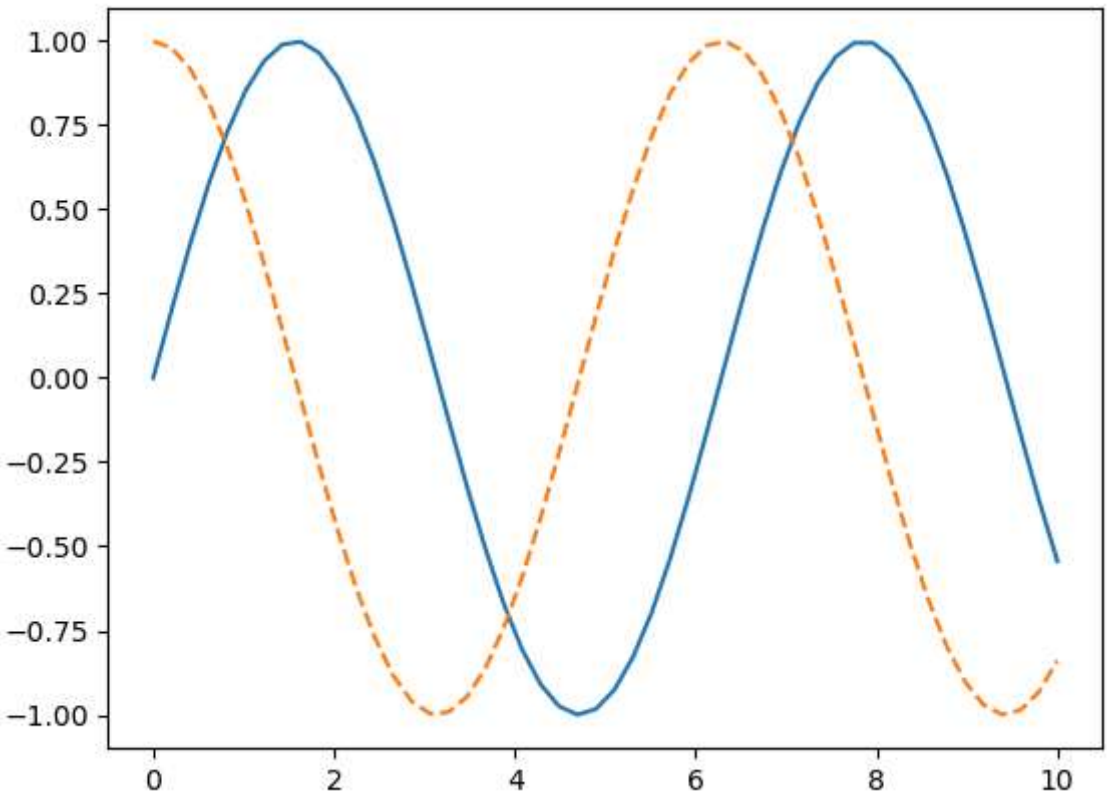
MATPLOTLIB

In [1]:
```
# Import Dependencies
import numpy as np
import pandas as pd
```

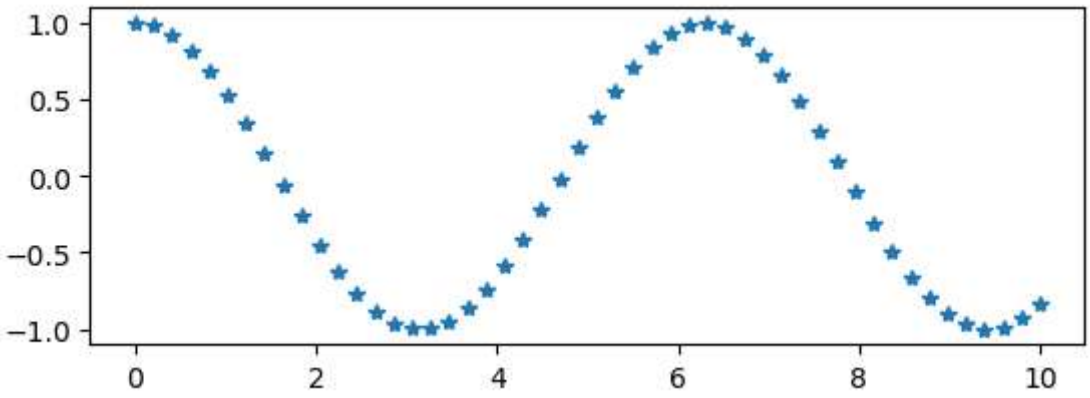
In [2]:
```
#import matplotlib
import matplotlib.pyplot as plt
```

In [6]:
```
#Displaying Plots in Matplotlib
%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()
```

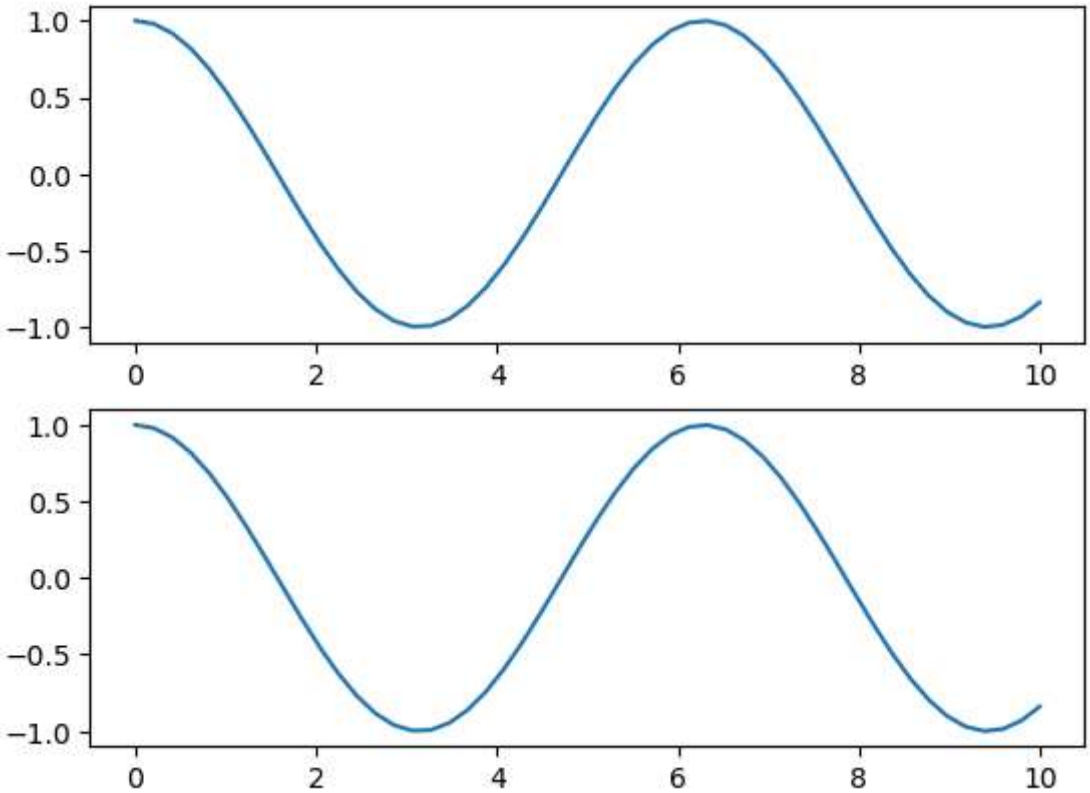


Pyplot API

In [12]:
```
# create the first of two panels and set current axis
plt.subplot(2, 1, 1) #(rows, columns,panel number)
plt.plot(x1, np.cos(x1),'*')
plt.show()
```



In [17]:
```
#create a plot figure
plt.figure()

#create the first of two panels and set current axis
plt.subplot(2, 1, 1) # (rows, columns, panel number)
plt.plot(x1, np.cos(x1))


# create the second of two panels and set current axis
plt.subplot(2, 1, 2)  # (rows, columns, panel number)
plt.plot(x1, np.cos(x1));
plt.show()
```

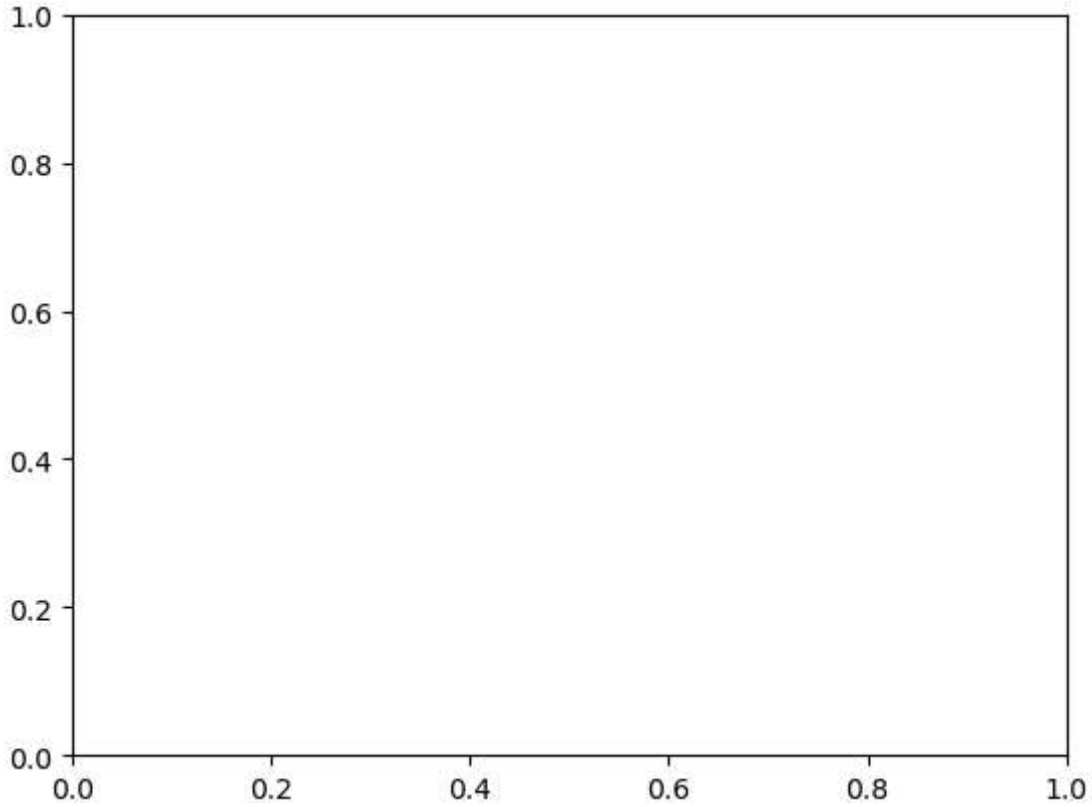


In [18]:
```
#get current figure information

print(plt.gcf())
```

```
Figure(640x480)
```

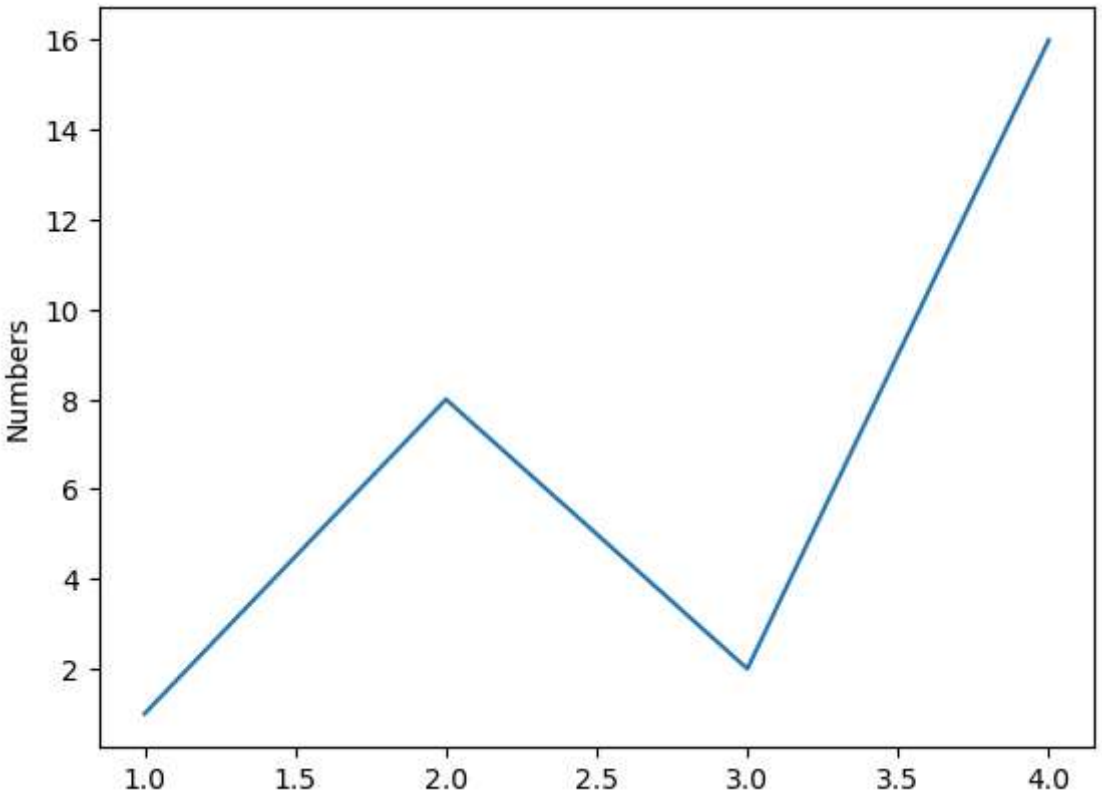
In [20]:
```
# get current axis information

print(plt.gca())
plt.show()
```

```
Axes(0.125,0.11;0.775x0.77)
```



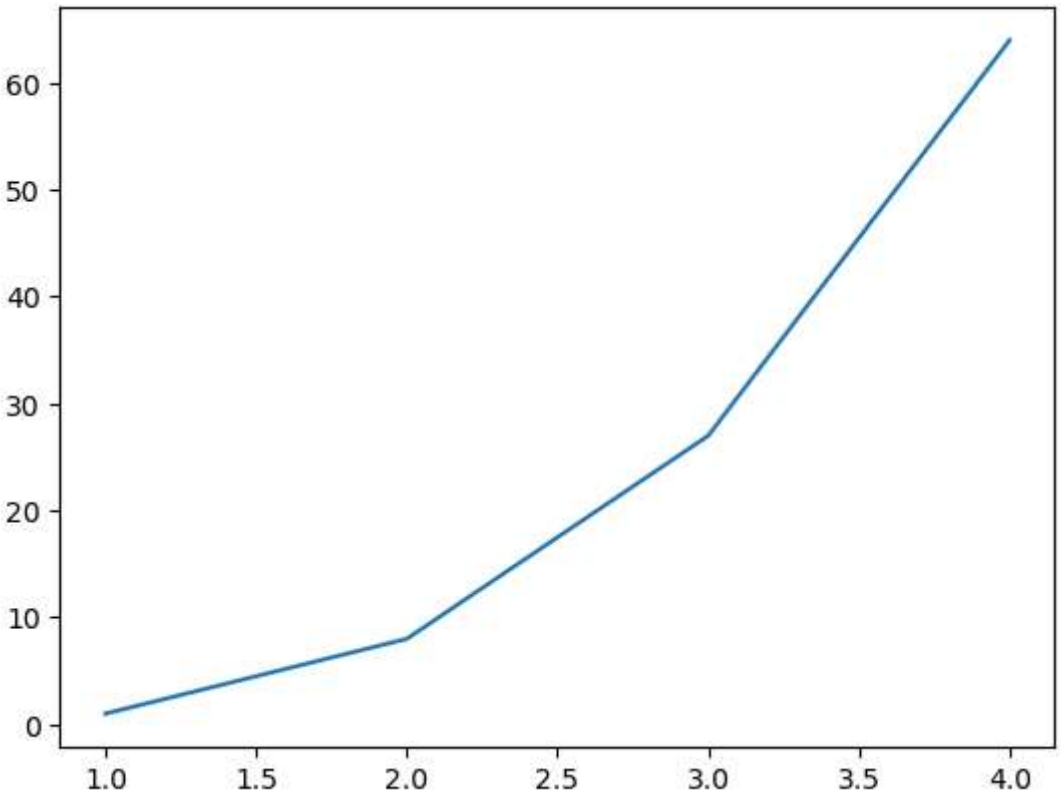
Visualization with Pyplot

In [21]:
```
plt.plot([1,2,3,4],[1,8,2,16])
plt.ylabel('Numbers')
plt.show()
```



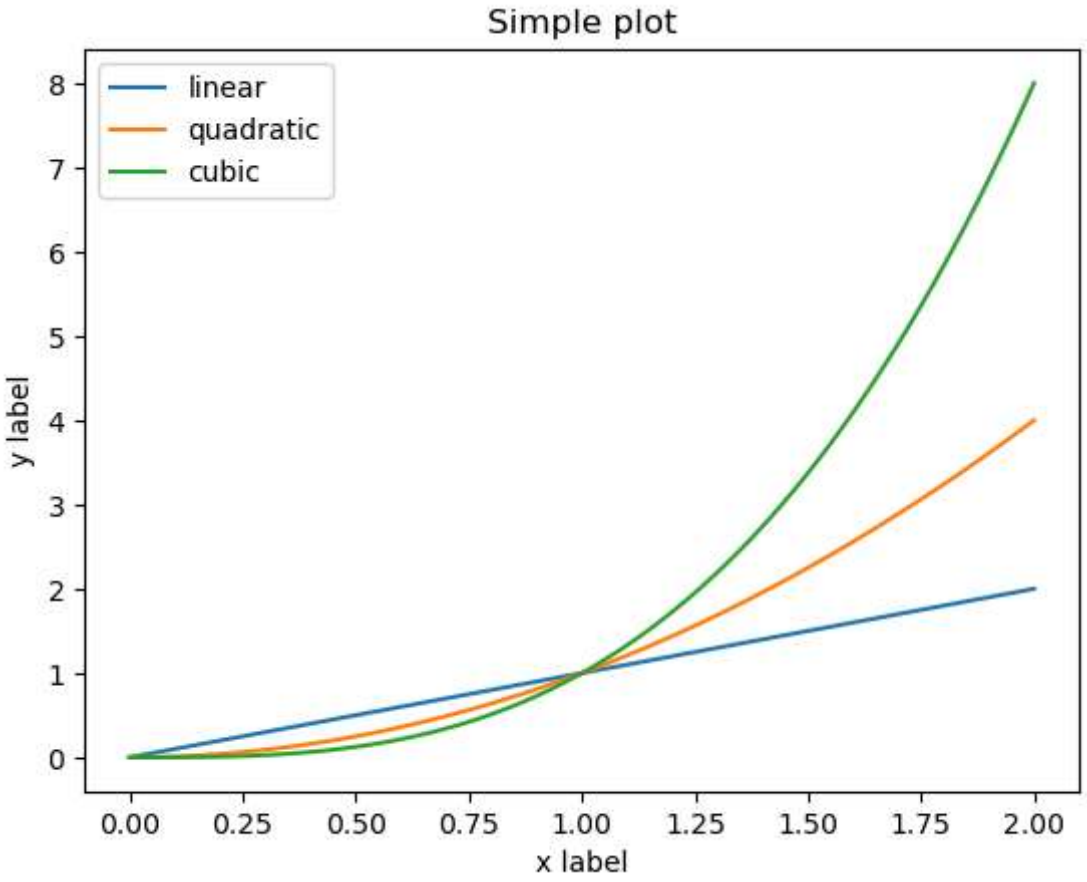
plot() - A versatile command

In [22]:
```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4],[1,8,27,64])
plt.show()
```

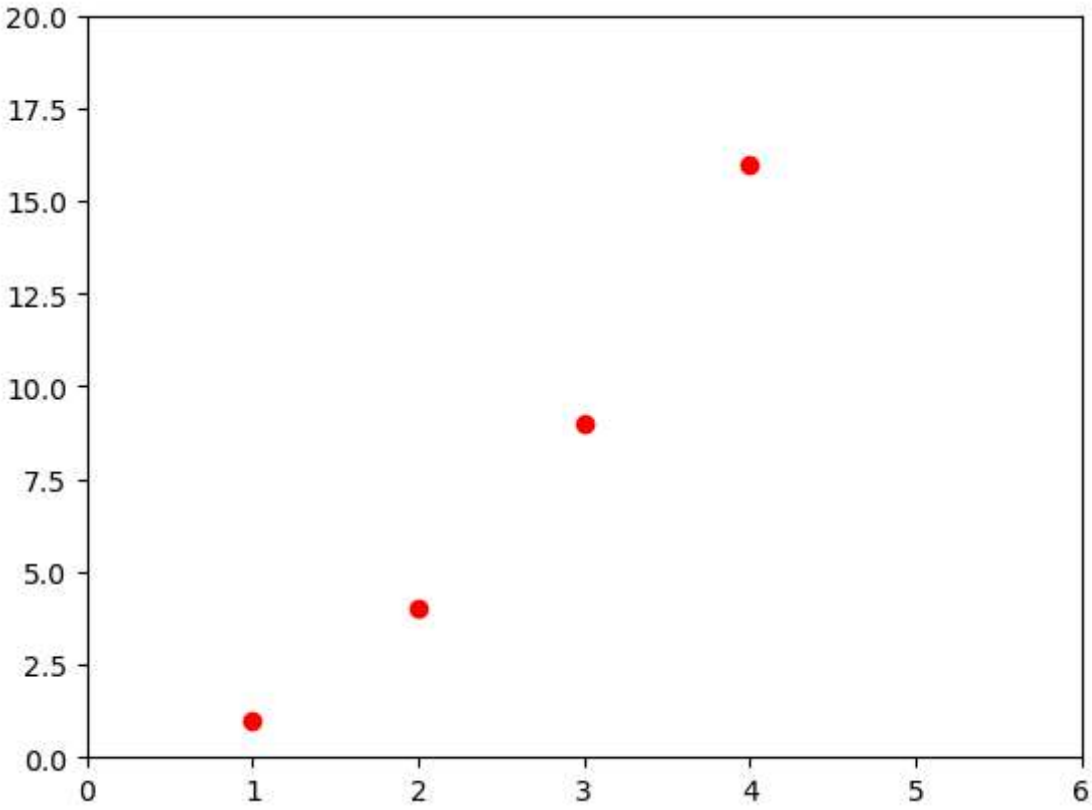


State-machine interface

In [23]:
```
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()
```

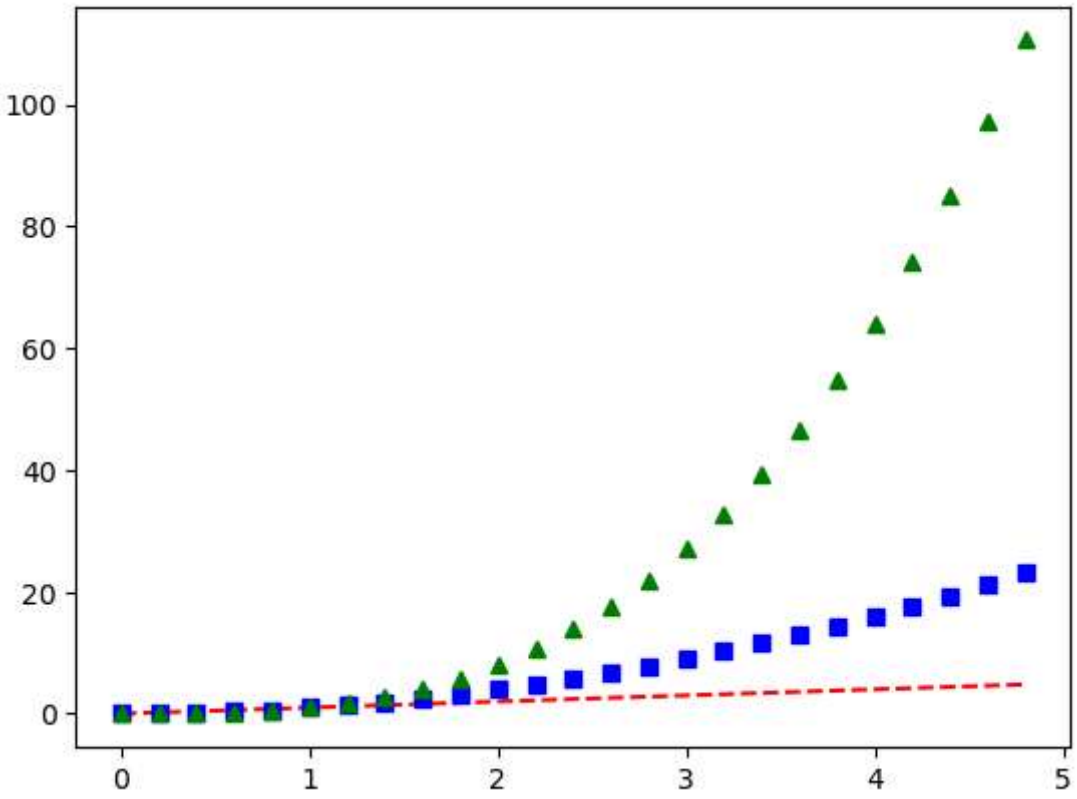


In [24]:
```
plt.plot([1,2,3,4],[1,4,9,16],'ro')
plt.axis([0,6,0,20])
plt.show()
```



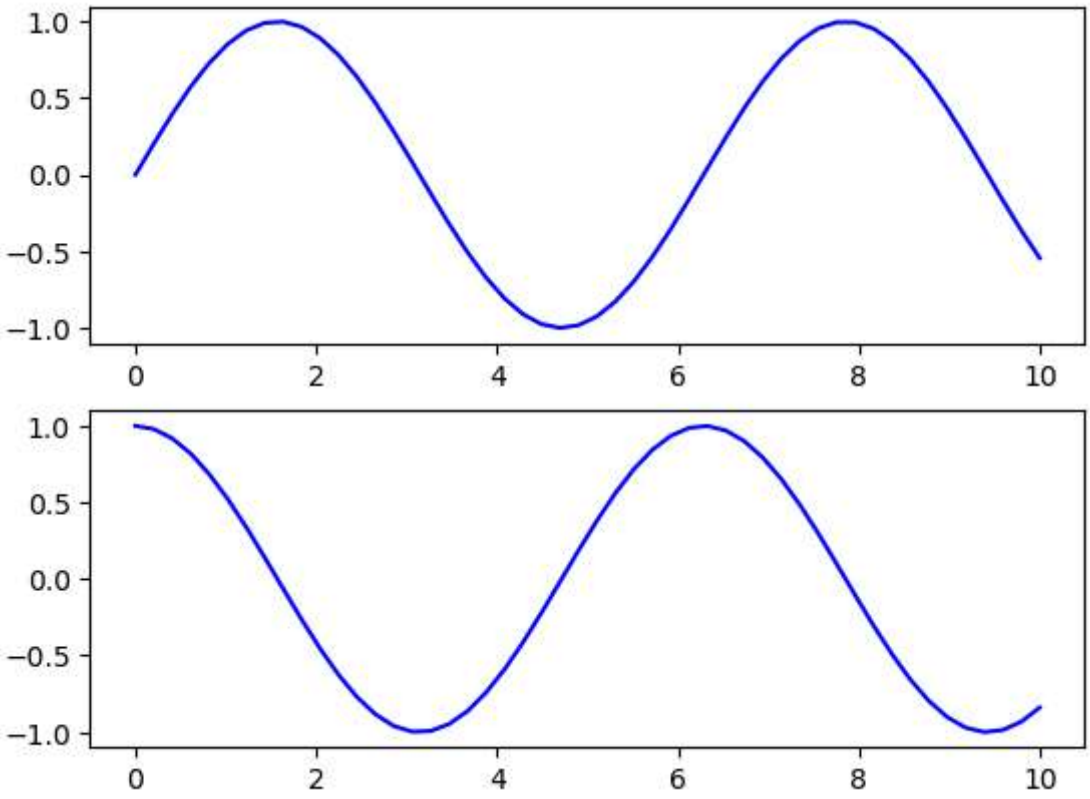
Working with NumPy arrays

In [25]:
```
# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



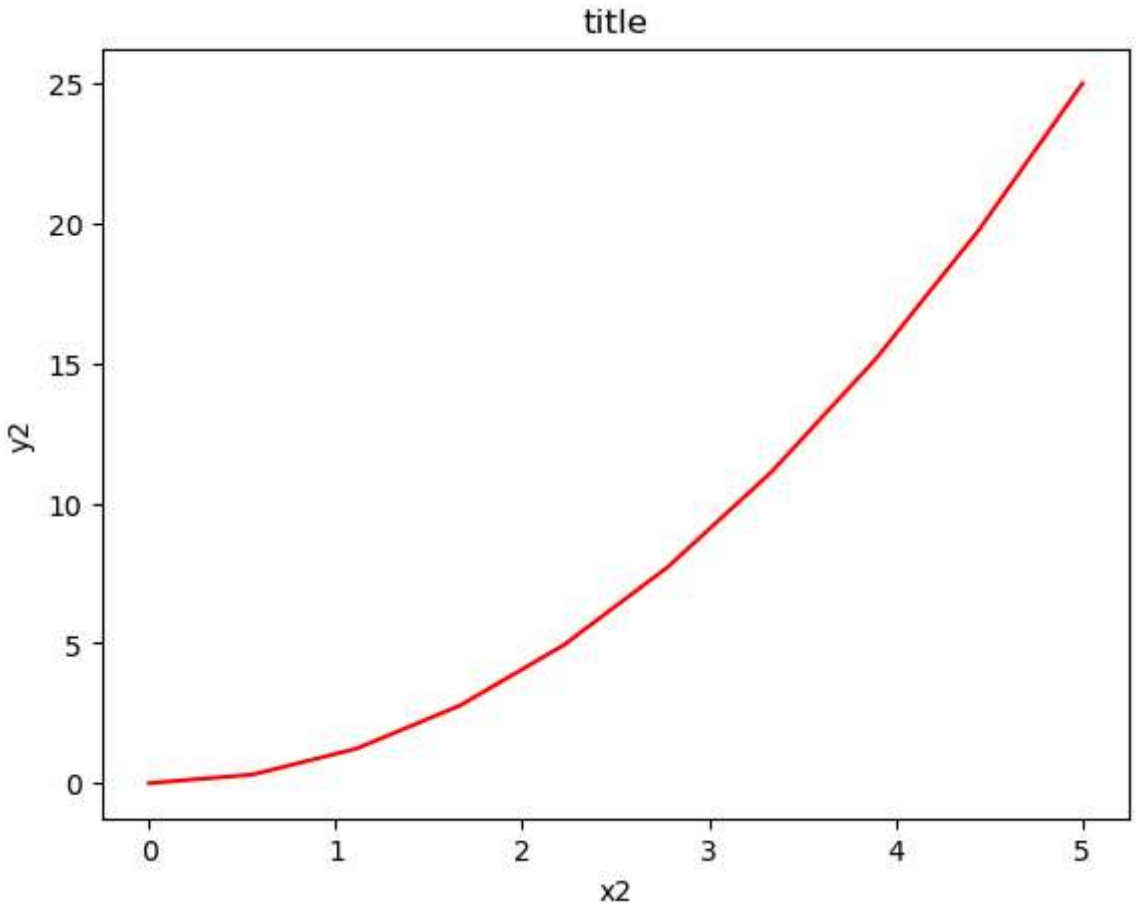
Object-Oriented API

In [28]:
```
 #First create a grid of plots
# ax will be an array of two Axes objects
fig, ax=plt.subplots(2)

#call plot() method on the appropriate object
ax[0].plot(x1, np.sin(x1), 'b-')
ax[1].plot(x1, np.cos(x1), 'b-');
plt.show()
```



Objects and Reference

In [29]:
```
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()
```



In [32]:
```
fig = plt.figure()

ax = plt.axes()

plt.show()
```

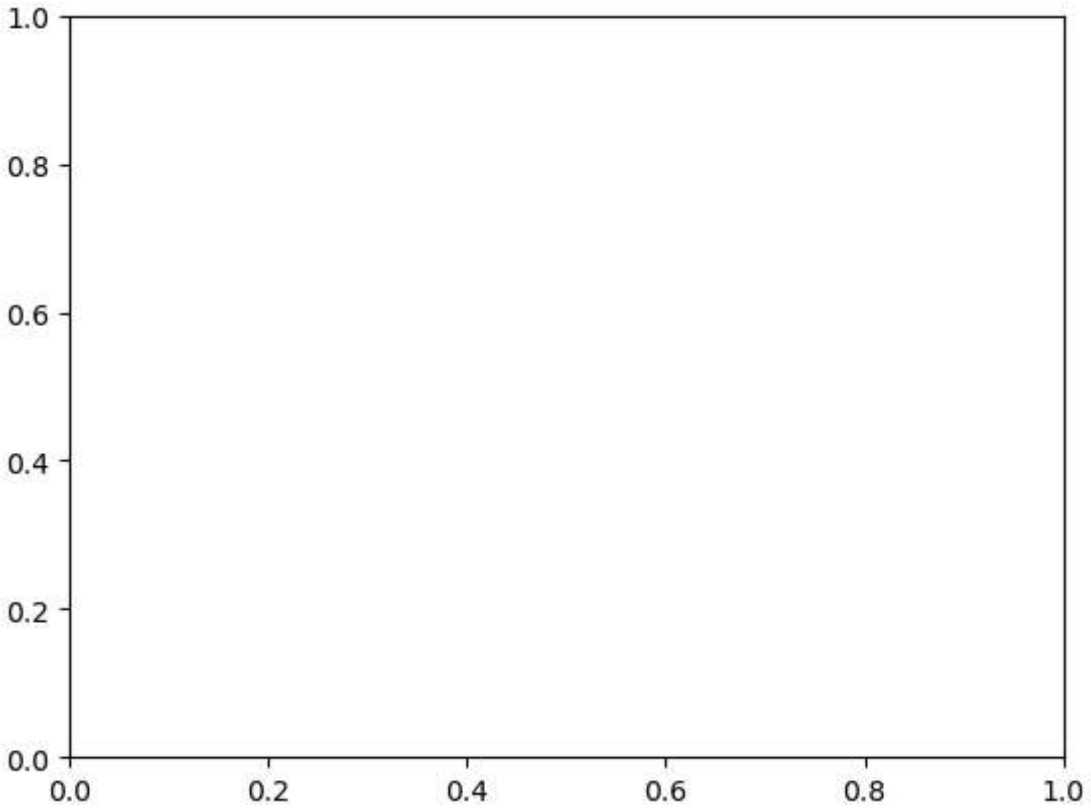
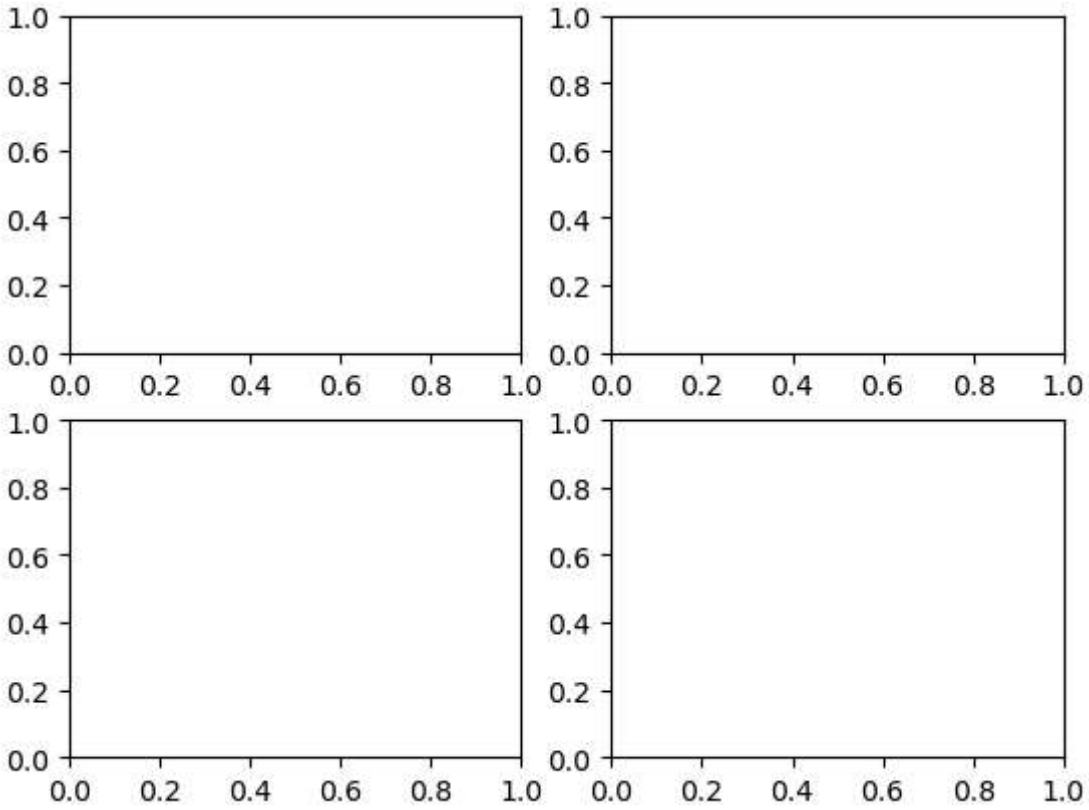


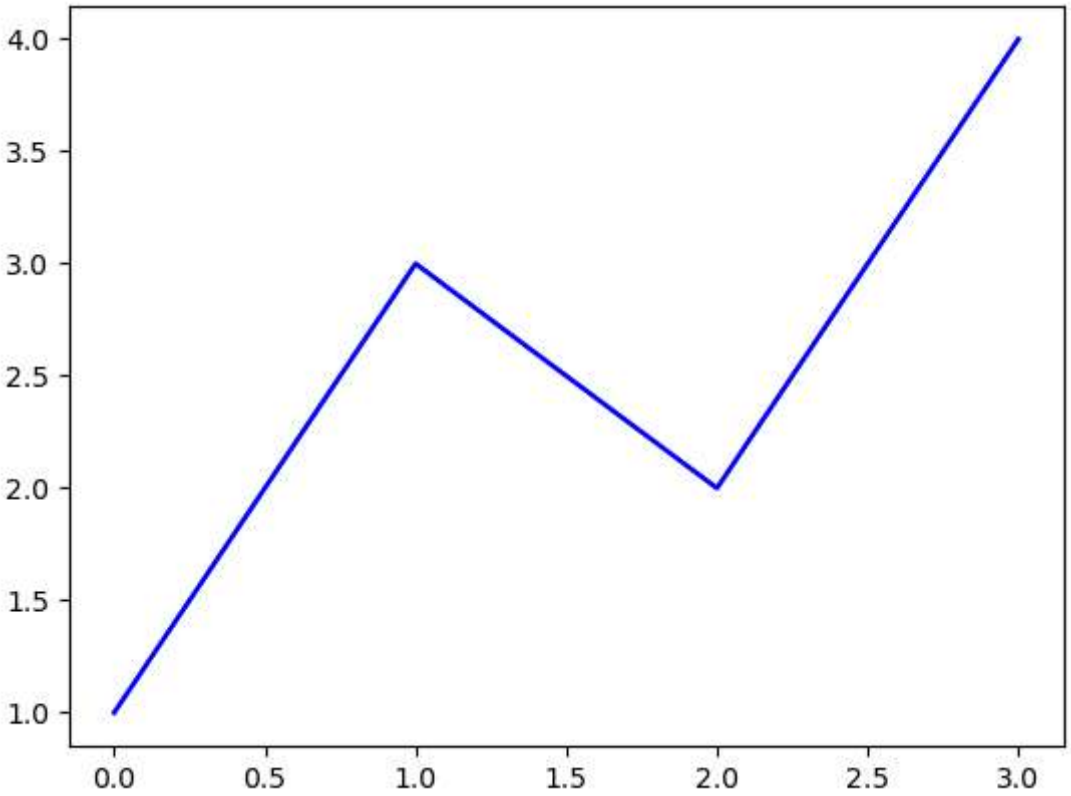
Figure and Subplots

In [33]:
```
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()
```



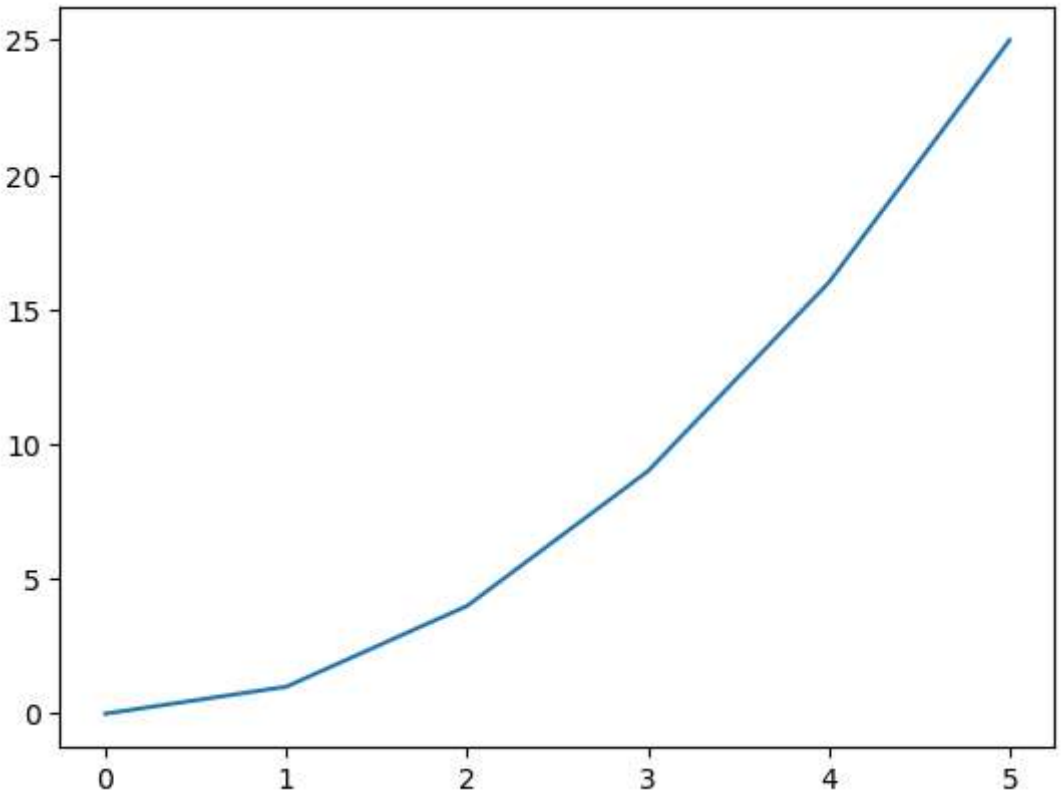
First plot with Matplotlib

In [35]:
```
plt.plot([1,3,2,4],'b-')
plt.show()
```

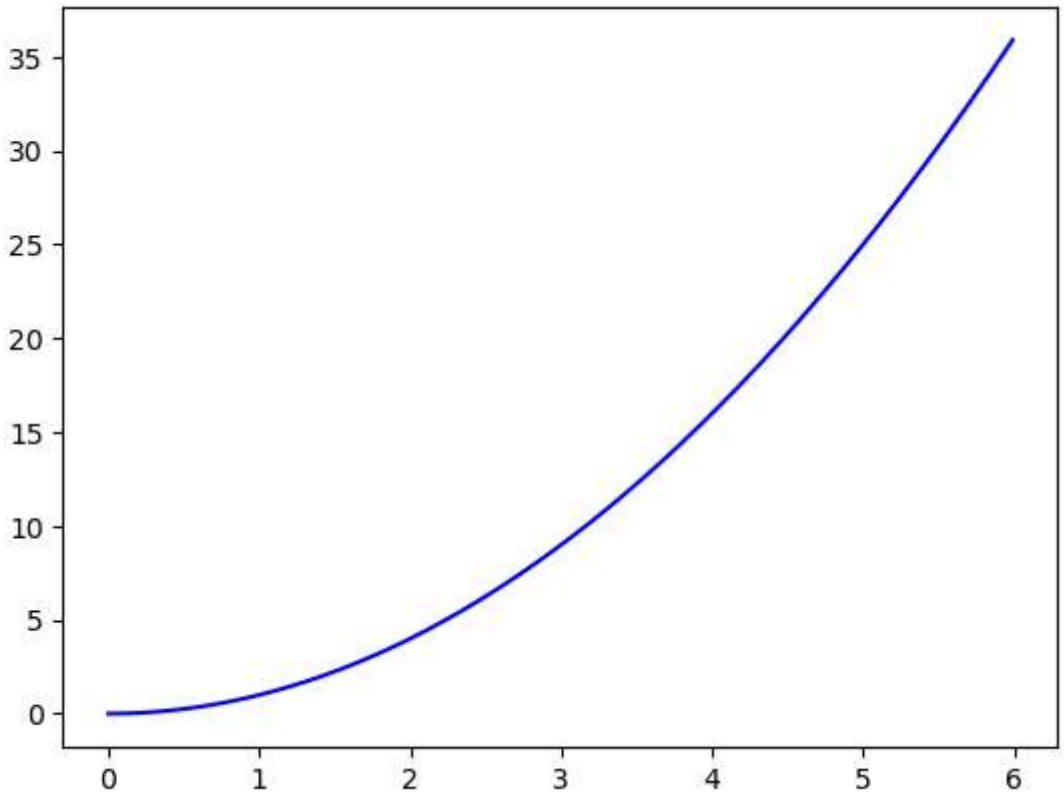


Specify both Lists

In [36]:
```
x3 = range(6)

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

plt.show()
```

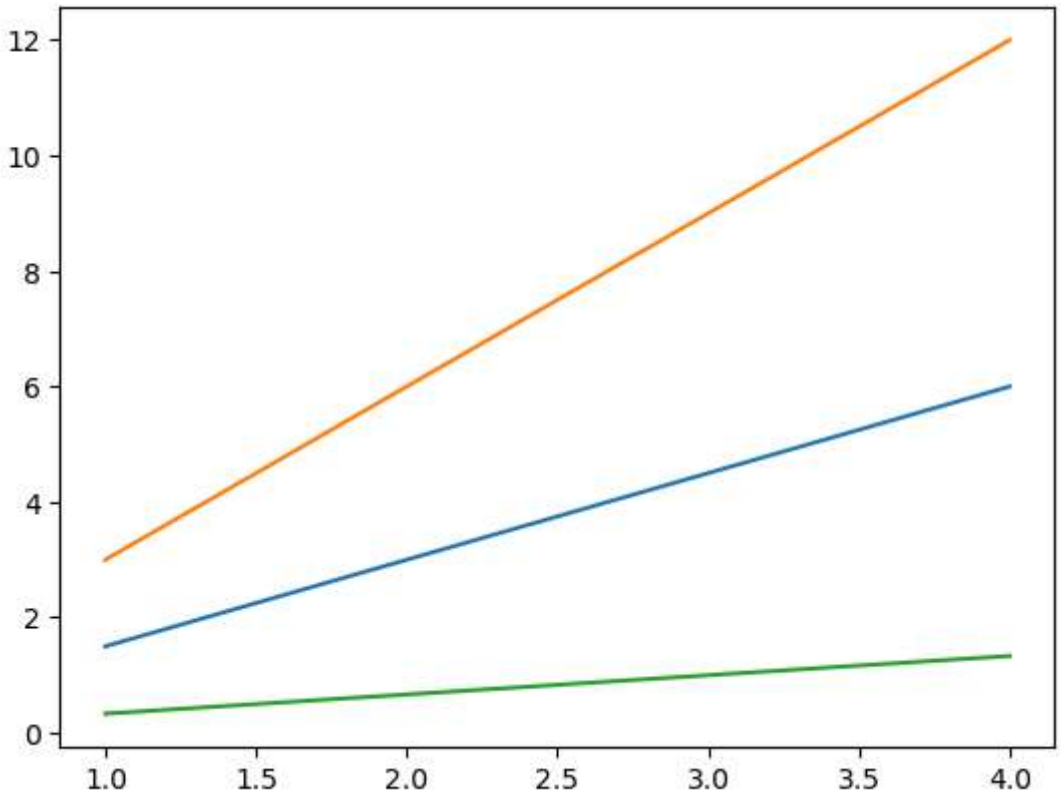


In [39]:
```
x3 = np.arange(0.0, 6.0, 0.01)

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

plt.show()
```



Multiline Plots

In [61]:
```
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()
```

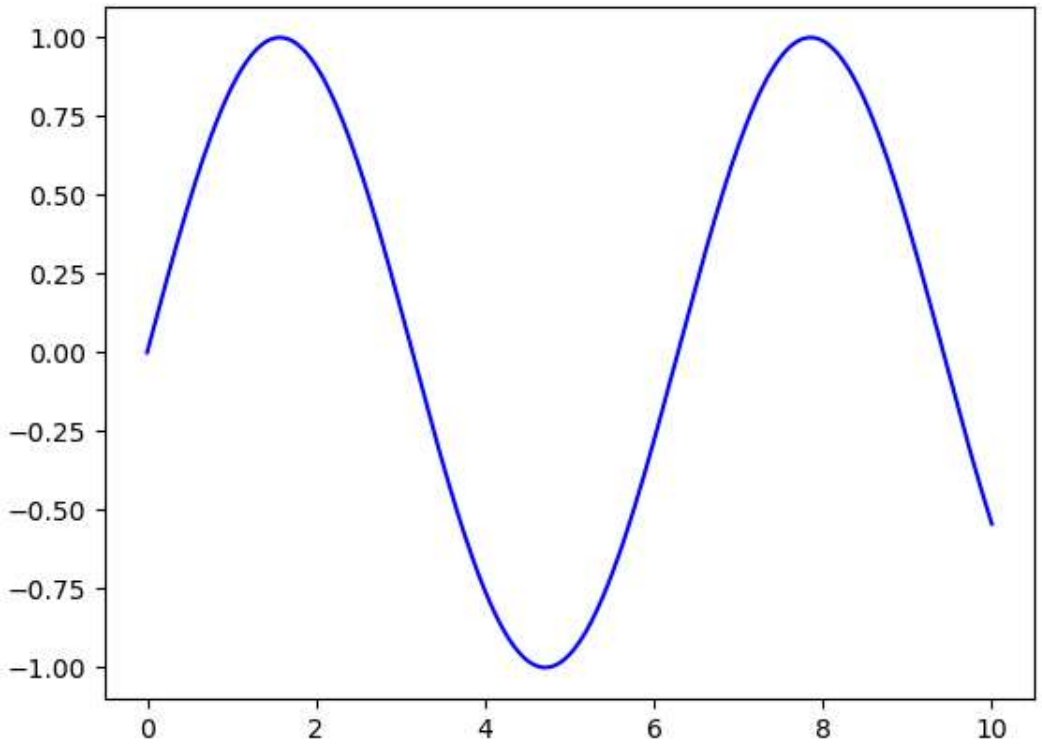


Saving the Plot

In [65]:
```
# Saving the figure

fig.savefig('plot1.png')
```

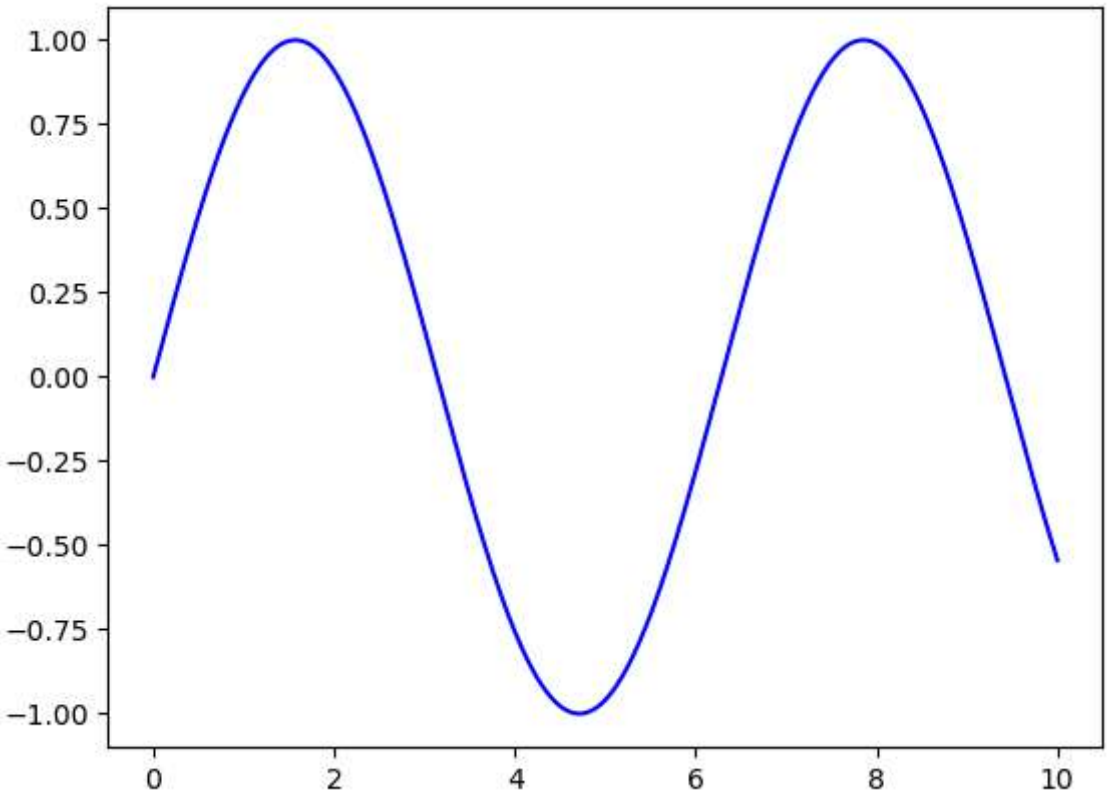
In [66]:
```
# Explore the contents of figure

from IPython.display import Image

Image('plot1.png')
```

Out[66]:



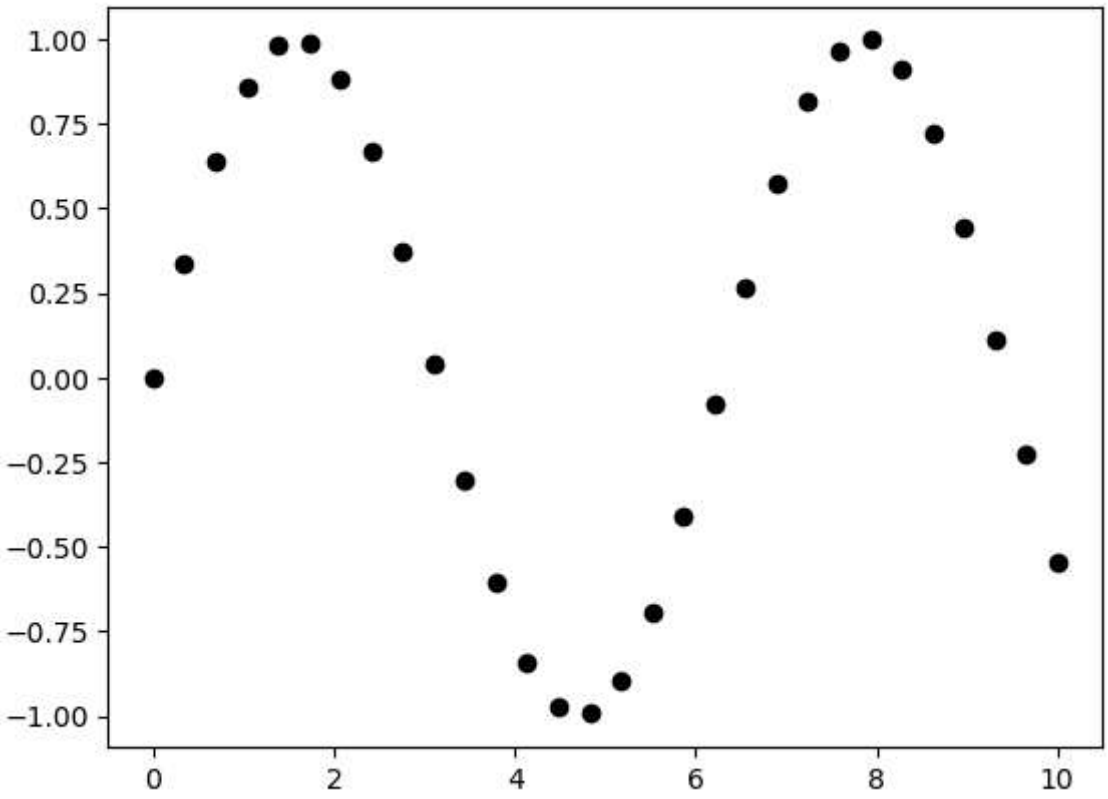
In [ ]:
```
Line plot
```

In [47]:
```
#create figure and axes first
fig = plt.figure()
ax = plt.axes()
#declarea variable x5
x5=np.linspace(0,10,1000)
#plot the sinusoid function
ax.plot(x5,np.sin(x5), 'b-');
plt.show()
```



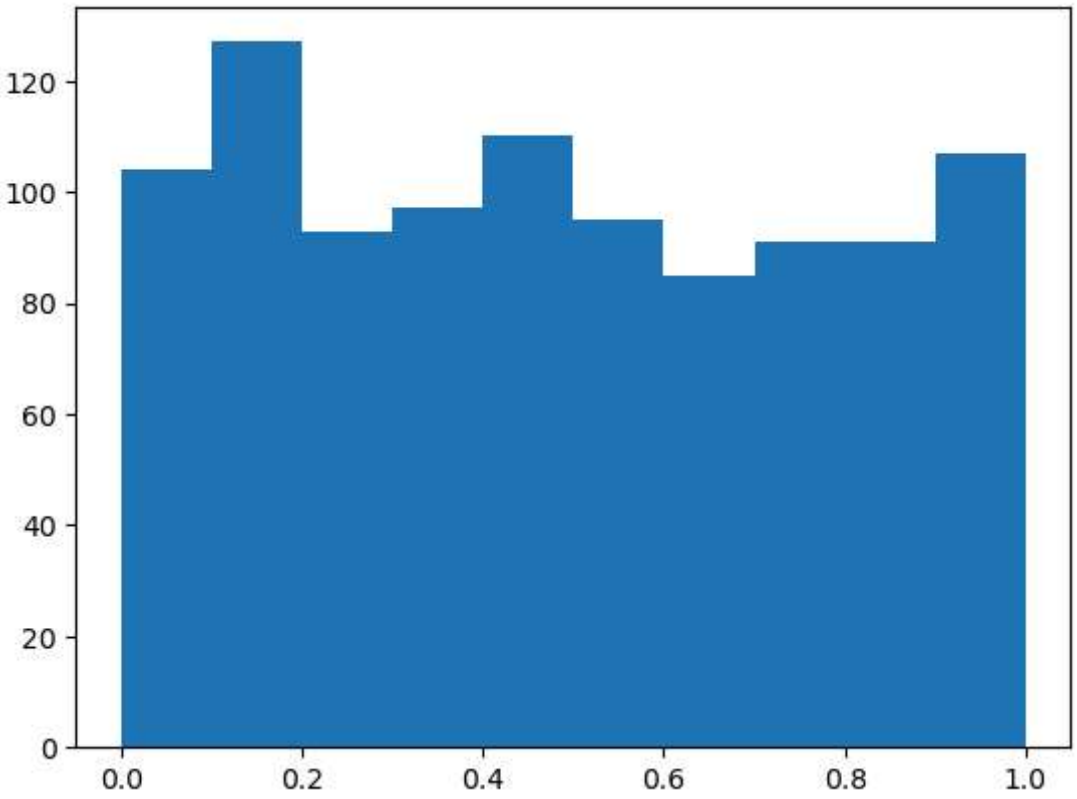
Scatter Plot

In [48]:
```
#Scatter Plot with plt.plot()

x7=np.linspace(0,10,30)
y7=np.sin(x7)
plt.plot(x7,y7,'o', color='black');
plt.show()
```



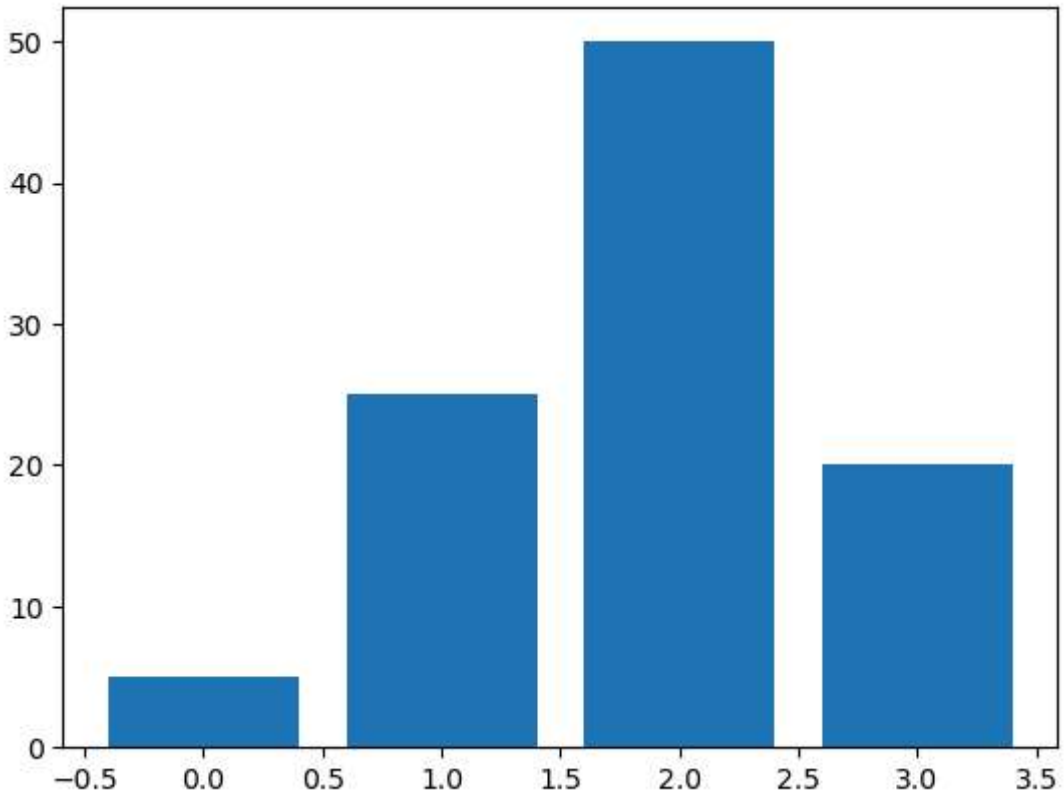
Histogram

In [56]:
```
data1 = np.random.rand(1000)
plt.hist(data1);
plt.show()
```



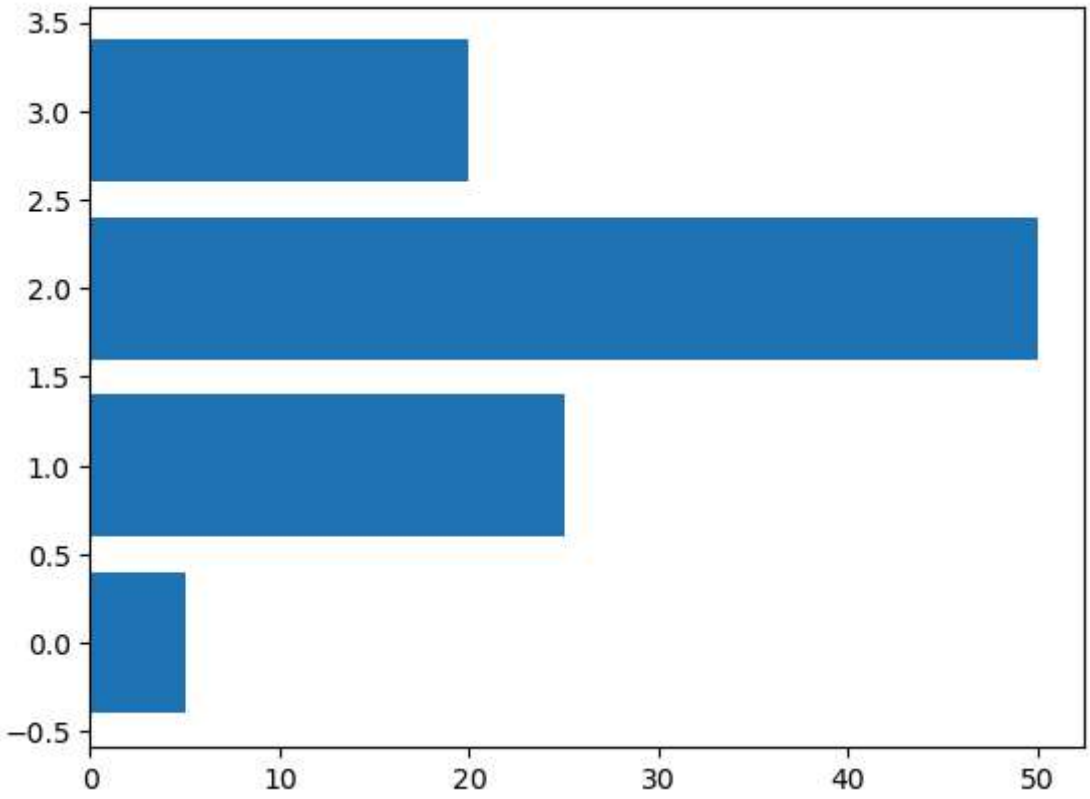
Bar Chart

In [57]:
```
data2 = [5. ,25. ,50. , 20.]
plt.bar(range(len(data2)), data2)
plt.show()
```



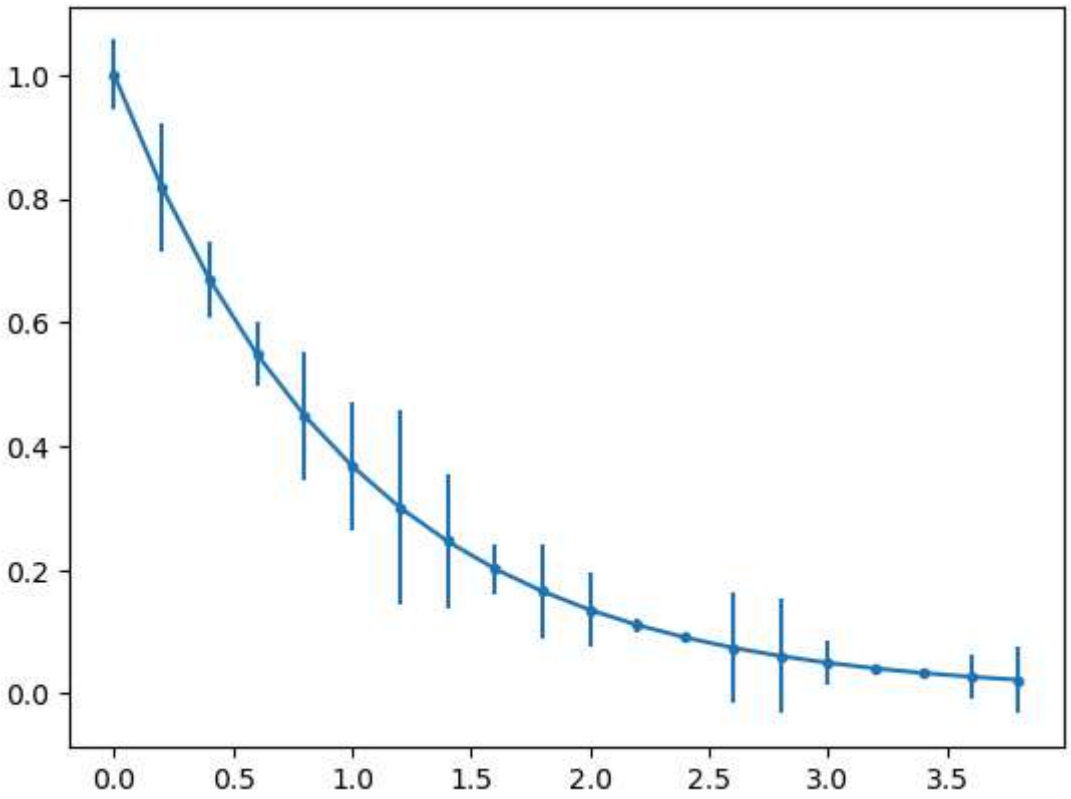
Horizontal Bar Chart

In [58]:
```
data2 = [5. ,25. ,50. , 20.]
plt.barh(range(len(data2)), data2)
plt.show()
```



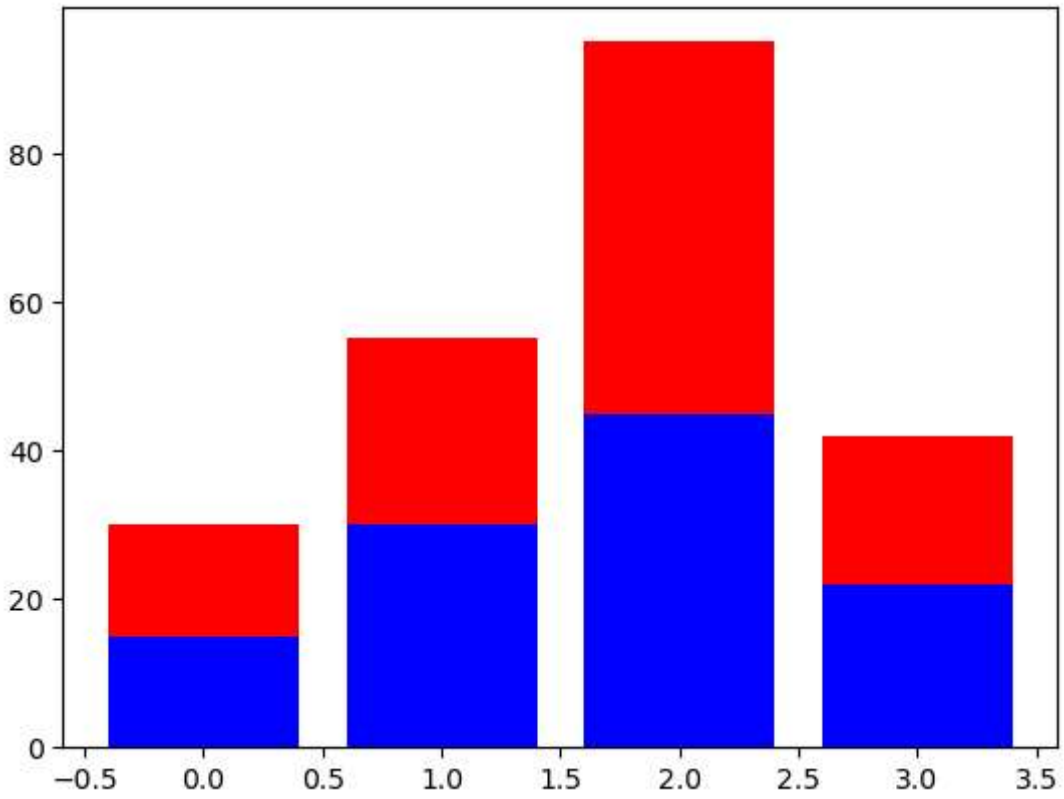
Error Bar Chart

In [68]:
```
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();
```



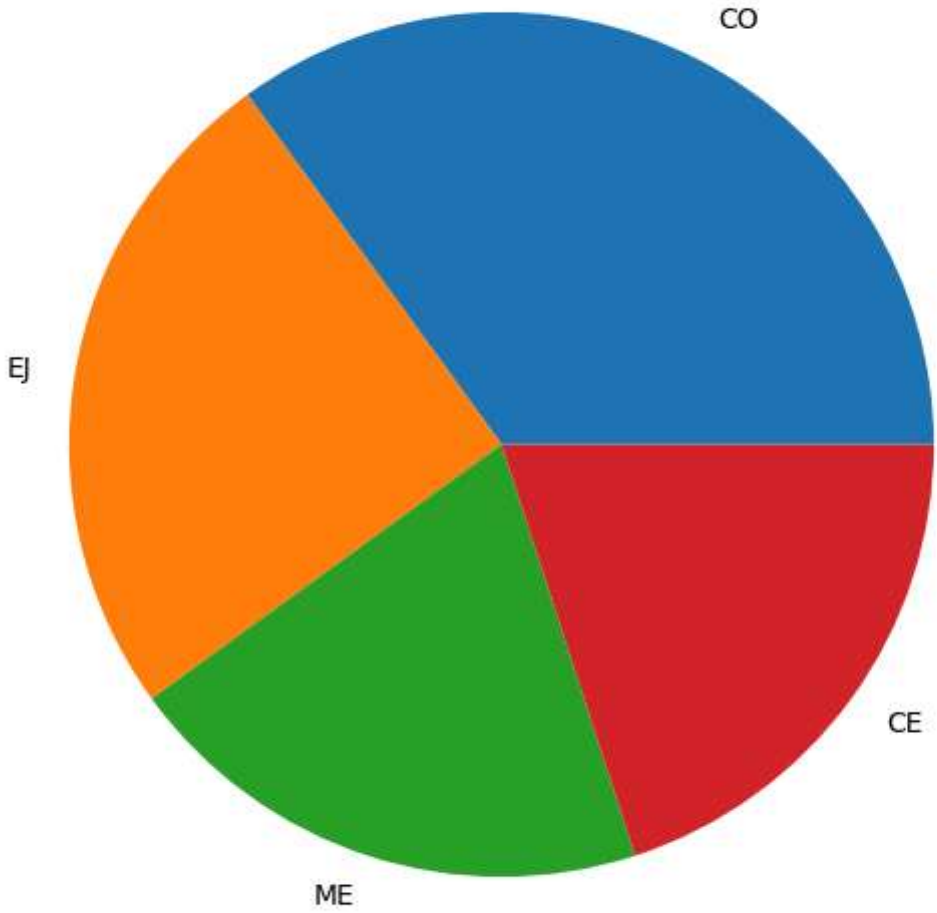
Stacked Bar Chart

In [69]:
```
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()
```



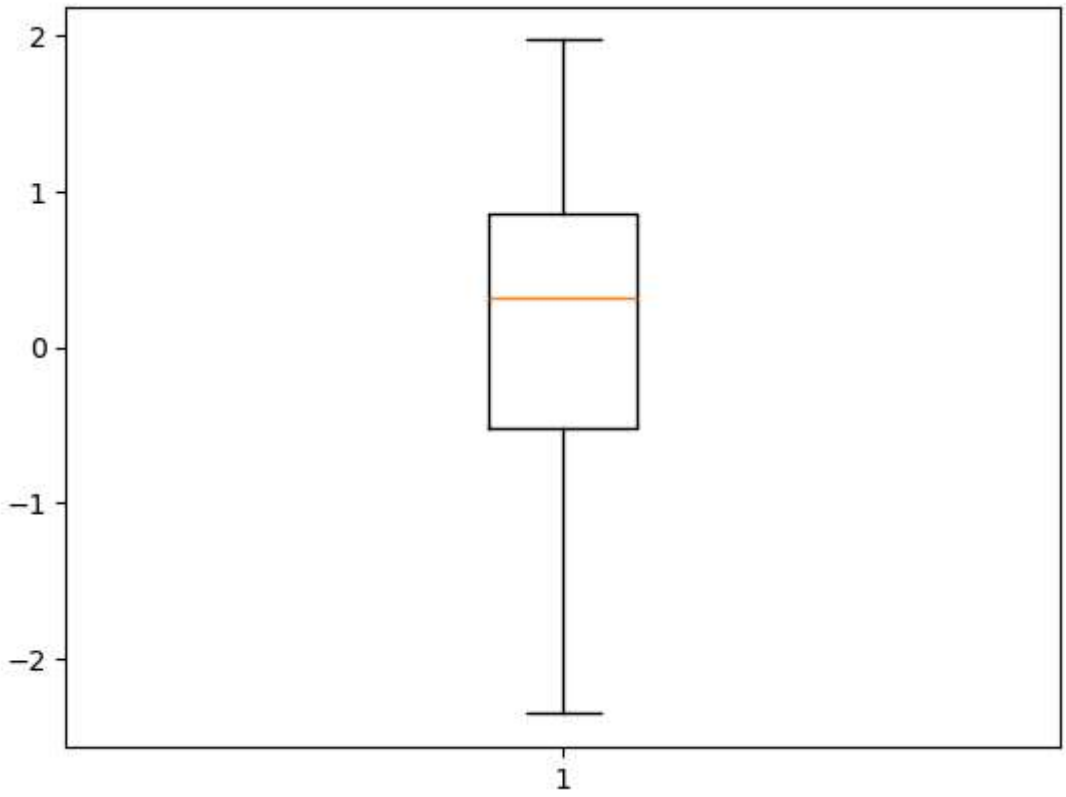
Pie Chart

In [70]:
```
plt.figure(figsize=(7,7))
x10=[35,25,20,20]
labels=['CO','EJ','ME','CE']
plt.pie(x10, labels=labels);
plt.show()
```



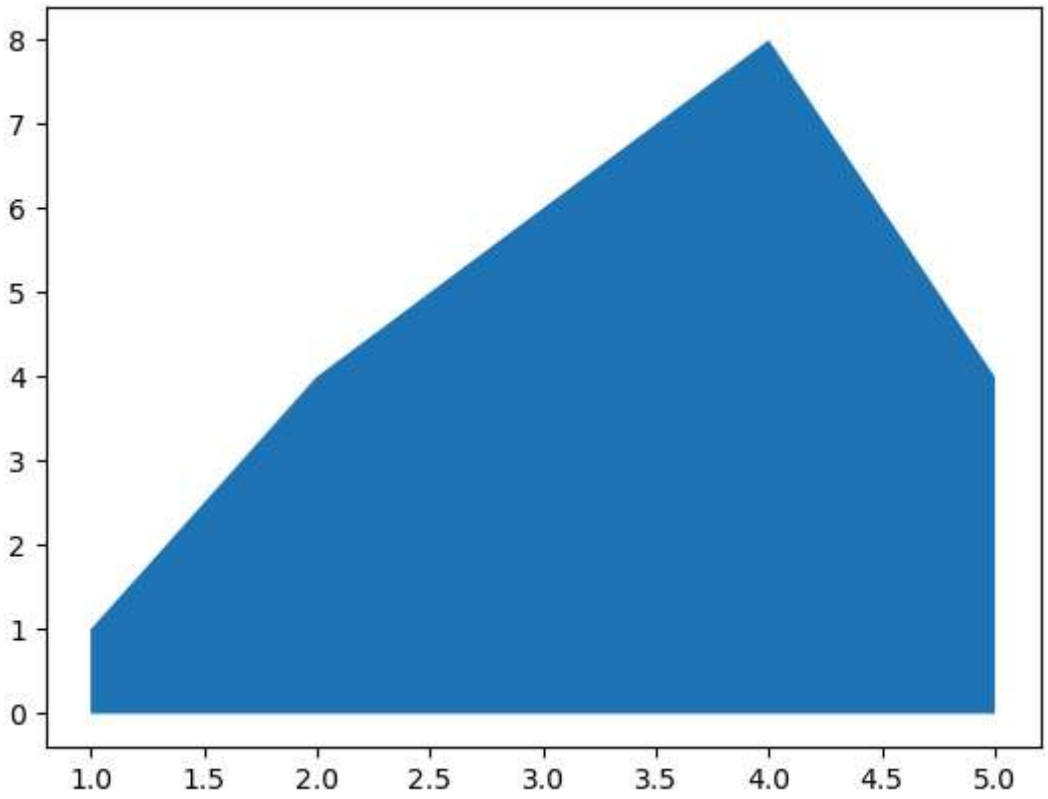
Boxplot

In [72]:
```
data3=np.random.randn(100)
plt.boxplot(data3)
plt.show();
```



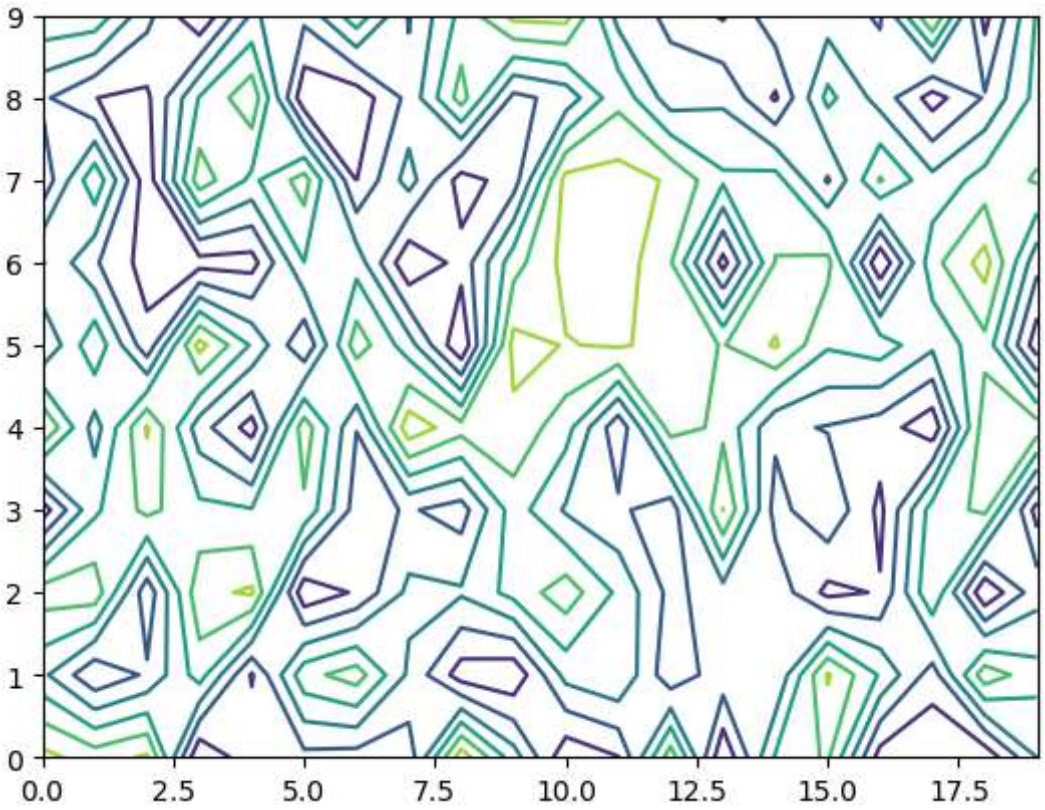
Area Chart

In [73]:
```
#create some data
x12=range(1,6)
y12=[1,4,6,8,4]
#Area plot
plt.fill_between(x12,y12)
plt.show()
```



Contour Plot

In [74]:
```
#create a matrix
matrix1=np.random.rand(10,20)
cp=plt.contour(matrix1)
plt.show()
```



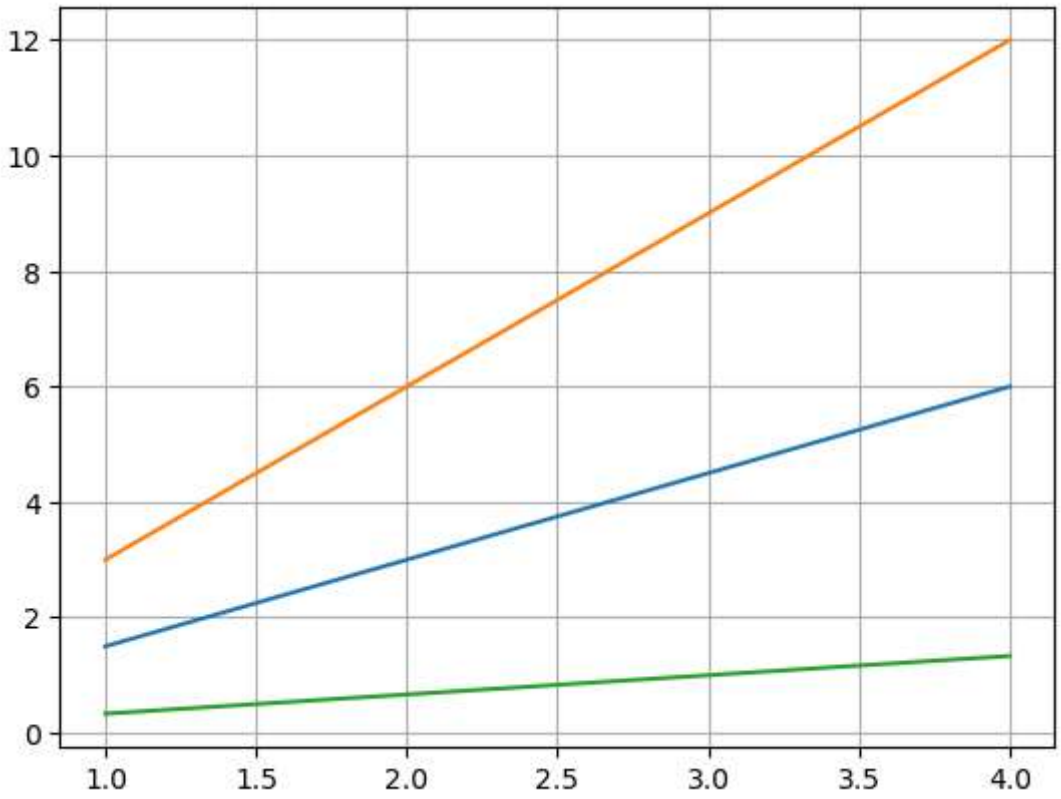
Styles with Matplotlib Plots

In [75]:
```
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-pastel', 'seaborn-v0_8-poster', 'seaborn-v0_8-talk', 'seaborn-v0_8-ticks', 'seaborn-v0_8-white', 'seaborn-v0_8-whitegrid', 'tableau-colorblind10']
```

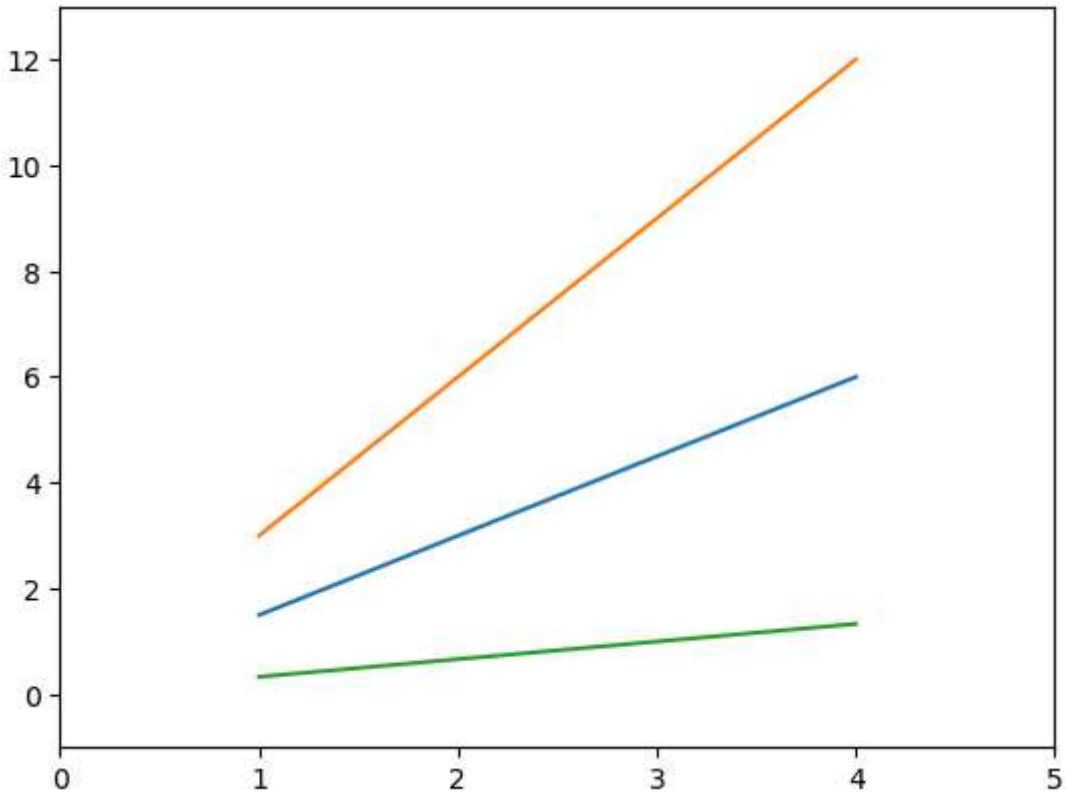
Adding a grid

In [77]:
```
x15=np.arange(1,5)
plt.plot(x15,x15*1.5,x15,x15*3.0,x15,x15/3.0)
plt.grid(True)
plt.show()
```

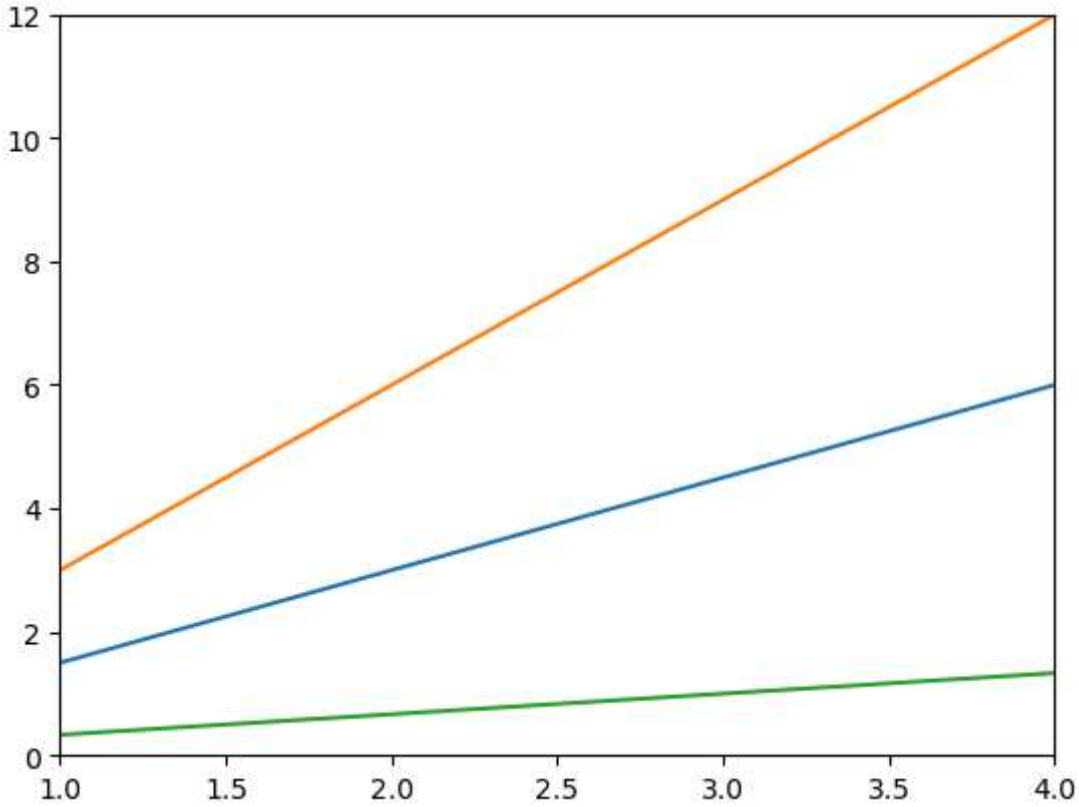


Handling axes

In [78]:
```
x15=np.arange(1,5)
plt.plot(x15,x15*1.5,x15,x15*3.0,x15,x15/3.0)
plt.axis()  #show the current axis limits values
plt.axis([0,5,-1,13])
plt.show()
```

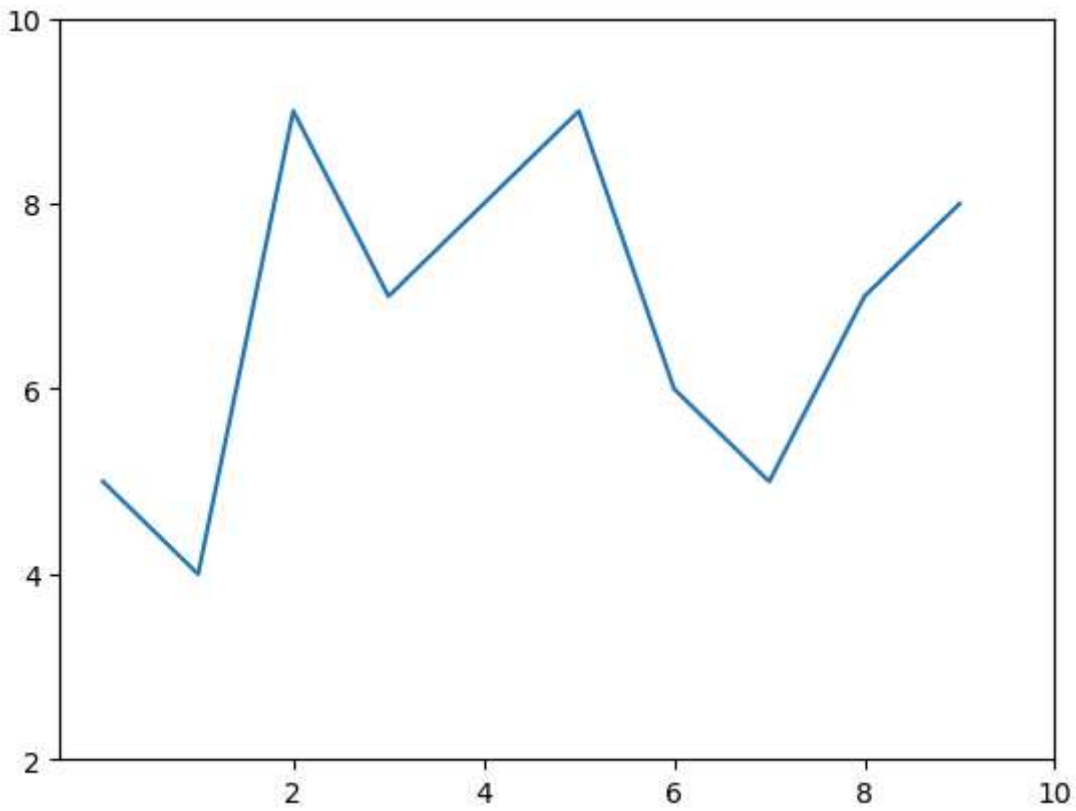


In [79]:
```
x15=np.arange(1,5)
plt.plot(x15,x15*1.5,x15,x15*3.0,x15,x15/3.0)
plt.xlim([1.0,4.0])
plt.ylim([0.0,12.0])
plt.show()
```



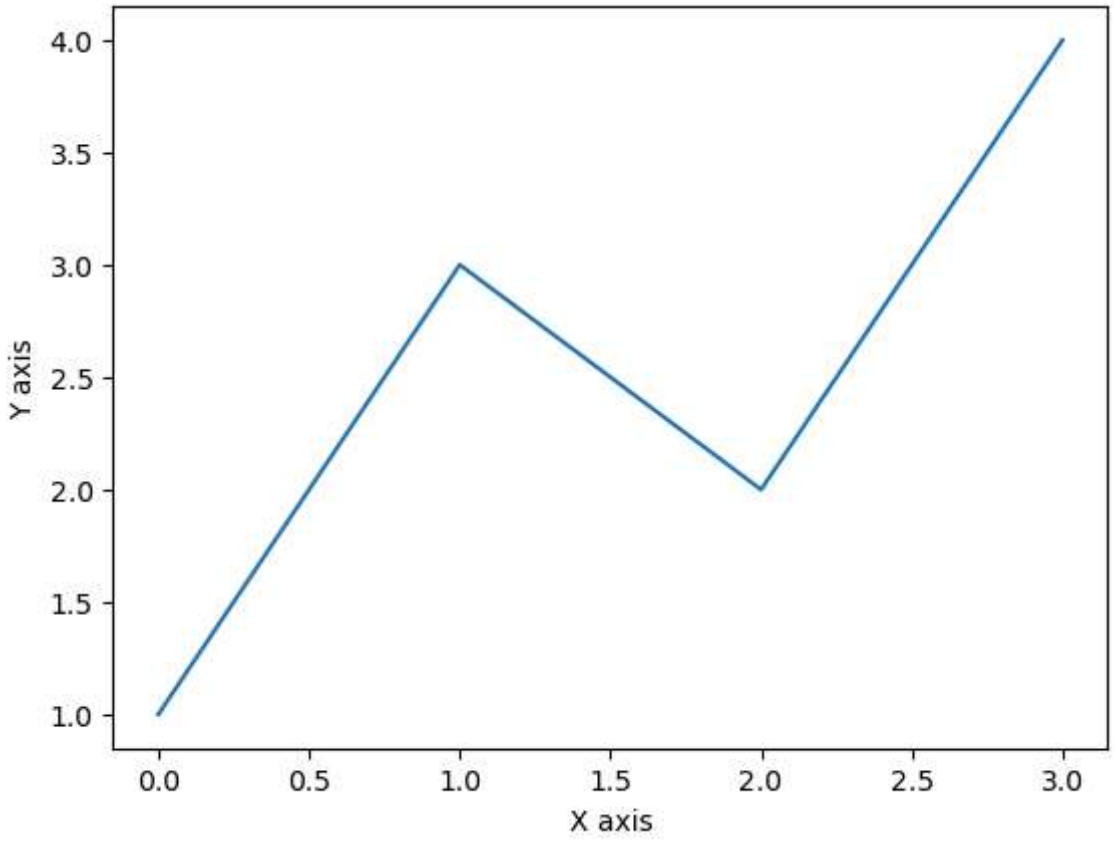
Handling X and Y ticks

In [80]:
```
u=[5,4,9,7,8,9,6,5,7,8]
plt.plot(u)
plt.xticks([2,4,6,8,10])
plt.yticks([2,4,6,8,10])
plt.show()
```



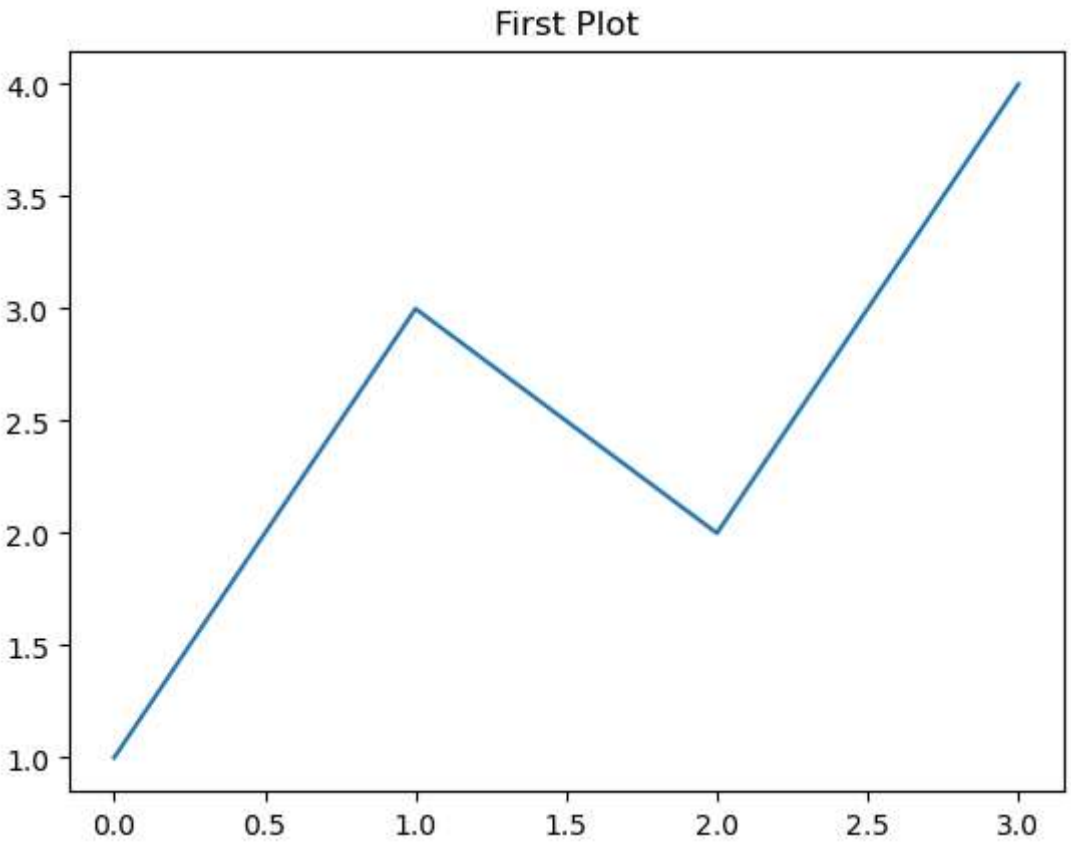
Adding labels

In [81]:
```
plt.plot([1,3,2,4])
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.show()
```



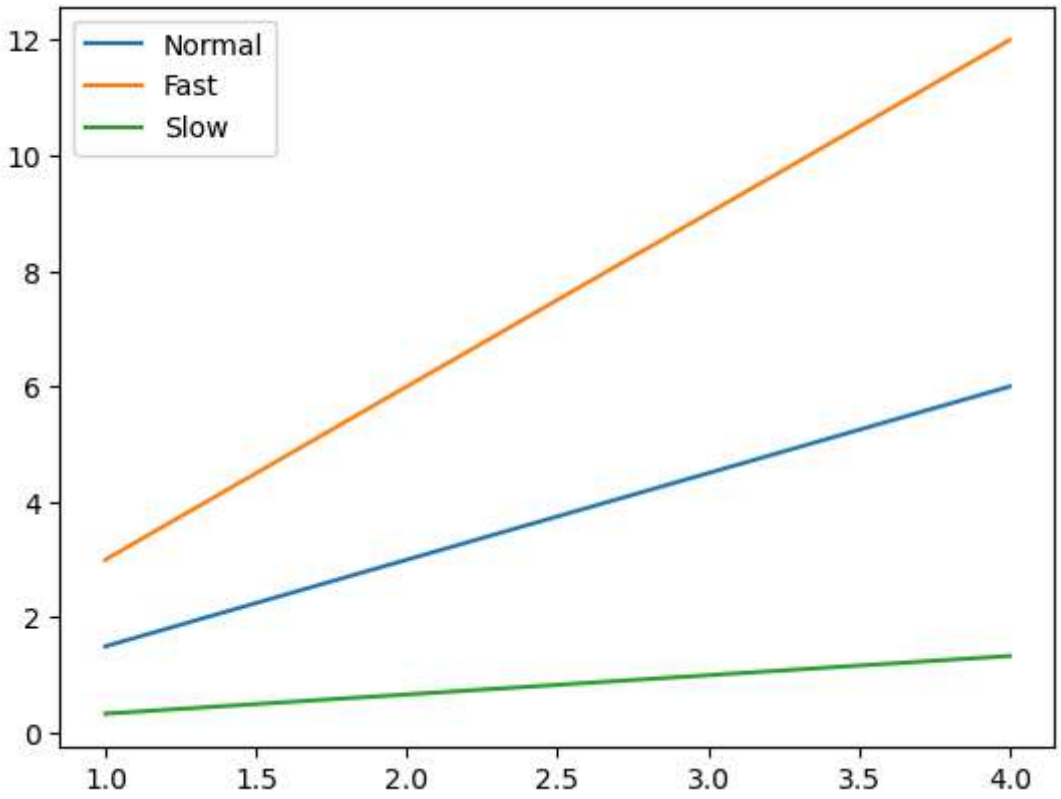
Adding a title

In [82]:
```
plt.plot([1,3,2,4])
plt.title('First Plot')
plt.show()
```

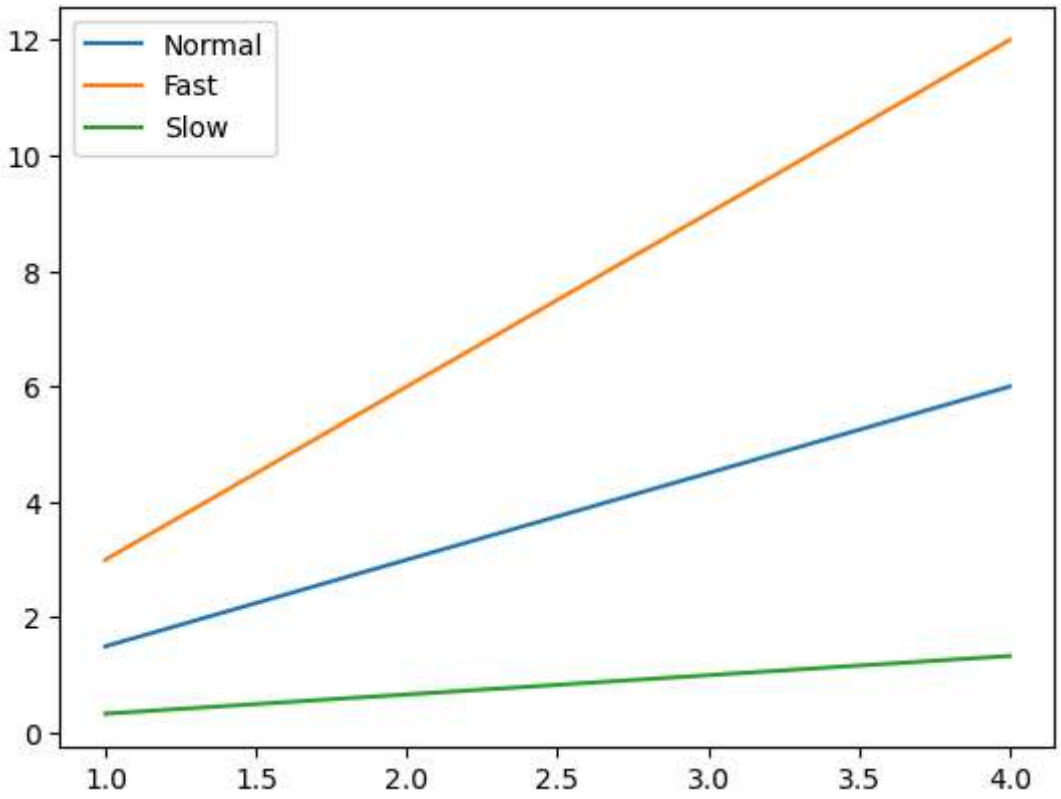


Adding a legend

In [83]:
```
x15=np.arange(1,5)

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()
```

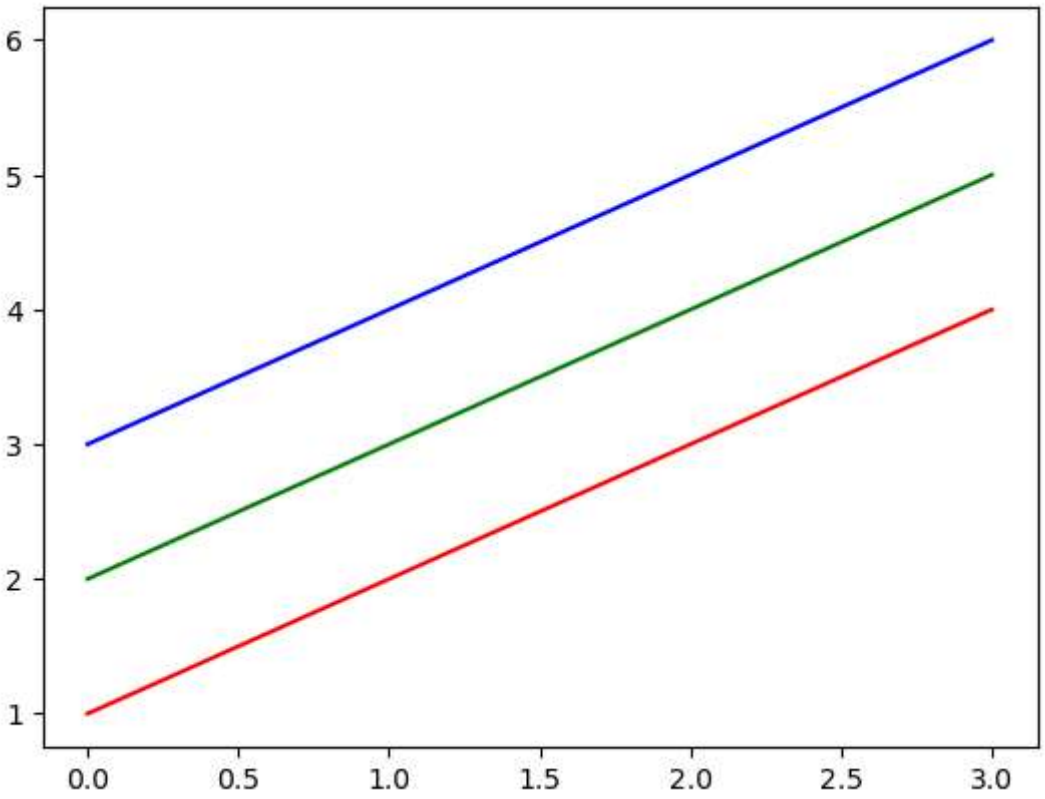


In [84]:
```
x15=np.arange(1,5)

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 [85]:
```
x16 = np.arange(1, 5)
plt.plot(x16, 'r')
plt.plot(x16+1, 'g')
plt.plot(x16+2, 'b')
plt.show()
```



Control line styles

In [86]:
```
x16=np.arange(1,5)
plt.plot(x16,'--',x16+1,'-.',x16+2,':')
plt.show()
```