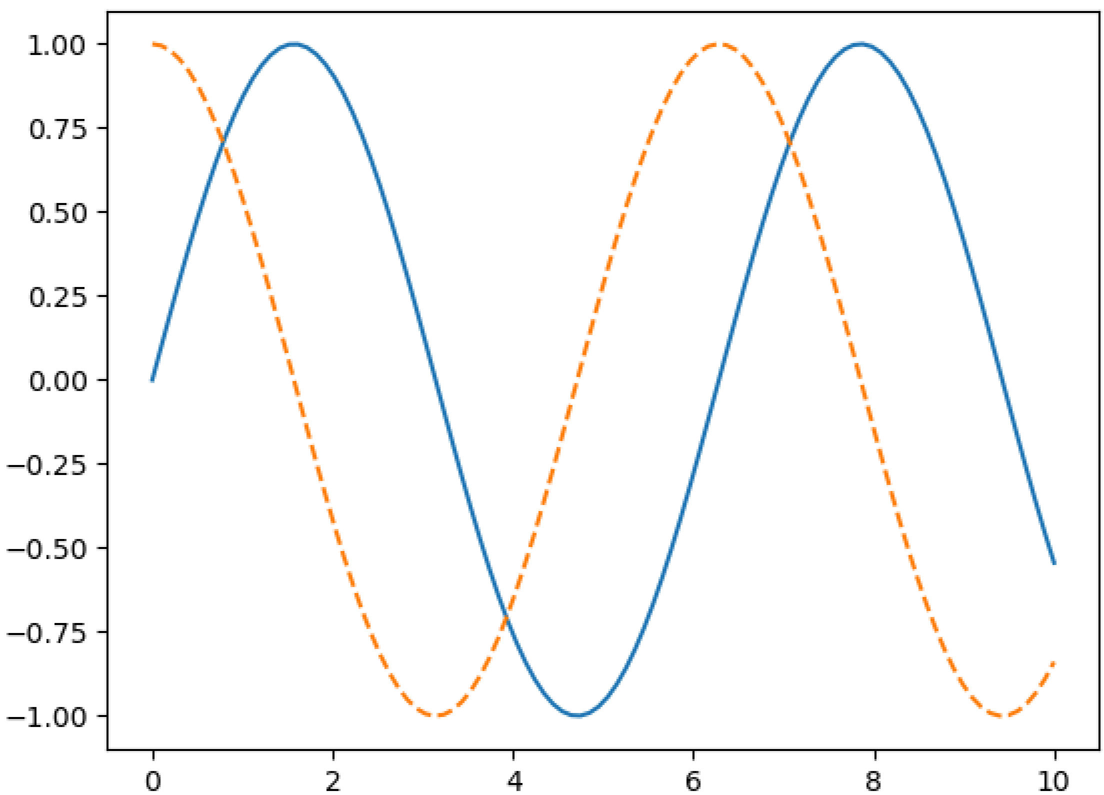
In [1]:
```
#Import Dependencies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
%matplotlib inline
```

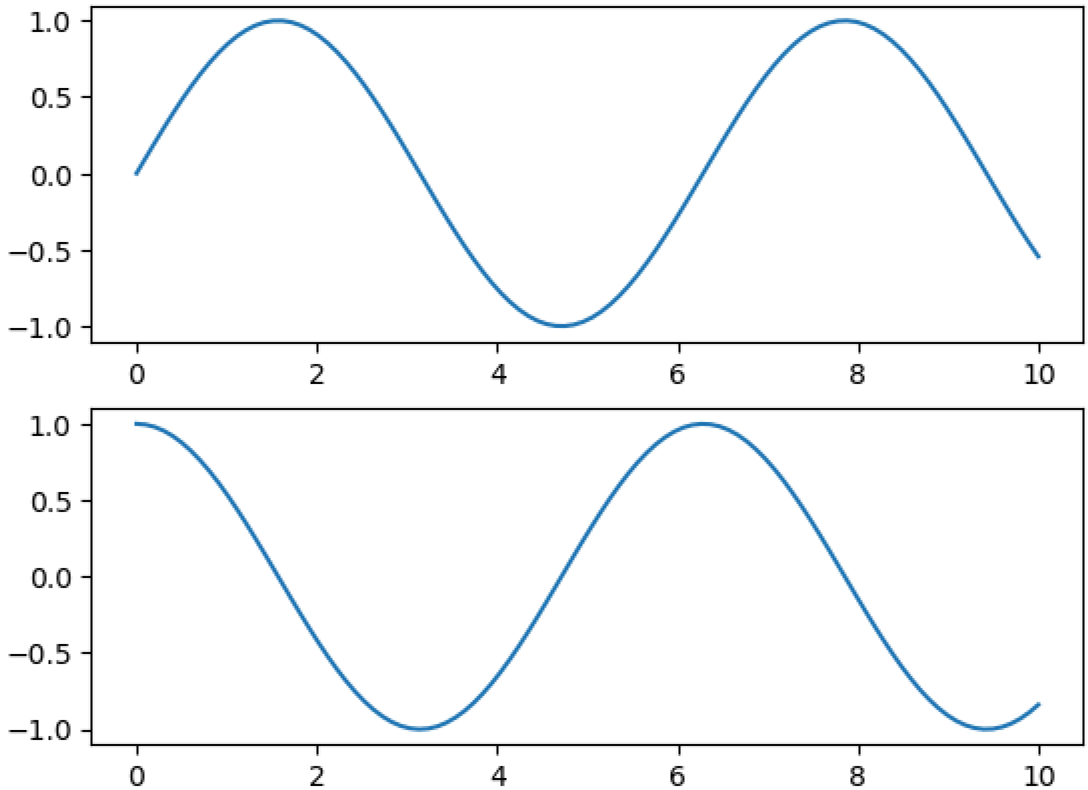
In [3]:
```
x1=np.linspace(0,10,100)
#create a plot figure
fig=plt.figure()
plt.plot(x1,np.sin(x1),'-')
plt.plot(x1,np.cos(x1),'--')
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x1c05d2b6a20>]
```



In [4]:
```
#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.sin(x1))
#create the second of two panels and set current axis
plt.subplot(2,1,2)
plt.plot(x1,np.cos(x1))
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x1c05d3c42f0>]
```

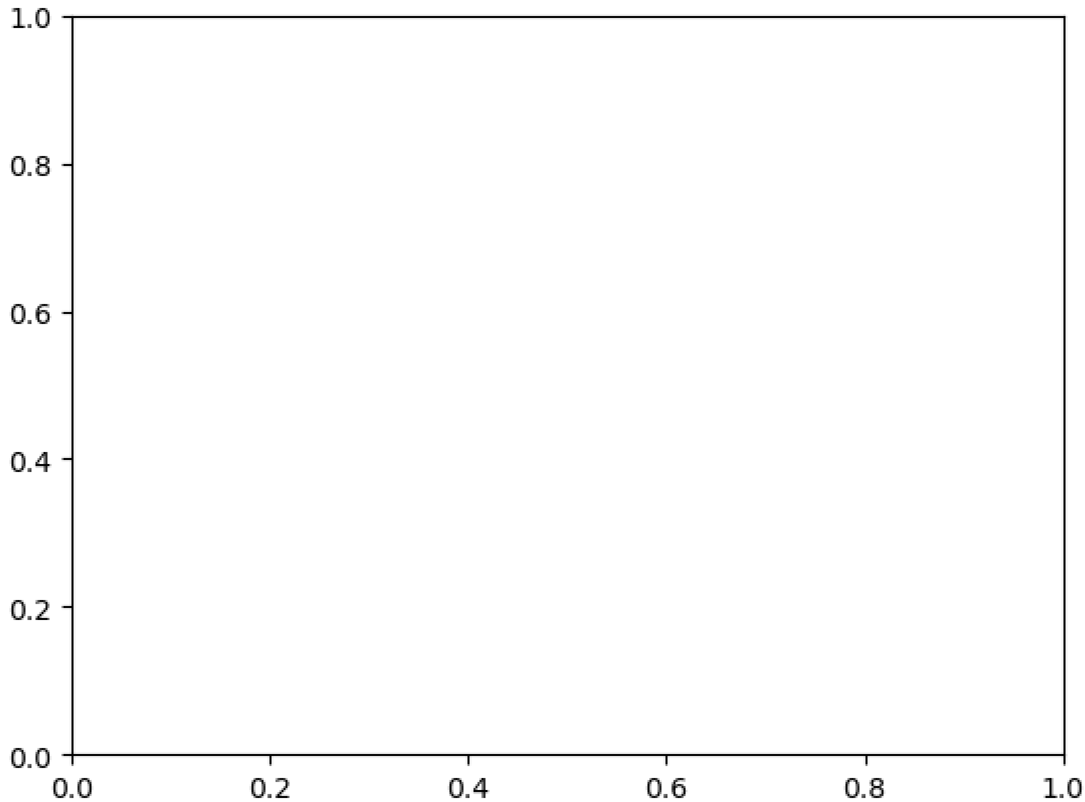


In [5]:
```
#get current figure info
print(plt.gcf())
```

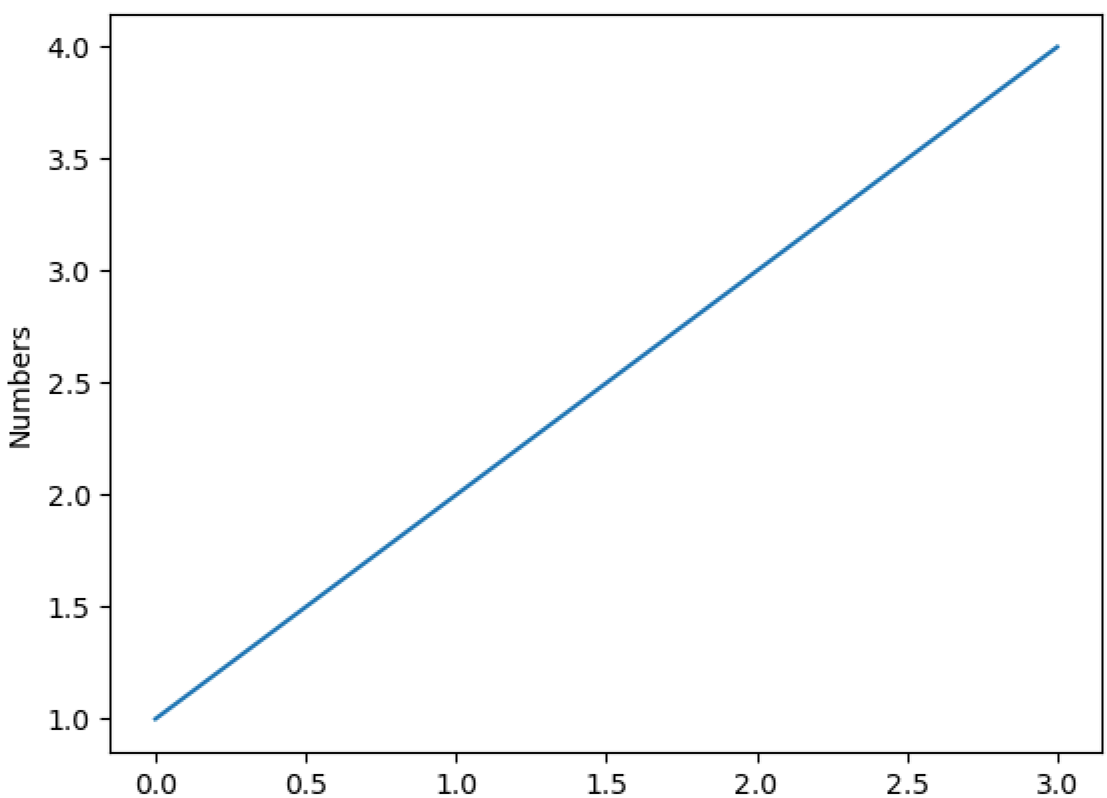
```
Figure(640x480)
<Figure size 640x480 with 0 Axes>
```

In [6]:
```
#get current axis information
print(plt.gca())
```

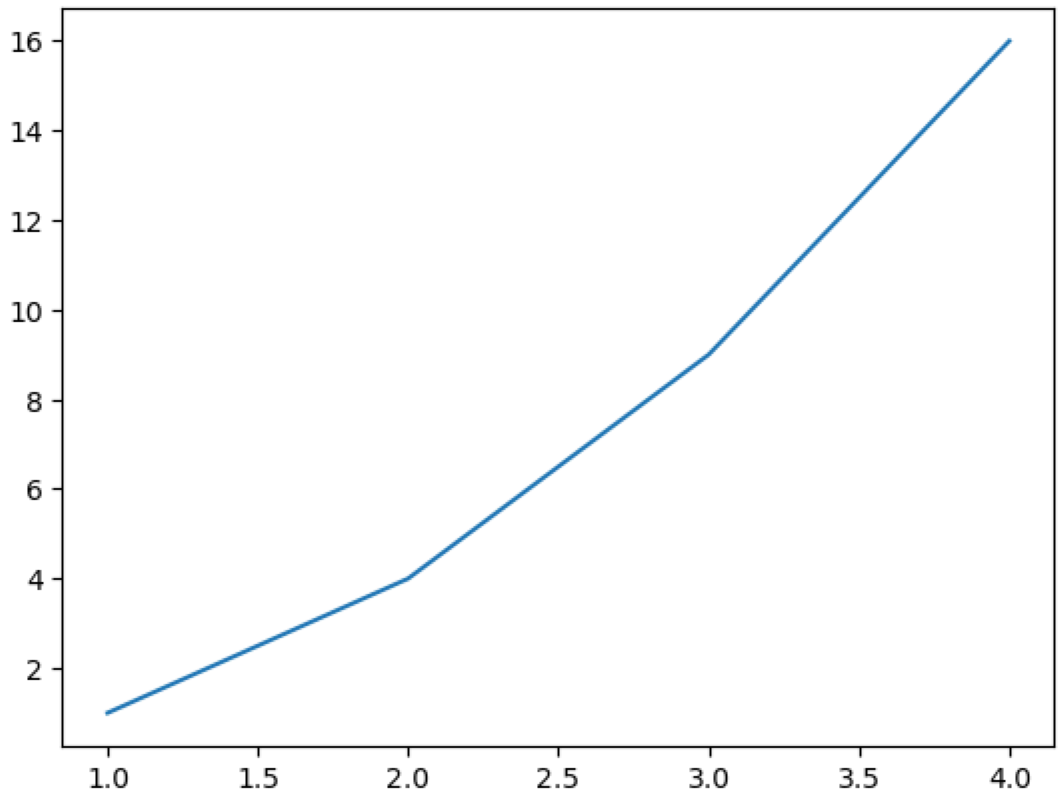
```
Axes(0.125,0.11;0.775x0.77)
```



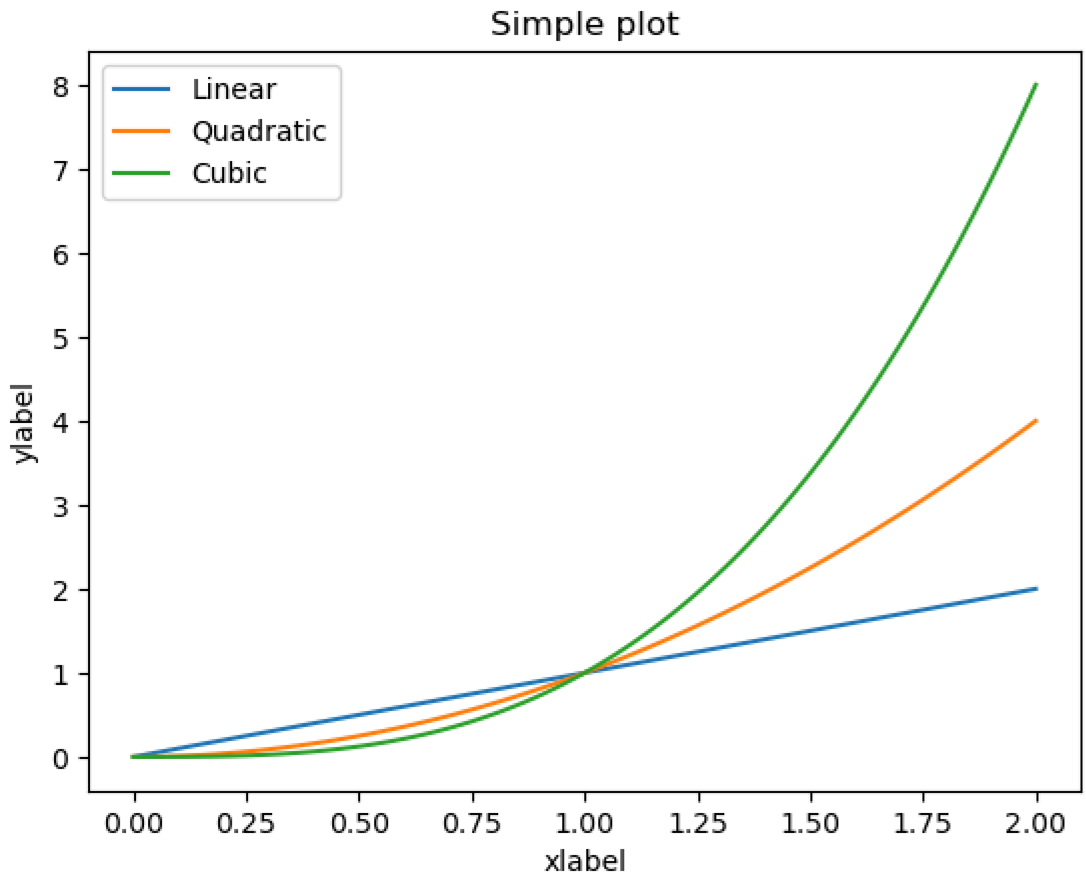
In [7]:
```
#Visulization with Pyplot
plt.plot([1,2,3,4])
plt.ylabel('Numbers')
plt.show()
```



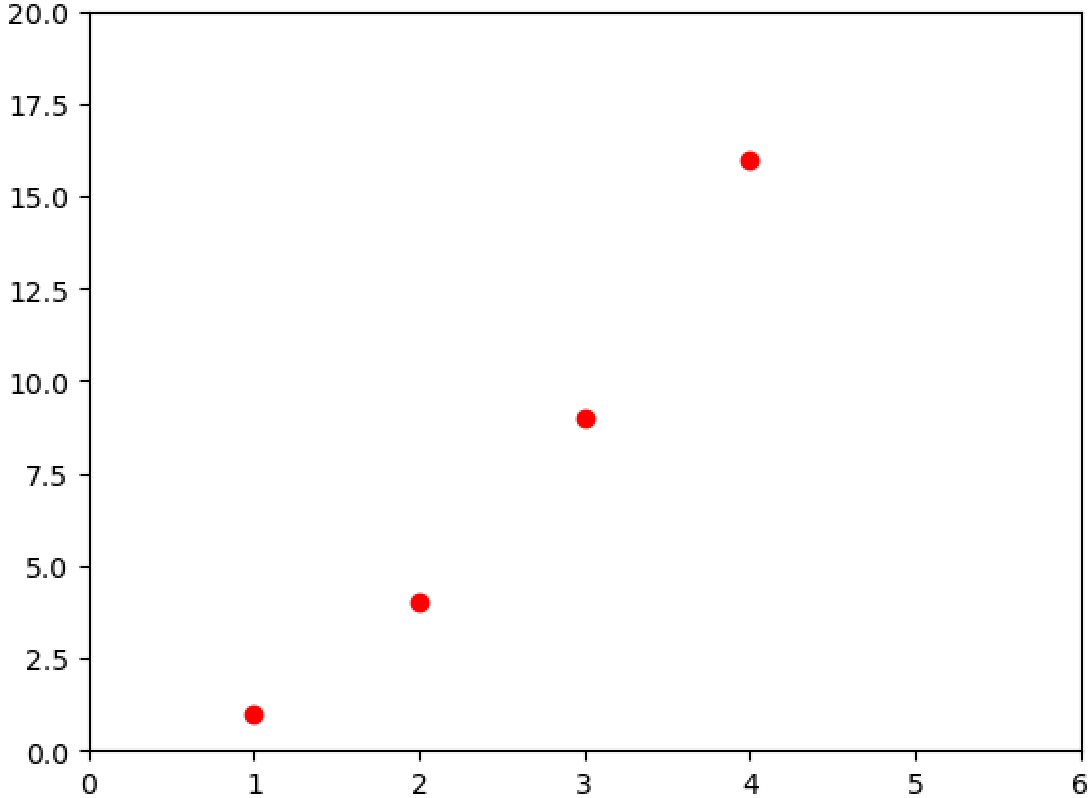
In [8]:
```
plt.plot([1,2,3,4],[1,4,9,16])
plt.show()
```



In [9]:
```
#State-Machine interface
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('xlabel')
plt.ylabel('ylabel')
plt.title('Simple plot')
plt.legend()
plt.show()
```

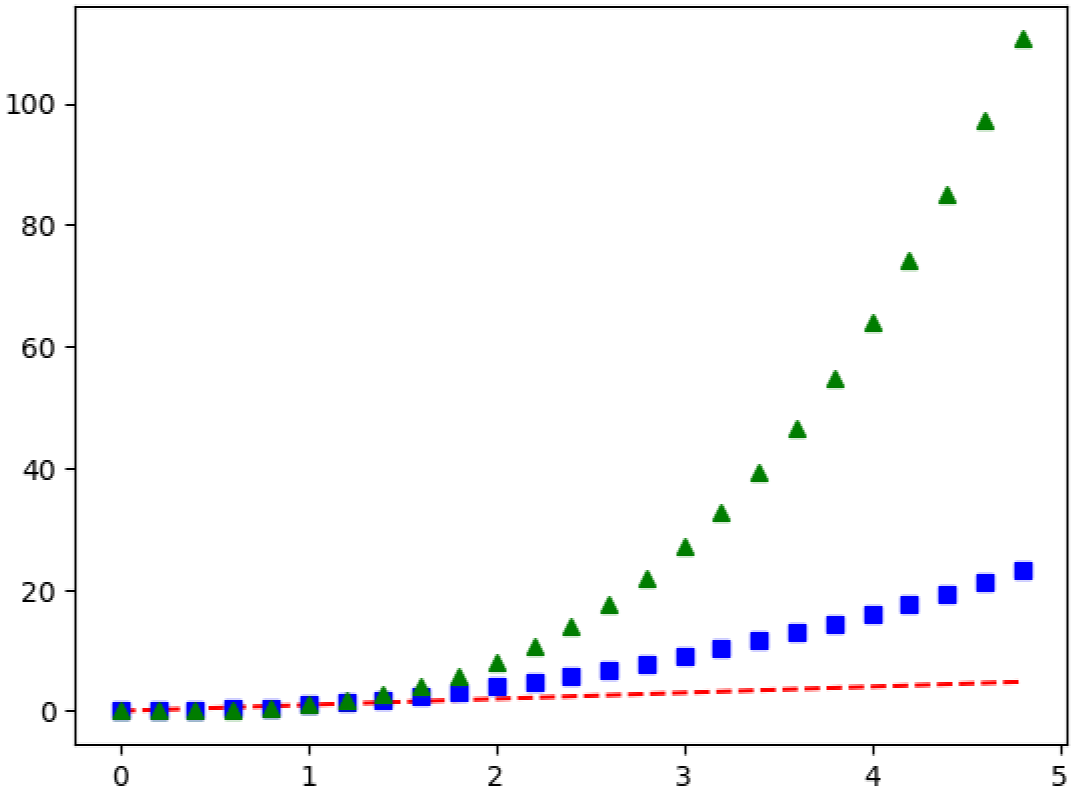


In [10]:
```
#Formatting the styles of plots
plt.plot([1,2,3,4],[1,4,9,16],'ro')
plt.axis([0,6,0,20])
#The axis() command in the example above takes a list of [xmin, xmax, ymin, ymax] a
plt.show()
```



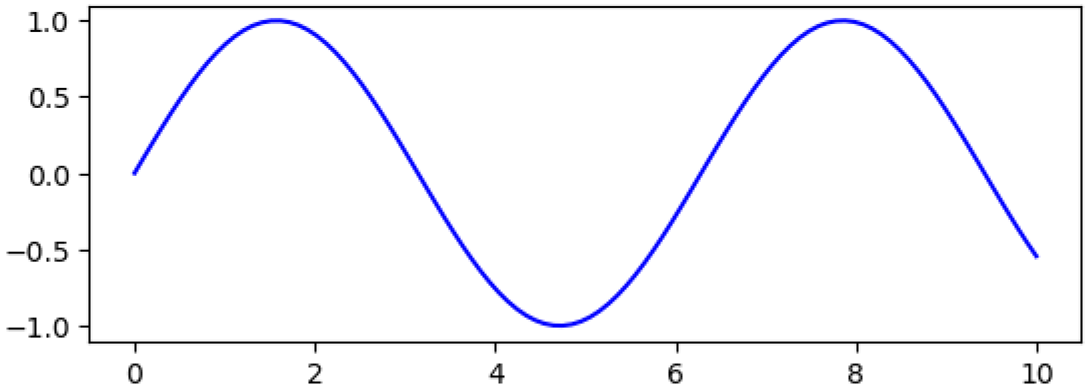
In [11]:
```
#Working with Numpy arrays
#evenly sampled time at 200ms intervals
t=np.arange(0.,5.,0.2)
#red dashes,blue squares and gree triangels
plt.plot(t,t,'r--',t,t**2,'bs',t,t**3,'g^')
```

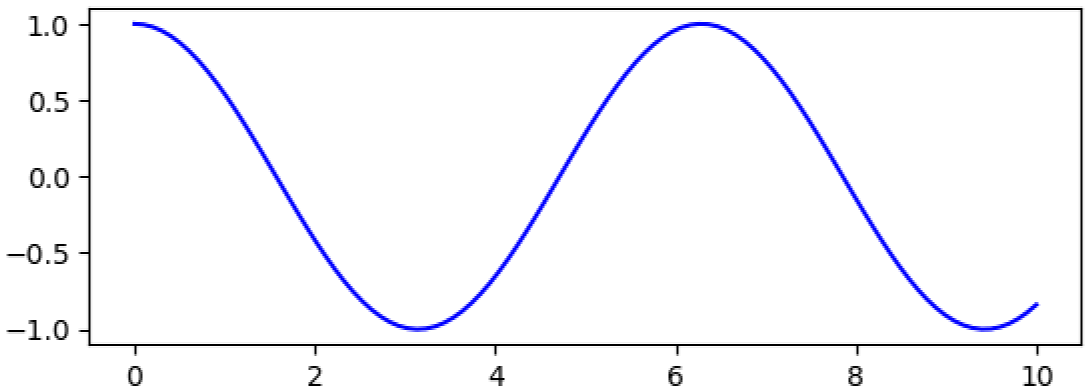
Out[11]:
```
[<matplotlib.lines.Line2D at 0x1c05d47a210>,
 <matplotlib.lines.Line2D at 0x1c05f540e90>,
 <matplotlib.lines.Line2D at 0x1c05d478770>]
```



In [12]:
```
#Object-Oriented Api
#first create a grid of plaots
#ax will be an array of two Axes objects
fig,ax=plt.subplots(2)
#call plot() methos on th apprpriate objects
ax[0].plot(x1,np.sin(x1),'b-')
ax[1].plot(x1,np.cos(x1),'b-')
```

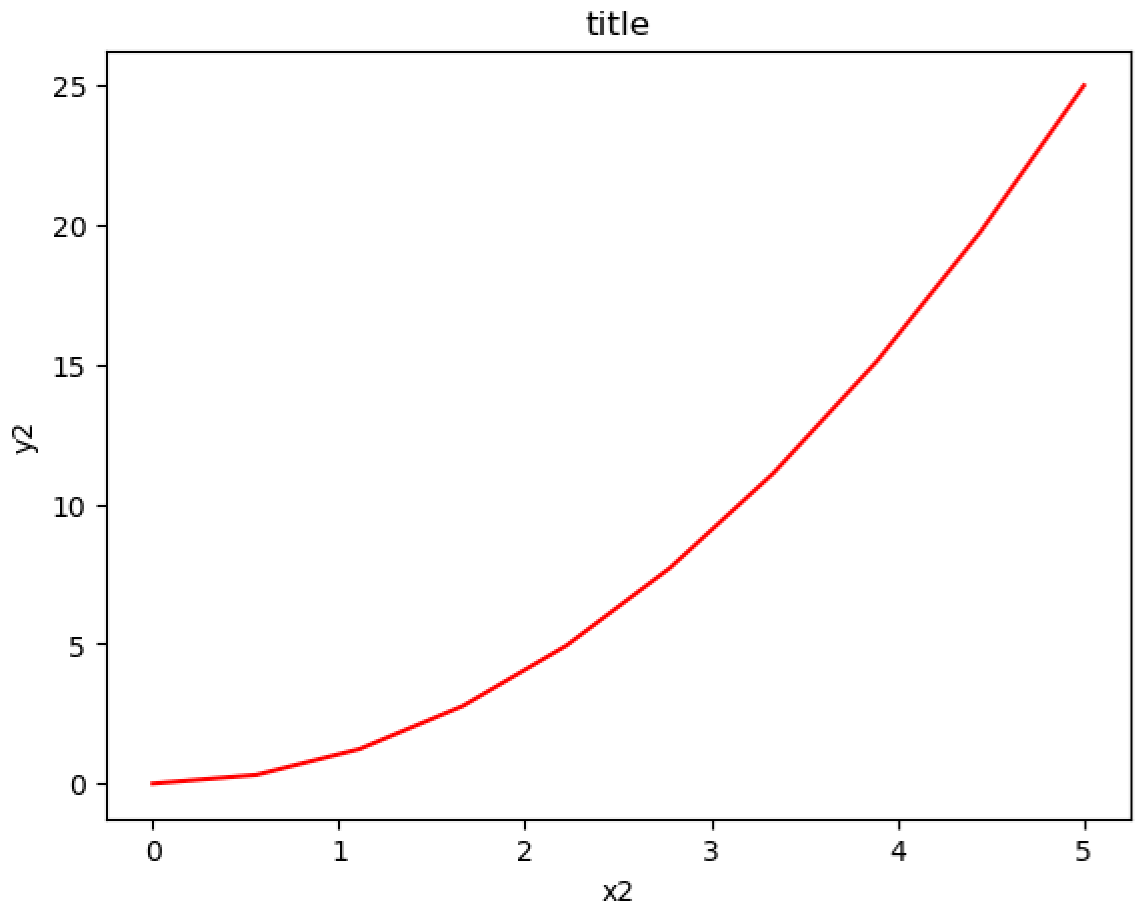
Out[12]: [<matplotlib.lines.Line2D at 0x1c05f5dc110>]



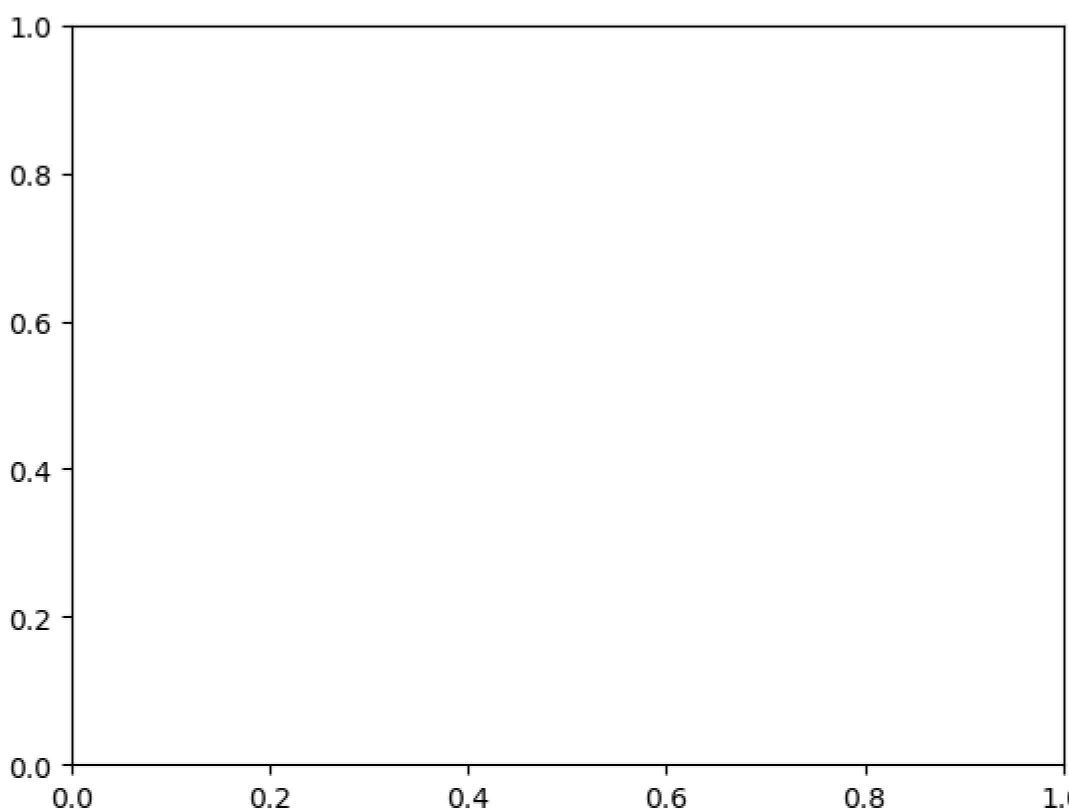


In [13]:
```
fig=plt.figure()
x2=np.linspace(0,5,10)
y2=x2**2
axes=fig.add_axes([0.1,0.1,0.8,0.8])
axes.plot(x2,y2,'r')
axes.set_xlabel('x2')
axes.set_ylabel('y2')
axes.set_title('title')
```

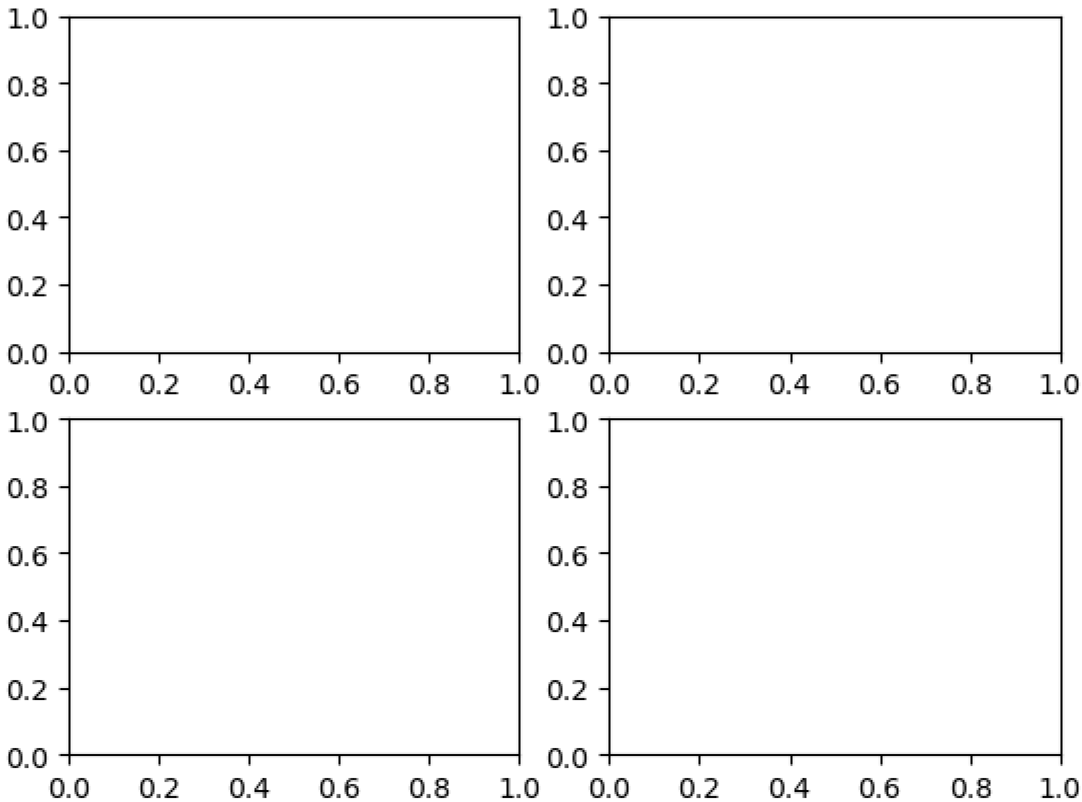
Out[13]:
```
Text(0.5, 1.0, 'title')
```



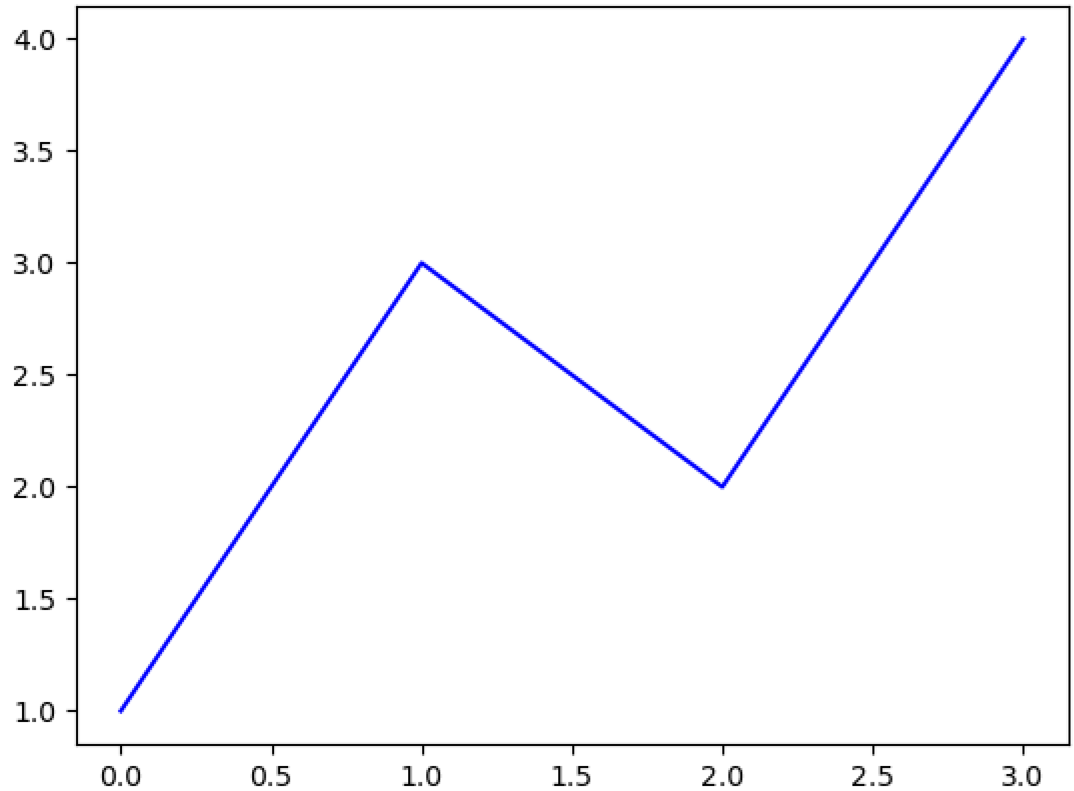
In [14]:
```
fig=plt.figure()
ax=plt.axes()
```



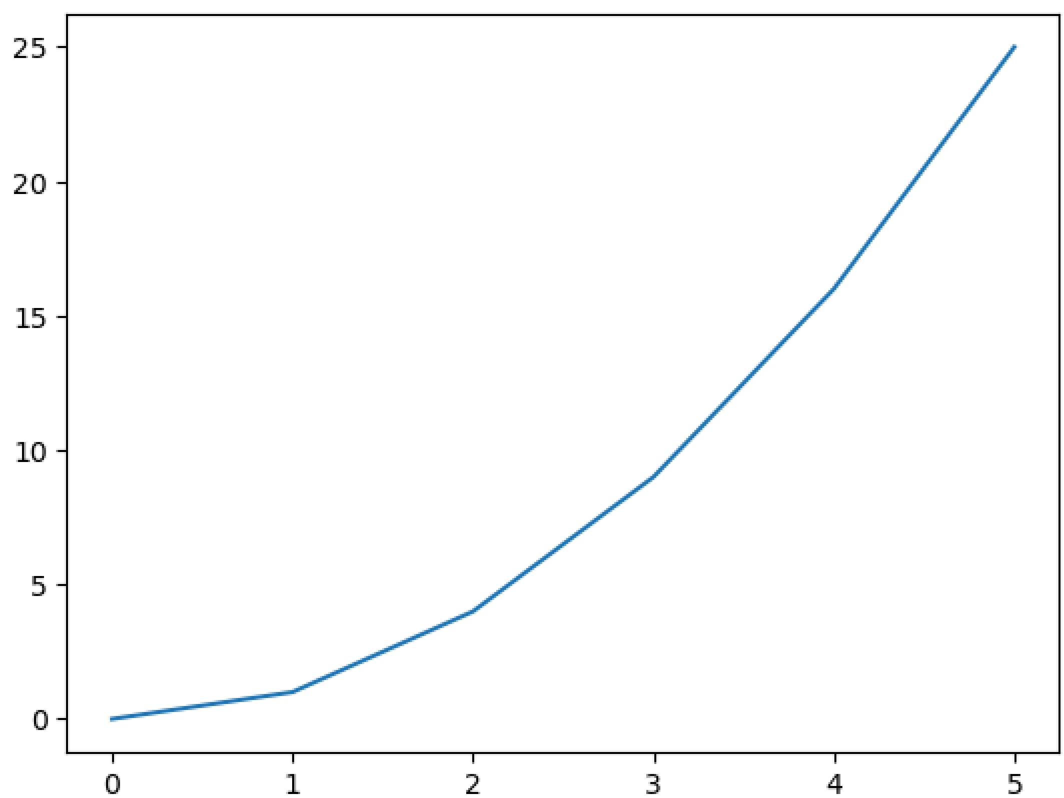
In [15]:
```
#Figure and Subplots
## plots in Matplolib reside within a figure object.
fig=plt.figure()
#Now ,I create one or more subplots usings fig.add_subplot() as follows
ax1=fig.add_subplot(2,2,1)
#The above command means that there are four subplots and im selecting the first on
#creating the other three subplots
ax2=fig.add_subplot(2,2,2)
ax3=fig.add_subplot(2,2,3)
ax4=fig.add_subplot(2,2,4)
```



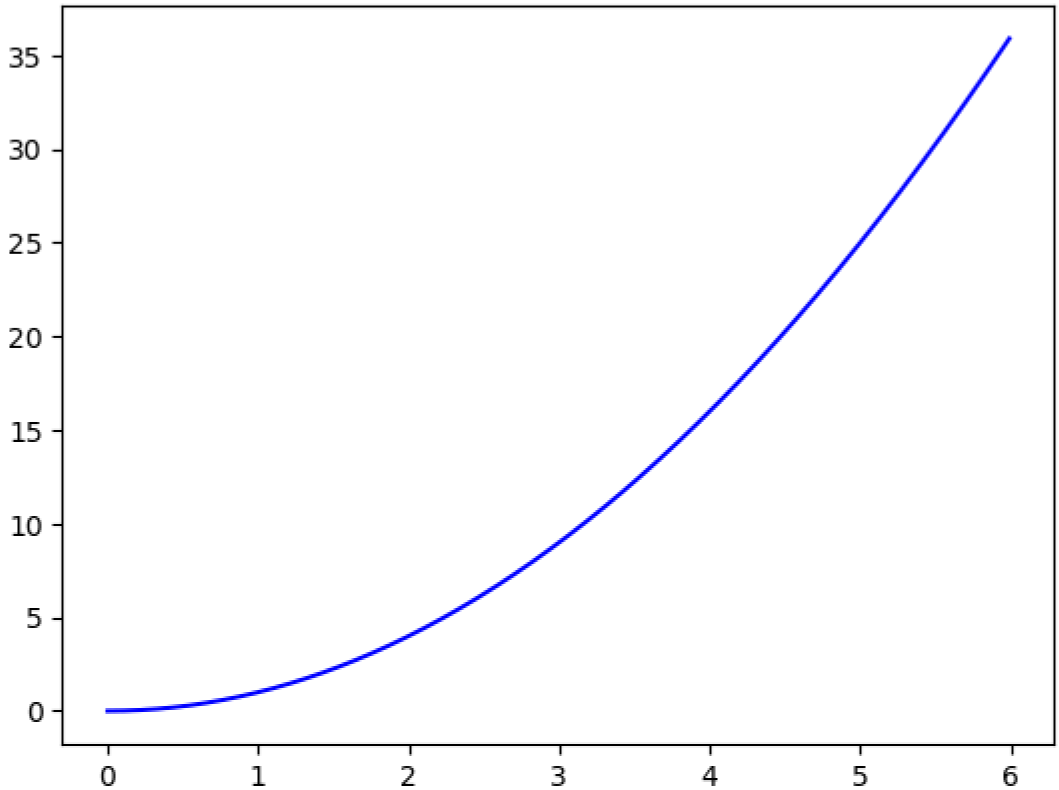
In [16]:
```
#First plot with Matplotlib
plt.plot([1,3,2,4],'b-')
plt.show()
```



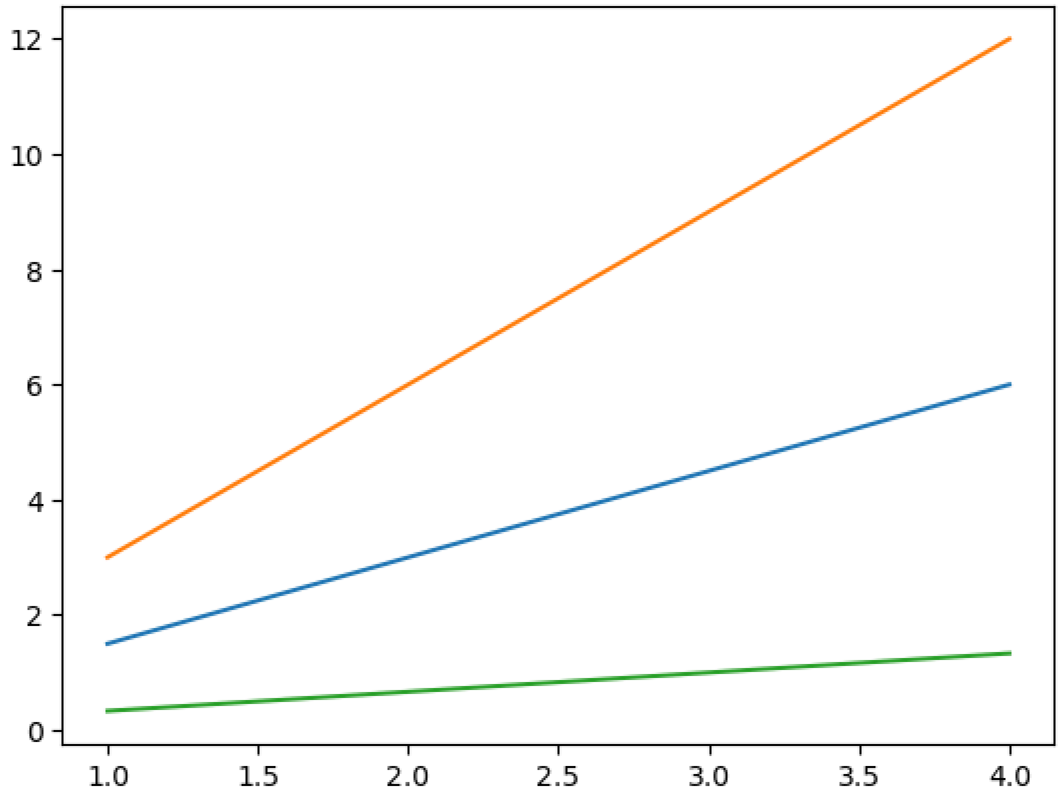
In [17]:
```
#Specify both Lists
x3=range(6)
plt.plot(x3,[xi**2 for xi in x3])
plt.show()
```



In [18]:
```
x3=np.arange(0.0,6.0,0.01)
plt.plot(x3,[xi**2 for xi in x3],'b-')
plt.show()
```

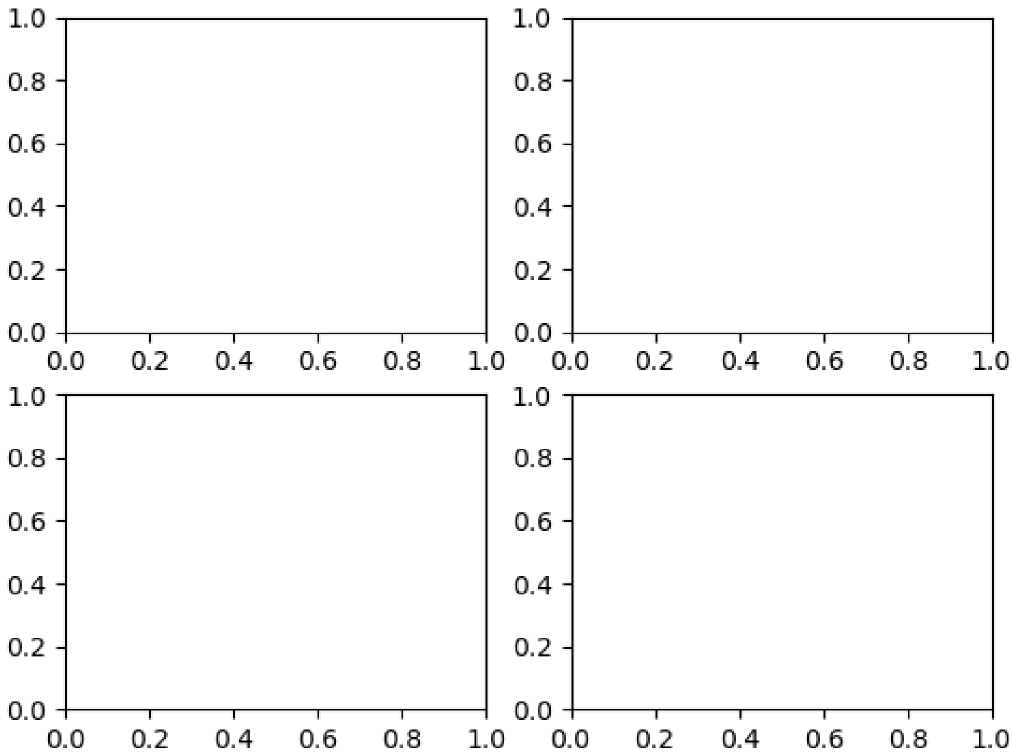


In [19]:
```
x4=range(1,5)
plt.plot(x4,[xi*1.5 for xi in x4])
plt.plot(x4,[xi*3 for xi in x4])
plt.plot(x4,[xi/3.0 for xi in x4])
plt.show()
```



In [20]:
```
#Saving the figure
fig.savefig('plot1.png')
#Explore the contents of figure
from IPython.display import Image
Image('plot1.png')
```

Out[20]:

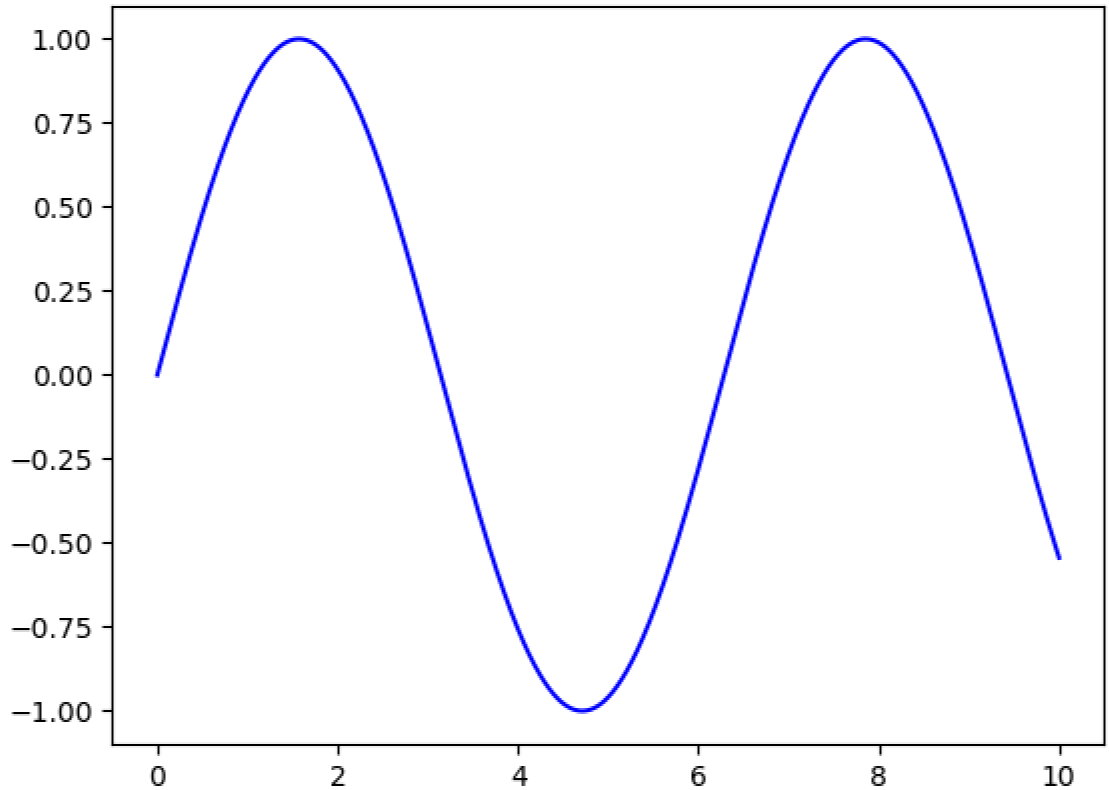


In [21]:
```
#Explore supported File formats
fig.canvas.get_supported_filetypes()
```

Out[21]:
```
{'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'}
```

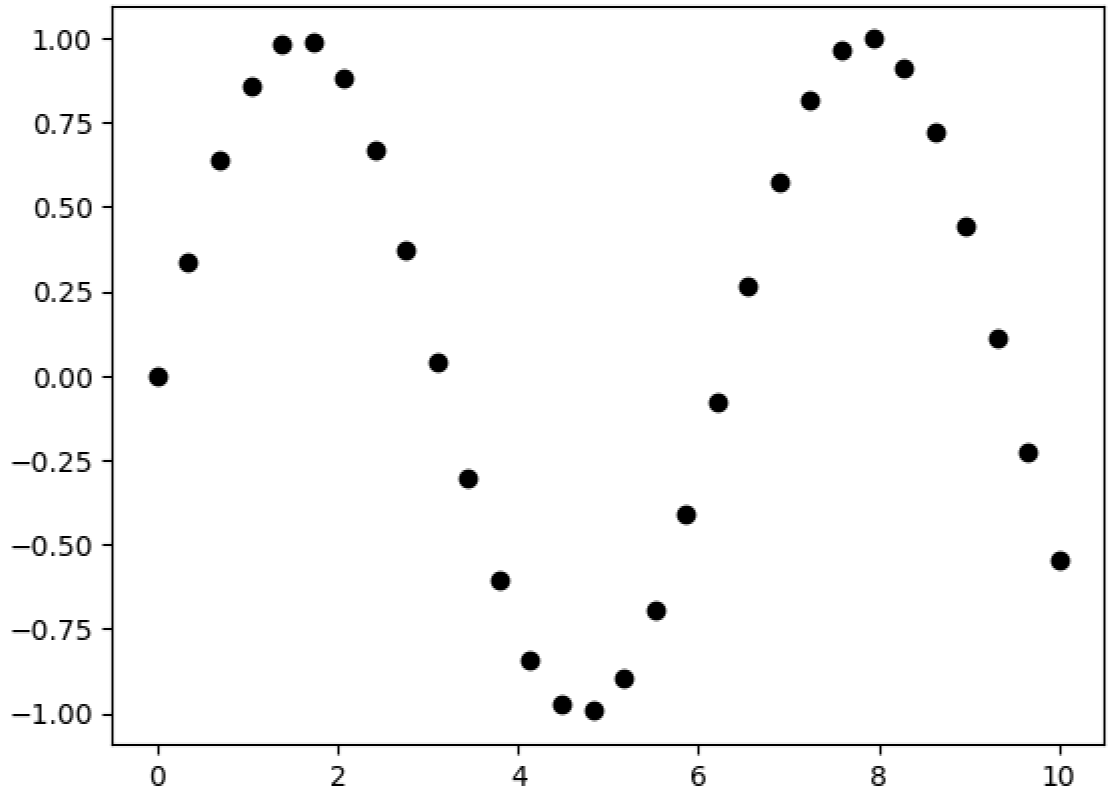
In [22]:
```
#Line Plot
#Creat figure and axes first
fig=plt.figure()
ax=plt.axes()
#Declare a variable x5
x5=np.linspace(0,10,1000)
#Plot the sinusoid function
ax.plot(x5,np.sin(x5),'b-')
```

Out[22]:
```
[<matplotlib.lines.Line2D at 0x1c060bf9d30>]
```



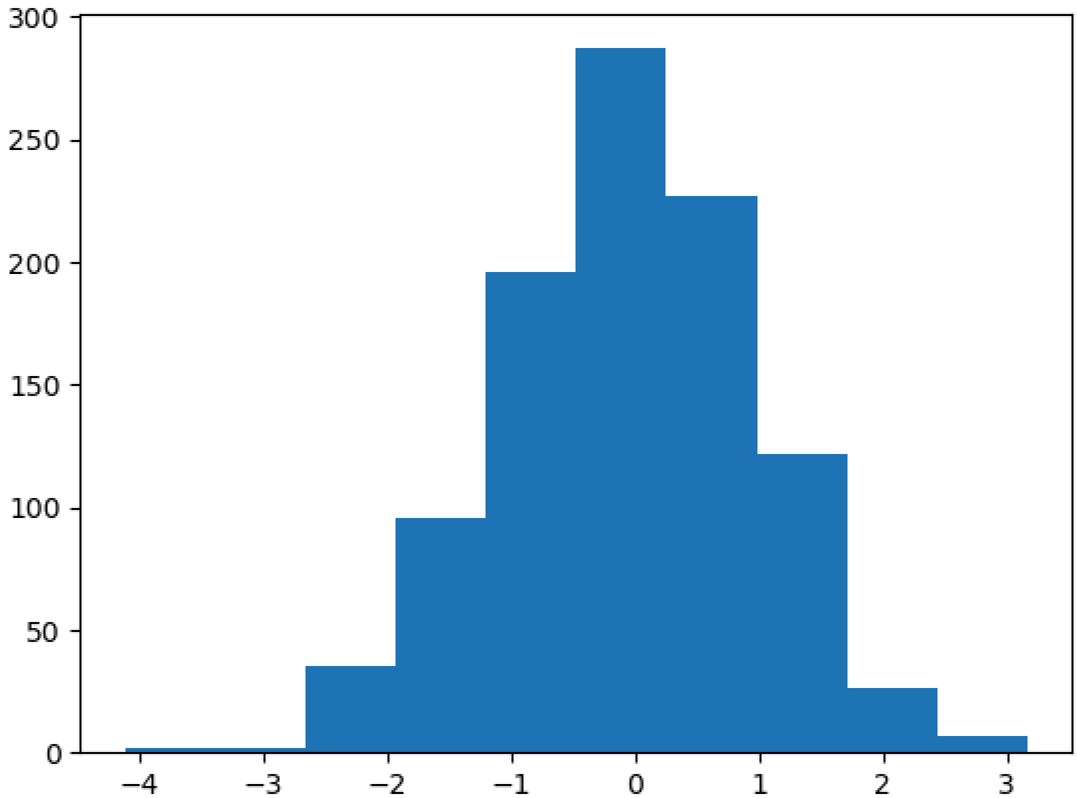
In [23]:
```
#Scatter olot using plt.plot()
x7=np.linspace(0,10,30)
y7=np.sin(x7)
plt.plot(x7,y7,'o',color='black')
```

Out[23]:
```
[<matplotlib.lines.Line2D at 0x1c060c4a9f0>]
```

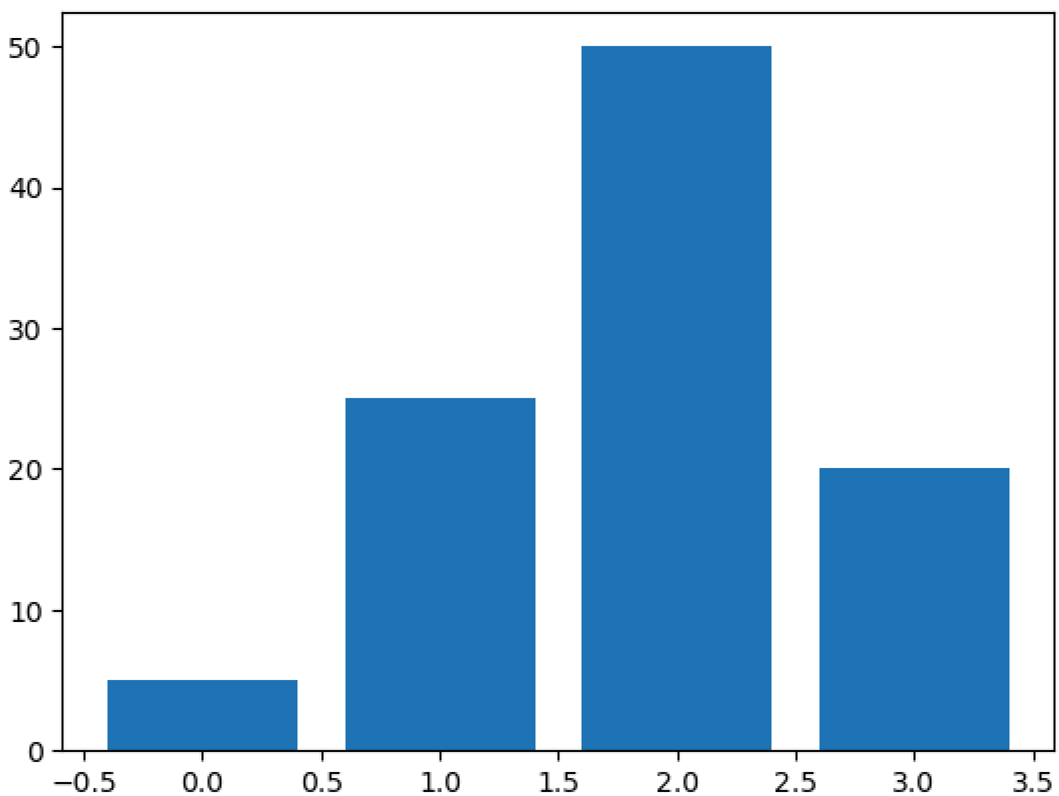


In [24]:
```
#Histogram
data1=np.random.randn(1000)
plt.hist(data1)
```

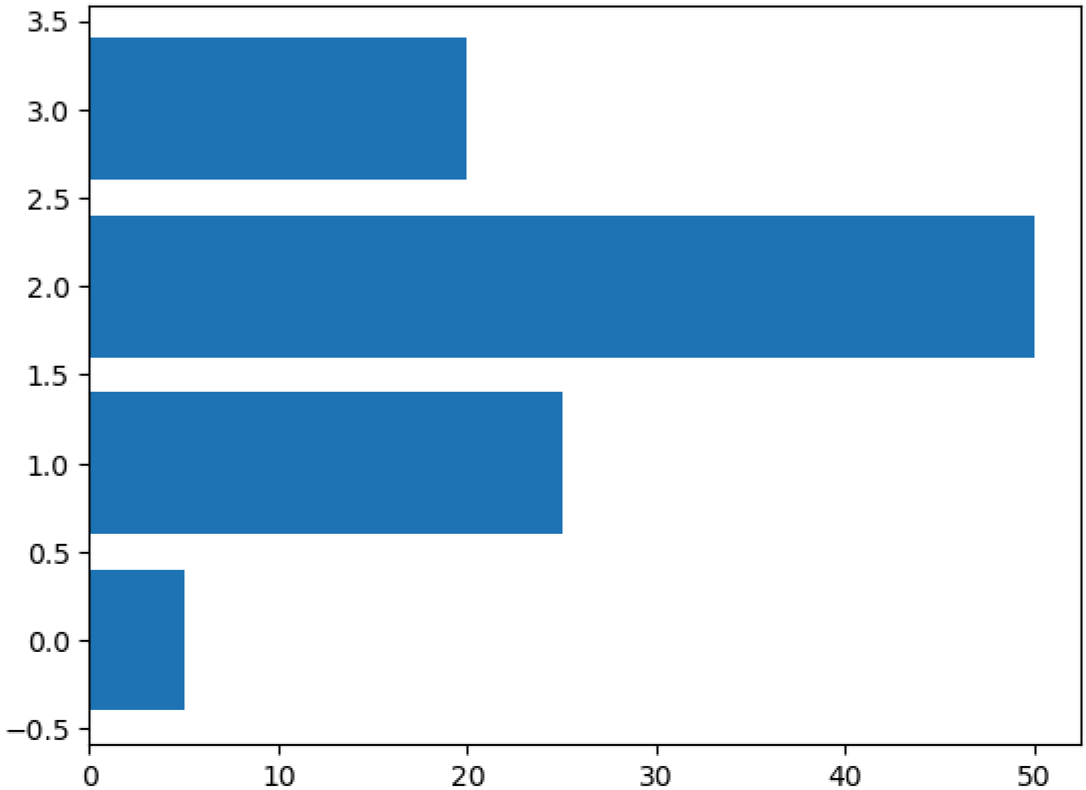
Out[24]:
```
(array([  2.,   2.,  35.,  96., 196., 287., 227., 122.,  26.,   7.]),
 array([-4.10775839, -3.38037395, -2.6529895 , -1.92560505, -1.19822061,
        -0.47083616,  0.25654829,  0.98393274,  1.71131718,  2.43870163,
         3.16608608]),
 <BarContainer object of 10 artists>)
```



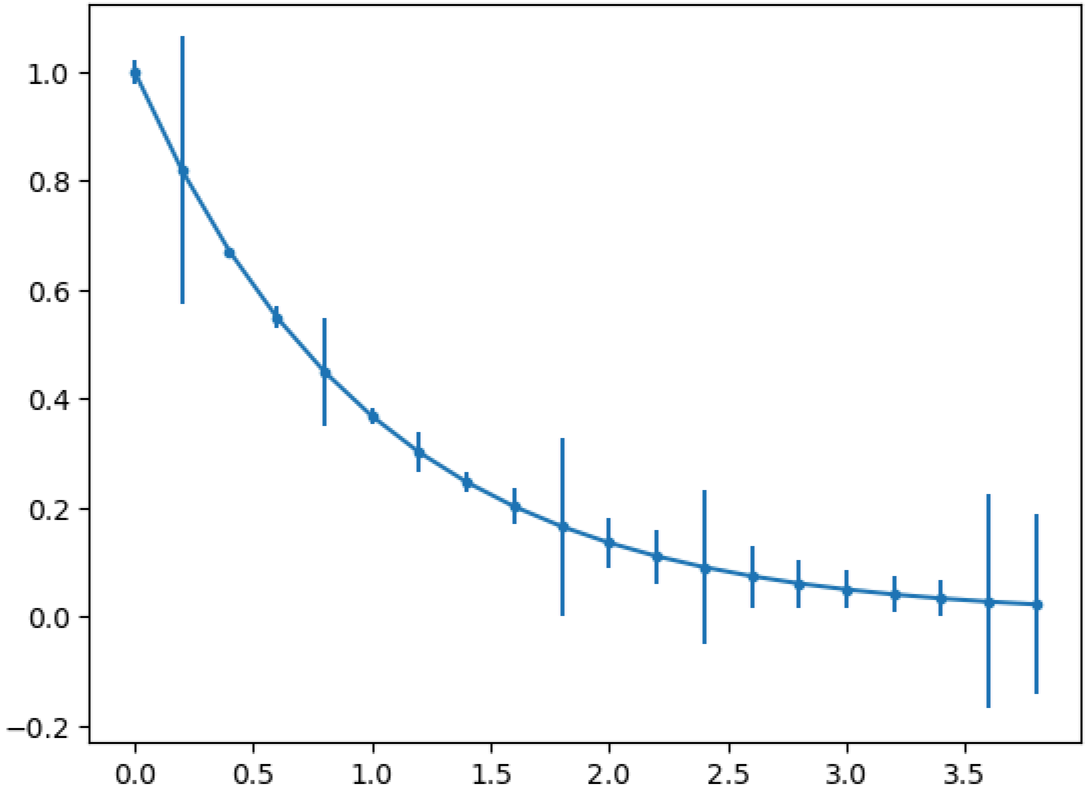
In [25]:
```
#Bar Chart
data2=[5.,25.,50.,20.]
plt.bar(range(len(data2)),data2)
plt.show()
```



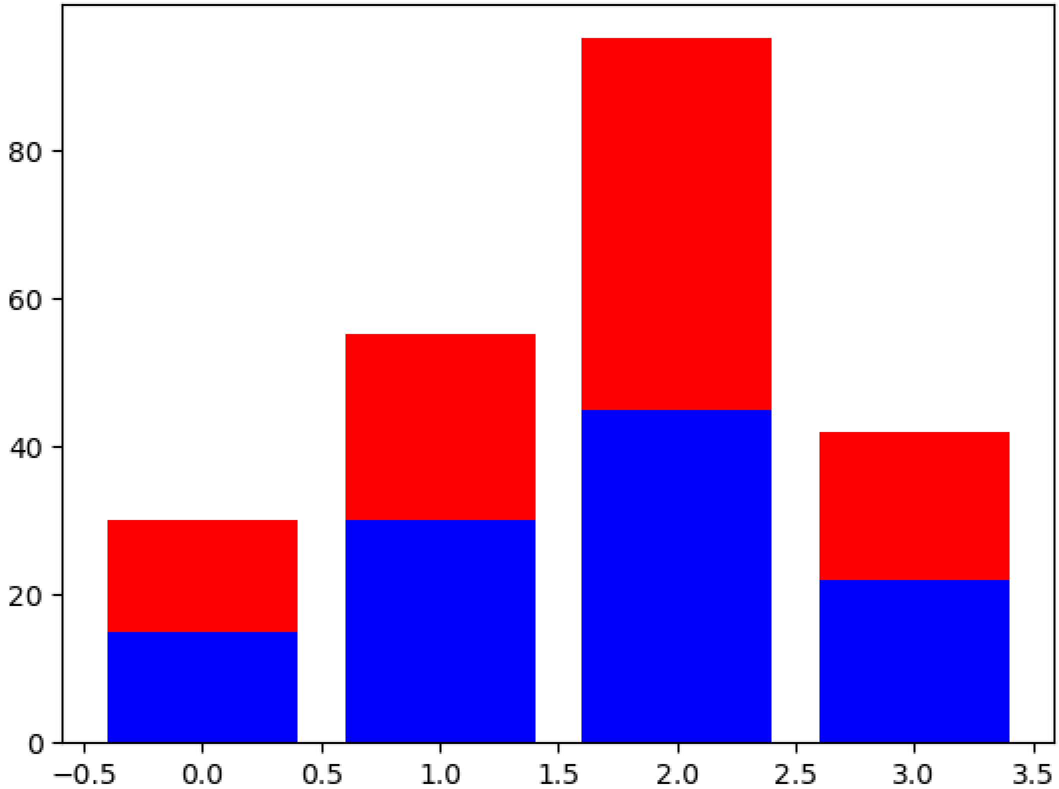
In [26]:
```
#Horizontal Bar chart
plt.barh(range(len(data2)),data2)
plt.show()
```



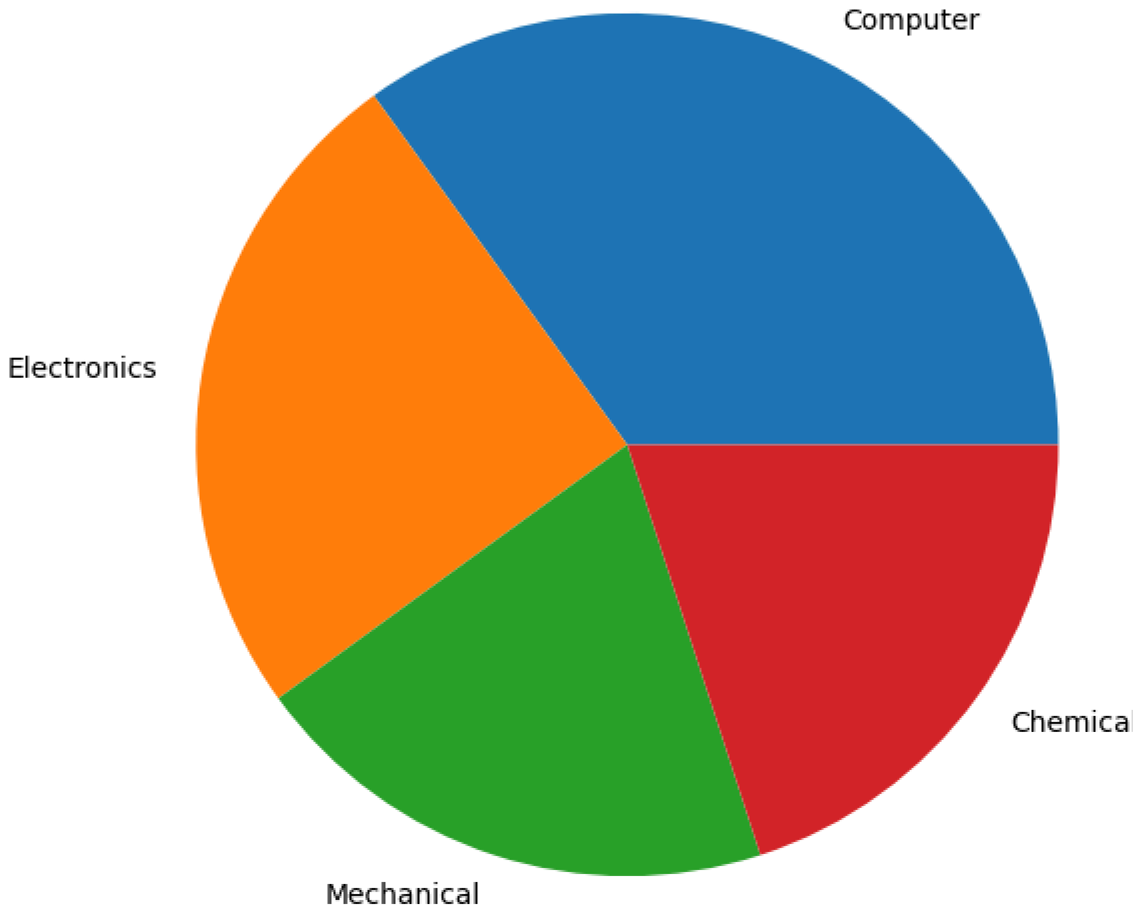
In [27]:
```
#Error Bar Chart
x9=np.arange(0,4,0.2)
y9=np.exp(-x9)
e1=0.1*np.abs(np.random.randn(len(y9)))
plt.errorbar(x9,y9,yerr=e1,fmt='.-')
plt.show()
```



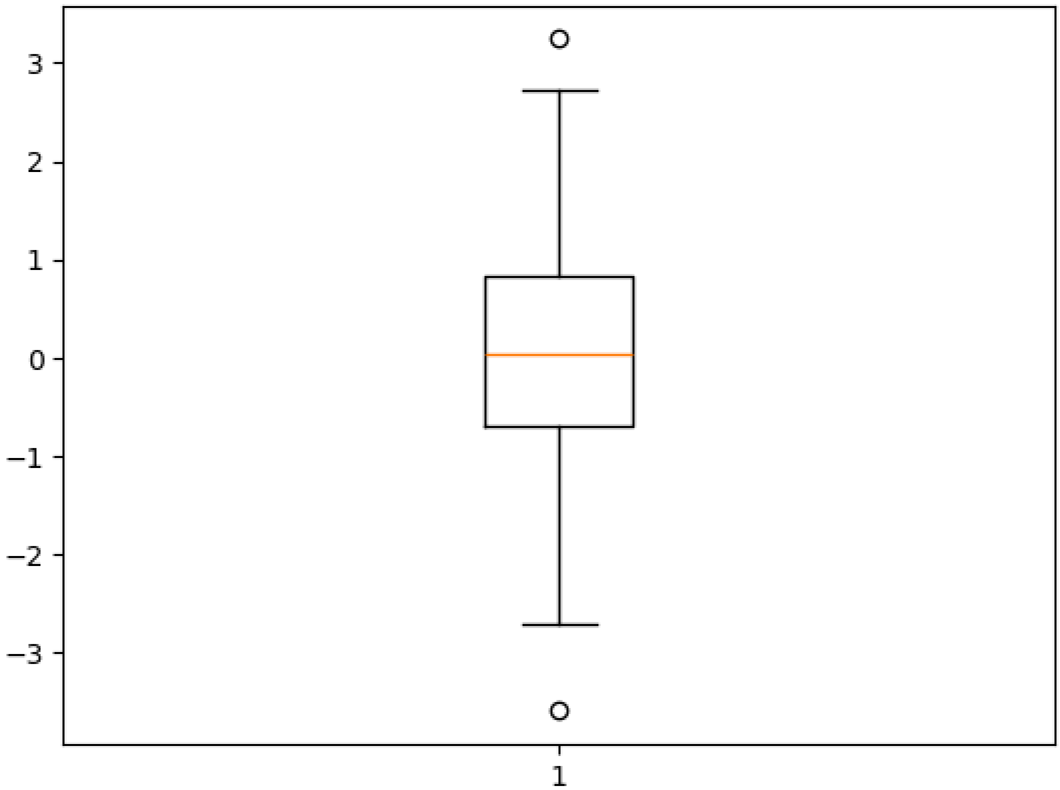
In [28]:
```
#Stacked Bar chart
A=[15,30,45,22]
B=[15,25,50,20]
z2=range(4)
plt.bar(z2,A,color='b')
plt.bar(z2,B,color='r',bottom=A)
plt.show()
```



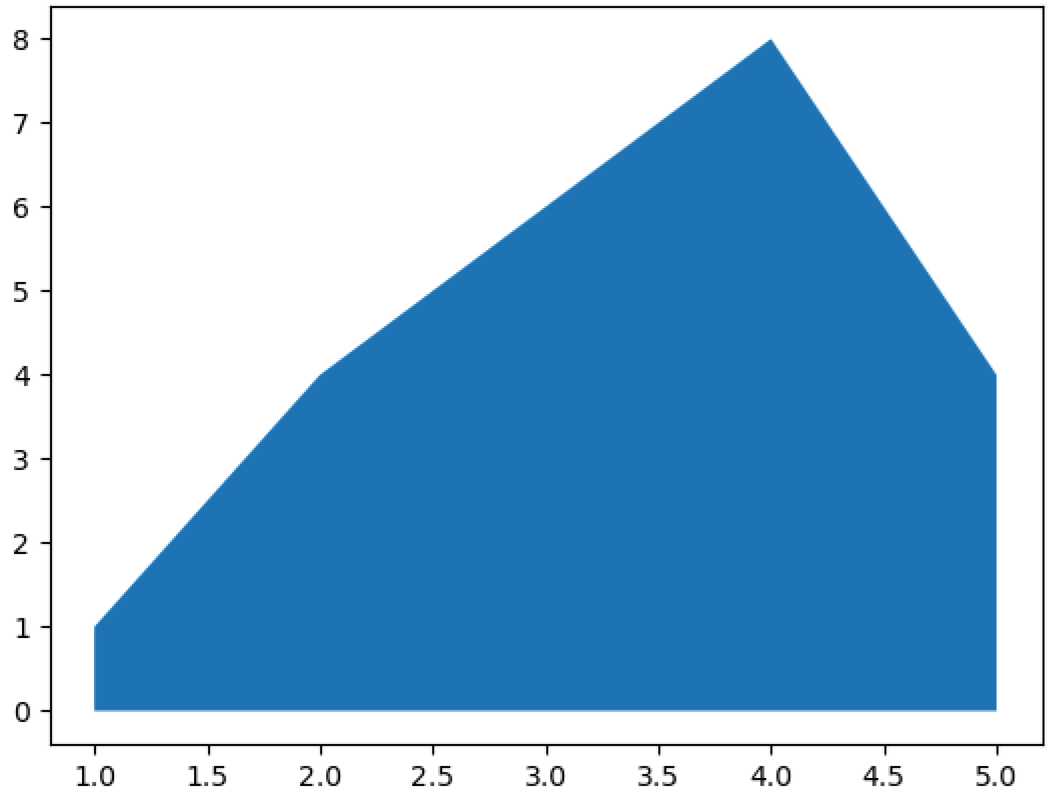
In [29]:
```
plt.figure(figsize=(7,7))
x10=[35,25,20,20]
labels=['Computer','Electronics','Mechanical','Chemical']
plt.pie(x10,labels=labels)
plt.show()
```



In [30]:
```
#Box plot
data3=np.random.randn(100)
plt.boxplot(data3)
plt.show()
```

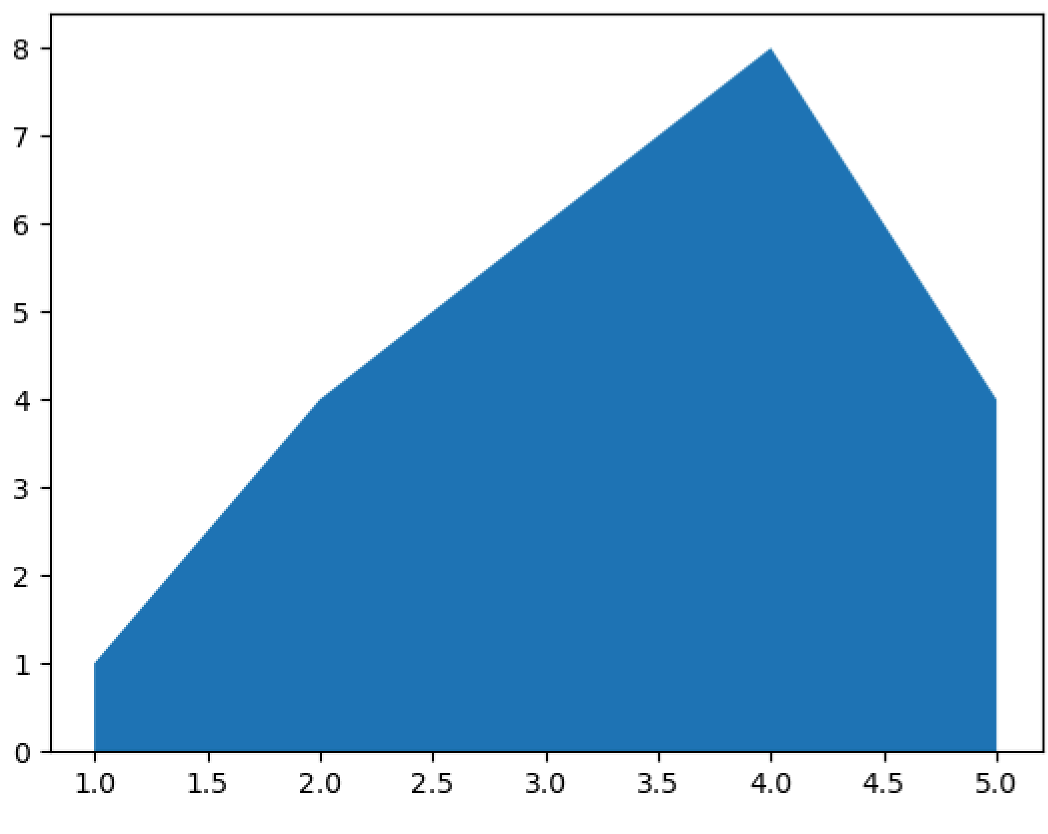


In [31]:
```
#Area chart
#Create some data
x12=range(1,6)
y12=[1,4,6,8,4]
#Area plot
plt.fill_between(x12,y12)
plt.show()
```



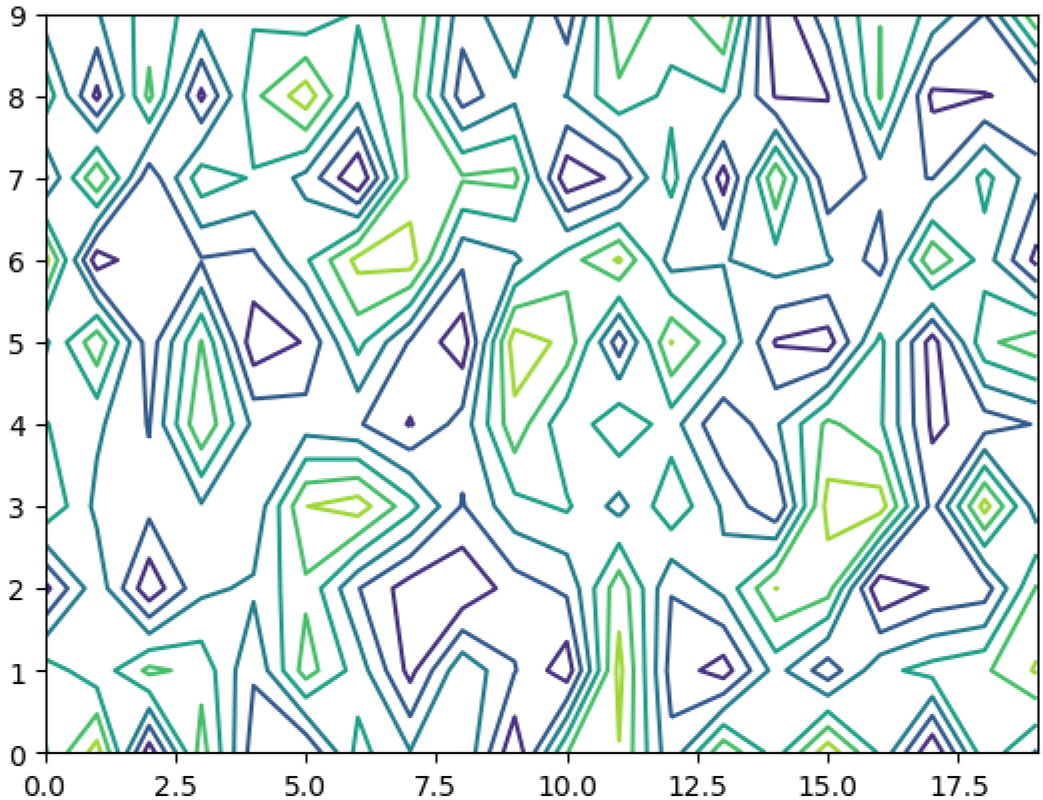
In [32]:
```
#Stack plot
plt.stackplot(x12,y12)
```

Out[32]: [<matplotlib.collections.PolyCollection at 0x1c05f76aae0>]



In [33]:
```
#Contour Plot
#Create a matrix

matrix1=np.random.rand(10,20)
cp=plt.contour(matrix1)
plt.show()
```

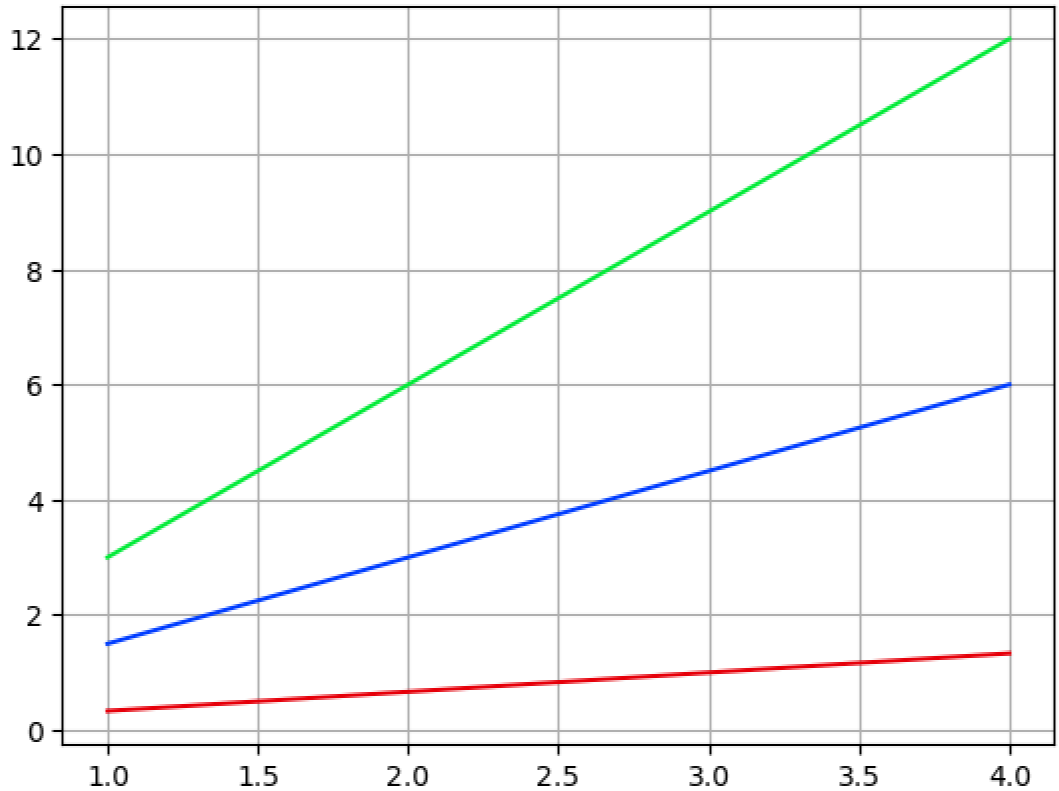


In [34]:
```
#Styles with Matplotlib
print(plt.style.available)
```

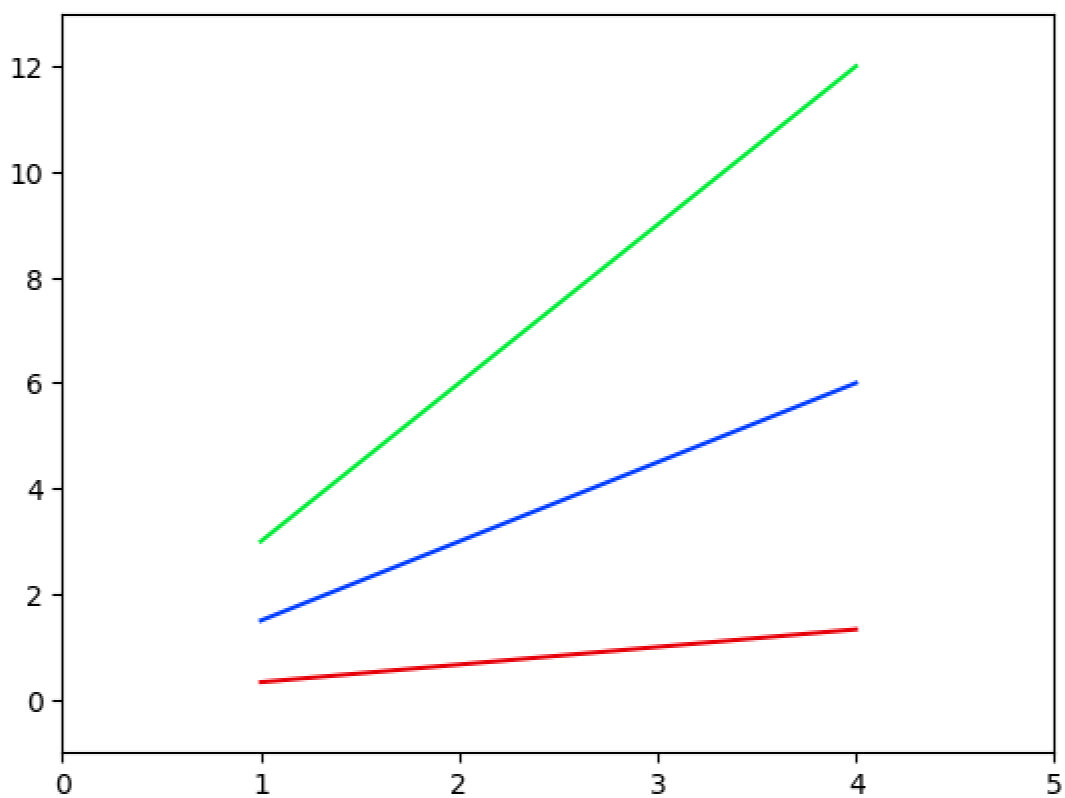
```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', 'b
mh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale',
'seaborn-v0_8', 'seaborn-v0_8-bright', 'seaborn-v0_8-colorblind', 'seaborn-v0_8-dar
k', 'seaborn-v0_8-dark-palette', 'seaborn-v0_8-darkgrid', 'seaborn-v0_8-deep', 'seab
orn-v0_8-muted', 'seaborn-v0_8-notebook', 'seaborn-v0_8-paper', 'seaborn-v0_8-paste
l', 'seaborn-v0_8-poster', 'seaborn-v0_8-talk', 'seaborn-v0_8-ticks', 'seaborn-v0_8-
white', 'seaborn-v0_8-whitegrid', 'tableau-colorblind10']
```

In [35]:
```
plt.style.use('seaborn-v0_8-bright')
```

In [36]:
```
#Adding a grid
x15=np.arange(1,5)
plt.plot(x15,x15*1.5,x15,x15*3.0,x15,x15/3.0)
plt.grid(True)
plt.show()
```

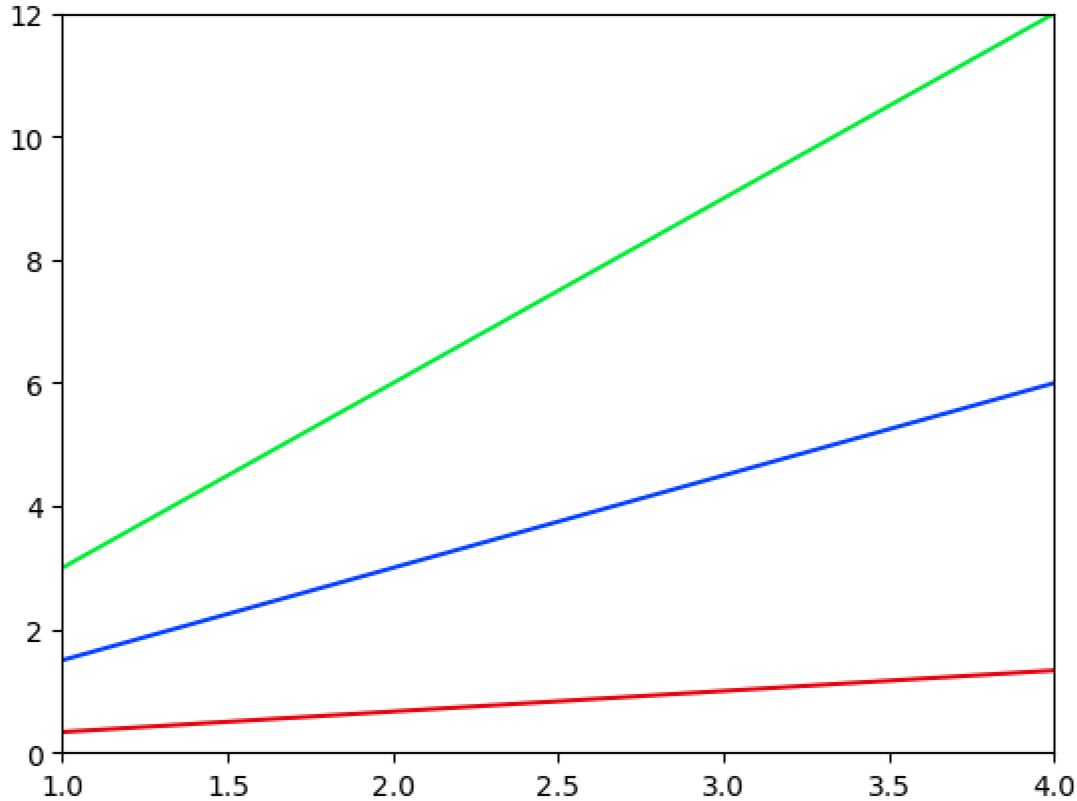


In [37]:
```
#Handling Axes
plt.plot(x15,x15*1.5,x15,x15*3.0,x15,x15/3.0)
plt.axis()
plt.axis([0,5,-1,13])
plt.show()
```

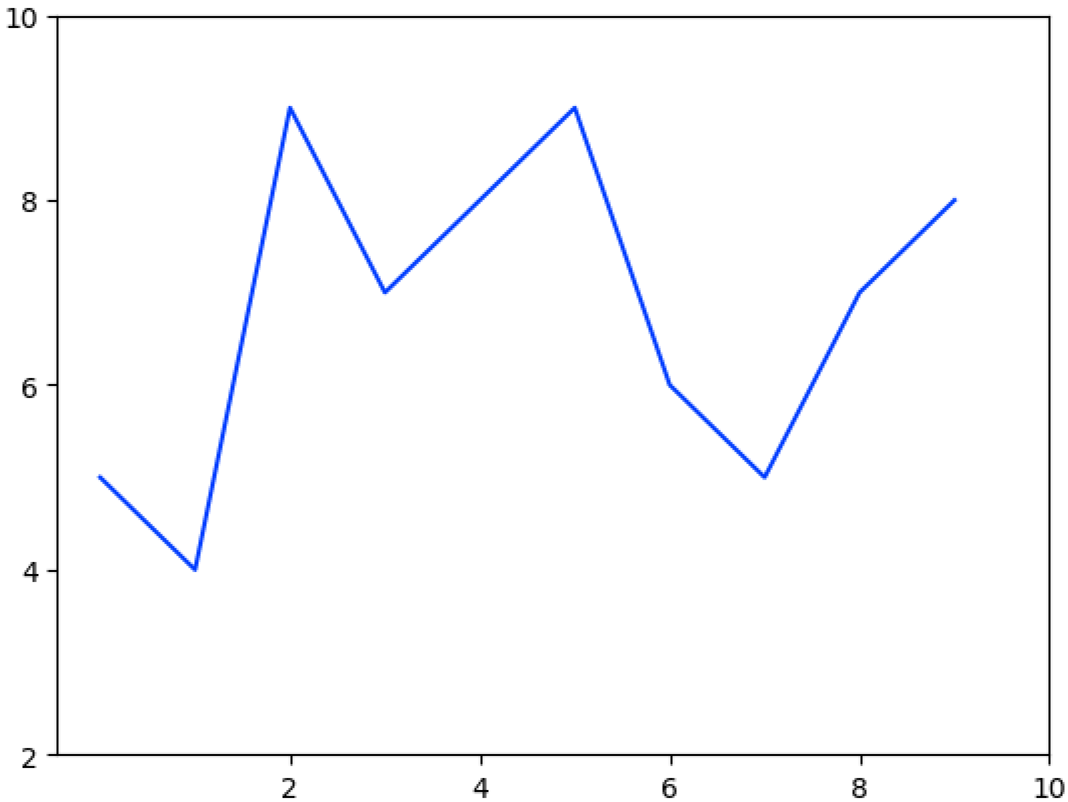


In [38]:
```
plt.plot(x15,x15*1.5,x15,x15*3.0,x15,x15/3.0)
plt.xlim([1.0,4.0])
plt.ylim([0.0,12.0])
```

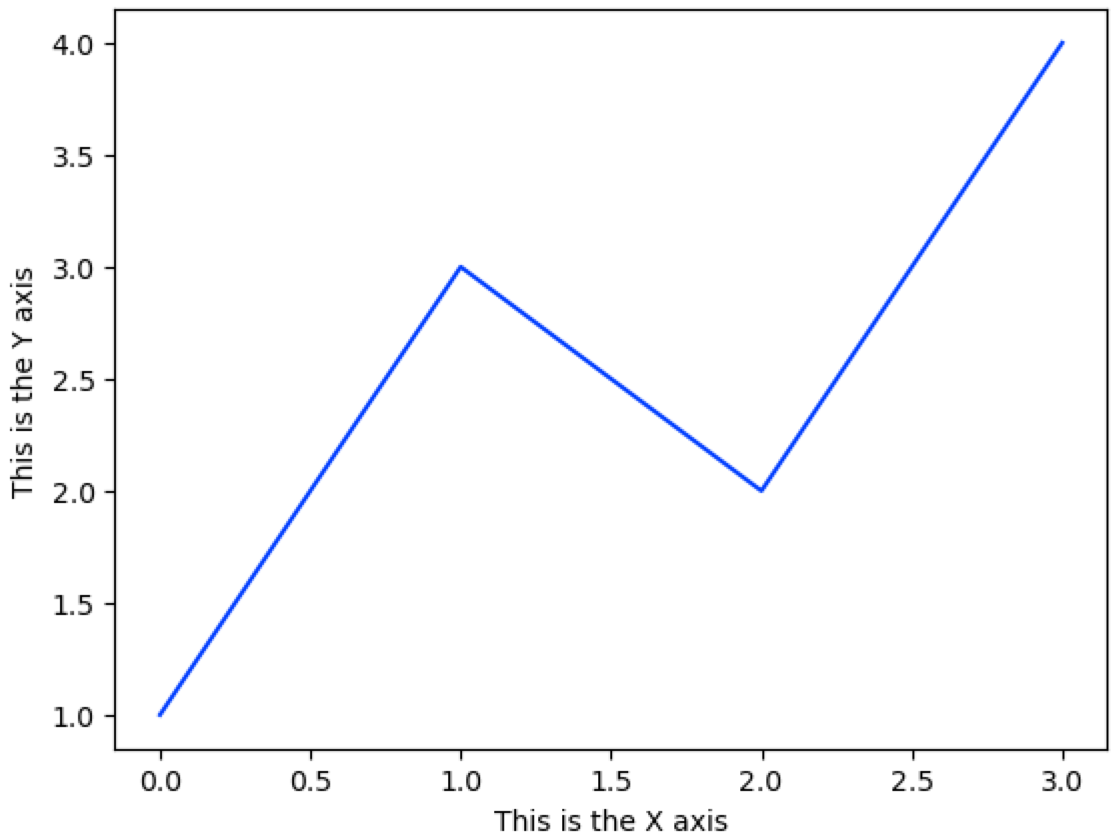
Out[38]: (0.0, 12.0)



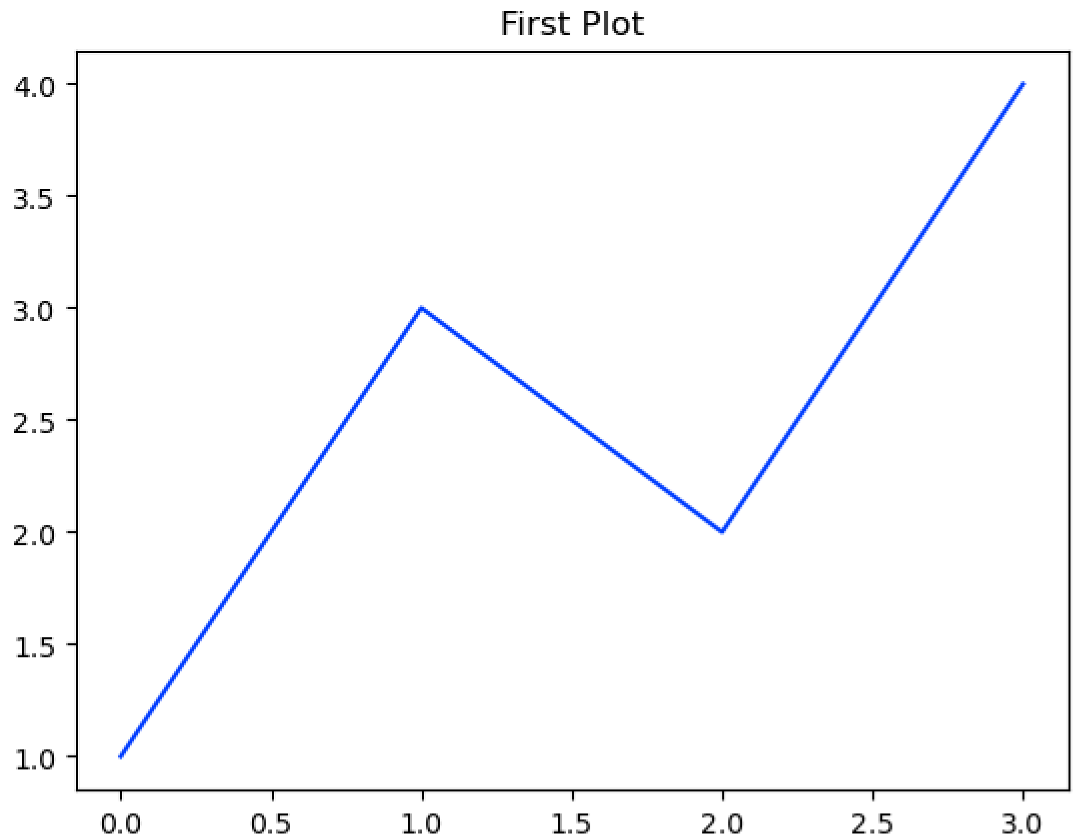
In [39]:
```
# Handling X and Y Ticks
u=[5,4,9,7,8,9,6,5,7,8]
plt.plot(u)
plt.xticks([2,4,6,8,10])
plt.yticks([2,4,6,8,10])
plt.show()
```



In [40]:
```
#Adding Labels
plt.plot([1, 3, 2, 4])
plt.xlabel('This is the X axis')
plt.ylabel('This is the Y axis')
plt.show()
```

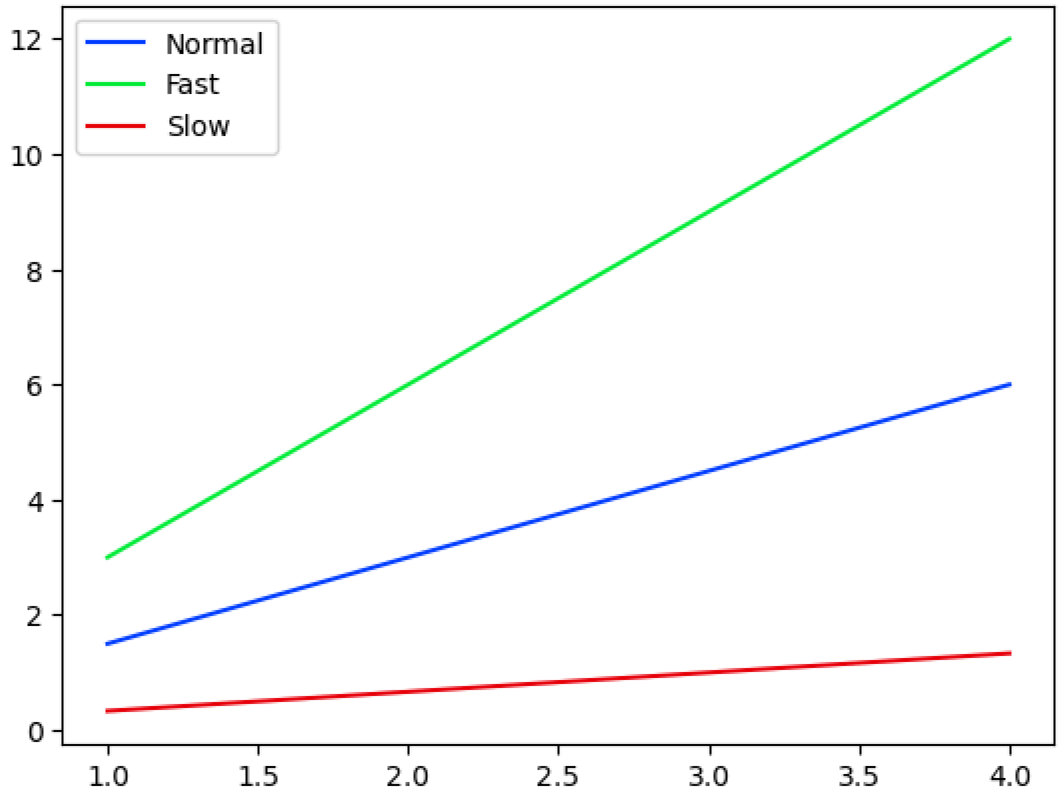


In [41]:
```
#Adding Title
plt.plot([1, 3, 2, 4])
plt.title('First Plot')
plt.show()
```



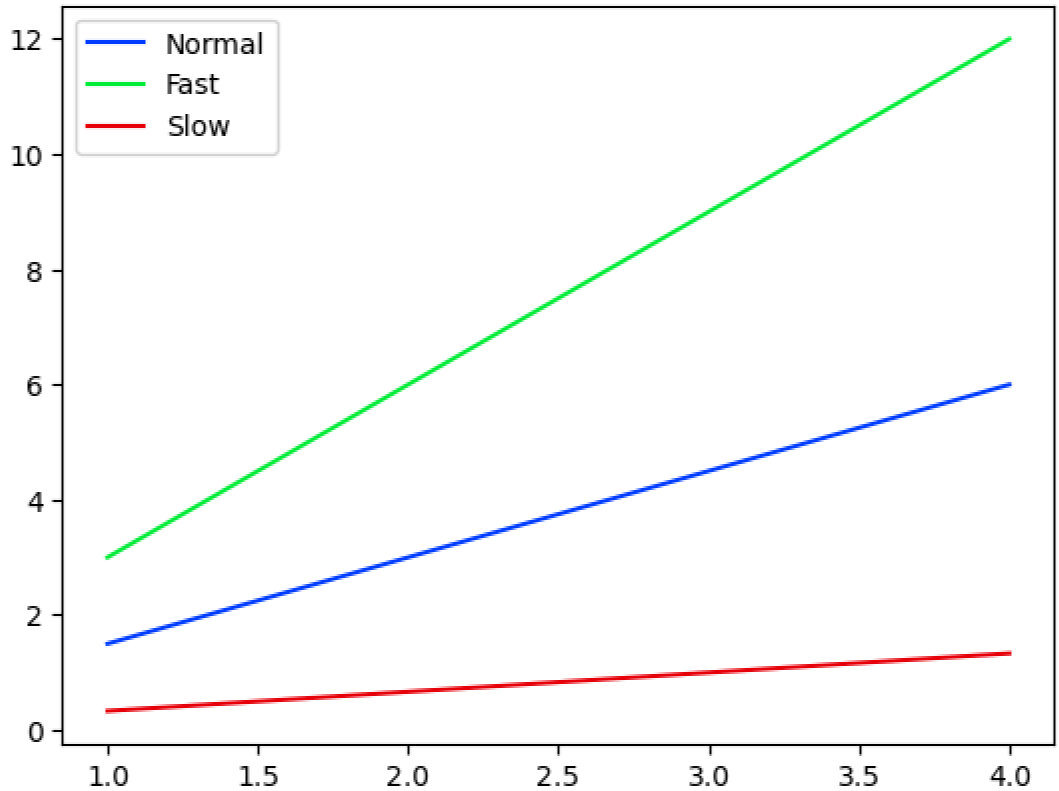
In [42]:
```
#Adding Legend
x15 = np.arange(1, 5)
fig, ax = plt.subplots()
ax.plot(x15, x15*1.5)
ax.plot(x15, x15*3.0)
ax.plot(x15, x15/3.0)
ax.legend(['Normal','Fast','Slow'])
```

Out[42]:
```
<matplotlib.legend.Legend at 0x1c05f523380>
```



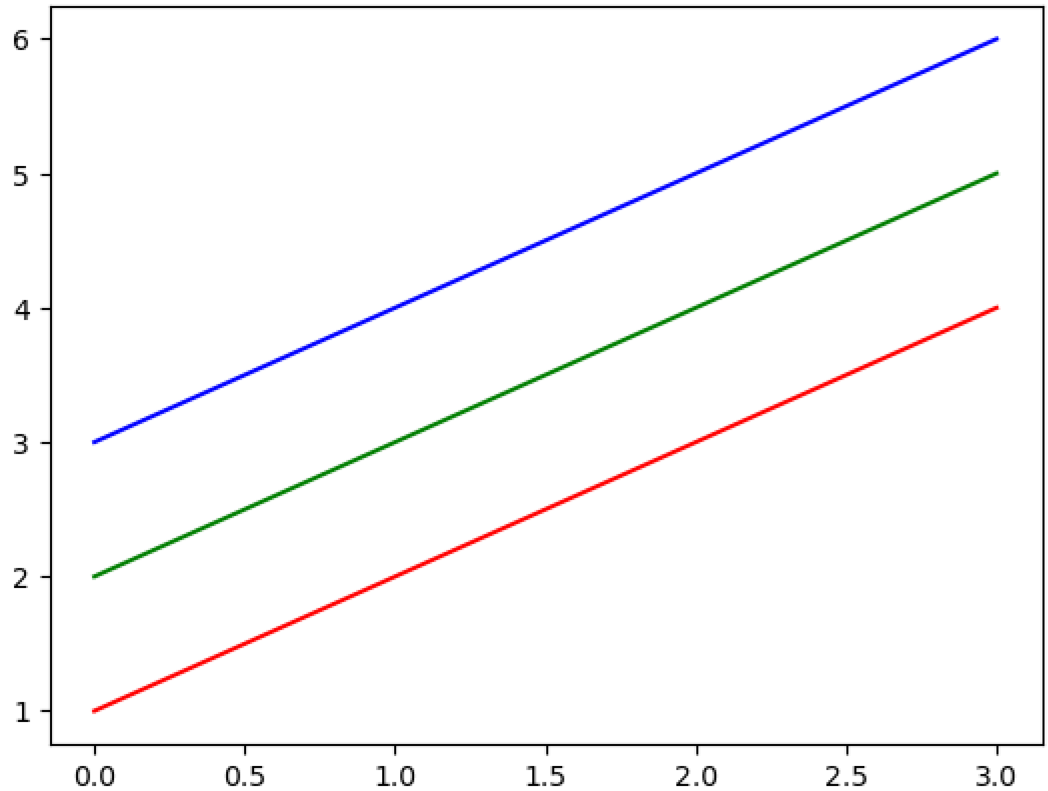
The above method follows the MATLAB API. It is prone to errors and unflexible if curves are added to or removed from the plot. It resulted in a wrongly labelled curve.

In [43]:
```
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();
```

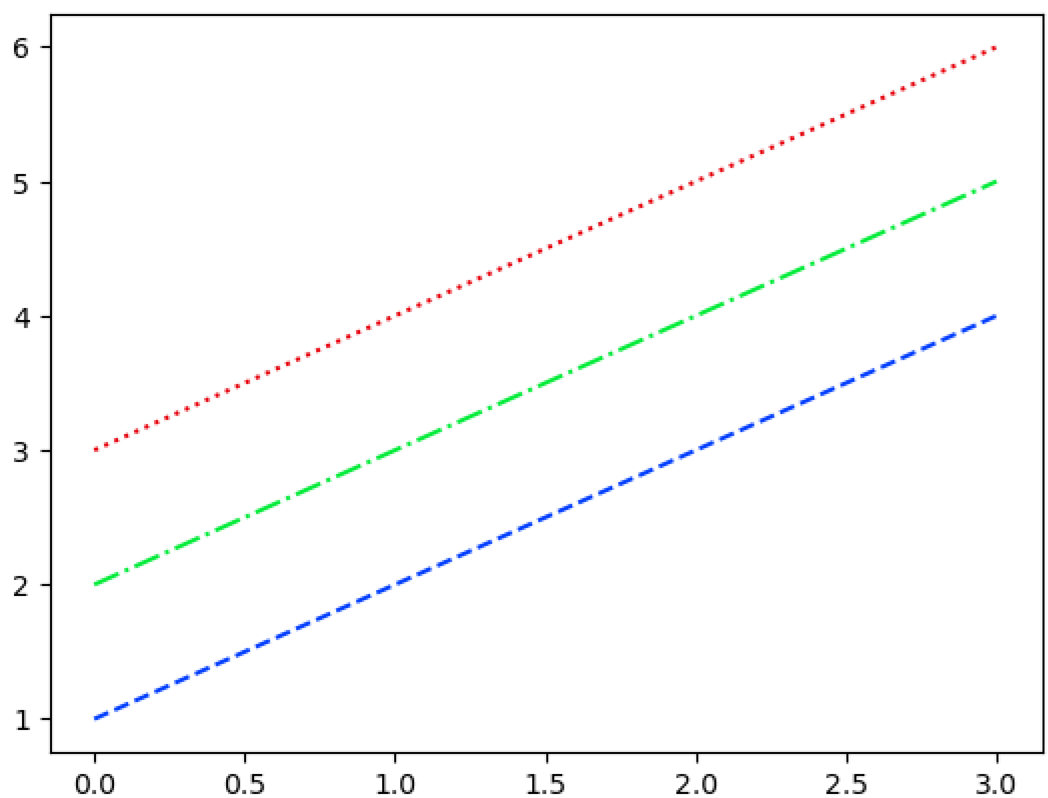


The legend function takes an optional keyword argument loc. It specifies the location of the legend to be drawn. The loc takes numerical codes for the various places the legend can be drawn. The most Out[131… In [134… common loc values are as follows:- ax.legend(loc=0) # let Matplotlib decide the optimal location 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 ax.legend(loc=5) # right ax.legend(loc=6) # center left ax.legend(loc=7) # center right ax.legend(loc=8) # lower center ax.legend(loc=9) # upper center ax.legend(loc=10) # center

In [44]:
```
#Control Colors
x16 = np.arange(1, 5)
plt.plot(x16, 'r')
plt.plot(x16+1, 'g')
plt.plot(x16+2, 'b')
plt.show()
```



In [45]:
```
#Controlines styles
x16 = np.arange(1, 5)
plt.plot(x16, '--', x16+1, '-.', x16+2, ':')
plt.show()
```



In [ ]: