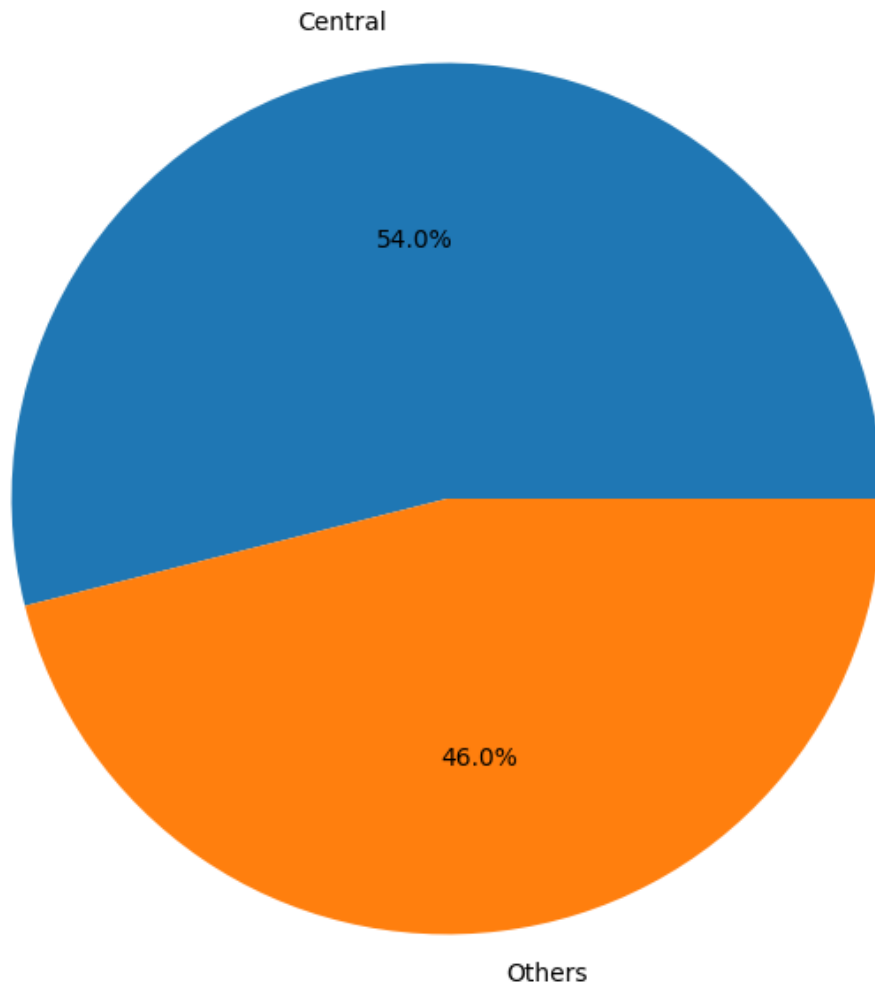
More male students are placed compared to female students. The no. of female students placed is half the amount of the no. of male students placed. In Not Placed, this proportion of difference between male and female Not Placed students is not as high as in the Placed scenario.

```
[ ]: labels = data['ssc_b'].value_counts().index
     sizes = data['ssc_b'].value_counts().values

     plt.figure(figsize = (8,8))
     plt.pie(sizes, labels=labels, autopct='%1.1f%%')
     plt.title("Distribution of Samples by 'ssc_b'",color = 'black',fontsize = 15)
```

```
[ ]: Text(0.5, 1.0, "Distribution of Samples by 'ssc_b'")
```

Distribution of Samples by 'ssc_b'



2.0.1 Follow the instructions to recreate the plots using this data:

2.0.2 Data

```
[ ]: import numpy as np
x = np.arange(0,150)
y = x*2
z = x**3
```

****** Import matplotlib.pyplot as plt and set %matplotlib inline if you are using the jupyter notebook. What command do you use if you aren't using the jupyter notebook?******

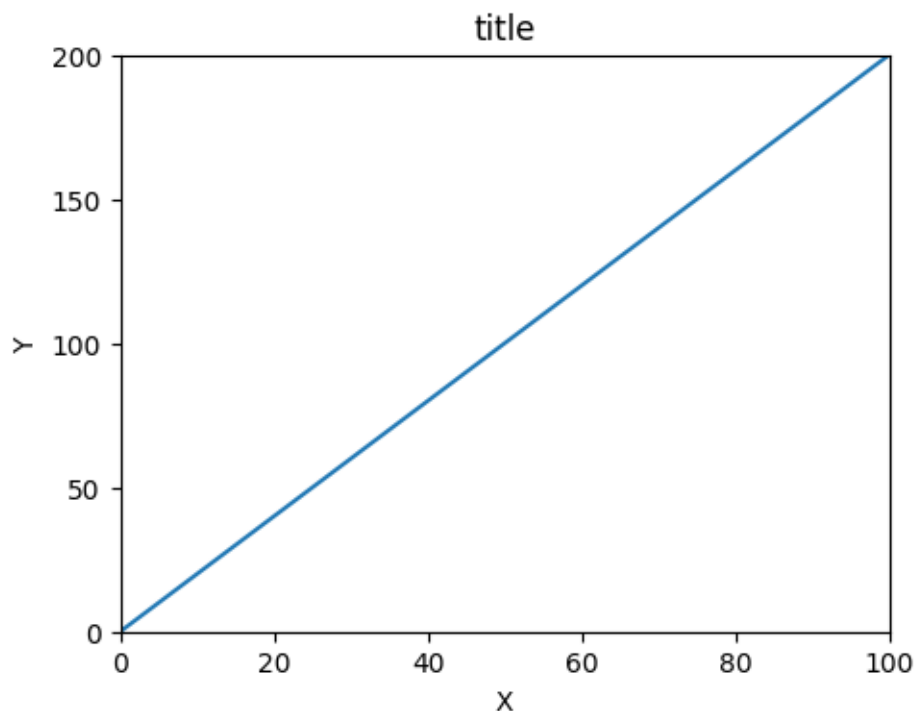
```
[ ]: import matplotlib.pyplot as plt
      %matplotlib inline
```

If you are not using Jupyter Notebook and want to display matplotlib plots, you can use the `plt.show()` command after creating the plot. This command will display the plot in a new window.

