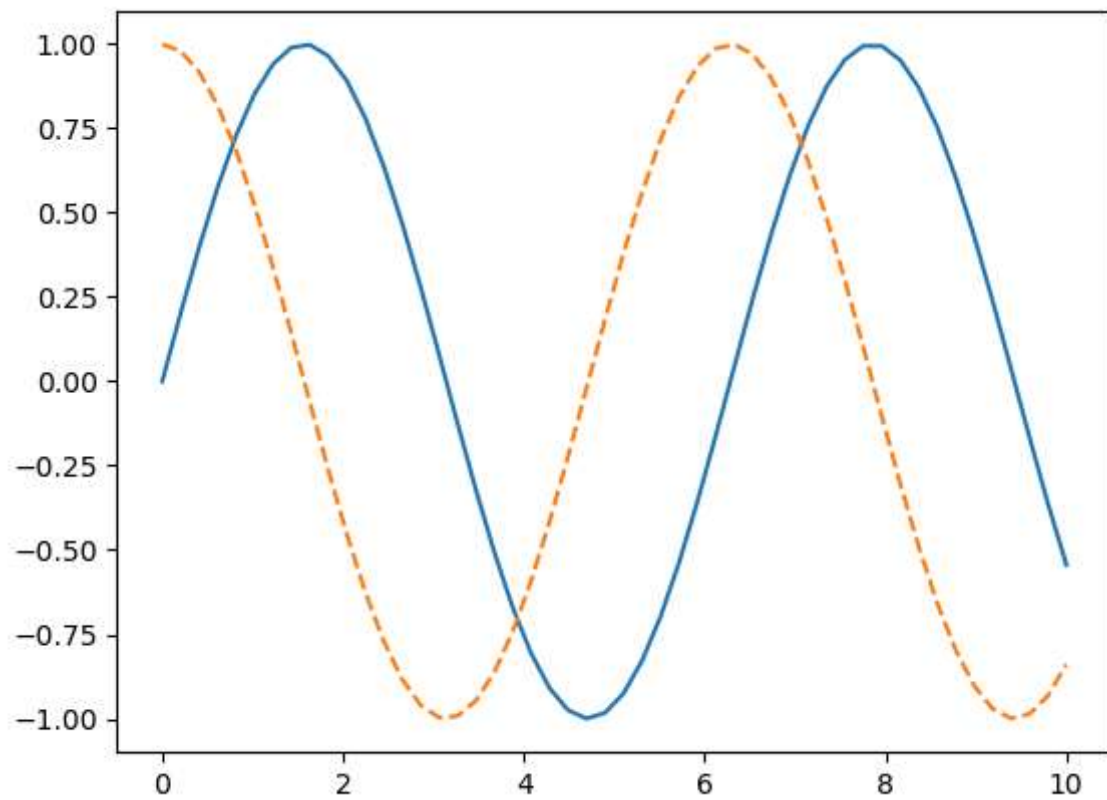


```
In [6]: import numpy as np
import matplotlib.pyplot as plt
```

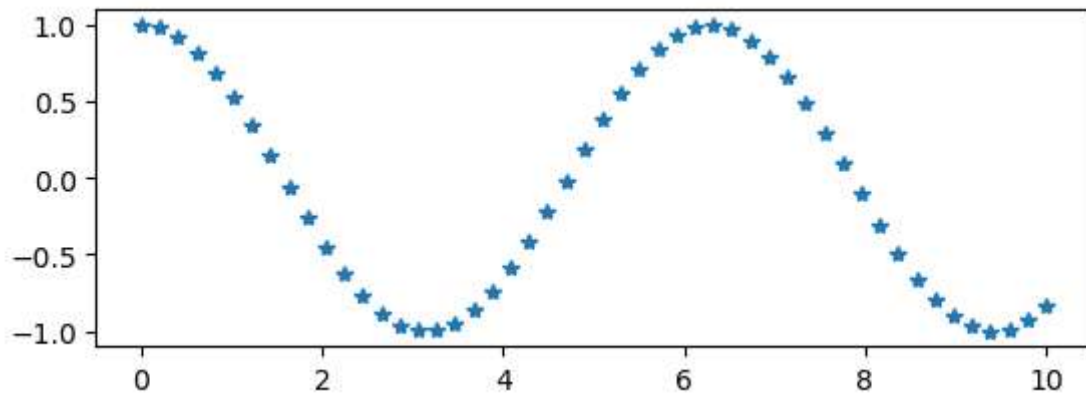
```
In [7]: %matplotlib inline
x1 = np.linspace(0, 10, 50)

# create a plot figure
#fig = plt.figure()

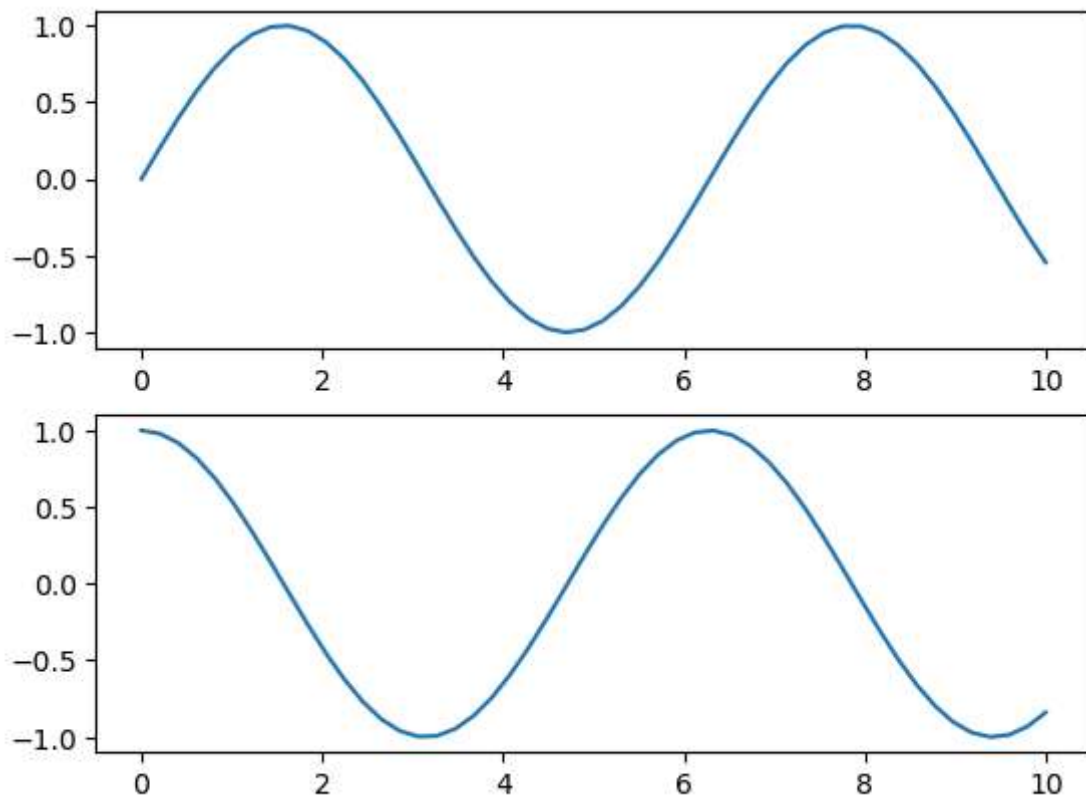
plt.plot(x1, np.sin(x1), '-')
plt.plot(x1, np.cos(x1), '--')
#plt.plot(x1, np.tan(x1), '--')
plt.show()
```



```
In [10]: plt.subplot(2, 1, 1)
plt.plot(x1, np.cos(x1), '*')
plt.show()
```



```
In [15]: plt.figure()
plt.subplot(2, 1, 1)
plt.plot(x1, np.sin(x1))
plt.subplot(2, 1, 2)
plt.plot(x1, np.cos(x1))
plt.show()
```

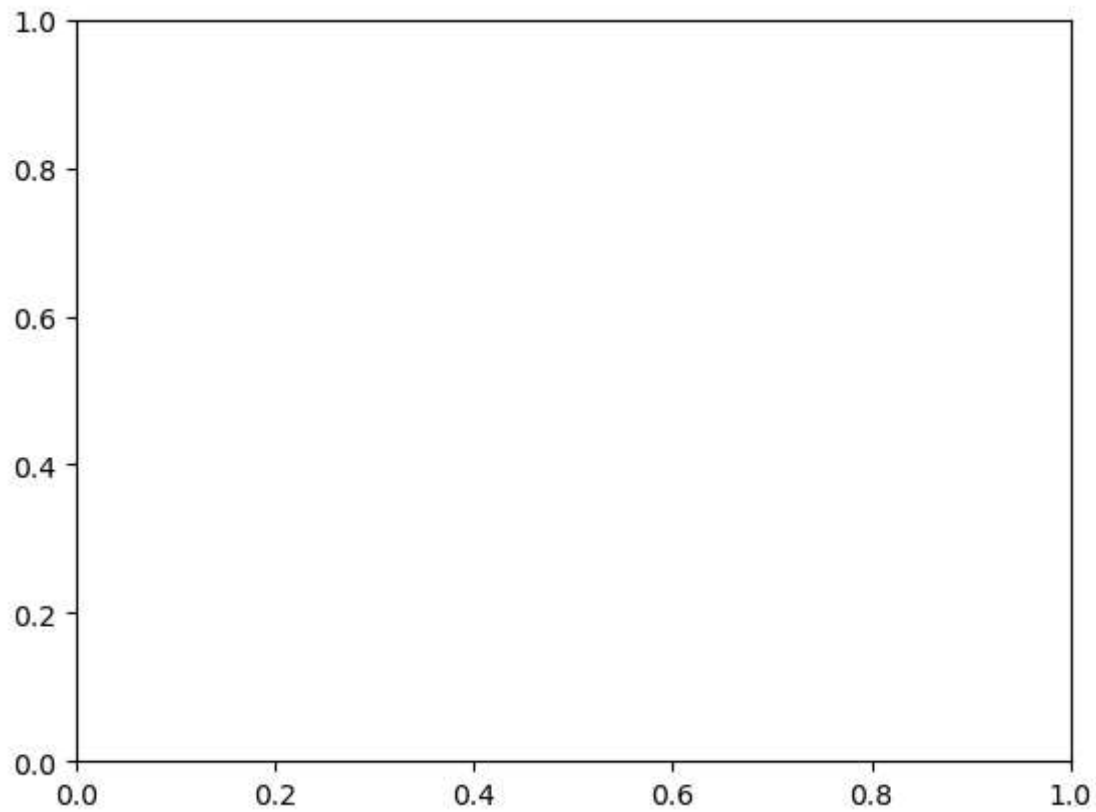


```
In [16]: print(plt.gcf())
```

Figure(640x480)

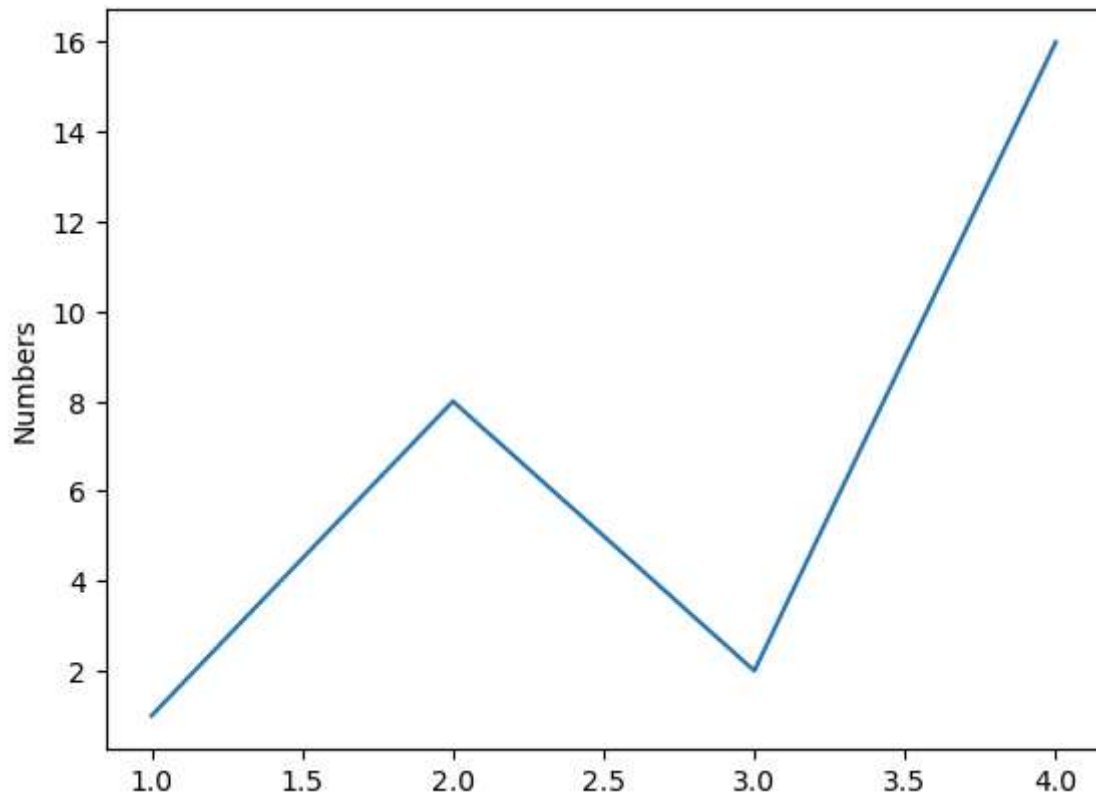
```
In [18]: print(plt.gca())
plt.show()
```

Axes(0.125,0.11;0.775x0.77)



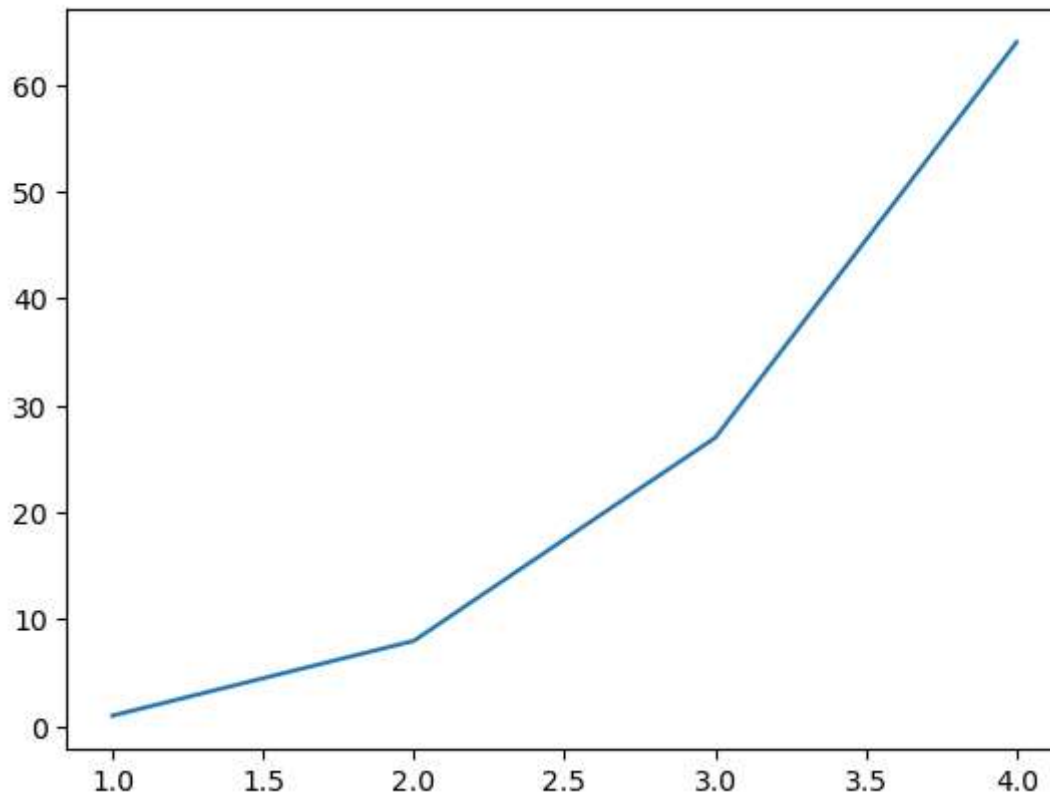
Visualization with Pyplot

```
In [20]: plt.plot([1,2,3,4], [1,8,2,16])  
plt.ylabel('Numbers')  
plt.show()
```



plot() - A versatile command

```
In [23]: import matplotlib.pyplot as plt
plt.plot([1,2,3,4],[1,8,27,64])
plt.show()
```



State-machine interface

```
In [24]: x = np.linspace(0, 2, 100)

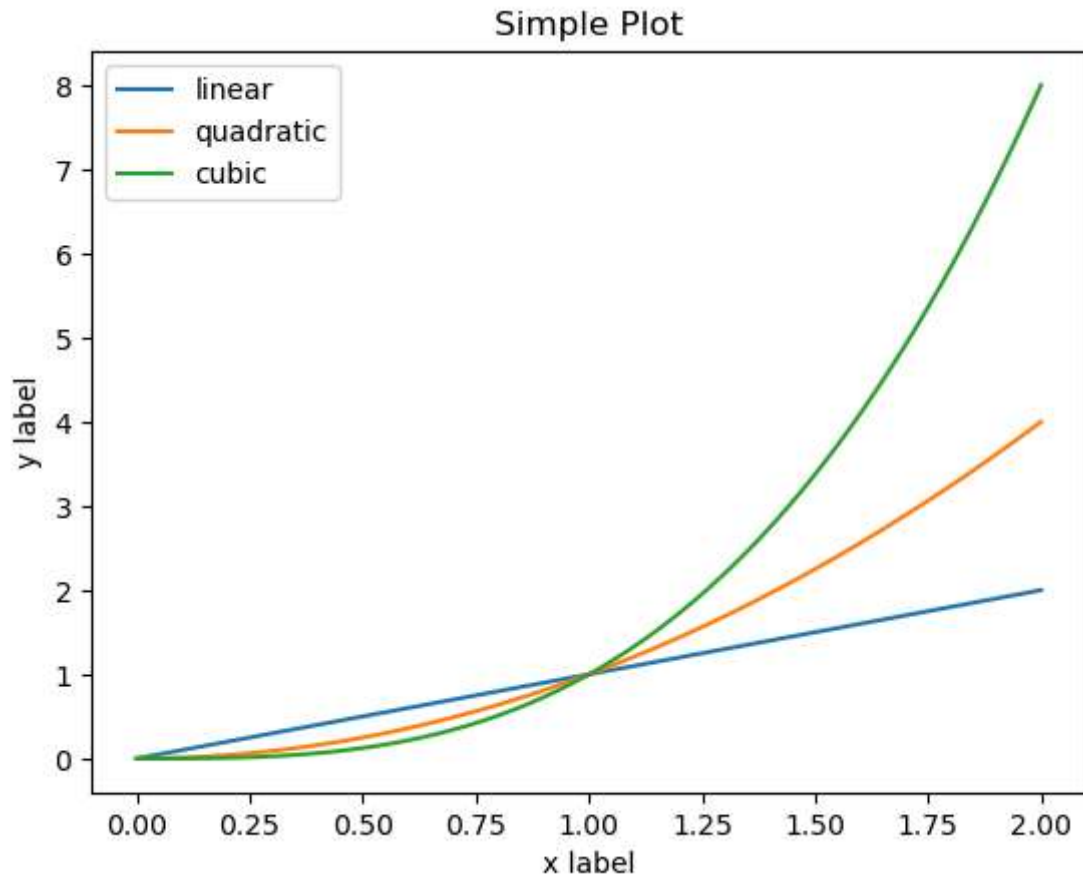
plt.plot(x, x, label='linear')
plt.plot(x, x**2, label='quadratic')
plt.plot(x, x**3, label='cubic')

plt.xlabel('x label')
plt.ylabel('y label')

plt.title("Simple Plot")

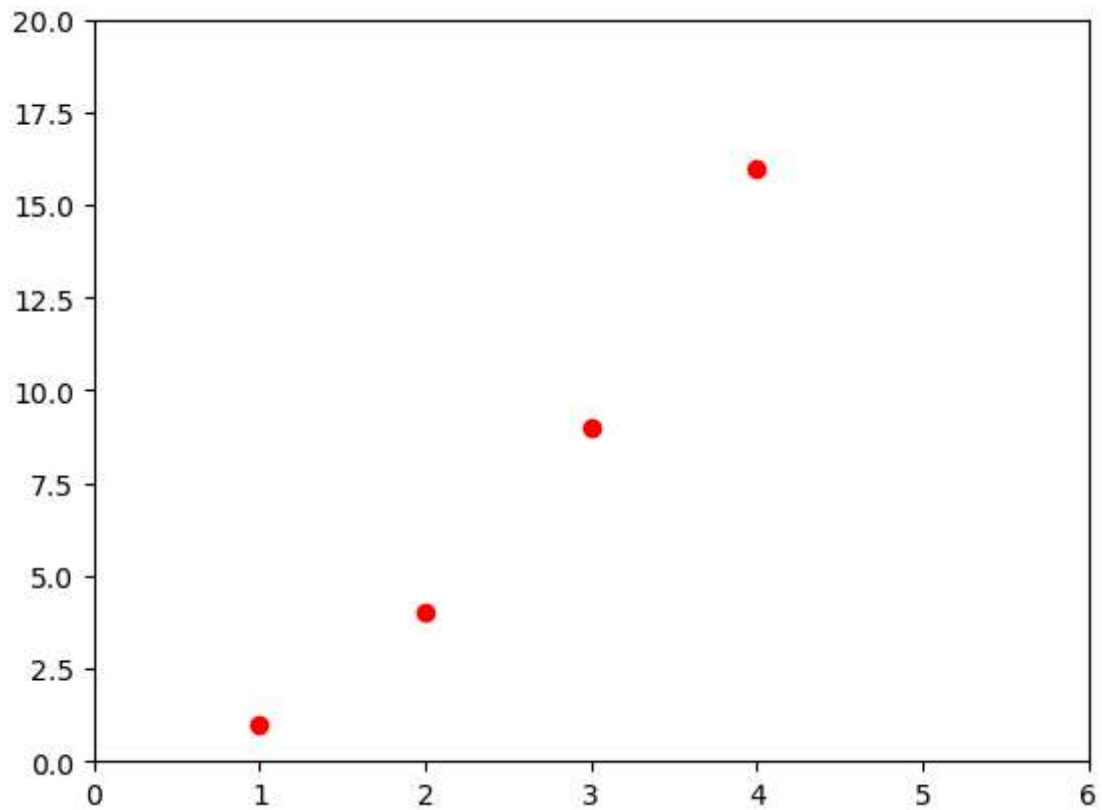
plt.legend()

plt.show()
```



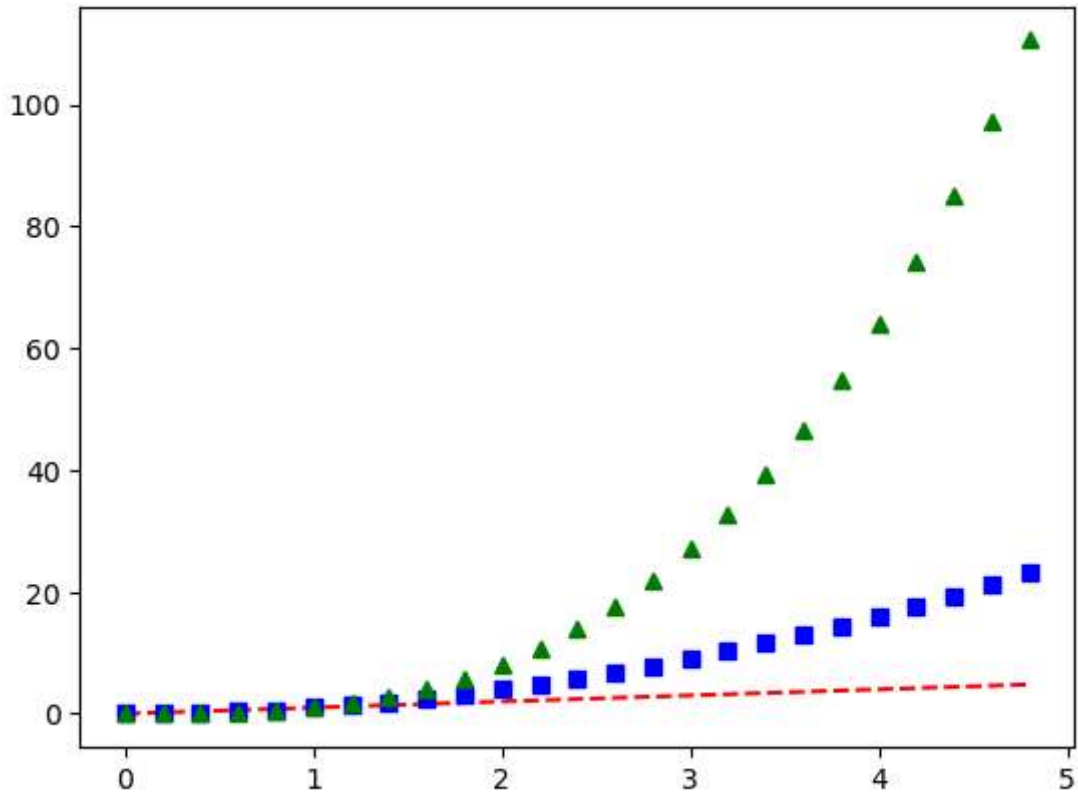
Formatting the style of plot

```
In [26]: plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')  
plt.axis([0, 6, 0, 20])  
plt.show()
```



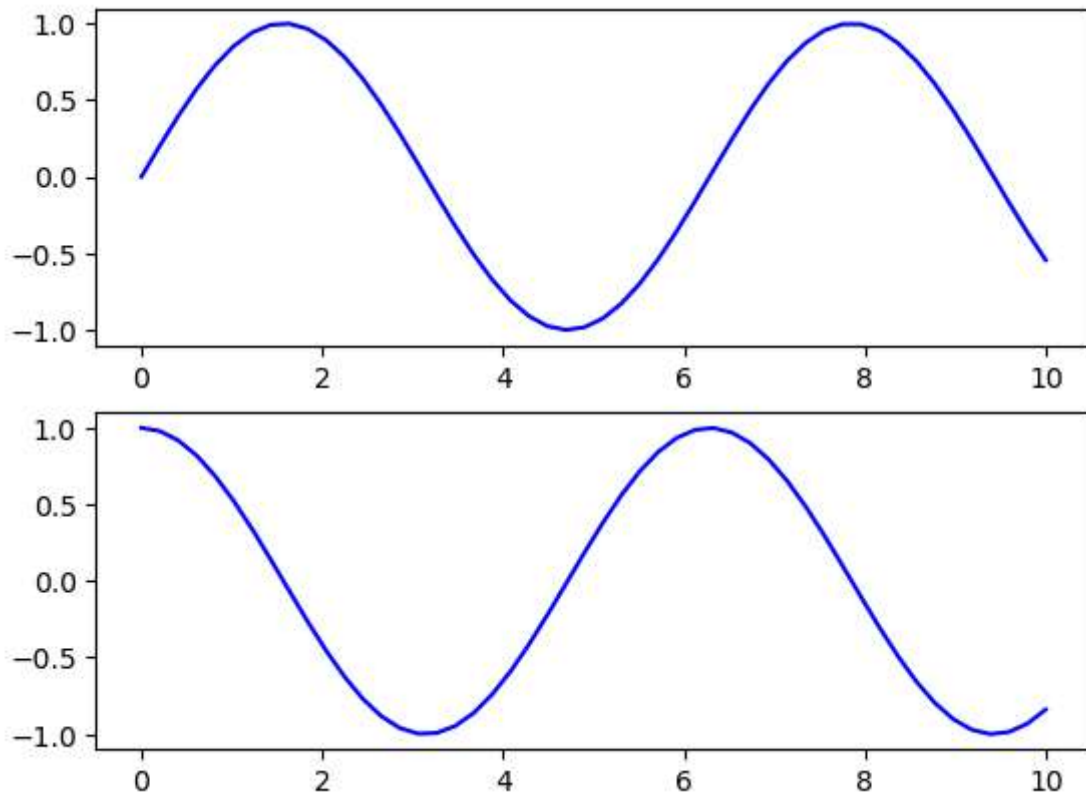
Working with NumPy arrays

```
In [27]: t = np.arange(0., 5., 0.2)
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



Object-Oriented API

```
In [33]: fig, ax = plt.subplots(2)
ax[0].plot(x1, np.sin(x1), 'b-')
ax[1].plot(x1, np.cos(x1), 'b-');
plt.show()
```

```
In [40]: fig = plt.figure()

x2 = np.linspace(0, 5, 10)
y2 = x2 ** 2

axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])

axes.plot(x2, y2, 'r')
axes.set_xlabel('x2')
axes.set_ylabel('y2')
axes.set_title('title')
plt.show()
```

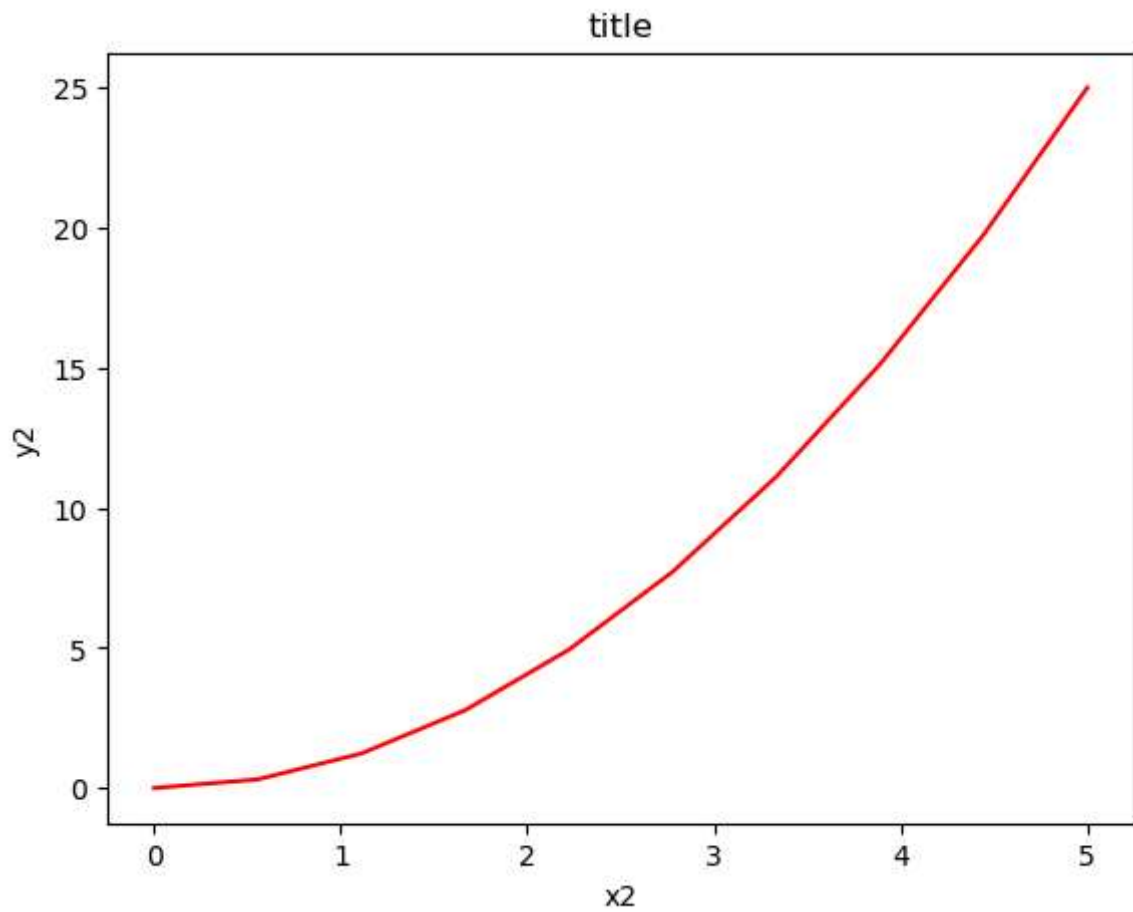


Figure and Axes