2.1 Exercise 1

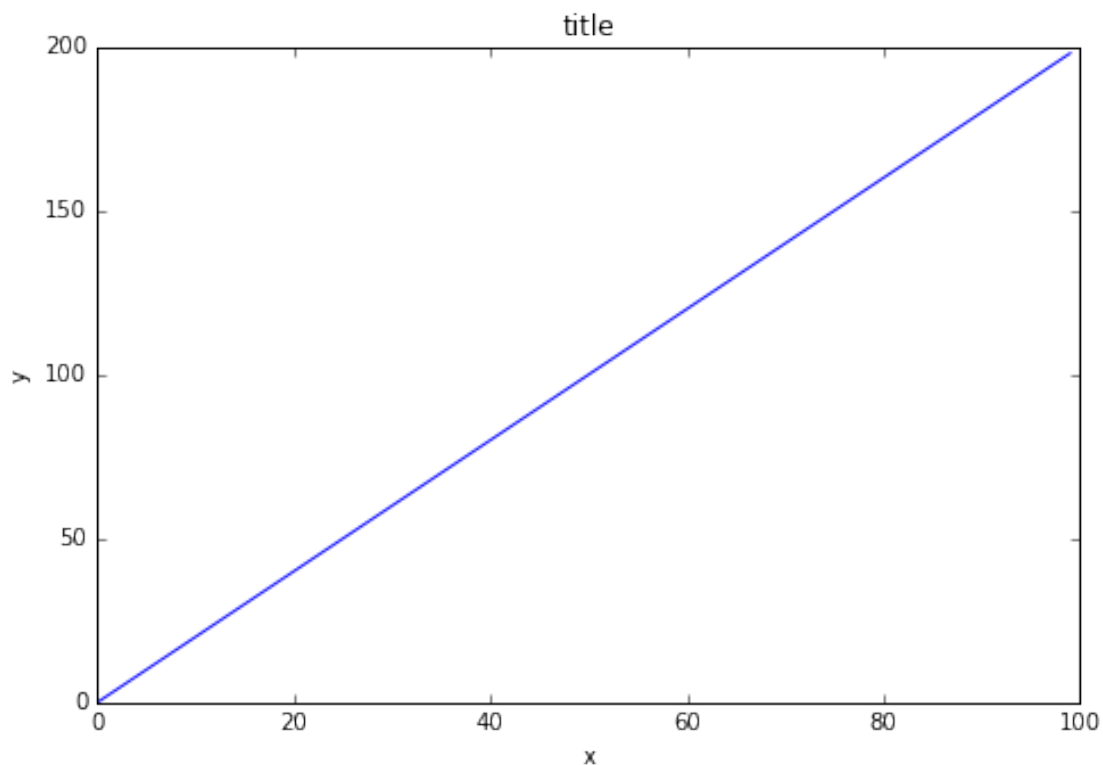
** Follow along with these steps: * Create a figure object called `fig` using `plt.figure()` * Use `add_axes` to add an axis to the figure canvas at `[0,0,2,2]`. Call this new axis `ax`. * Plot `(x,y)` on that axes and set the labels and titles to match the plot below:**

```
[ ]: fig = plt.figure(figsize=(2, 1.5))
      ax = fig.add_axes([0,0,2,2])
      ax.plot(x, y)
      ax.set_xlabel('X')
      ax.set_ylabel('Y')
      ax.set_title('title')
      ax.set_xlim([0, 100])
      ax.set_ylim([0, 200])
      ax.set_yticks([0, 50, 100, 150, 200])
      plt.show()
```



```
[ ]:
```

```
[ ]: <matplotlib.text.Text at 0x111534c50>
```

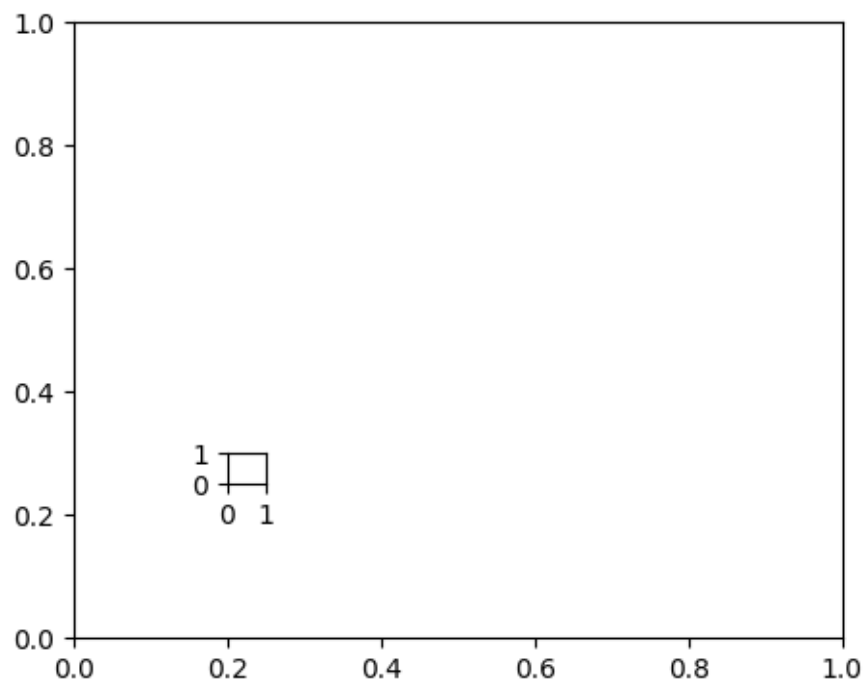


2.2 Exercise 2

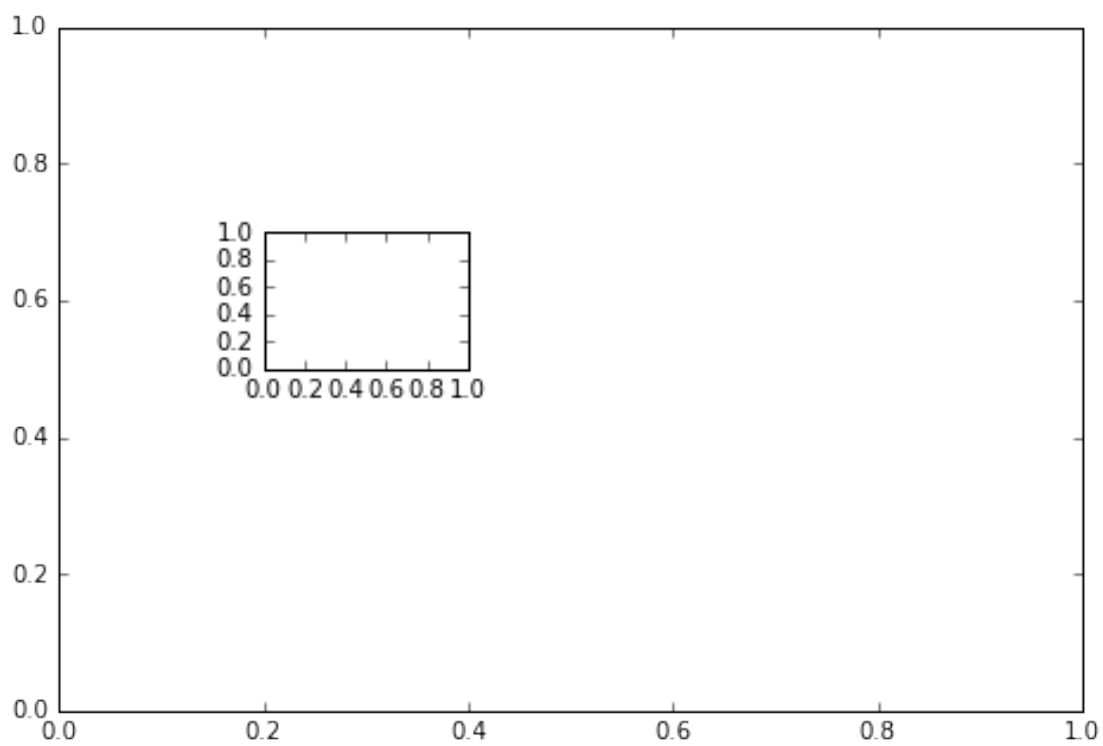
**** Create a figure object and put two axes on it, ax1 and ax2. Located at [0,0,2,2] and [0.4,0.5,.1,.1] respectively.****

```
[ ]: import matplotlib.pyplot as plt

fig = plt.figure(figsize=(2,1.6))
ax1 = fig.add_axes([0,0,2,2])
ax2 = fig.add_axes([0.4,0.5,0.1,0.1])
plt.show()
```



[]:



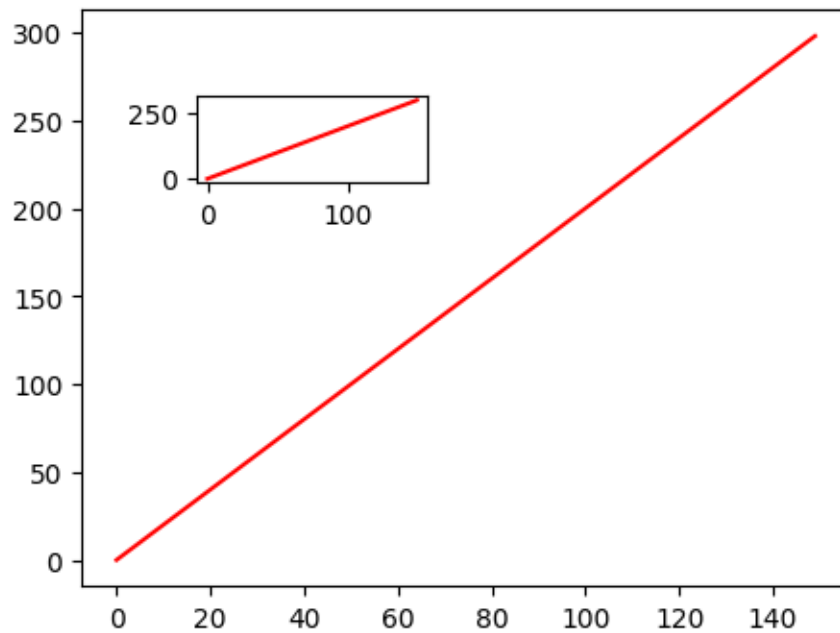
**** Now plot (x,y) on both axes. And call your figure object to show it.****

```
[ ]: # Create blank canvas
fig = plt.figure(figsize=(2, 1.5))

# Main Plot
ax1 = fig.add_axes([0, 0, 2, 2]) #[left, bottom, width, height]
ax1.plot(x, y, color='red')

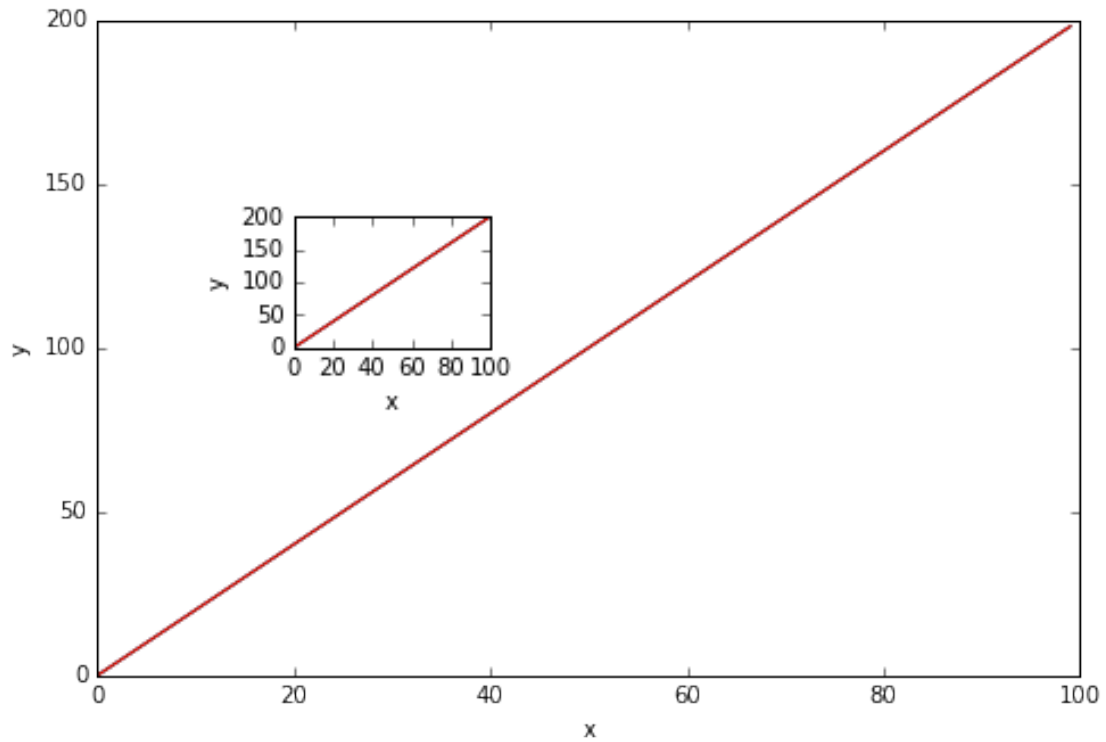
# Inset Plot
ax2 = fig.add_axes([0.3, 1.4, 0.6, 0.3]) #[left, bottom, width, height]
ax2.plot(x, y, color='red')
```

```
[ ]: [
```



```
[ ]:
```

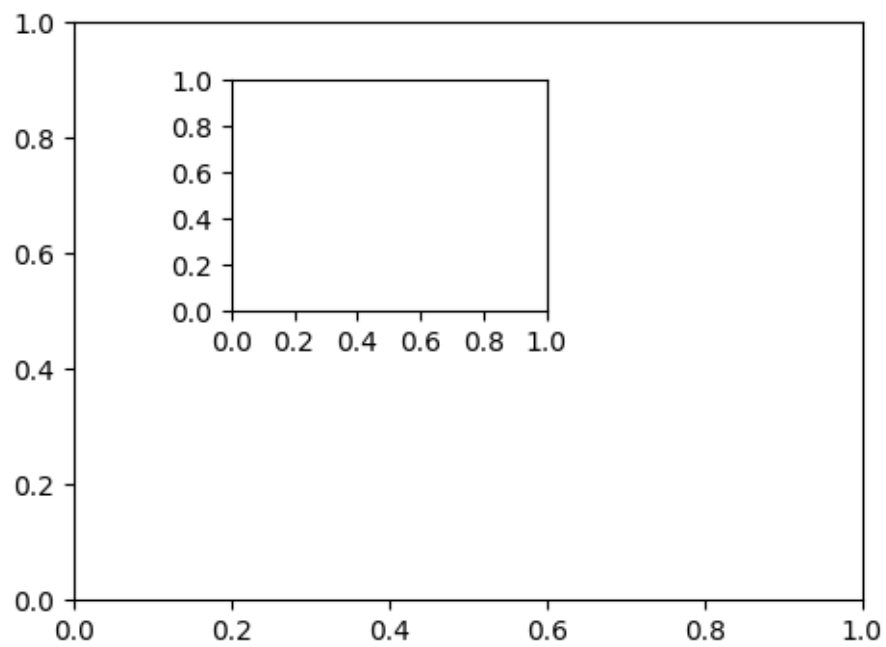
```
[ ]:
```



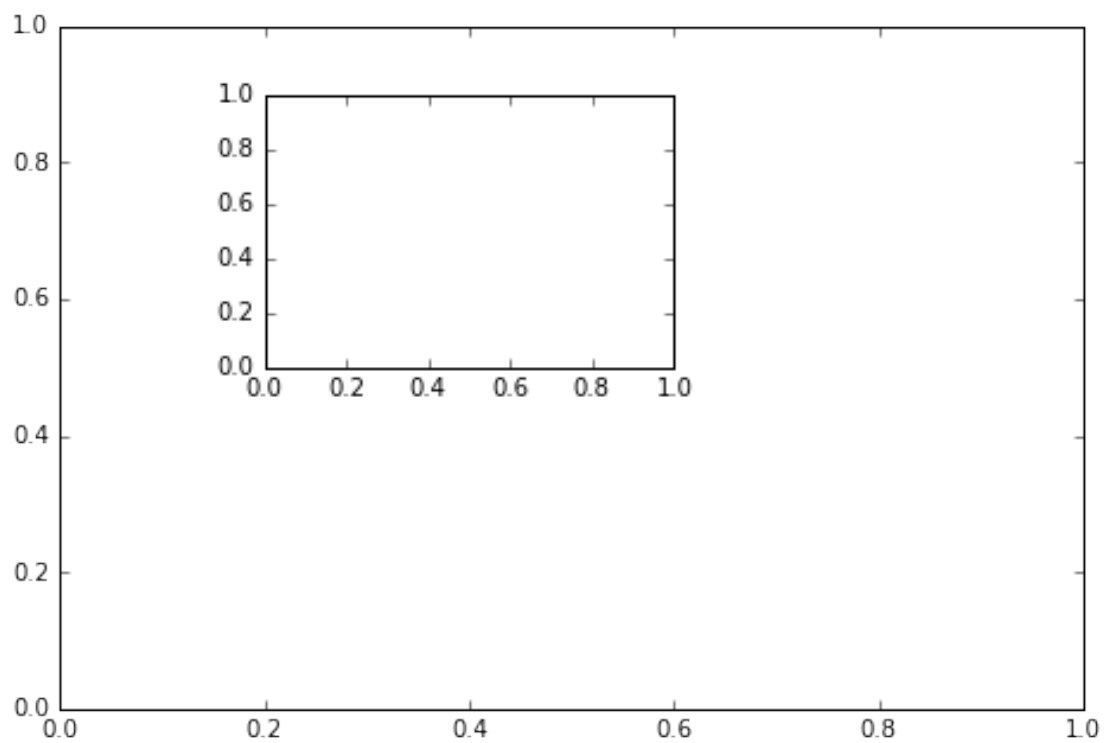
2.3 Exercise 3

** Create the plot below by adding two axes to a figure object at $[0,0,1,1]$ and $[0.2,0.5,.4,.4]$ **

```
[ ]: fig = plt.figure(figsize=(4.1, 3))
ax1 = fig.add_axes([0, 0, 1, 1])
ax1.set_xticks([0.0, 0.2, 0.4, 0.6, 0.8,1.0])
ax2 = fig.add_axes([0.2, 0.5, .4, .4])
ax2.set_xticks([0.0, 0.2, 0.4, 0.6, 0.8,1.0])
ax2.set_yticks([0.0, 0.2, 0.4, 0.6, 0.8,1.0])
plt.show()
```

[]:



**** Now use x,y, and z arrays to recreate the plot below. Notice the xlimits and y limits on the inserted plot:****

```
[ ]: import matplotlib.pyplot as plt
import numpy as np

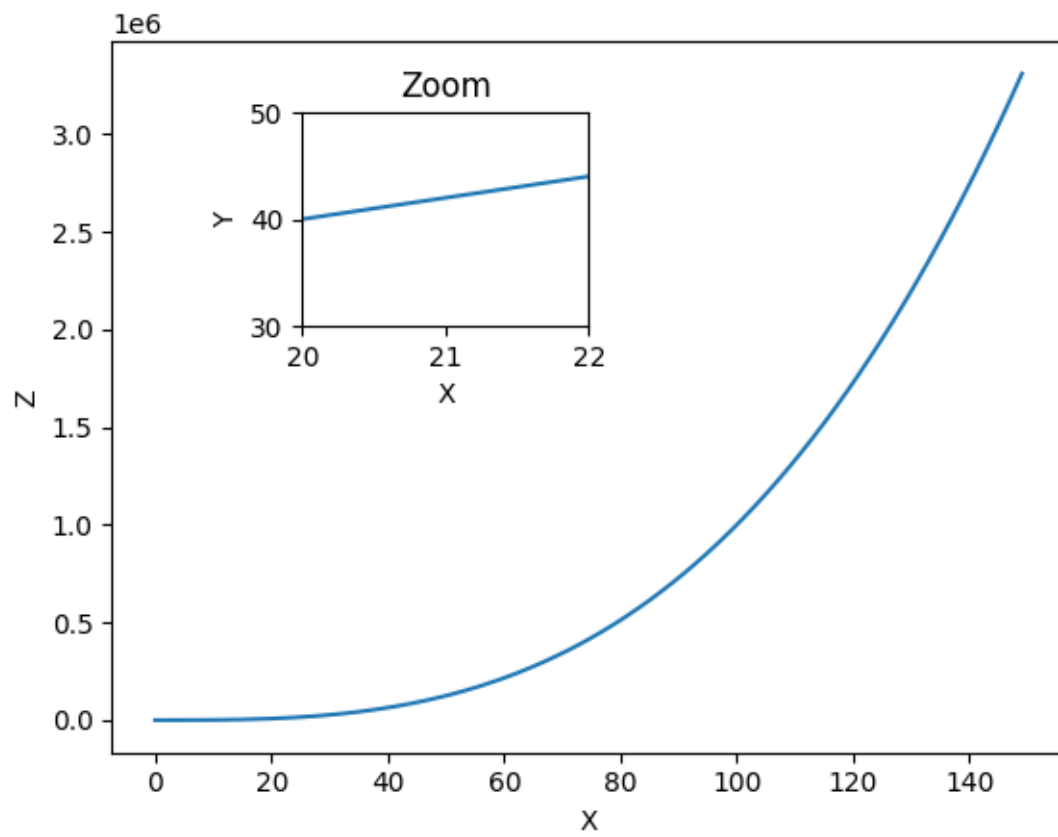
# Create figure and axis objects
fig, ax = plt.subplots()
ax2 = ax.inset_axes([0.2, 0.6, 0.3, 0.3])

# Define data
x = np.arange(0,150)
y = x*2
z = x**3

# Plot data on main axis
ax.plot(x, z)
ax.set_xlabel('X')
ax.set_ylabel('Z')

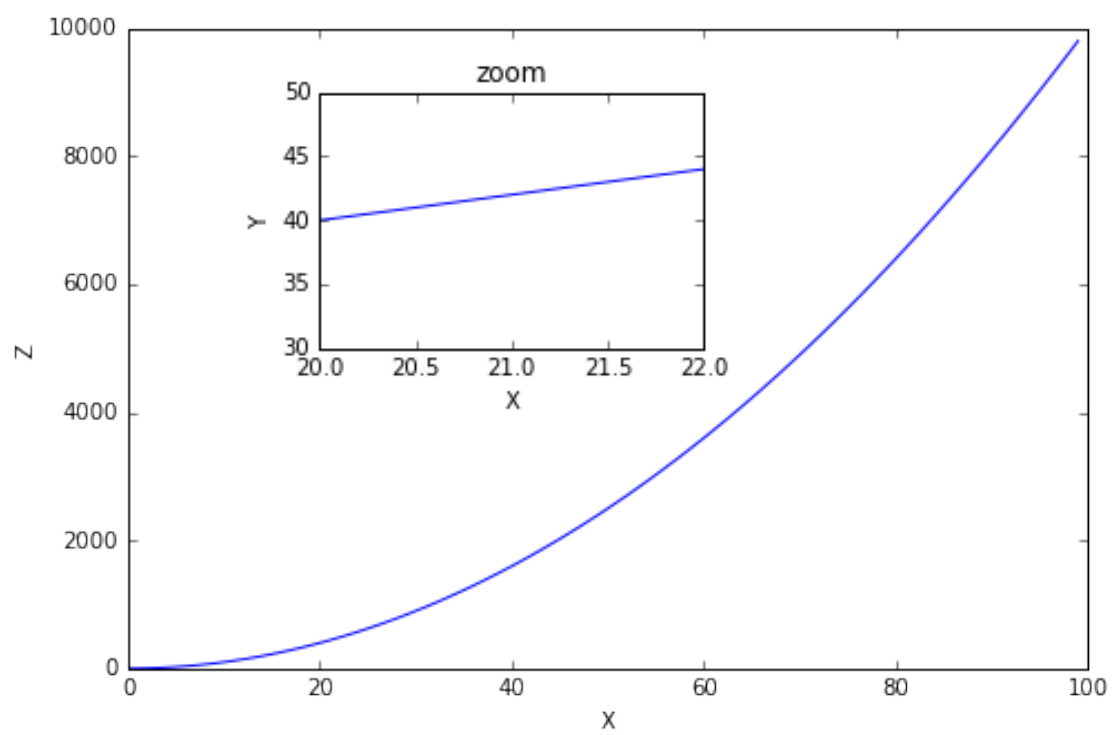
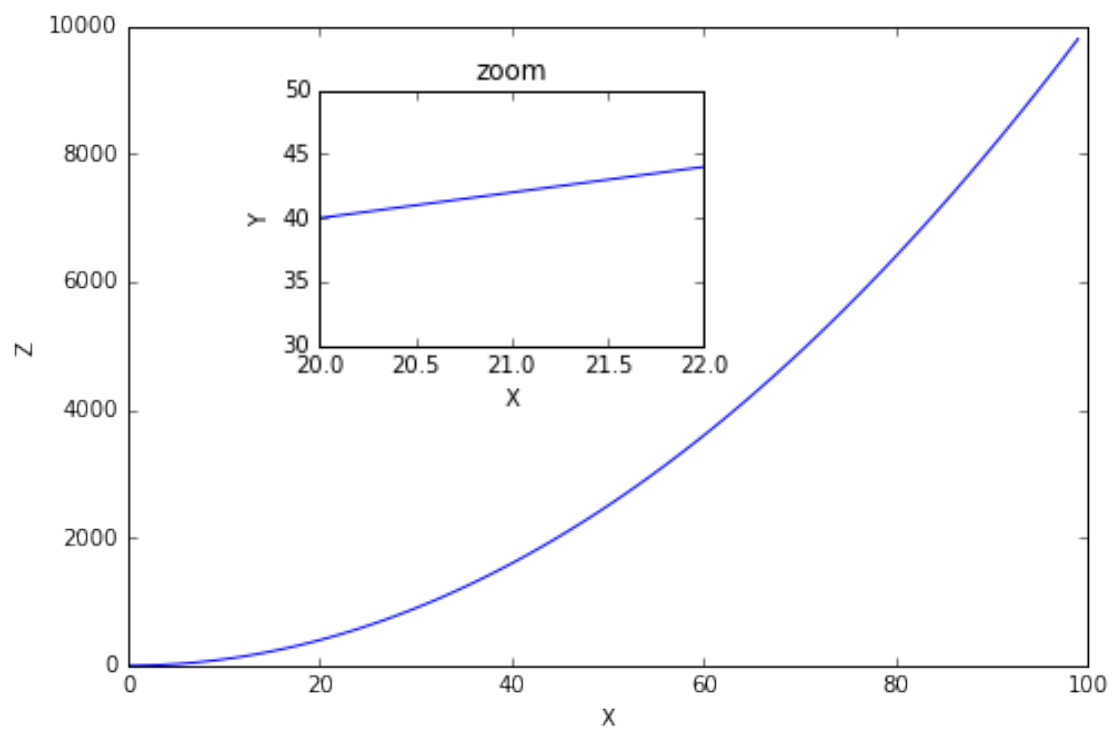
# Plot data on inset axis
ax2.plot(x, y)
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_title('Zoom')
ax2.set_xlim(20,22)
ax2.set_ylim(30,50)

# Show the plot
plt.show()
```



[]:

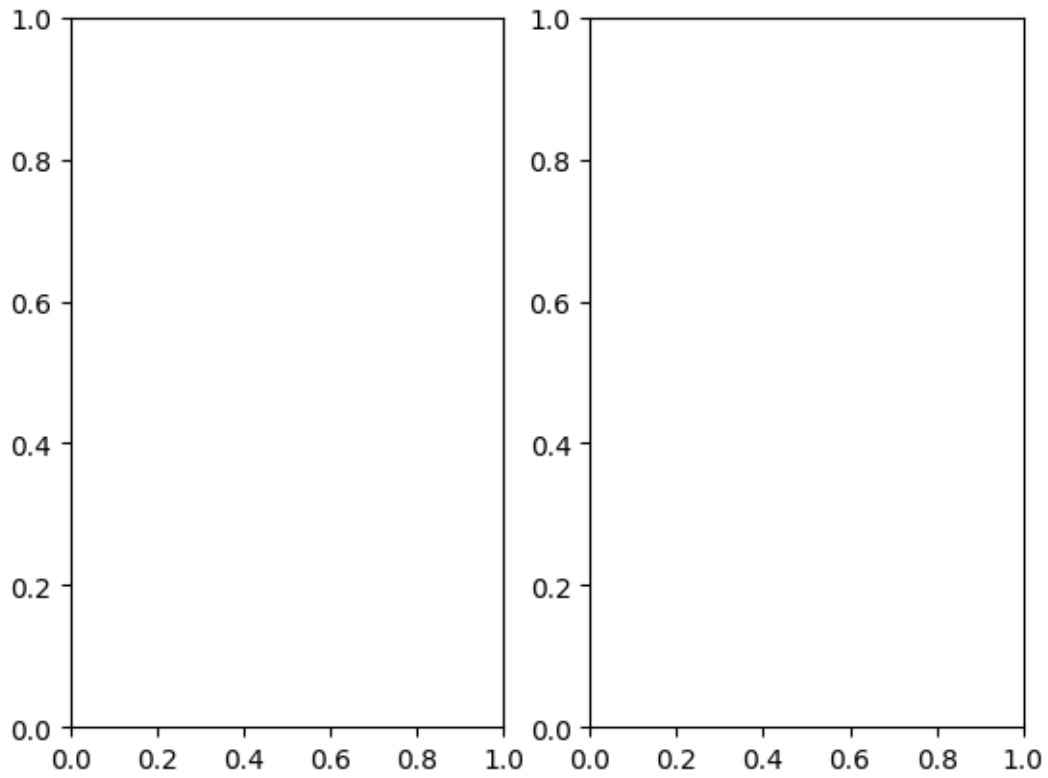
[]:



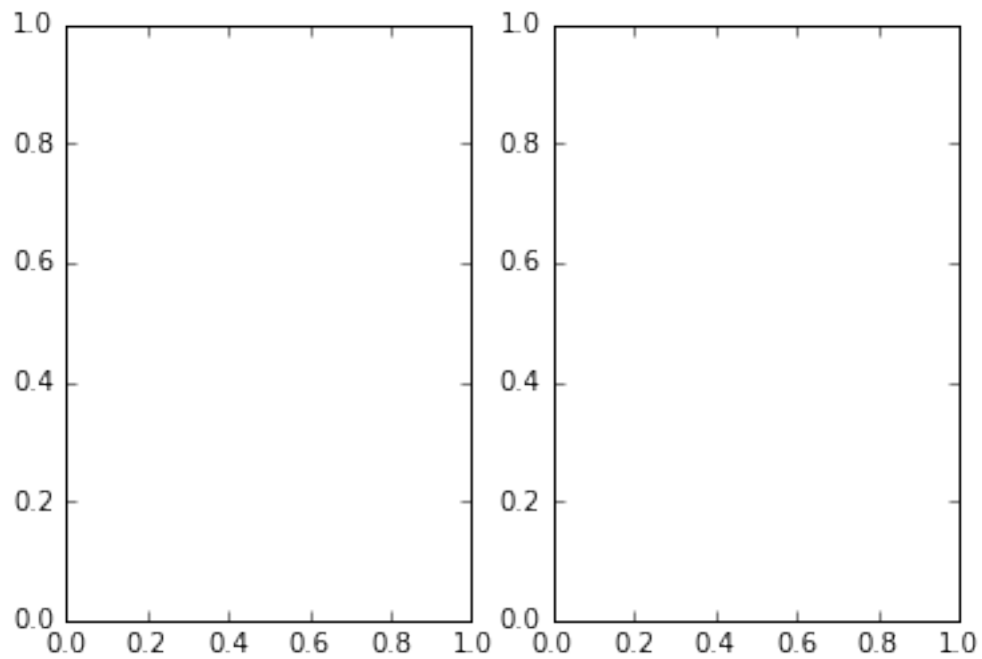
2.4 Exercise 4

**** Use `plt.subplots(nrows=1, ncols=2)` to create the plot below.****

```
[ ]: fig, axes = plt.subplots(nrows=1, ncols=2)
```



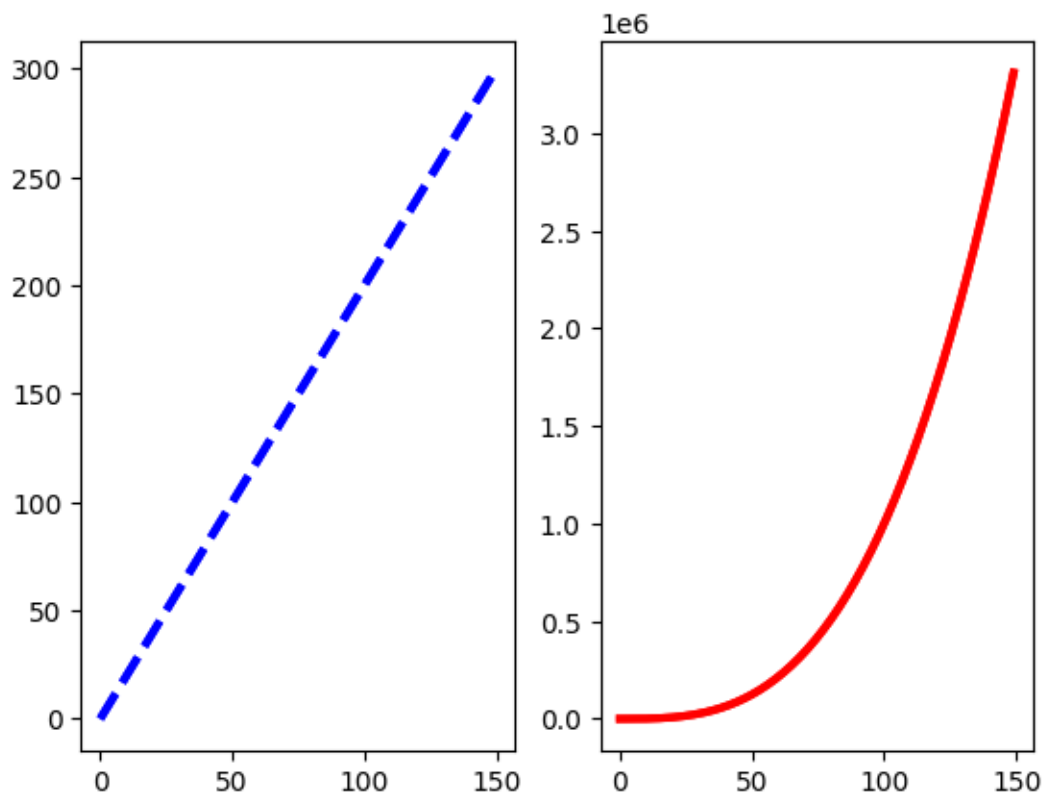
```
[ ]:
```



**** Now plot (x,y) and (x,z) on the axes. Play around with the linewidth and style****

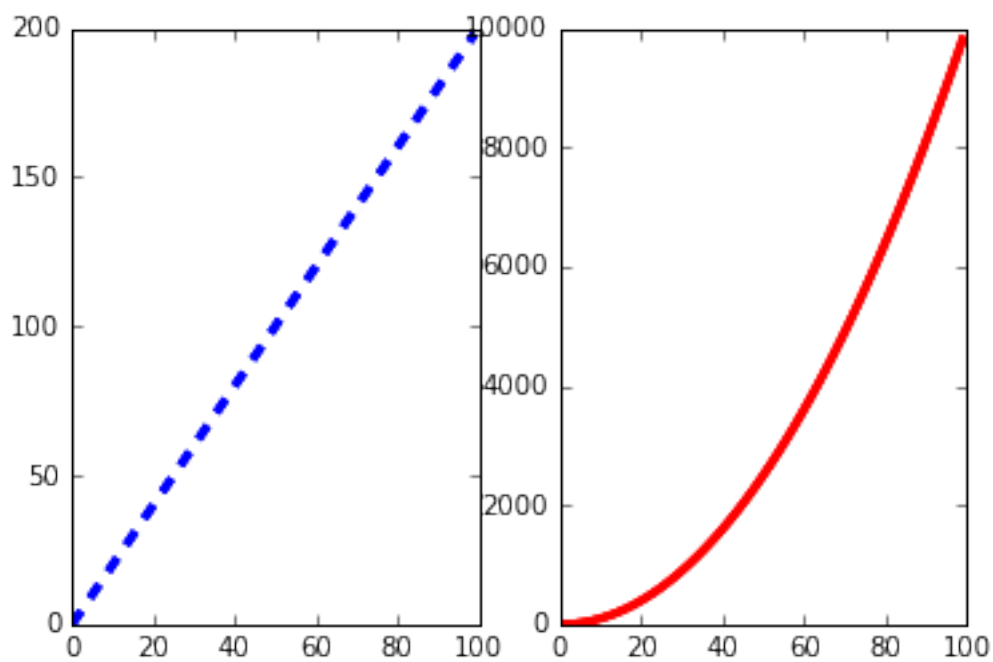
```
[ ]: axes[0].plot(x,y,color="blue",lw=3,ls='--')  
axes[1].plot(x,z,color="red",lw=3,ls='-')  
fig
```

```
[ ]:
```



[]:

[]:

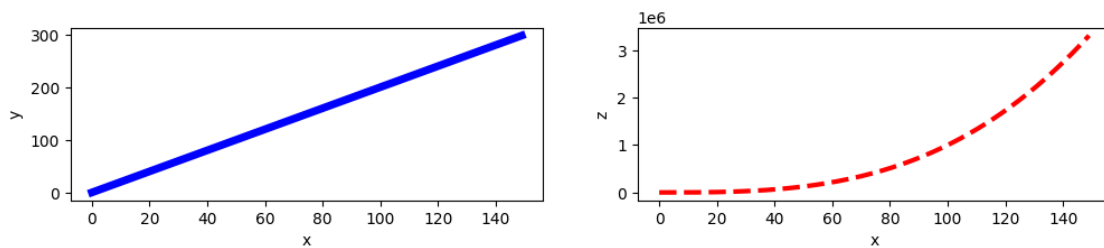


**** See if you can resize the plot by adding the `figsize()` argument in `plt.subplots()` are copying and pasting your previous code.****

```
[ ]: fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 2))
axes[0].plot(x, y, color="blue", lw=5)
axes[0].set_xlabel('x')
axes[0].set_ylabel('y')

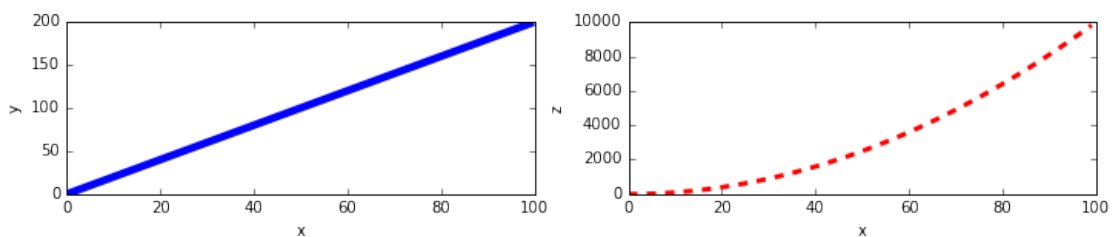
axes[1].plot(x, z, color="red", lw=3, ls='--')
axes[1].set_xlabel('x')
axes[1].set_ylabel('z')
```

```
[ ]: Text(0, 0.5, 'z')
```



```
[ ]:
```

```
[ ]: <matplotlib.text.Text at 0x1141b4ba8>
```



Conclusions: We have successfully implemented matplotlib library in this practical.