```
In [39]: fig = plt.figure()  
         ax = plt.axes()  
         plt.show()
```

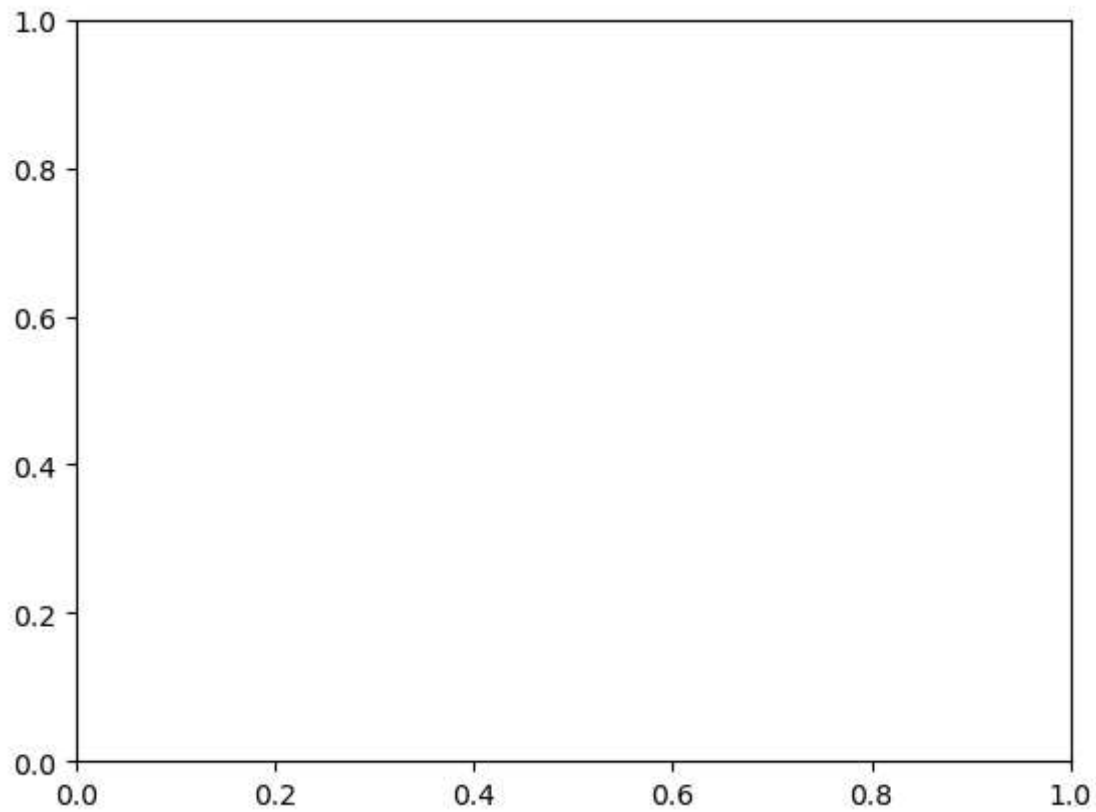
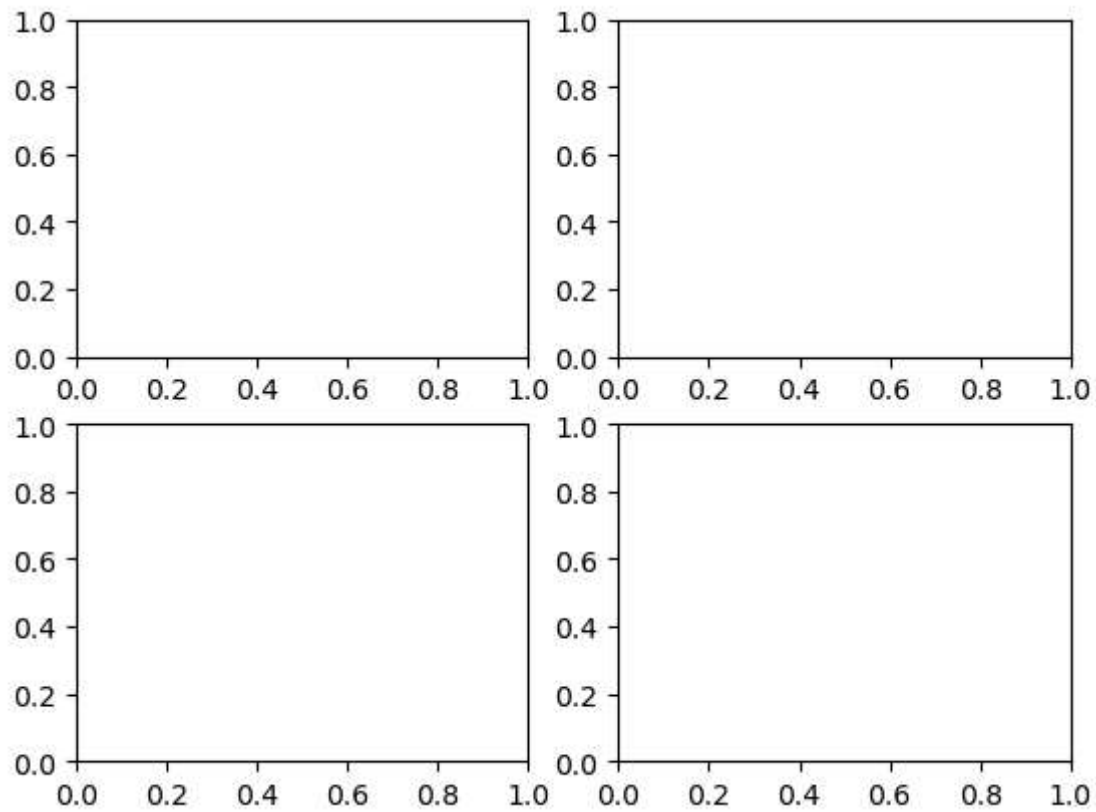


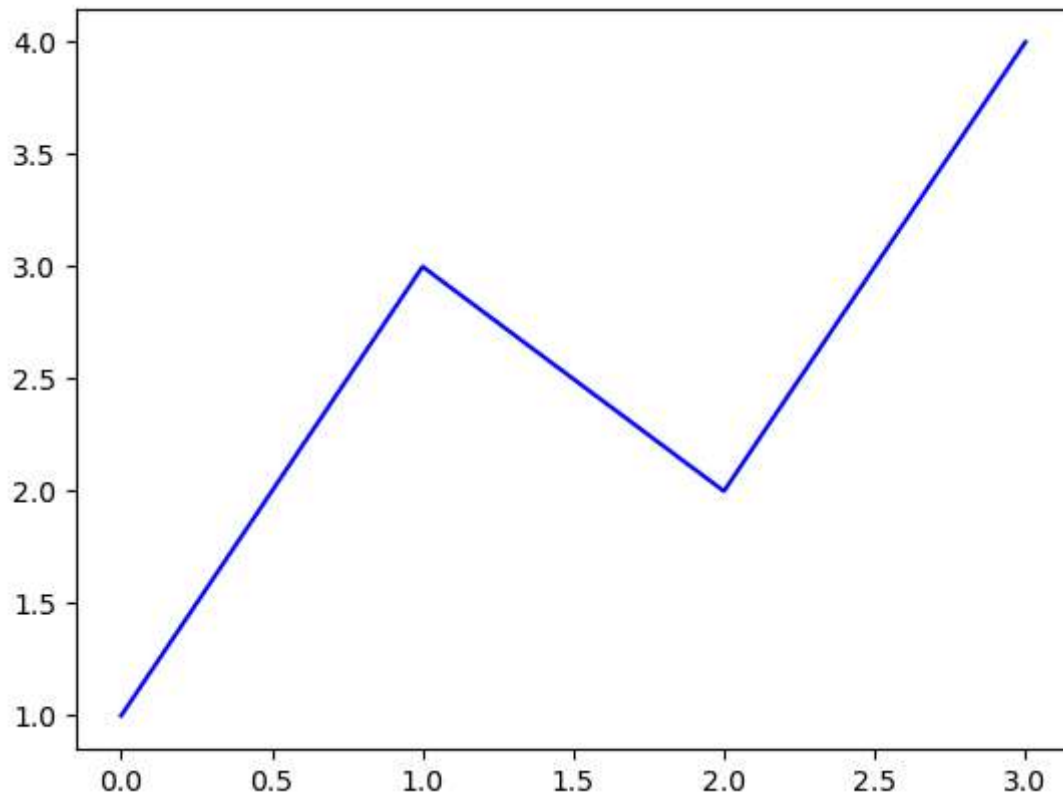
Figure and Subplots

```
In [43]: fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
ax4 = fig.add_subplot(2, 2, 4)
plt.show()
```



First plot with Matplotlib

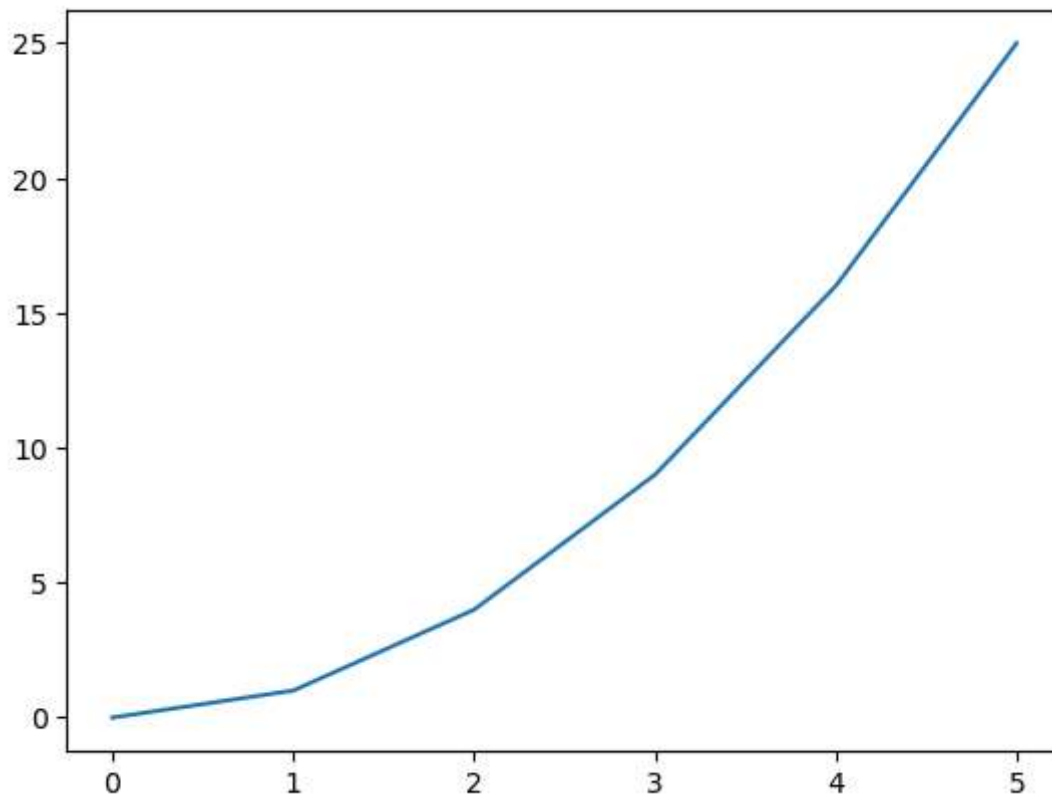
```
In [44]: plt.plot([1, 3, 2, 4], 'b-')  
plt.show()
```



```
In [45]: x3 = range(6)

plt.plot(x3, [xi**2 for xi in x3])

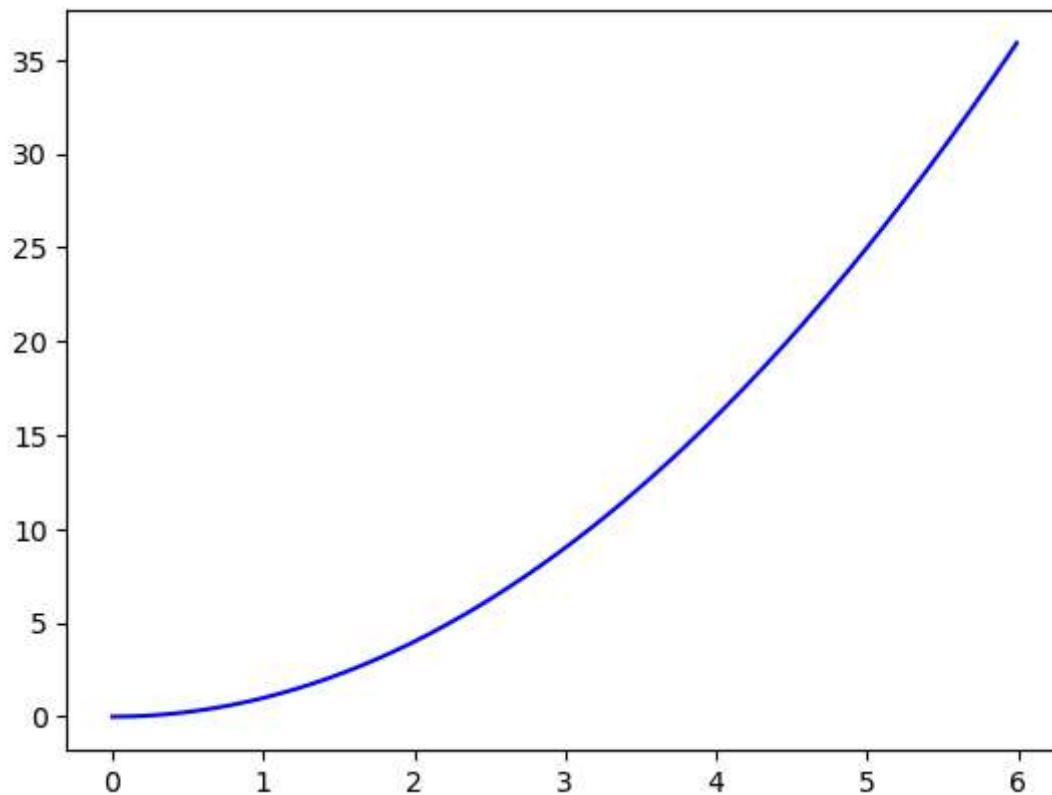
plt.show()
```



```
In [46]: x3 = np.arange(0.0, 6.0, 0.01)

plt.plot(x3, [xi**2 for xi in x3], 'b-')

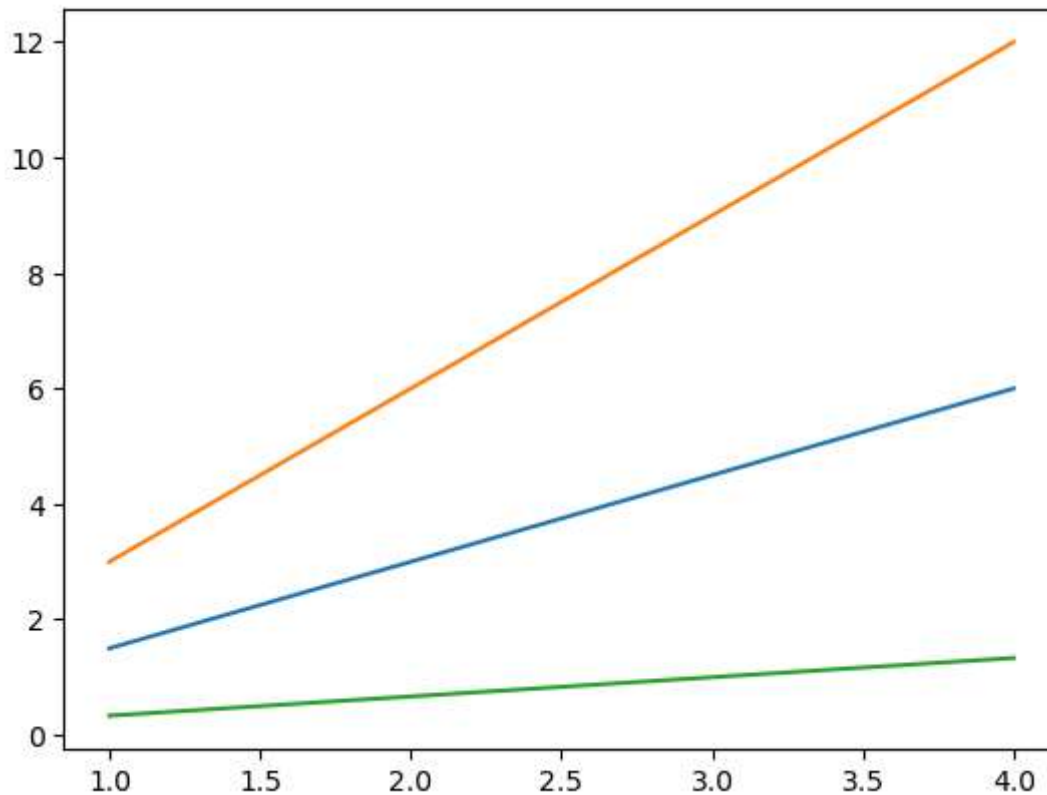
plt.show()
```



Multiline Plots

```
In [51]: x4 = range(1, 5)

plt.plot(x4, [xi*1.5 for xi in x4])
plt.plot(x4, [xi*3 for xi in x4])
plt.plot(x4, [xi/3.0 for xi in x4])
plt.show()
```



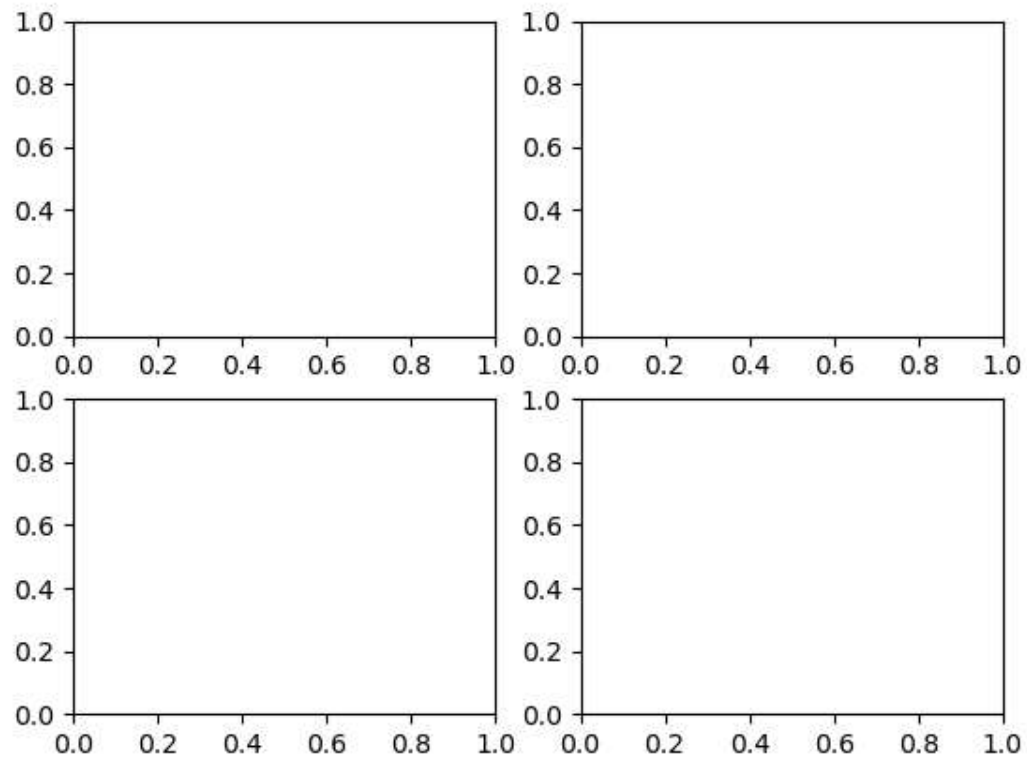
Saving The Plot

```
In [60]: fig.canvas.get_supported_filetypes()
```

```
Out[60]: {'eps': 'Encapsulated Postscript',  
          'jpg': 'Joint Photographic Experts Group',  
          'jpeg': 'Joint Photographic Experts Group',  
          'pdf': 'Portable Document Format',  
          'pgf': 'PGF code for LaTeX',  
          'png': 'Portable Network Graphics',  
          'ps': 'Postscript',  
          'raw': 'Raw RGBA bitmap',  
          'rgba': 'Raw RGBA bitmap',  
          'svg': 'Scalable Vector Graphics',  
          'svgz': 'Scalable Vector Graphics',  
          'tif': 'Tagged Image File Format',  
          'tiff': 'Tagged Image File Format',  
          'webp': 'WebP Image Format'}
```

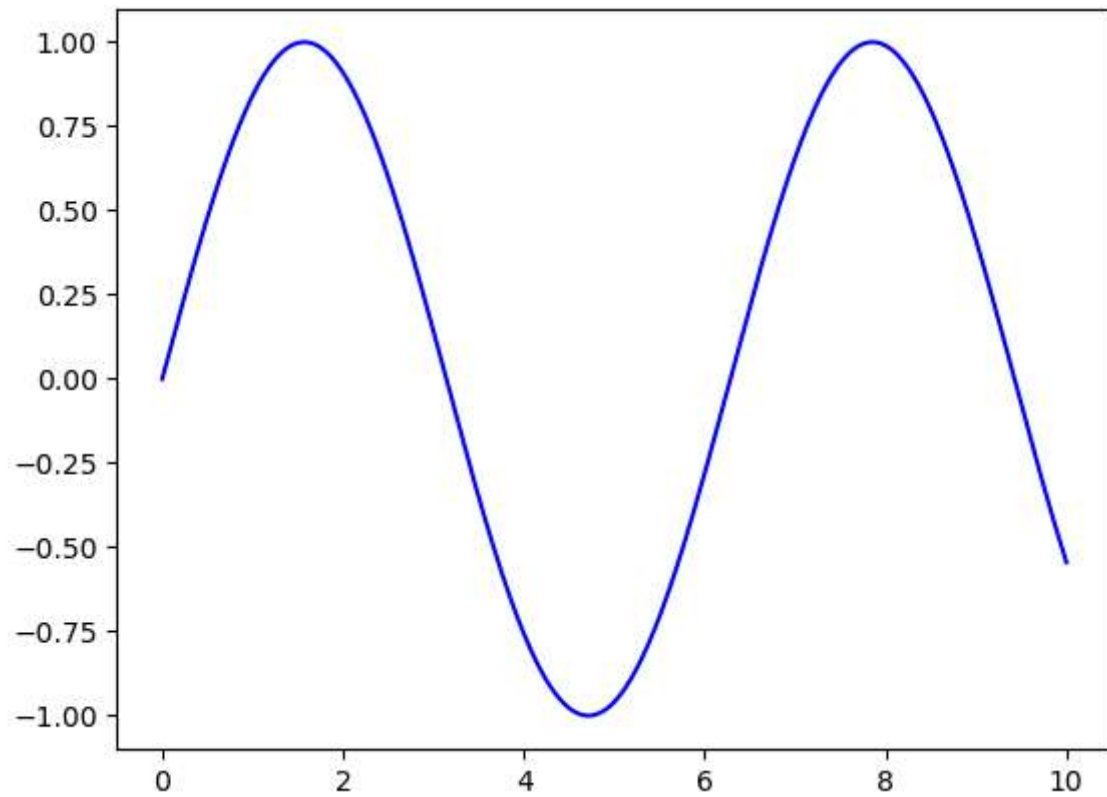
```
In [61]: fig.savefig('plot1.png')  
from IPython.display import Image  
Image('plot1.png')
```

Out[61]:



Line Plot

```
In [64]: fig = plt.figure()
ax = plt.axes()
x5 = np.linspace(0, 10, 1000)
ax.plot(x5, np.sin(x5), 'b-');
plt.show()
```

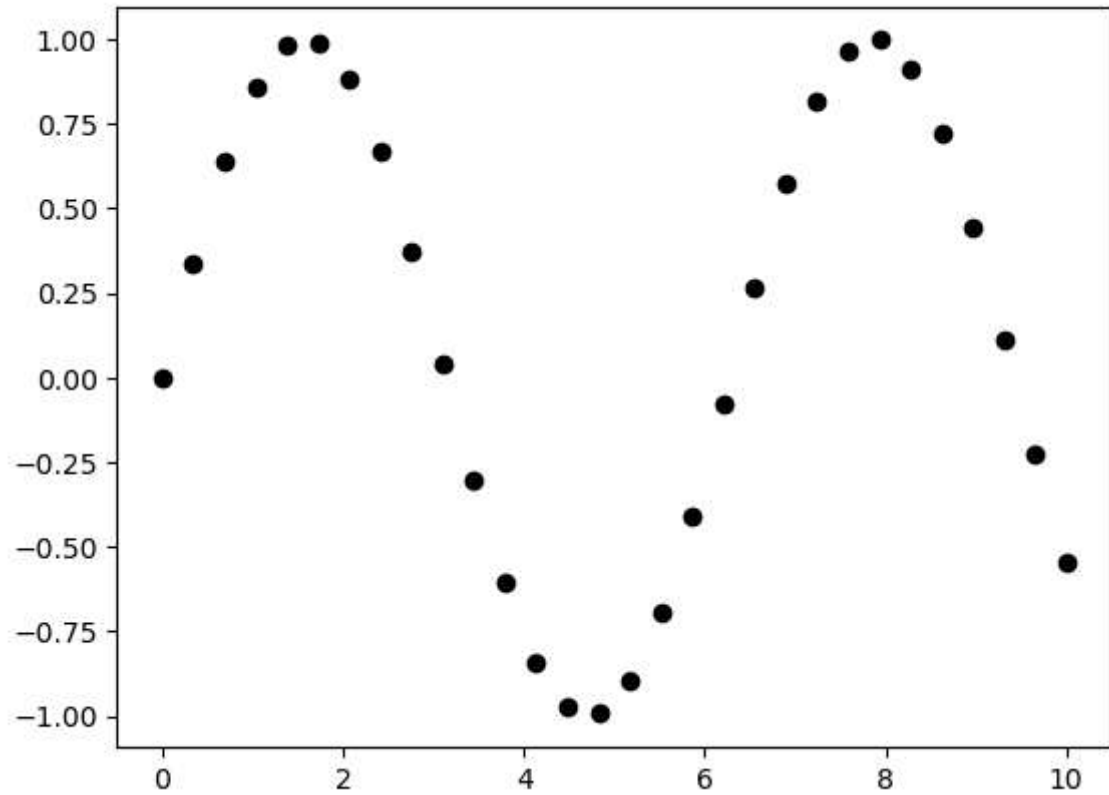



Scatter plot

```
In [70]: x7 = np.linspace(0, 10, 30)

y7 = np.sin(x7)

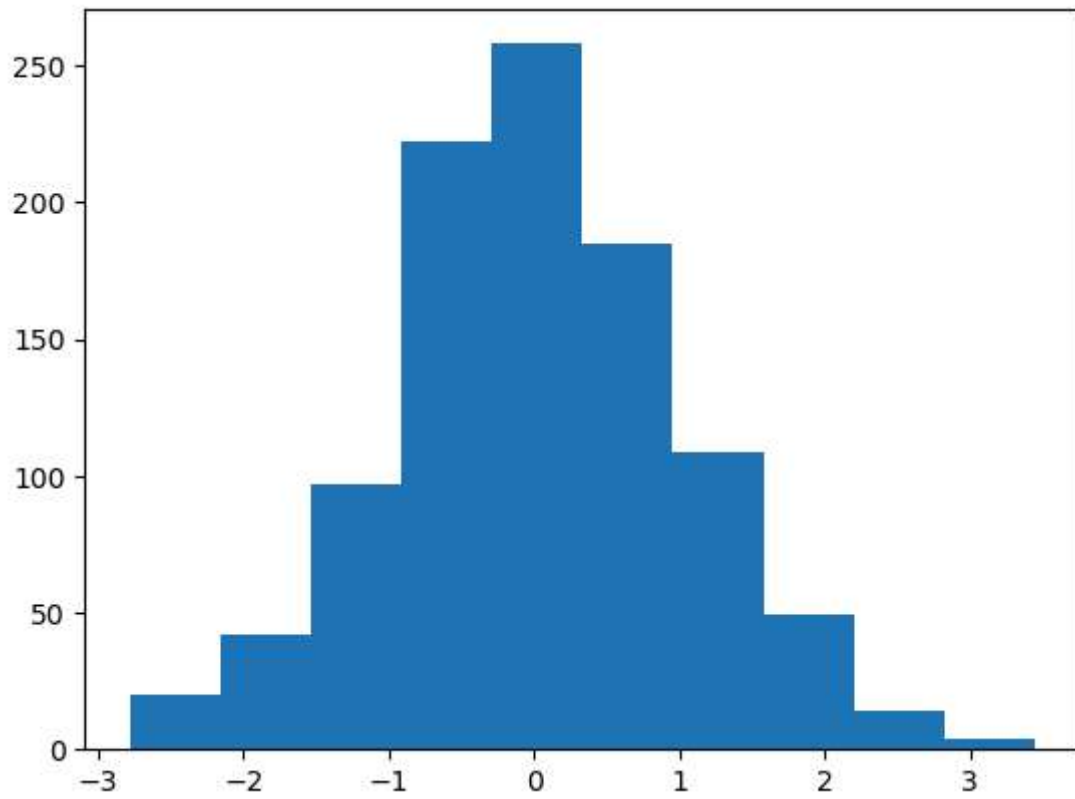
plt.plot(x7, y7, 'o', color = 'black');
plt.show()
```



Histogram

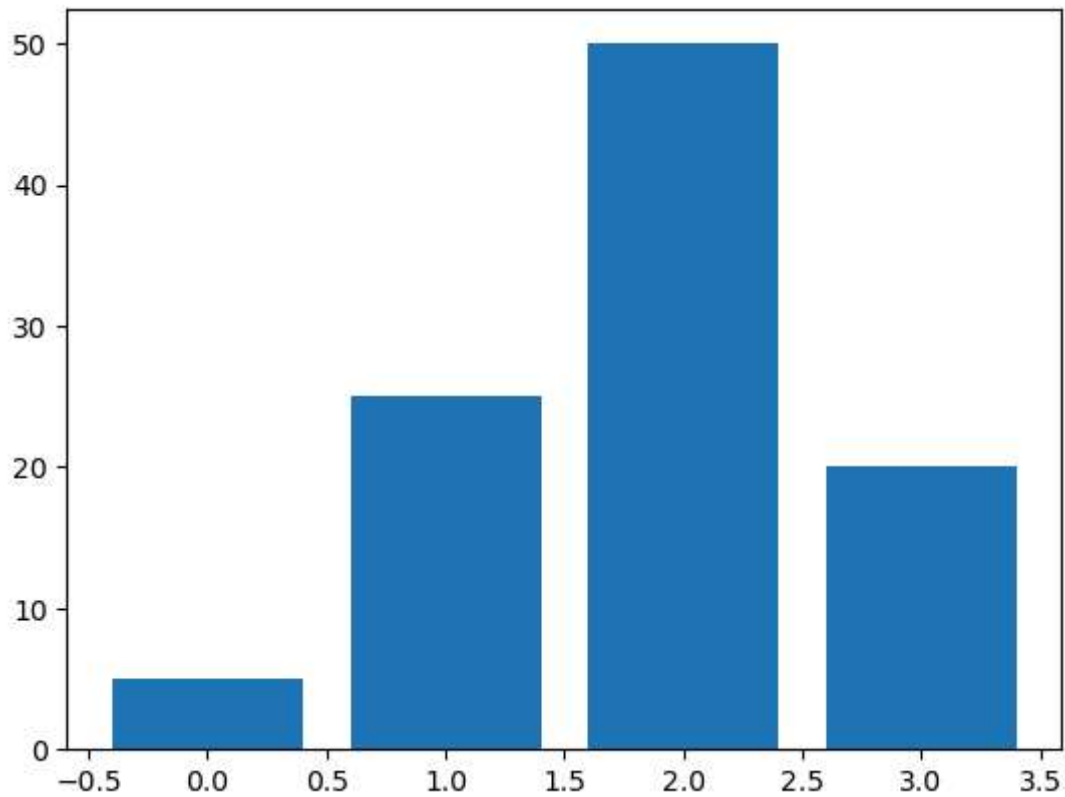
```
In [73]: data1 = np.random.randn(1000)

plt.hist(data1);
plt.show()
```



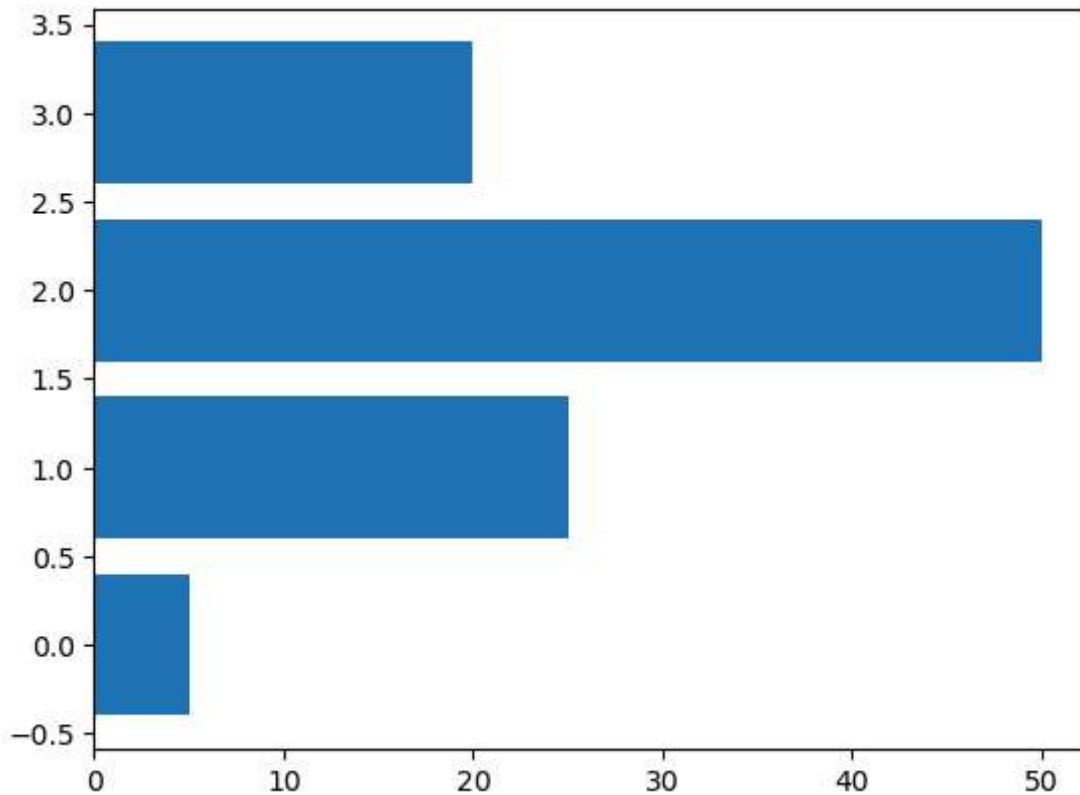
Bar Chart

```
In [74]: data2 = [5, 25, 50, 20.]  
plt.bar(range(len(data2)), data2)  
plt.show()
```



Horizontal Bar Chart

```
In [76]: data2 = [5., 25., 50., 20.]  
plt.barh(range(len(data2)), data2)  
plt.show()
```



Error Bar Chart

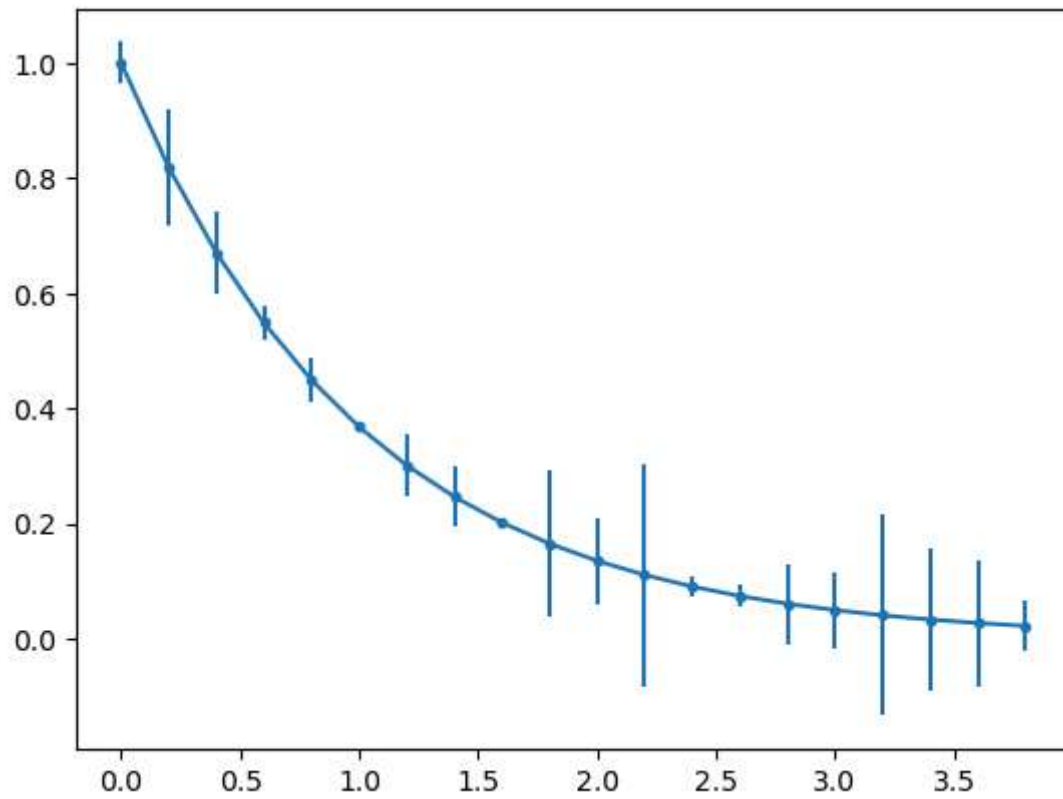
```
In [77]: x9 = np.arange(0, 4, 0.2)

y9 = np.exp(-x9)

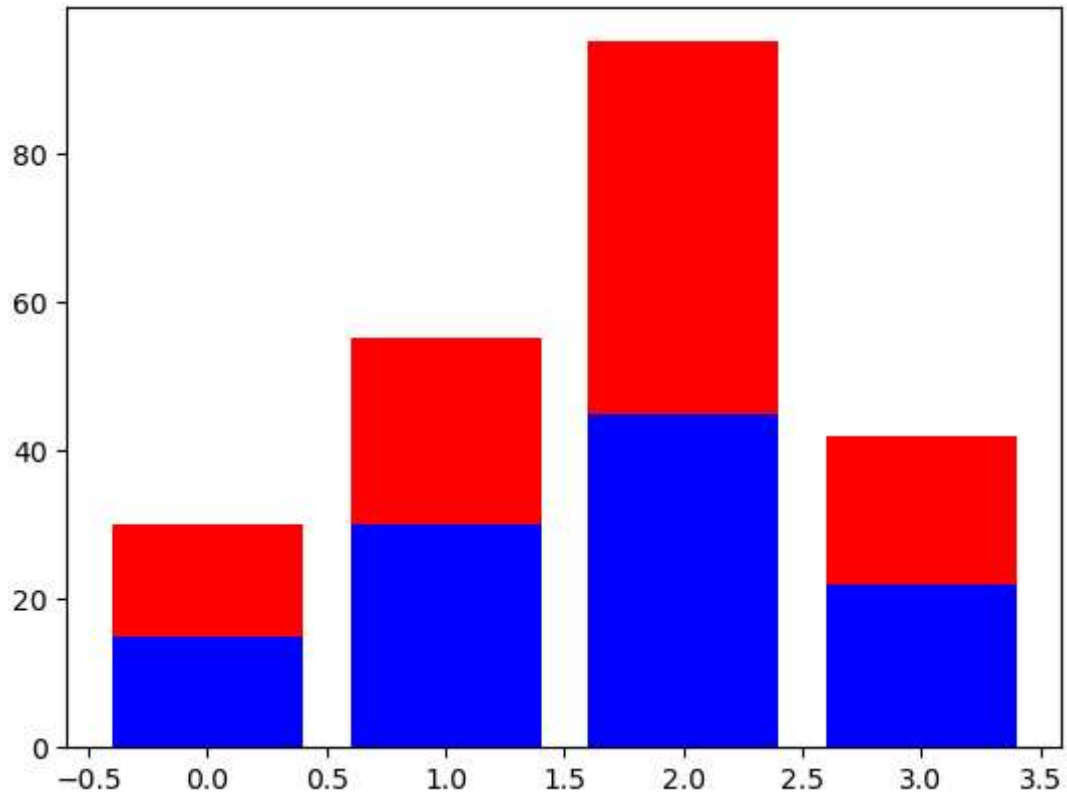
e1 = 0.1 * np.abs(np.random.randn(len(y9)))

plt.errorbar(x9, y9, yerr = e1, fmt = '.-')

plt.show();
```

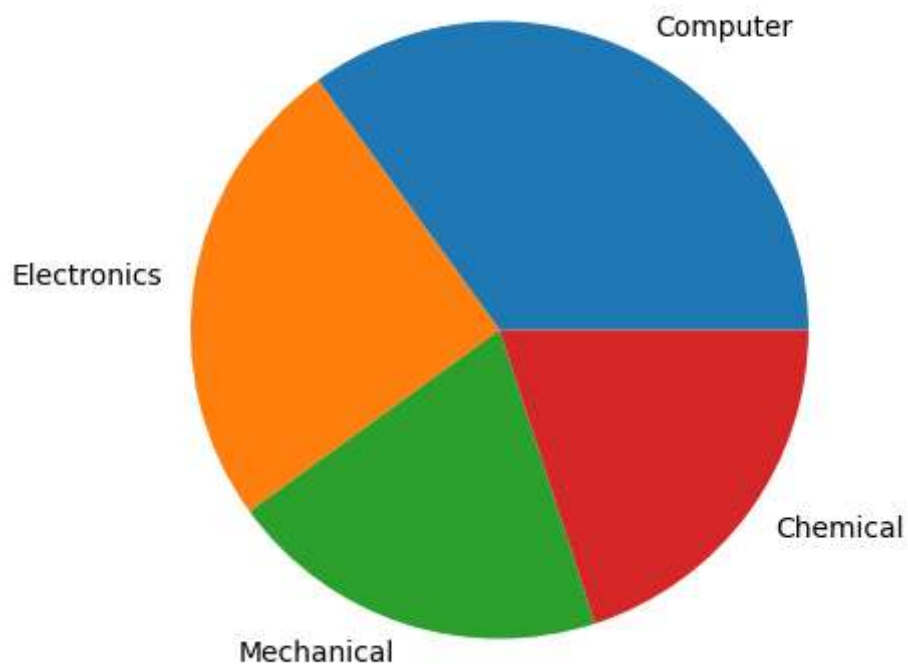


```
In [79]: A = [15., 30., 45., 22.]  
B = [15., 25., 50., 20.]  
z2 = range(4)  
plt.bar(z2, A, color = 'b')  
plt.bar(z2, B, color = 'r', bottom = A)  
plt.show()
```



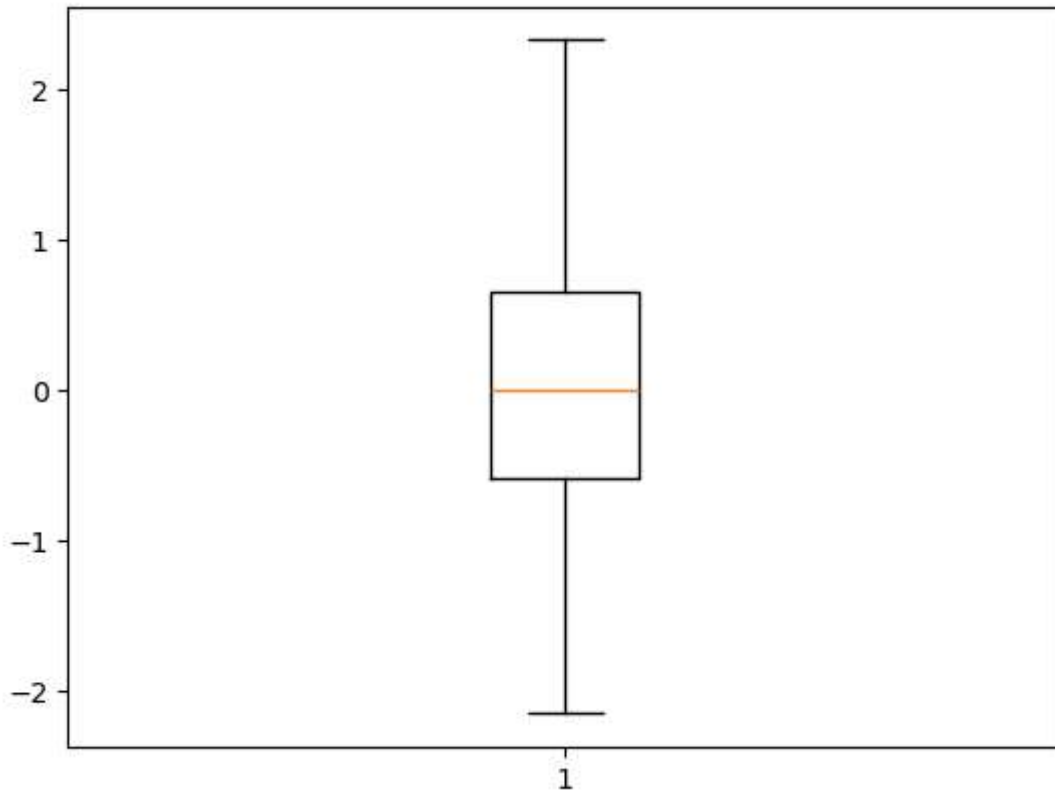
Pie Chart

```
In [82]: plt.figure(figsize=(5,5))  
  
x10 = [35, 25, 20, 20]  
  
labels = ['Computer', 'Electronics', 'Mechanical', 'Chemical']  
  
plt.pie(x10, labels=labels);  
  
plt.show()
```



BoxPlot

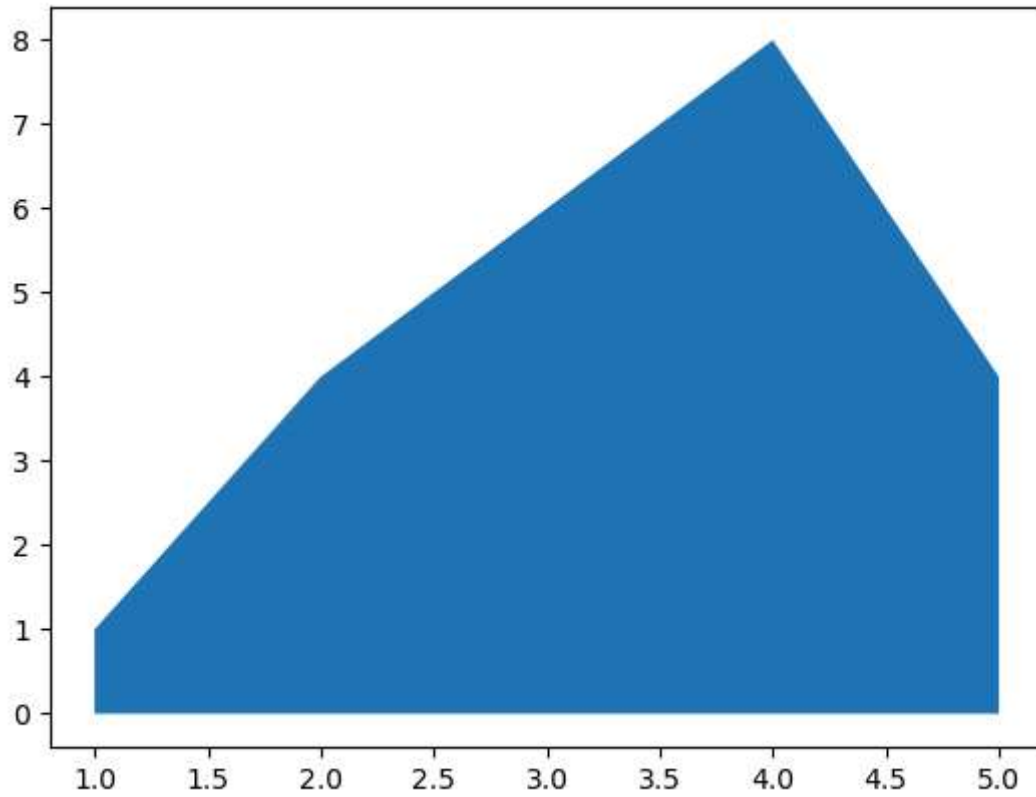
```
In [83]: data3 = np.random.randn(100)
plt.boxplot(data3)
plt.show()
```

Area Chat

```
In [84]: x12 = range(1, 6)
          y12 = [1, 4, 6, 8, 4]

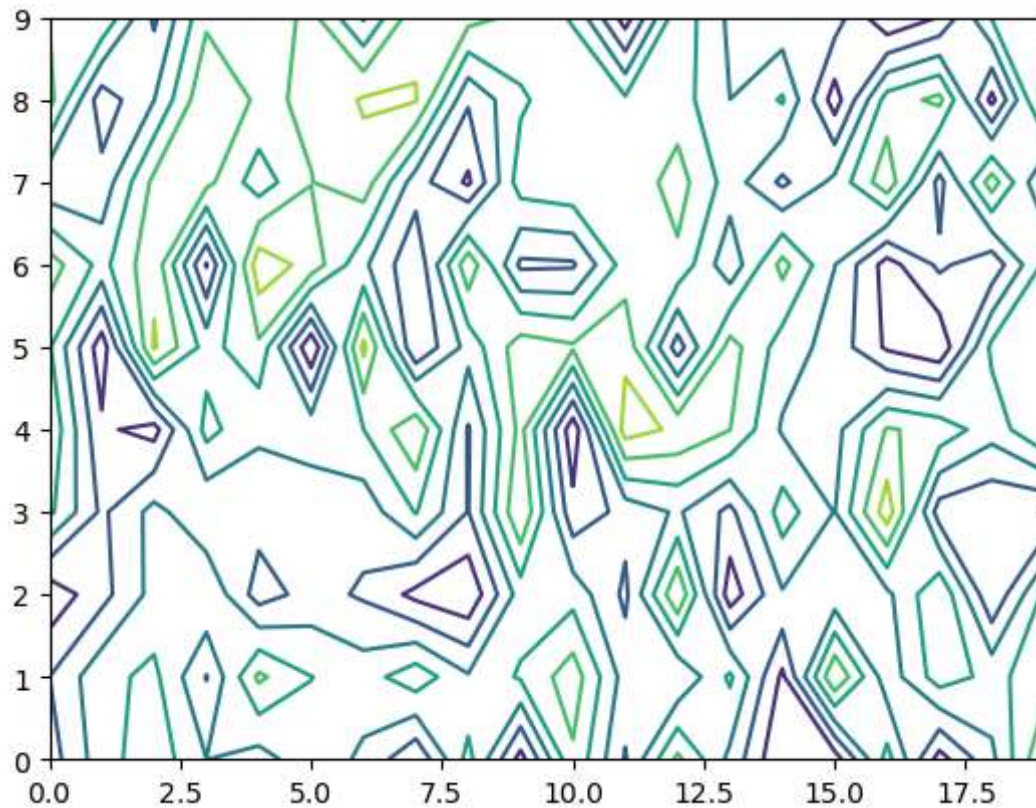
          plt.fill_between(x12, y12)
          plt.show()
```



Contour Plot

```
In [86]: # Create a matrix
matrix1 = np.random.rand(10, 20)

cp = plt.contour(matrix1)
plt.show()
```



styles with Matplotlib Plots

In [89]: *# View list of all available styles*

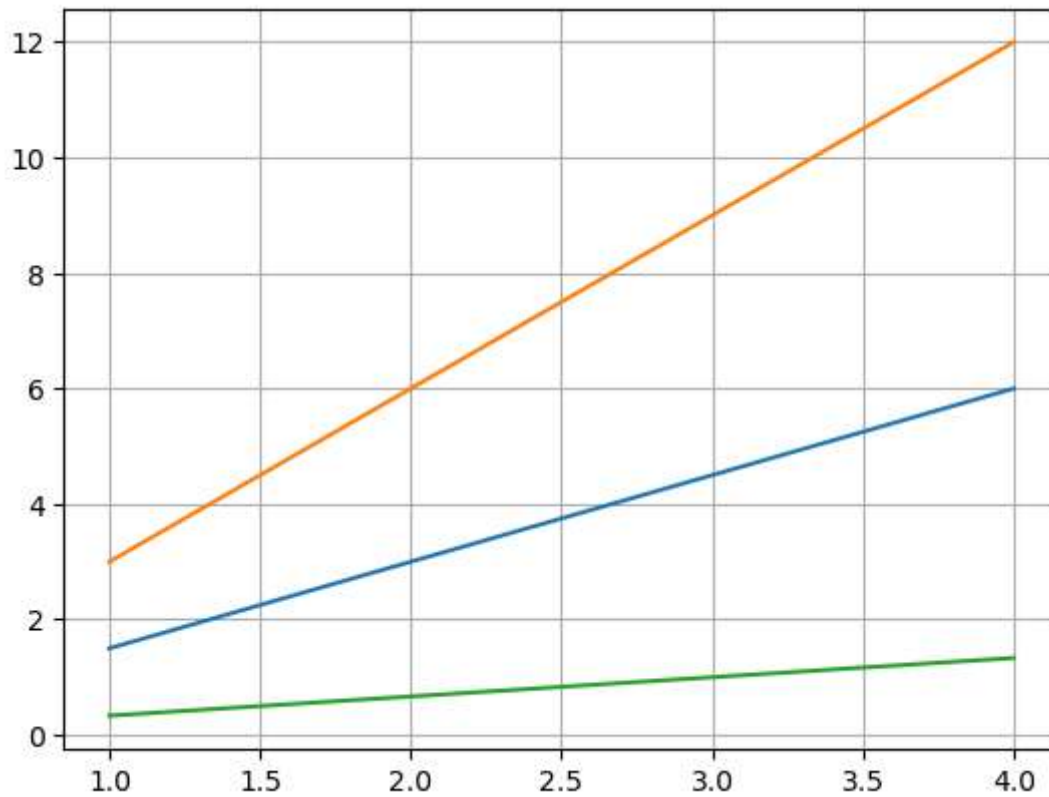
```
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn-v0_8', 'seaborn-v0_8-bright', 'seaborn-v0_8-colorblind', 'seaborn-v0_8-dark', 'seaborn-v0_8-dark-palette', 'seaborn-v0_8-darkgrid', 'seaborn-v0_8-deep', 'seaborn-v0_8-muted', 'seaborn-v0_8-notebook', 'seaborn-v0_8-paper', 'seaborn-v0_8-pastel1', 'seaborn-v0_8-poster', 'seaborn-v0_8-talk', 'seaborn-v0_8-ticks', 'seaborn-v0_8-white', 'seaborn-v0_8-whitegrid', 'tableau-colorblind10']
```

In [93]: `x15 = np.arange(1, 5)`

```
plt.plot(x15, x15*1.5, x15, x15*3.0, x15, x15/3.0)
```

```
plt.grid(True)
plt.show()
```



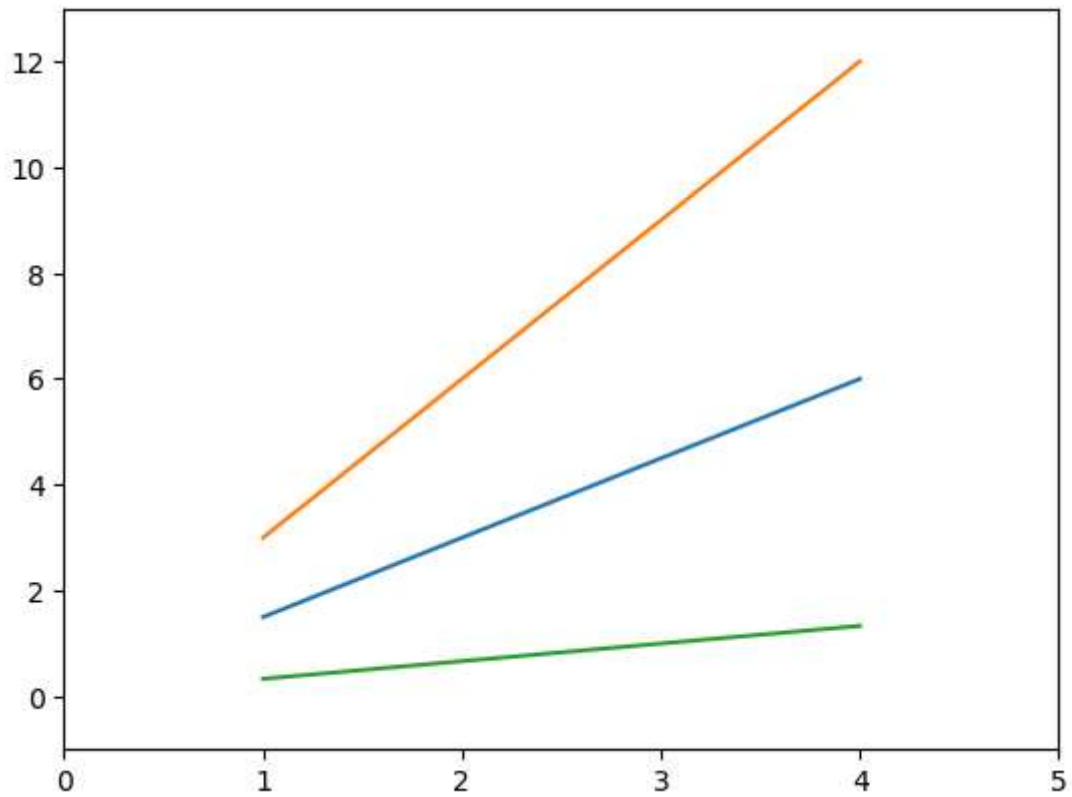
Handling Axes

```
In [94]: x15 = np.arange(1, 5)

plt.plot(x15, x15*1.5, x15, x15*3.0, x15, x15/3.0)
plt.axis() # shows the current axis limits values

plt.axis([0, 5, -1, 13])

plt.show()
```

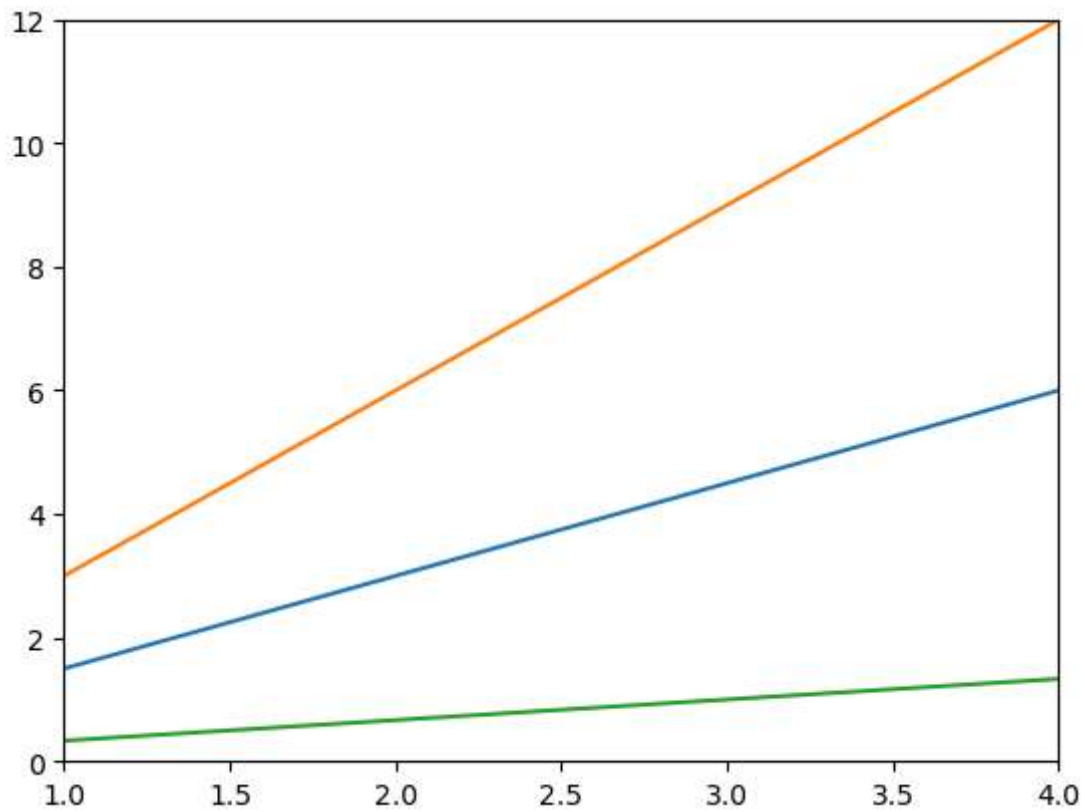


```
In [98]: x15 = np.arange(1, 5)

plt.plot(x15, x15*1.5, x15, x15*3.0, x15, x15/3.0)

plt.xlim([1.0, 4.0])

plt.ylim([0.0, 12.0])
plt.show()
```



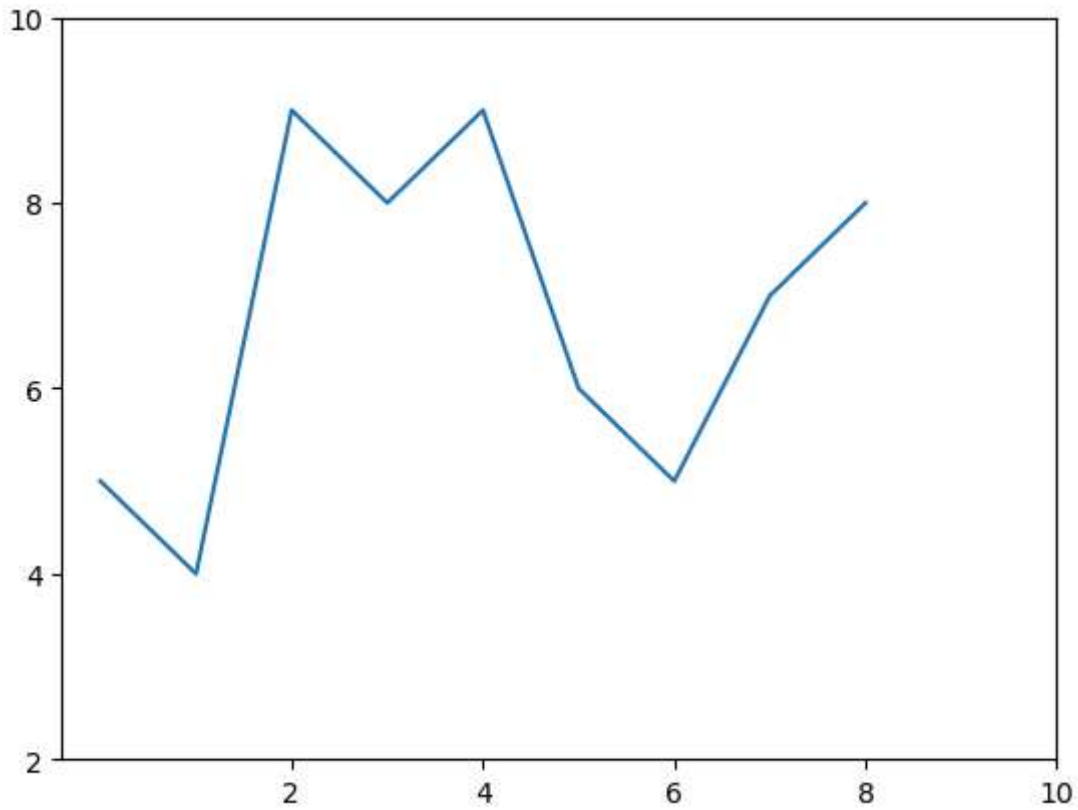
Handling X and Y ticks

```
In [99]: u = [5, 4, 9, 8, 9, 6, 5, 7, 8]

plt.plot(u)

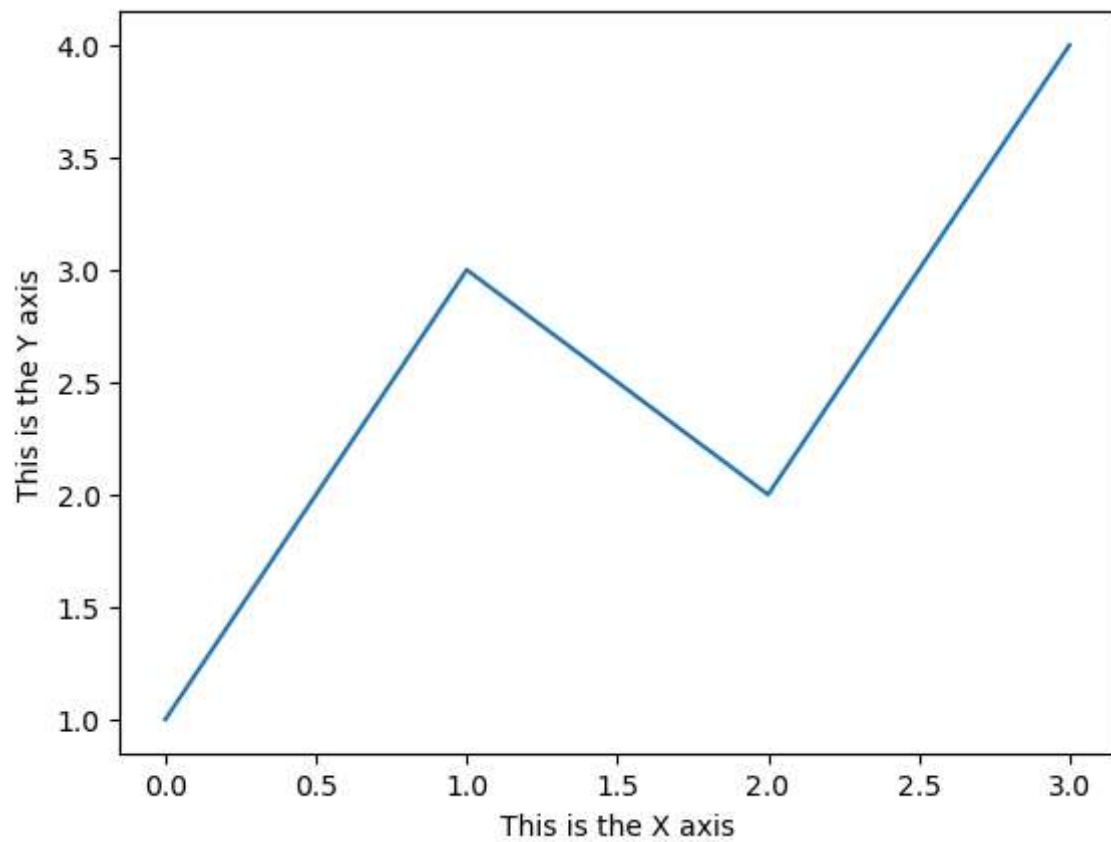
plt.xticks([2, 4, 6, 8, 10])
plt.yticks([2, 4, 6, 8, 10])

plt.show()
```



Adding Labels

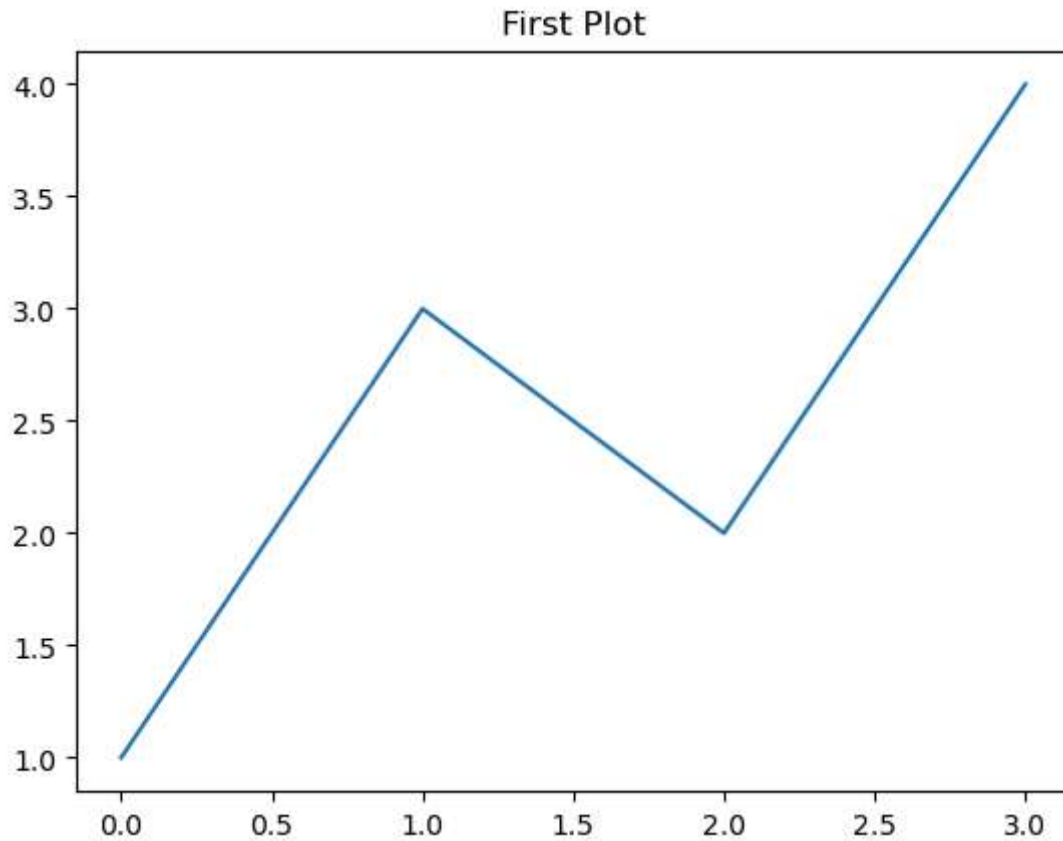
```
In [100... plt.plot([1, 3, 2, 4])  
  
plt.xlabel('This is the X axis')  
plt.ylabel('This is the Y axis')  
  
plt.show()
```



Adding a Title

In [101...

```
plt.plot([1, 3, 2, 4])  
  
plt.title('First Plot')  
  
plt.show()
```

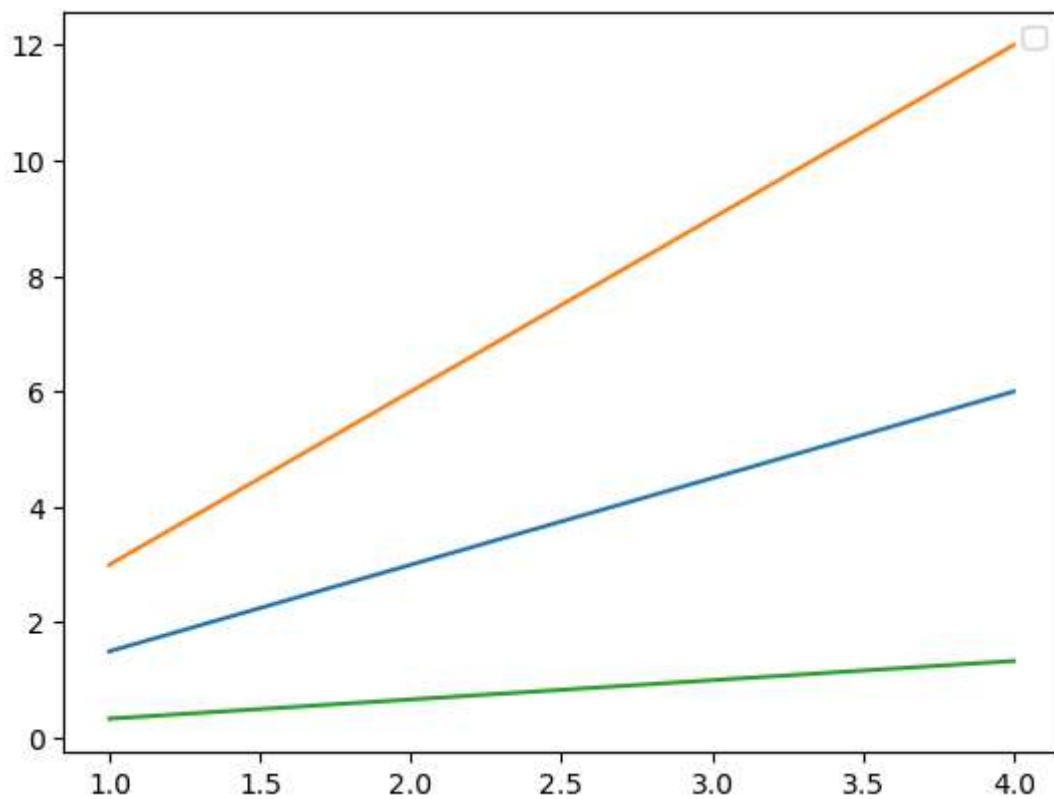
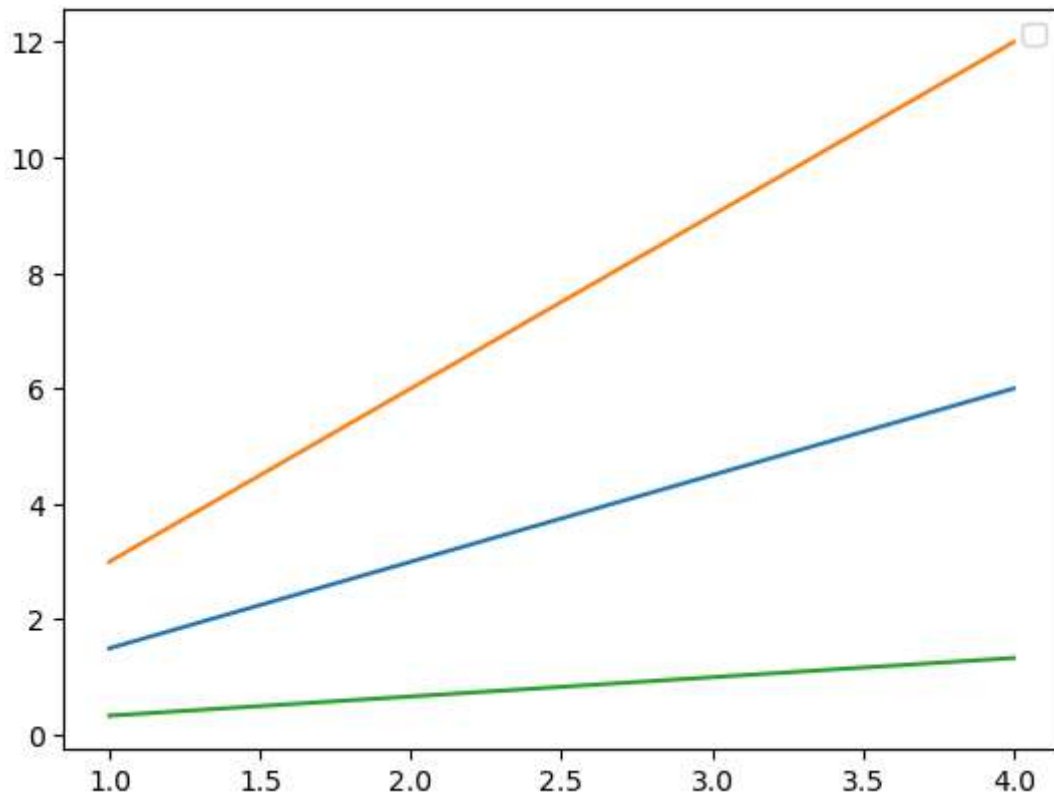
Adding a legend

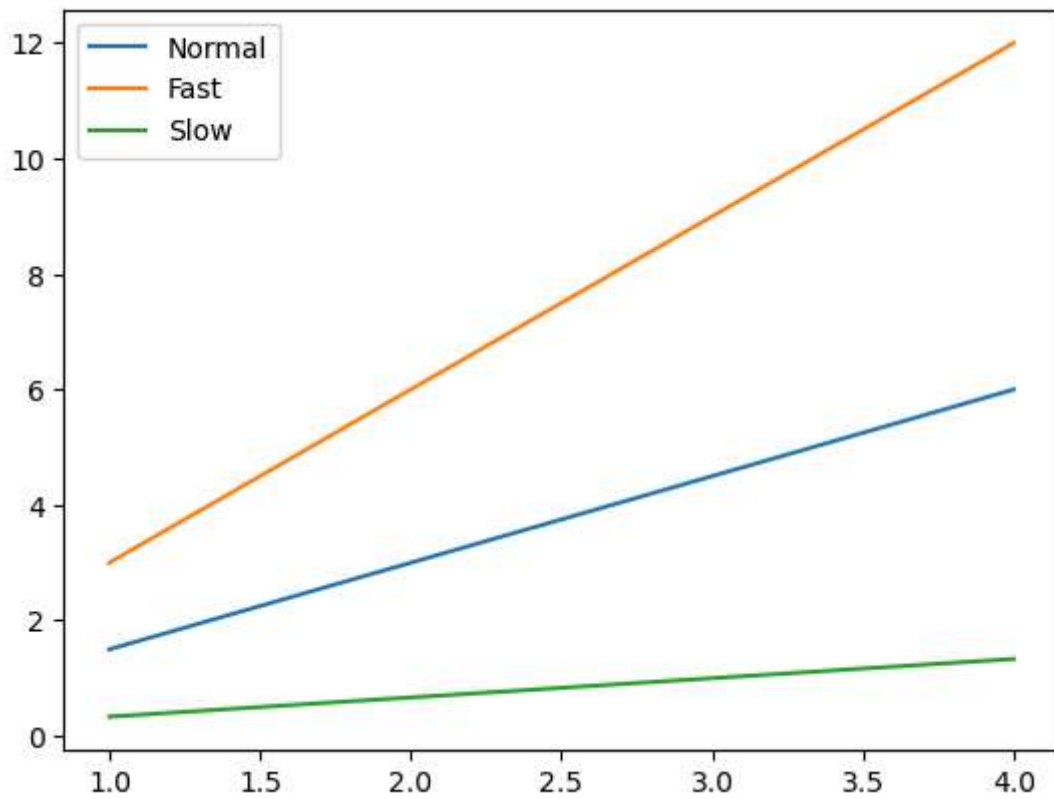
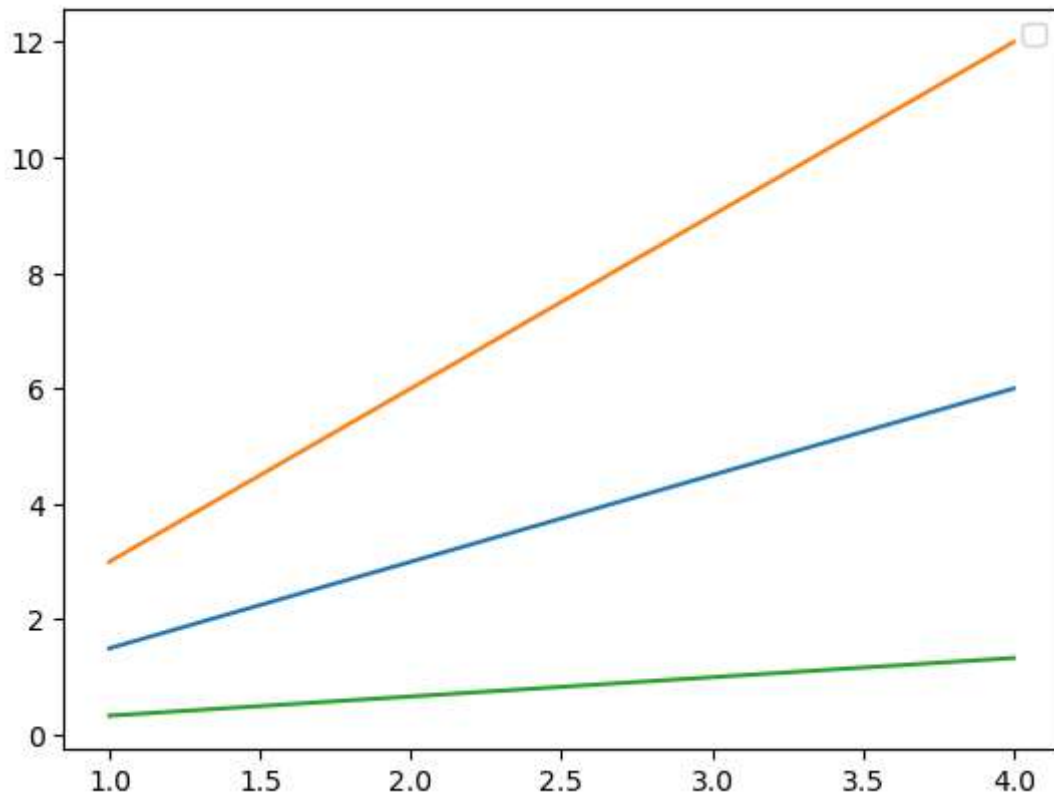
```
In [113... fig, ax = plt.subplots()

ax.plot(x15, x15*1.5)
ax.plot(x15, x15*3.0)
ax.plot(x15, x15/3.0)

ax.legend(['Normal', 'Fast', 'Slow']);

plt.show()
```



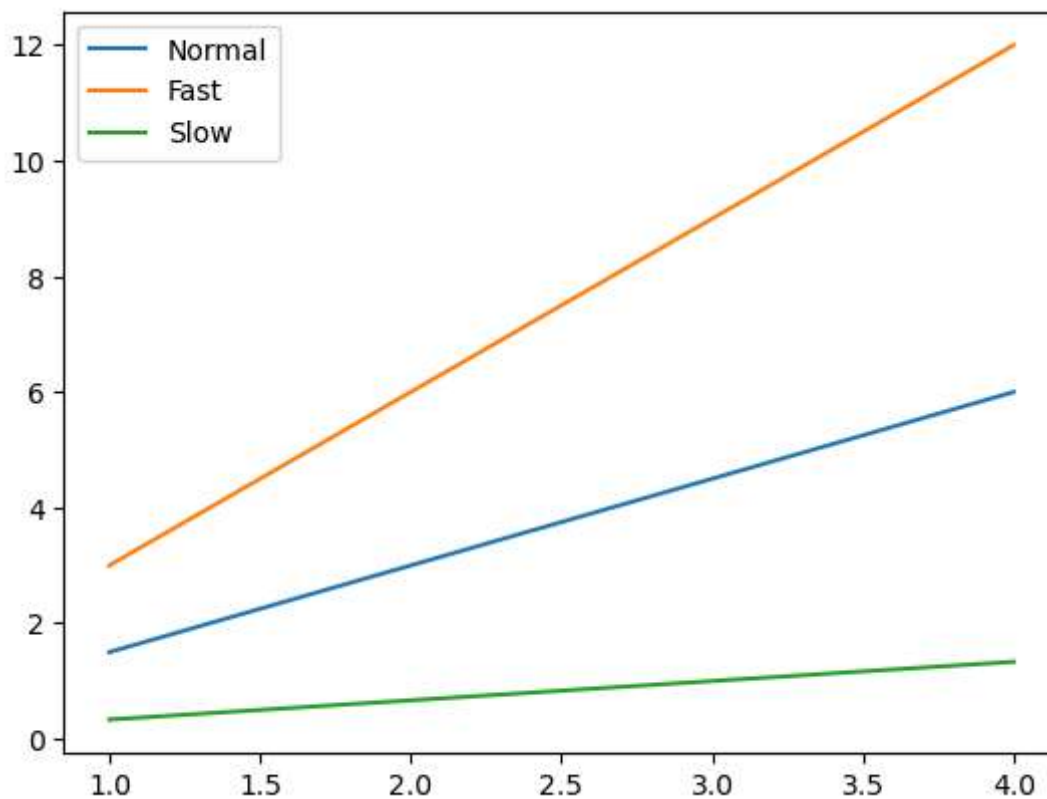
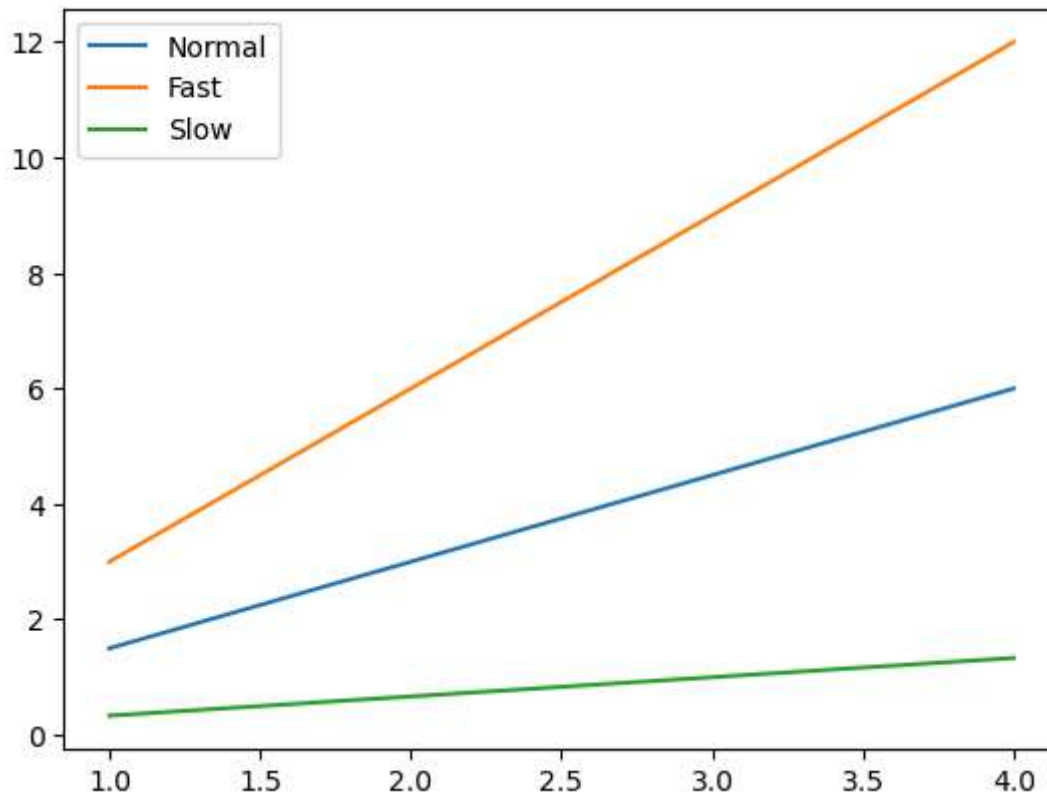


```
In [115... x15 = np.arange(1, 5)

fig, ax = plt.subplots()

ax.plot(x15, x15*1.5, label='Normal')
ax.plot(x15, x15*3.0, label='Fast')
```

```
ax.plot(x15, x15/3.0, label='Slow')  
  
ax.legend();  
  
plt.show()
```

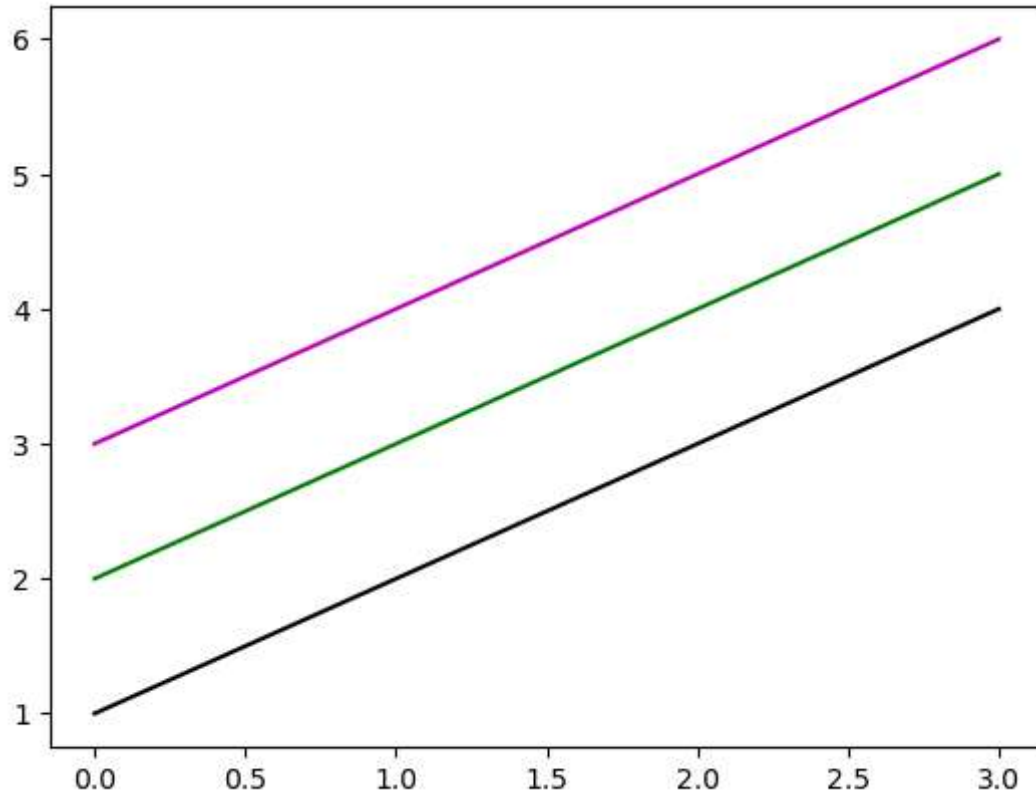


Control colours

```
In [118... x16 = np.arange(1, 5)

plt.plot(x16, 'k')
plt.plot(x16+1, 'g')
plt.plot(x16+2, 'm')

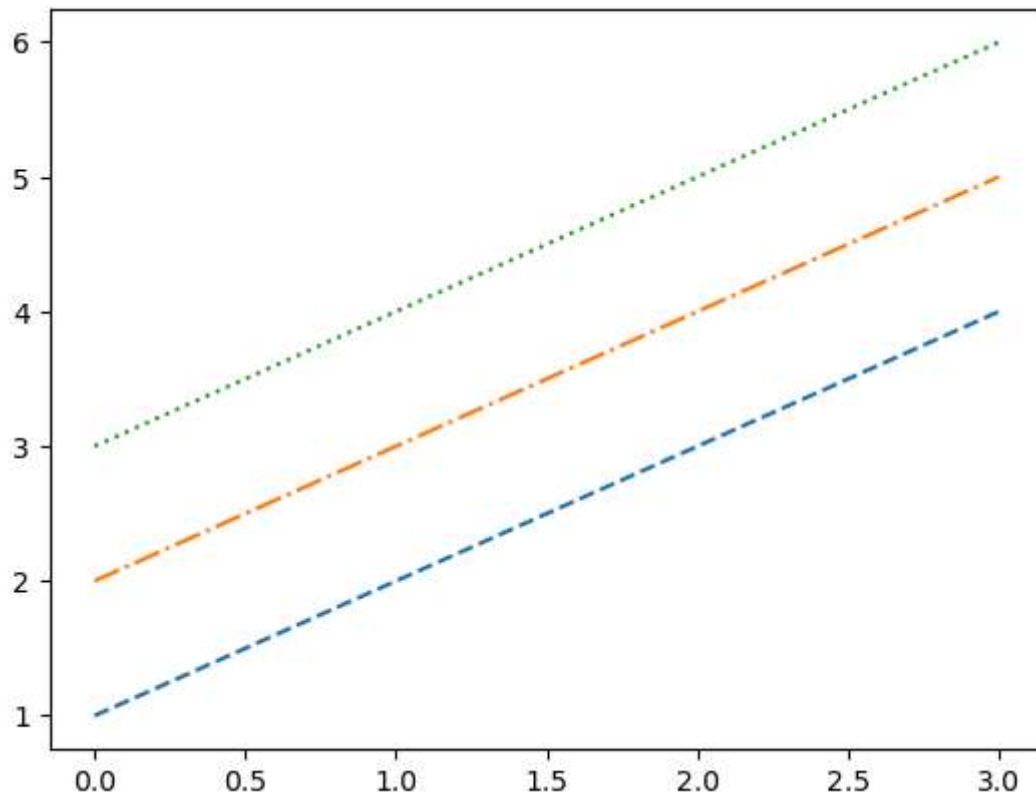
plt.show()
```



Control line styles

```
In [119... x16 = np.arange(1, 5)

plt.plot(x16, '--', x16+1, '-.', x16+2, ':')
plt.show()
```



In []: