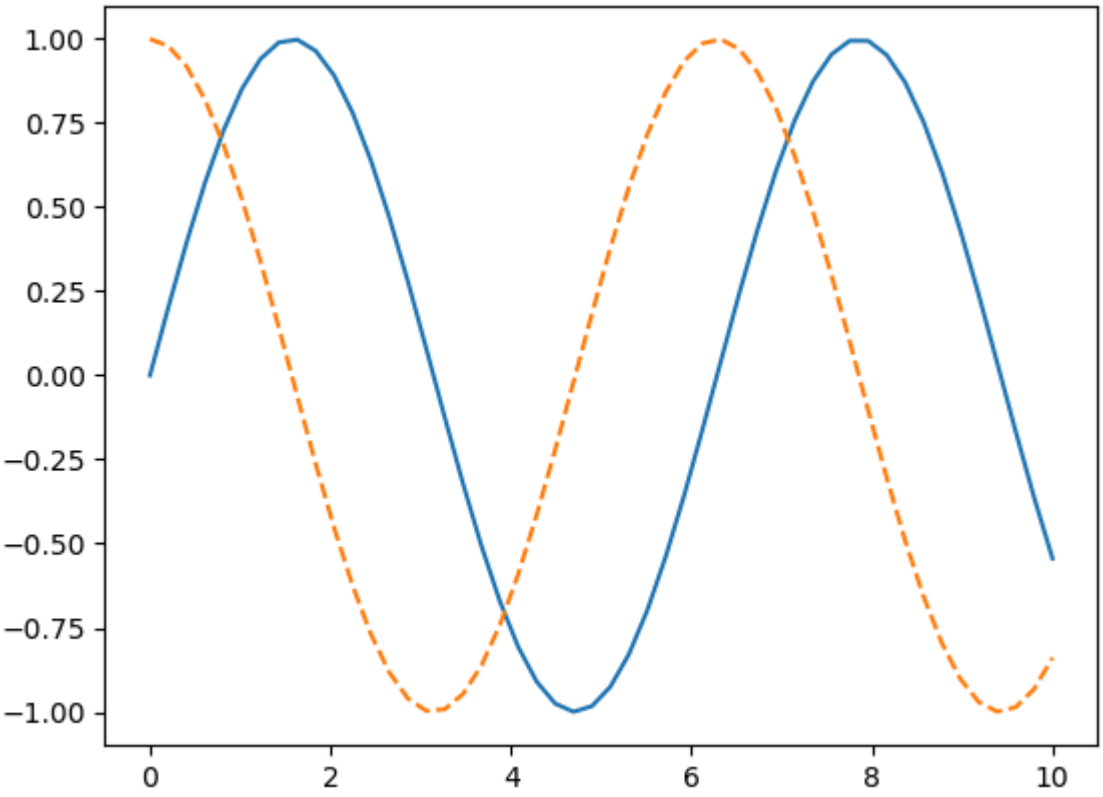
MATPLOTLIB

In [1]:
```
import numpy as np
```

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

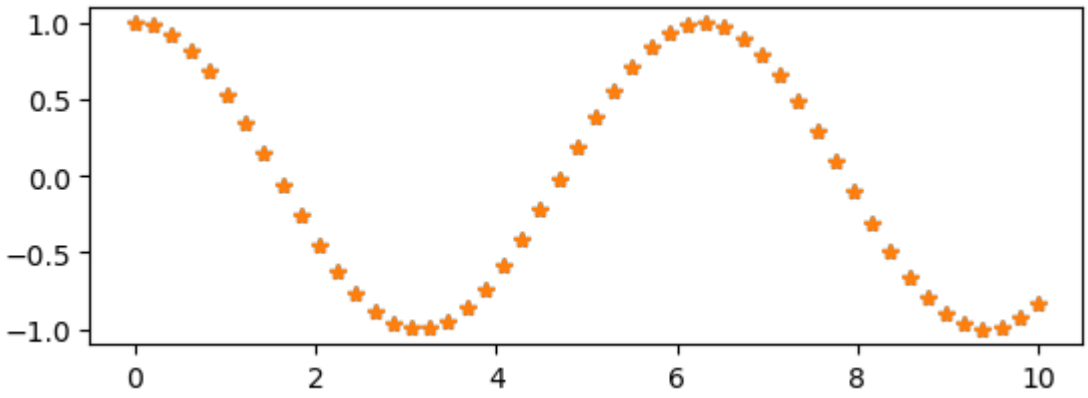
1. Displaying Plots in Matplotlib

In [4]:
```
%matplotlib inline
x1 = np.linspace(0,10,50)  #fig = plt.figure()

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

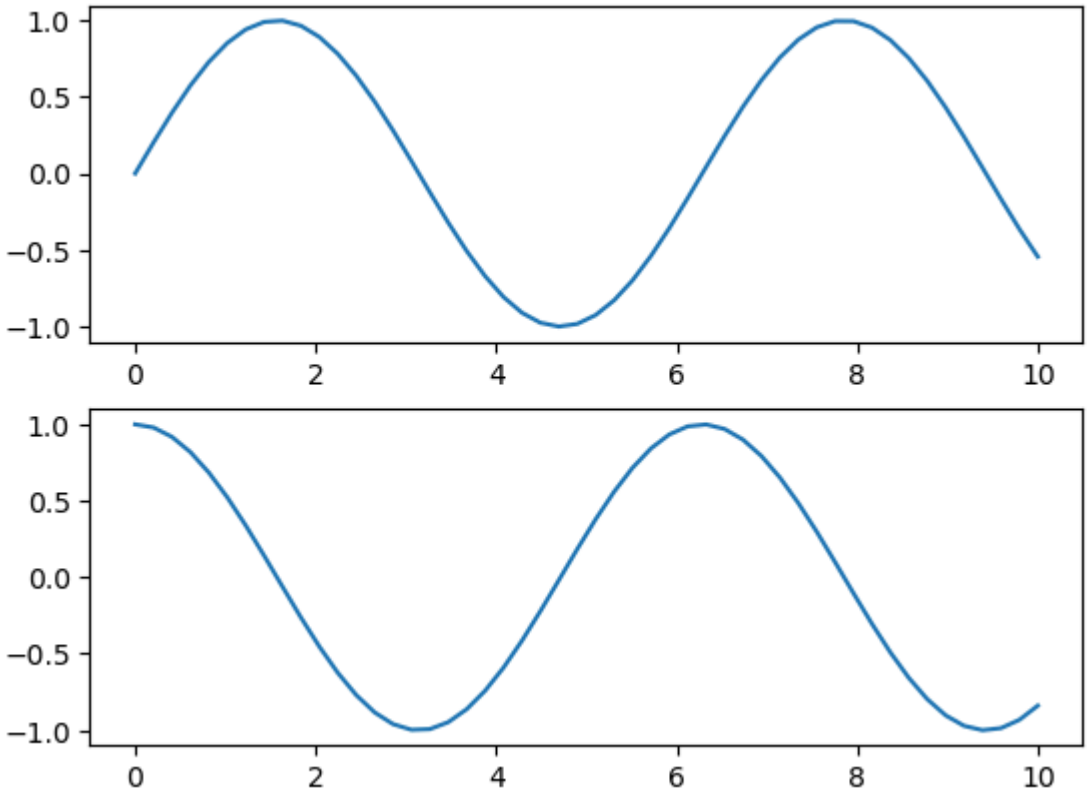


2. Pyplot API

In [6]:
```
# 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 [7]:
```
# 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))
plt.show()
```



In [8]:
```
print(plt.gcf())              # get current figure information
```

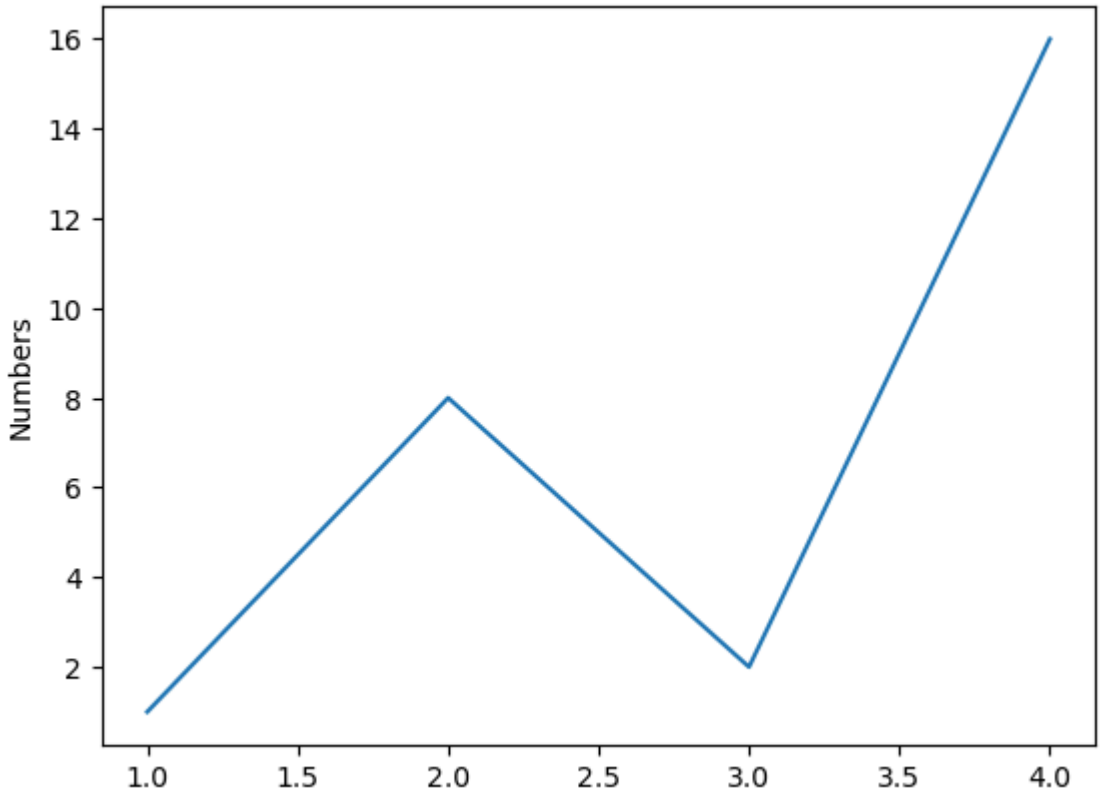
```
Figure(640x480)
```

In [9]:
```
print(plt.gca())                  # get current axis
```

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

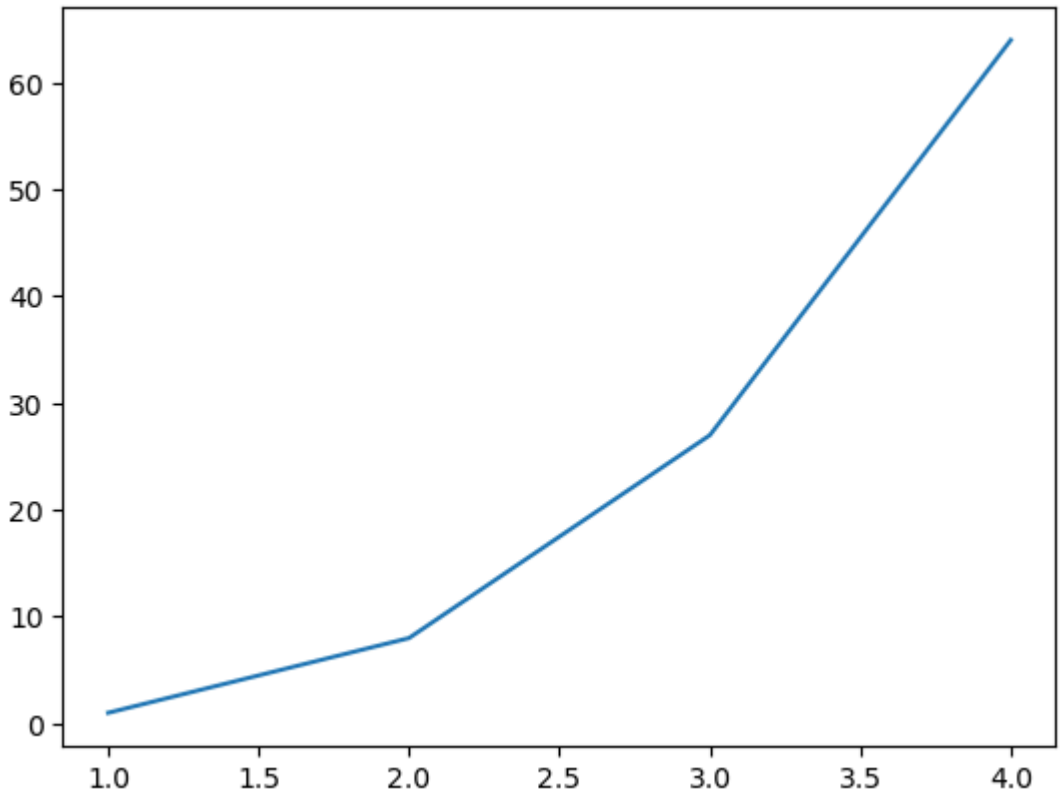
Visualization with pyplot

In [10]:
```
plt.plot([1,2,3,4], [1,8,2,16])
plt.ylabel('Numbers')
plt.show()
```



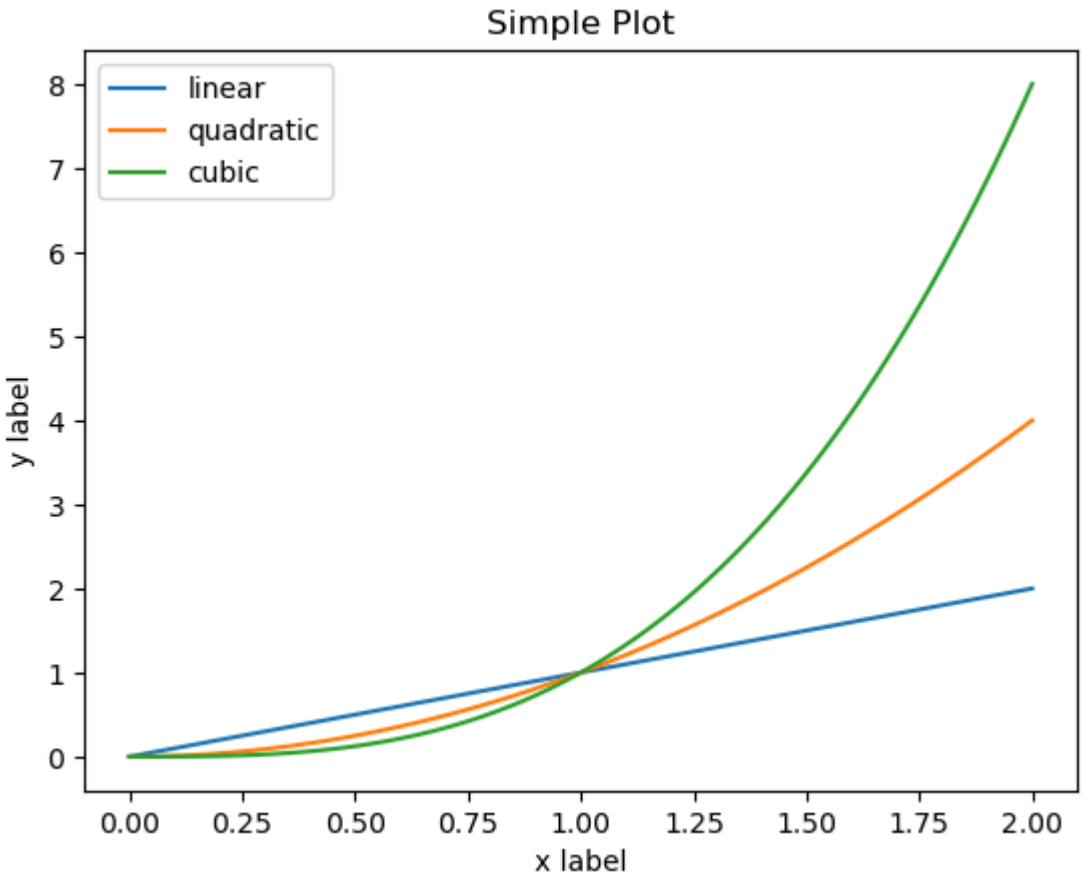
plot()-A versatile command

In [11]:
```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 8, 27, 64])
plt.show()
```



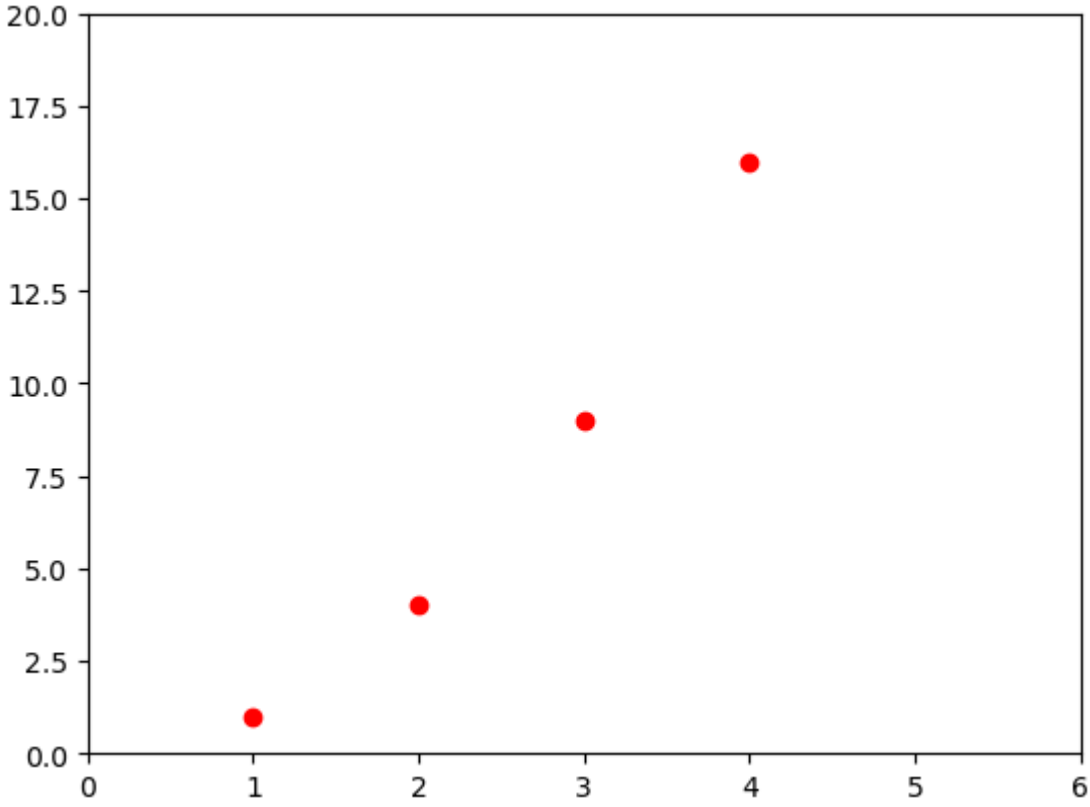
State-machine interface

In [12]:
```
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()
```



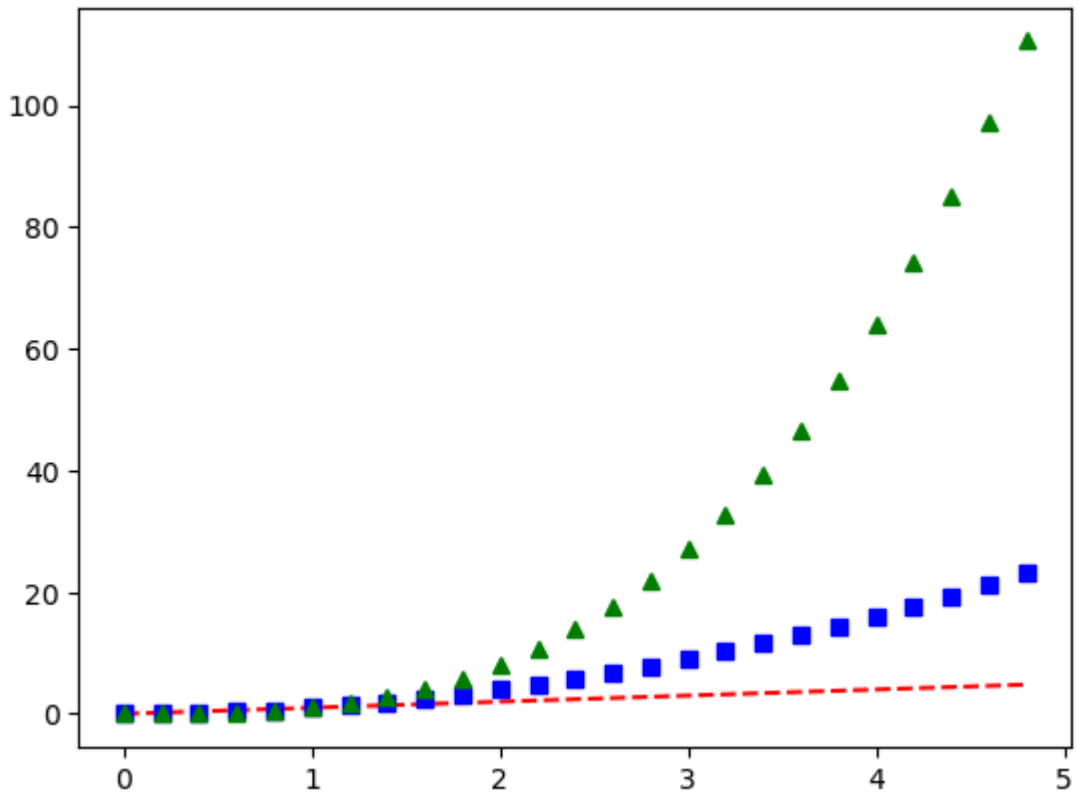
Formatting the style of plot

In [13]:
```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```



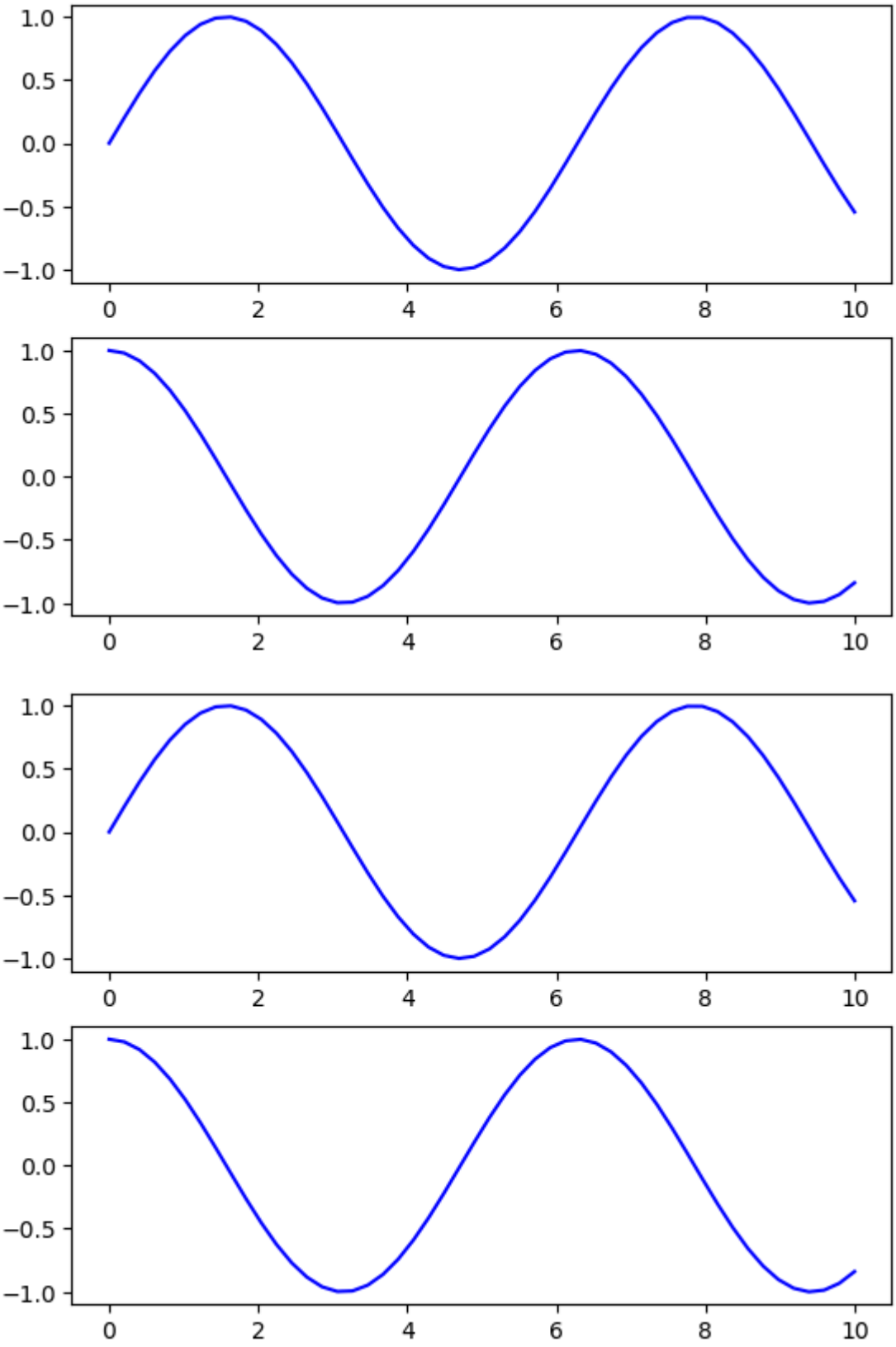
Working with NumPy arrays

In [14]:
```
# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



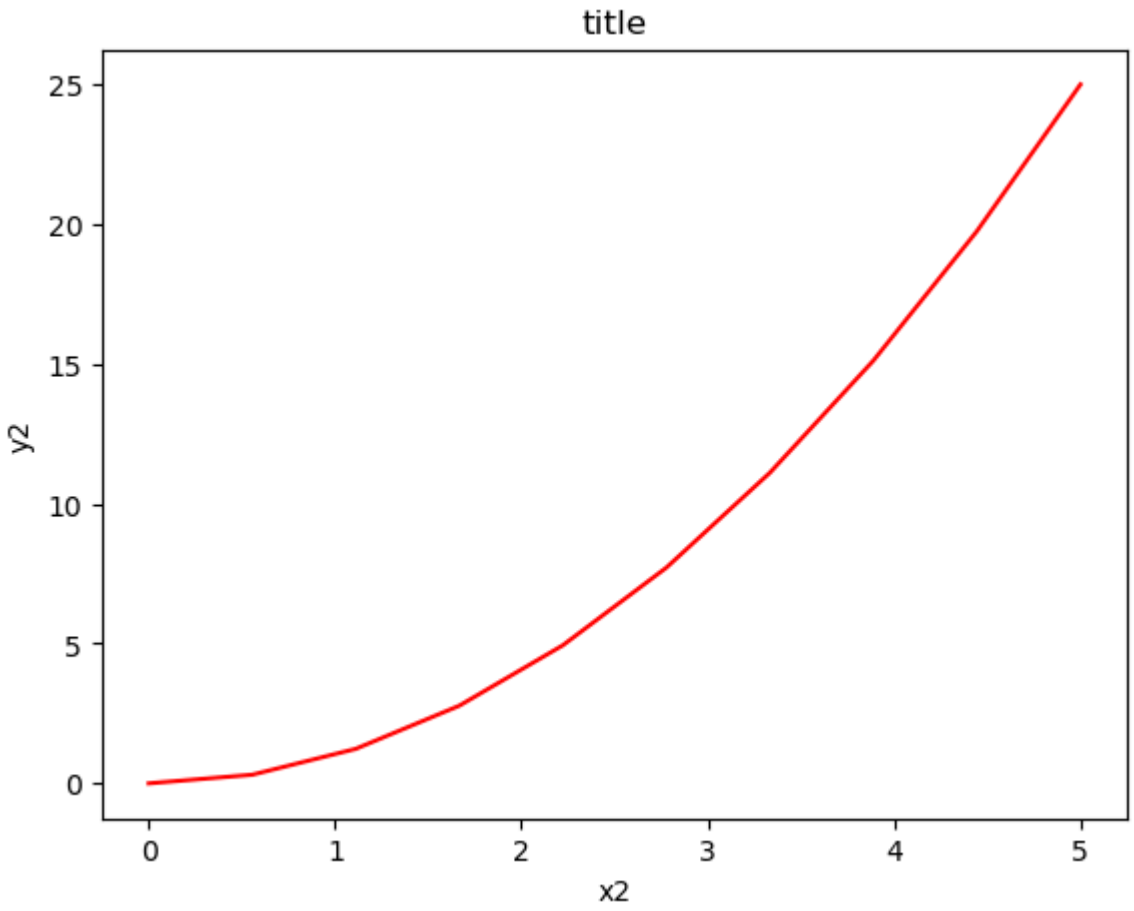
3.Object-Oriented API

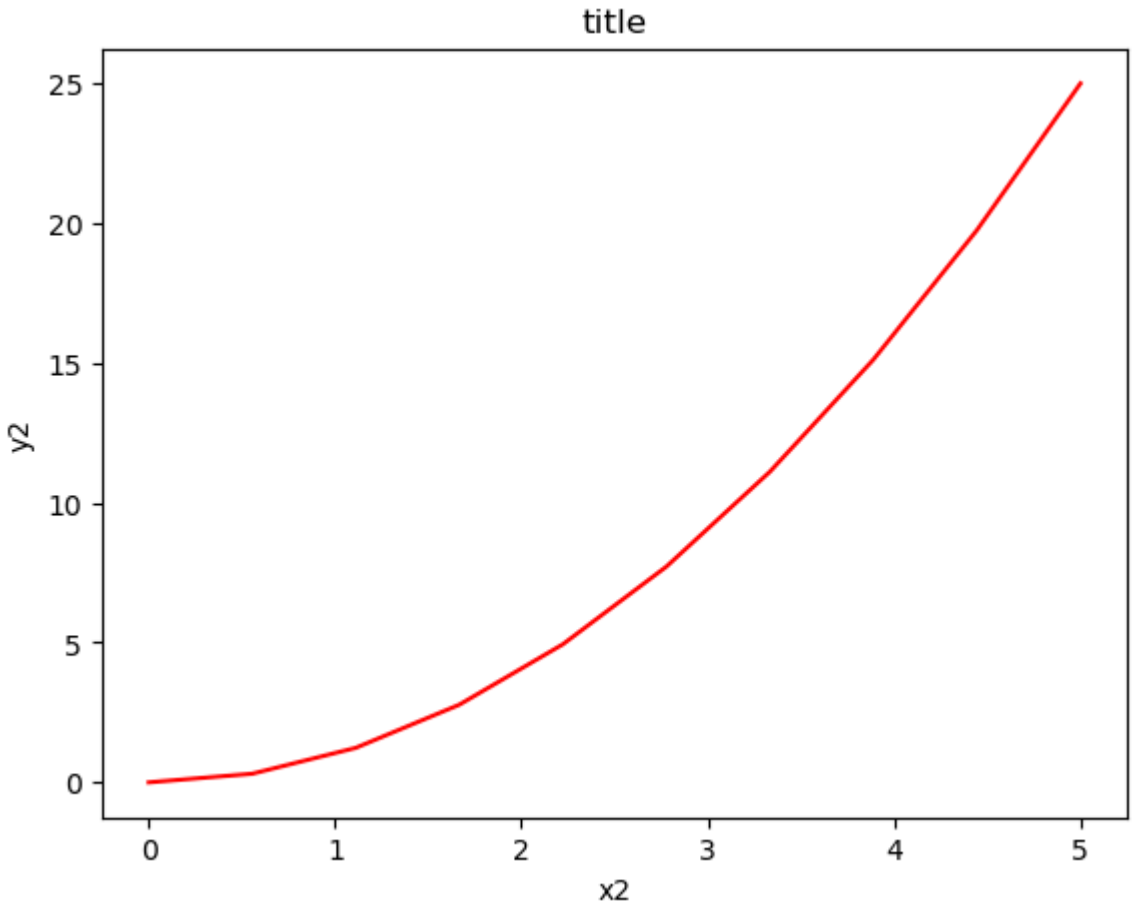
In [16]:
```
# ax will be an array of two Axes objects
fig, ax = plt.subplots(2)

ax[0].plot(x1, np.sin(x1), 'b-')
ax[1].plot(x1, np.cos(x1), 'b-');
plt.show()
```



Objects and Reference

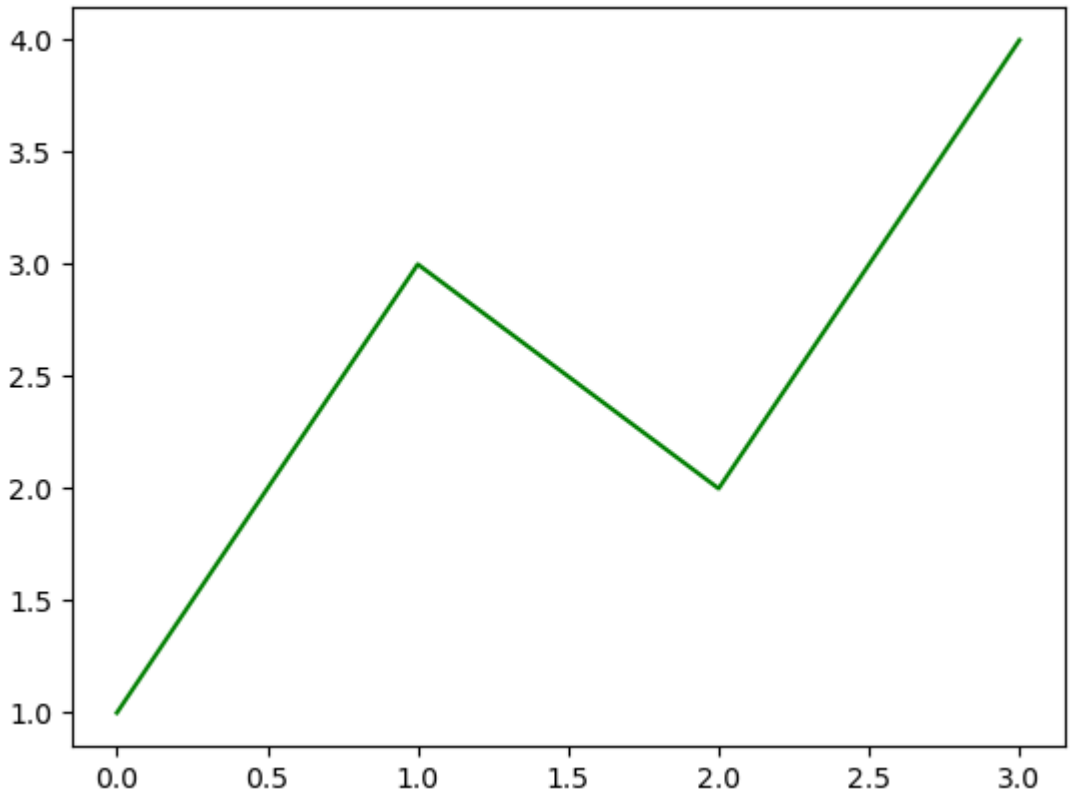
In [18]:
```
fig = plt.figure()

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()
```





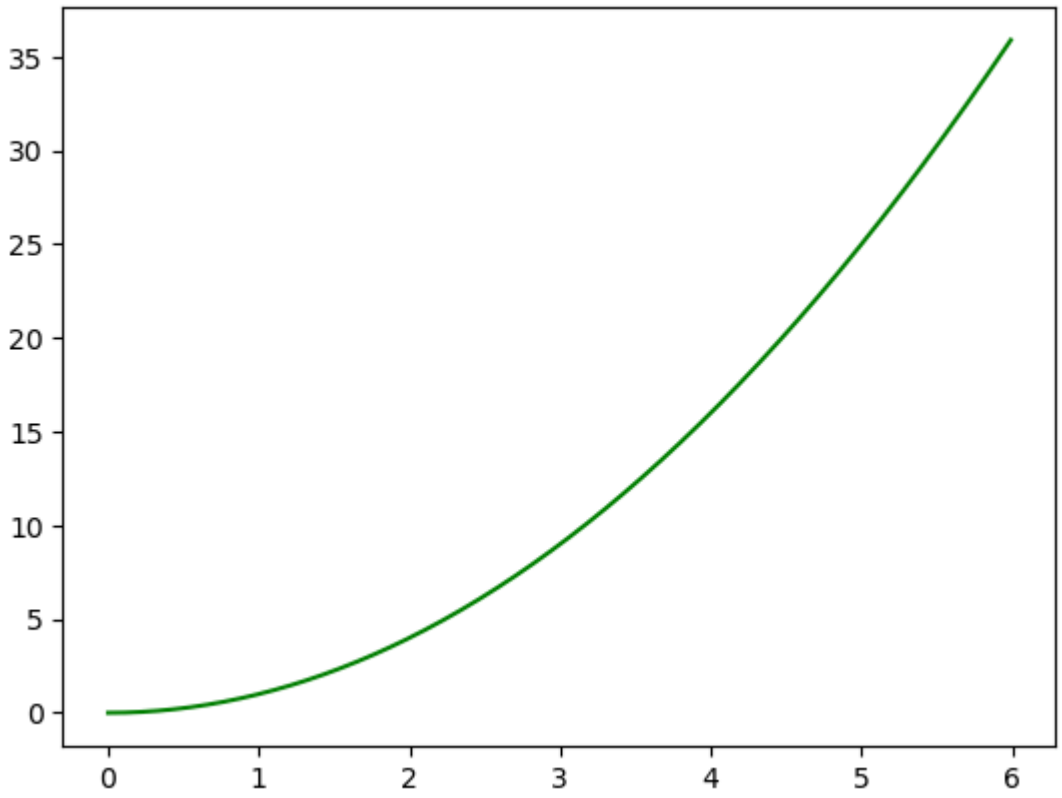
4.First plot with Matplotlib

In [22]:
```
plt.plot([1,3,2,4],'g-')
plt.show()
```



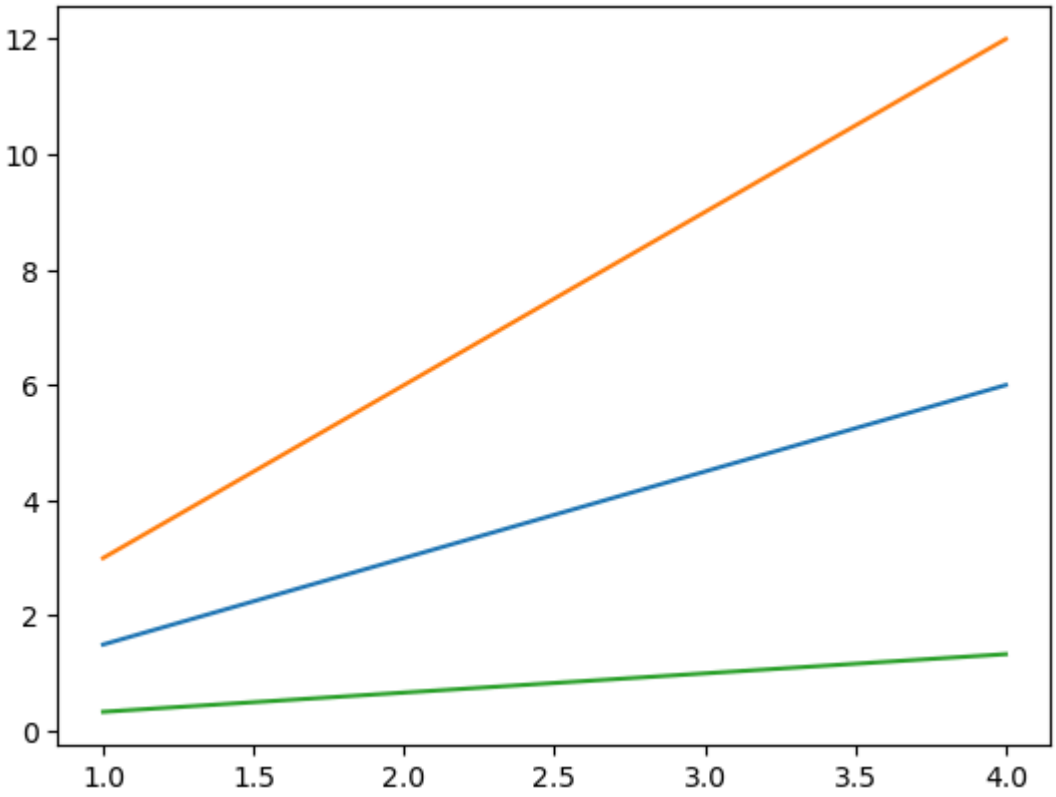
Specify both Lists

In [23]:
```
x3 = np.arange(0.0,6.0,0.01)
plt.plot(x3,[xi**2 for xi in x3],'g-')
plt.show()
```



5.Multiline Plots

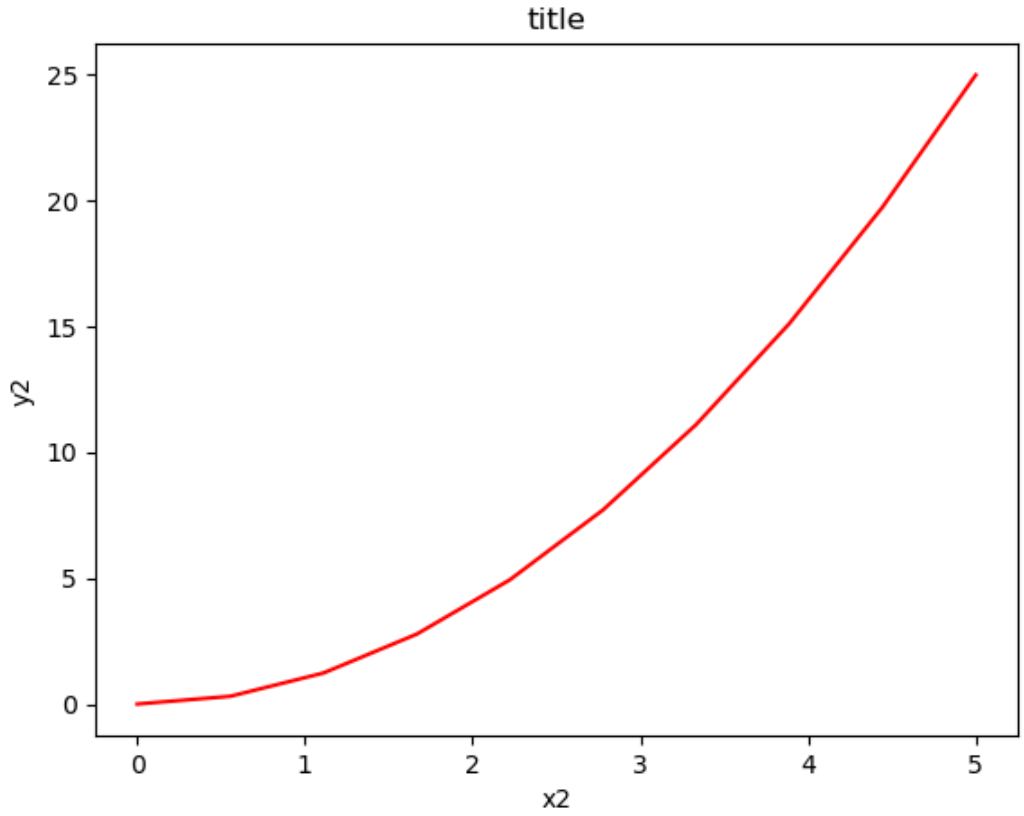
In [25]:
```
x4 = range(1,5)

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()
```



6.Saving the Plot

In [26]:
```
                    # Saving the figure
fig.savefig('plot1.png')

from IPython.display import Image

Image('plot1.png')
```

Out[26]:

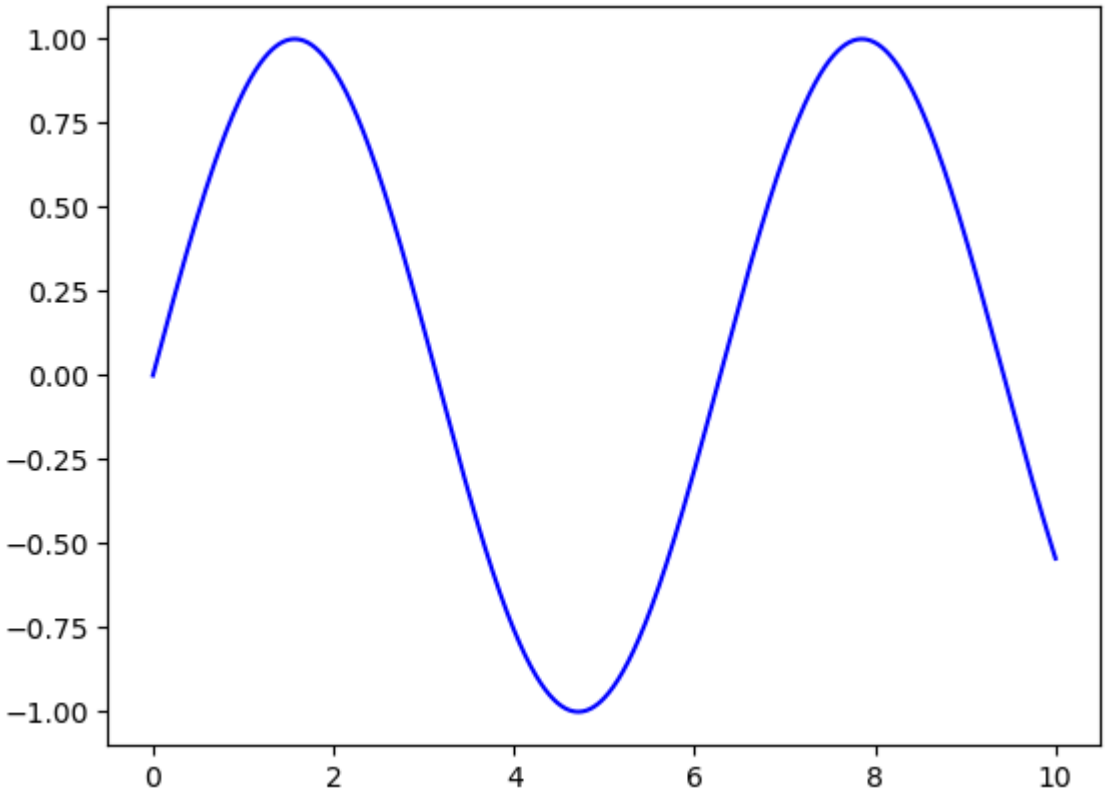


In [27]:
```
# Explore supported file formats
fig.canvas.get_supported_filetypes()
```

Out[27]:
```
{'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'}
```

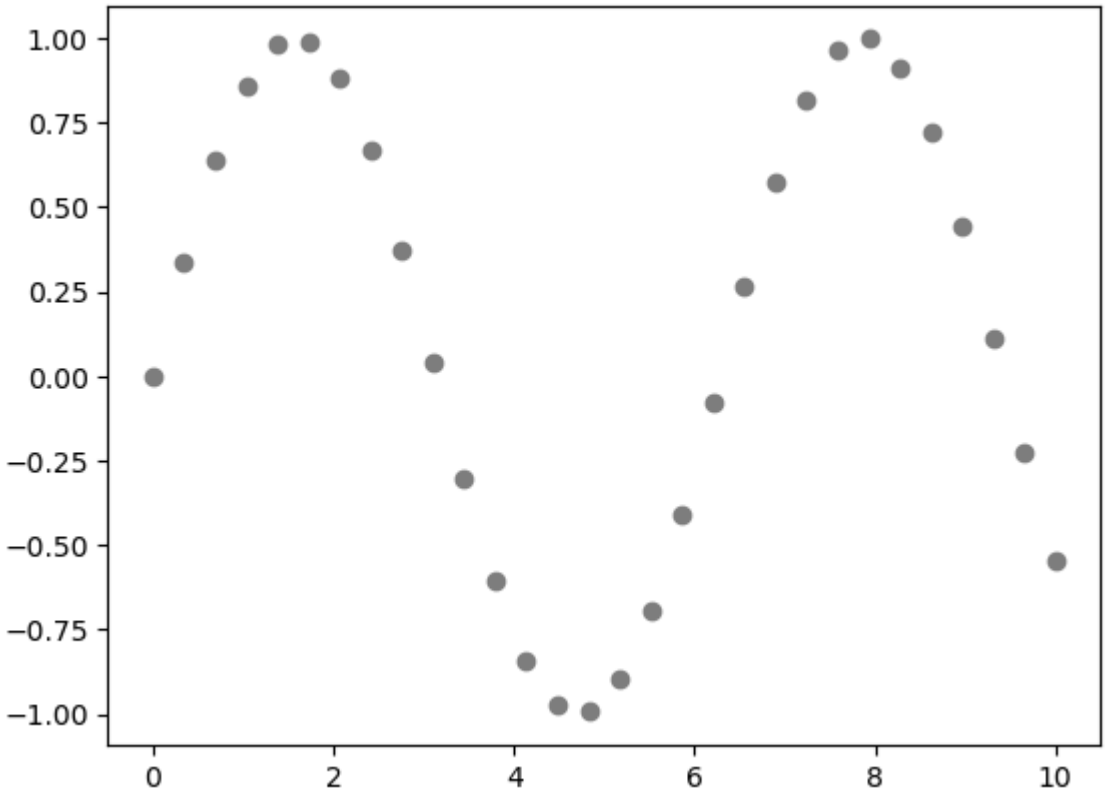
7.Line Plot

In [28]:
```
fig = plt.figure()

ax = plt.axes()

x5 = np.linspace(0, 10, 1000)
# Plot the sinusoid function
ax.plot(x5, np.sin(x5), 'b-')
plt.show()
```



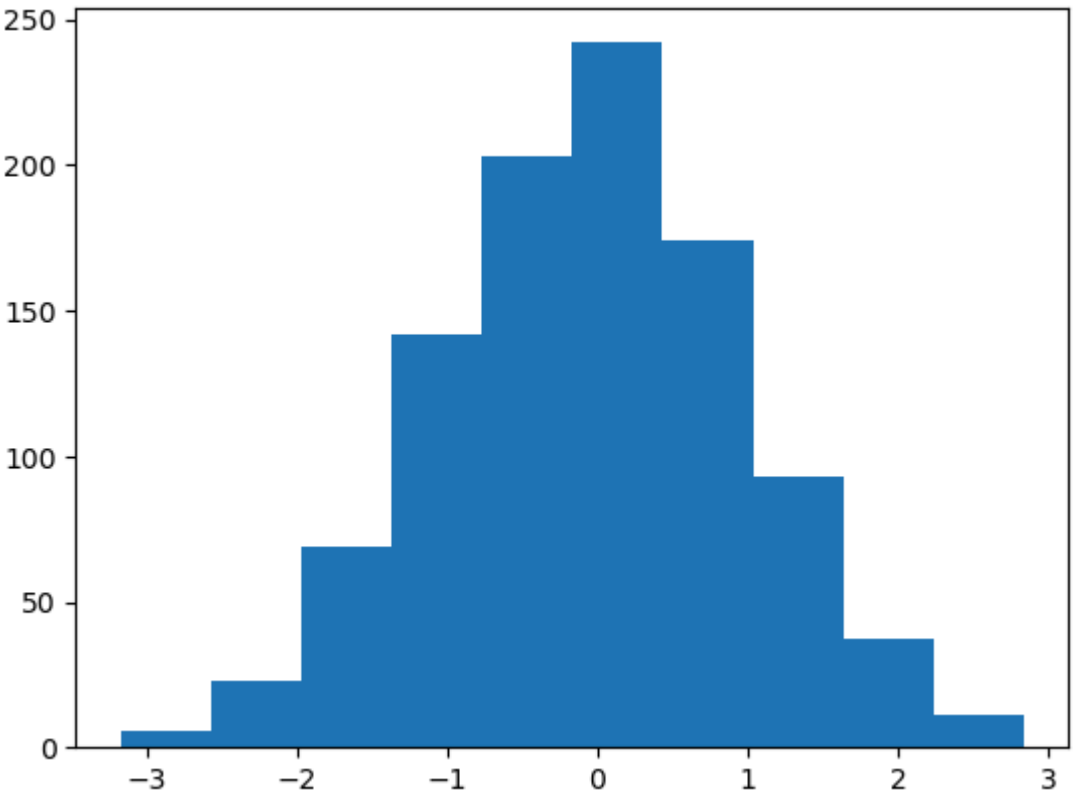
8.Scatter Plot

In [31]:
```
x7 = np.linspace(0, 10, 30)

y7 = np.sin(x7)

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



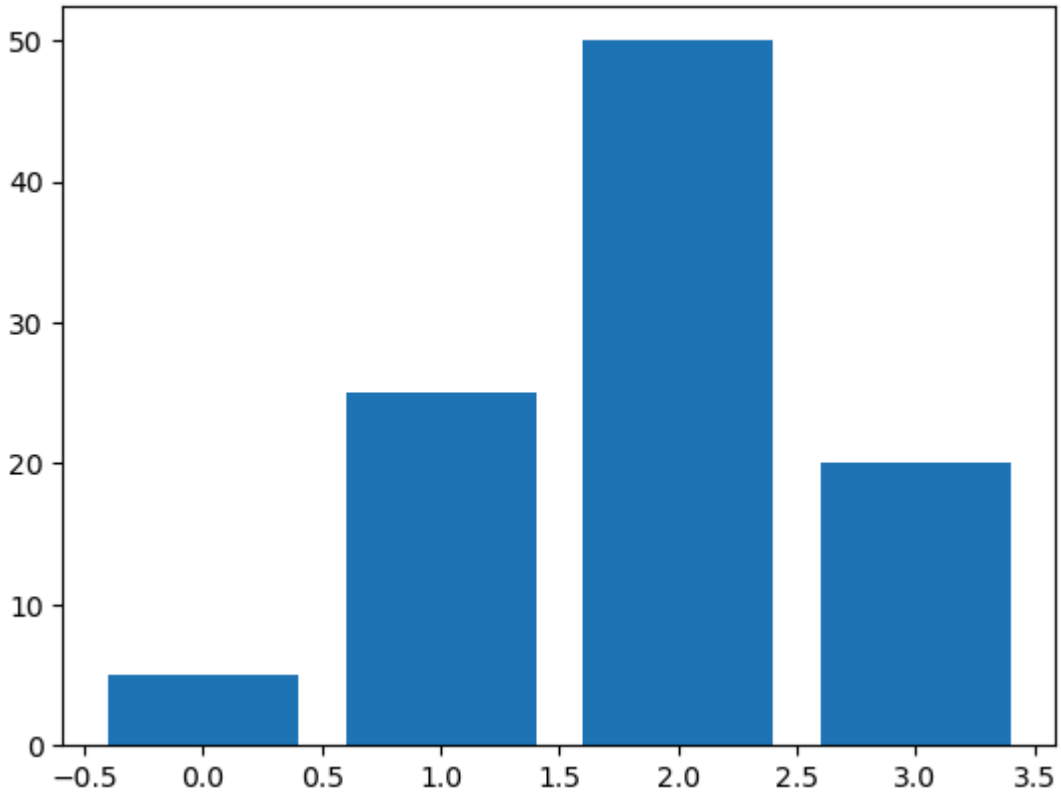
9.Histogram

In [34]:
```
data1 = np.random.randn(1000)

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



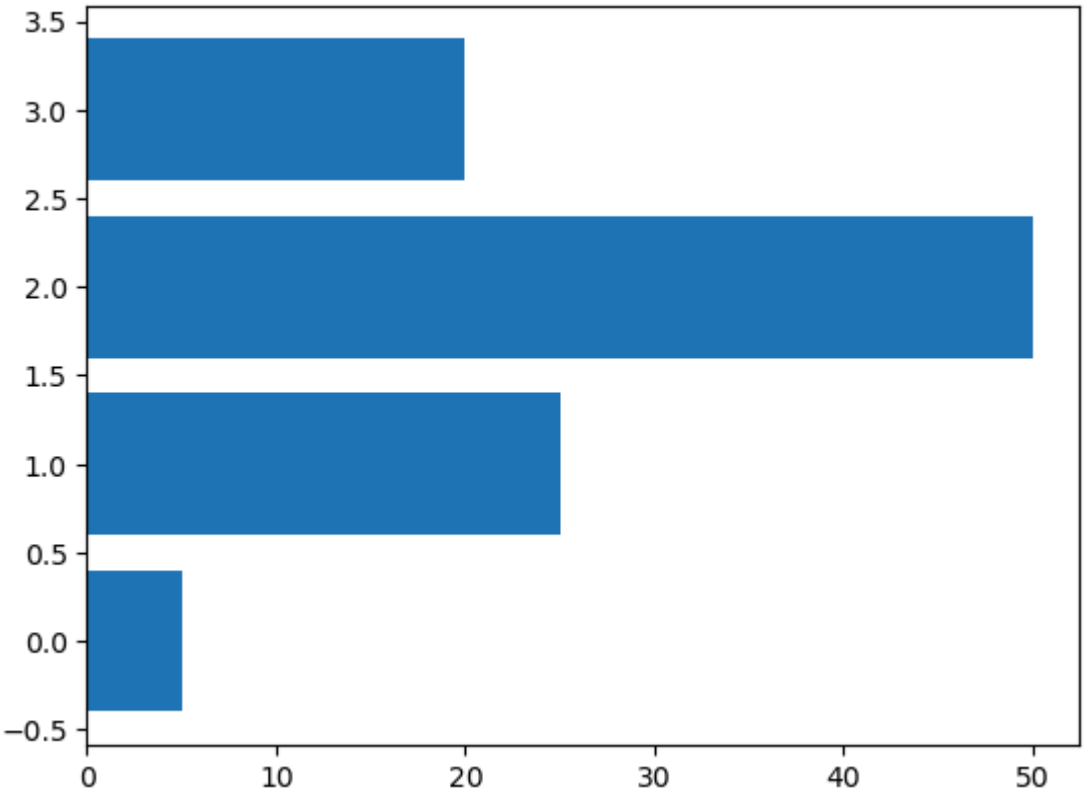
10.Bar Chart

In [35]:
```
data2 = [5. , 25. , 50. , 20.]

plt.bar(range(len(data2)), data2)

plt.show()
```



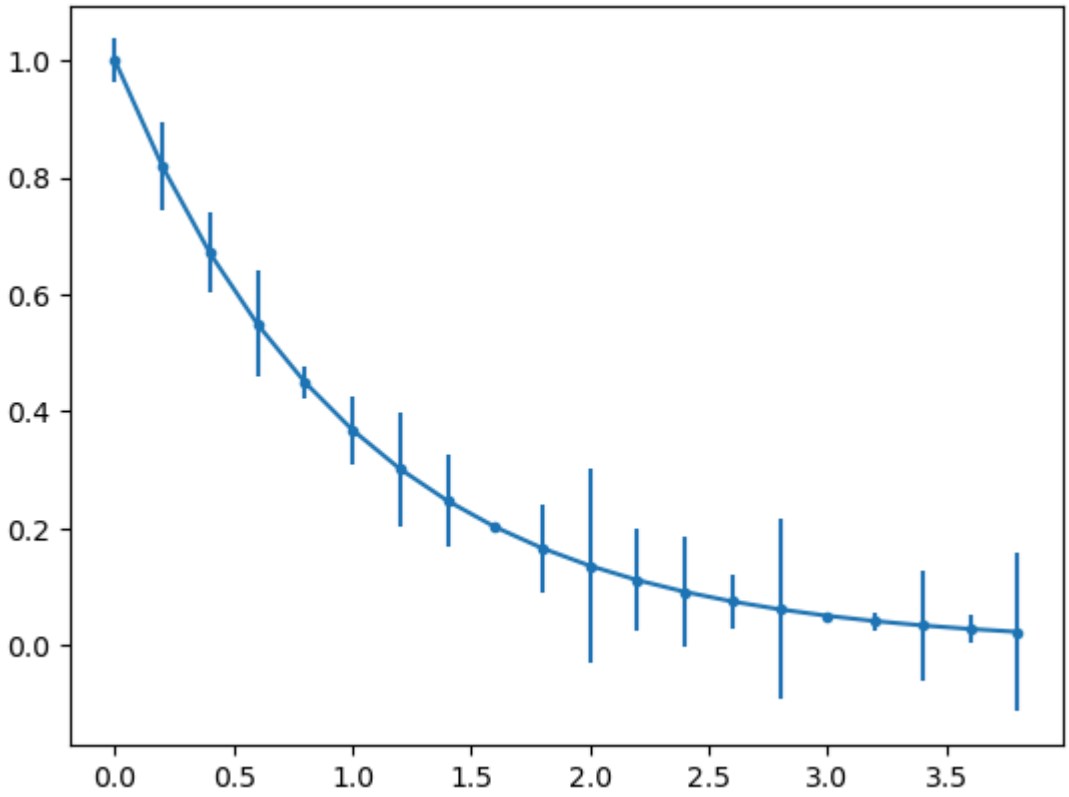
11.Horizontal Bar Chart

In [36]:
```
data2 = [5. , 25. , 50. , 20.]

plt.barh(range(len(data2)), data2)

plt.show()
```



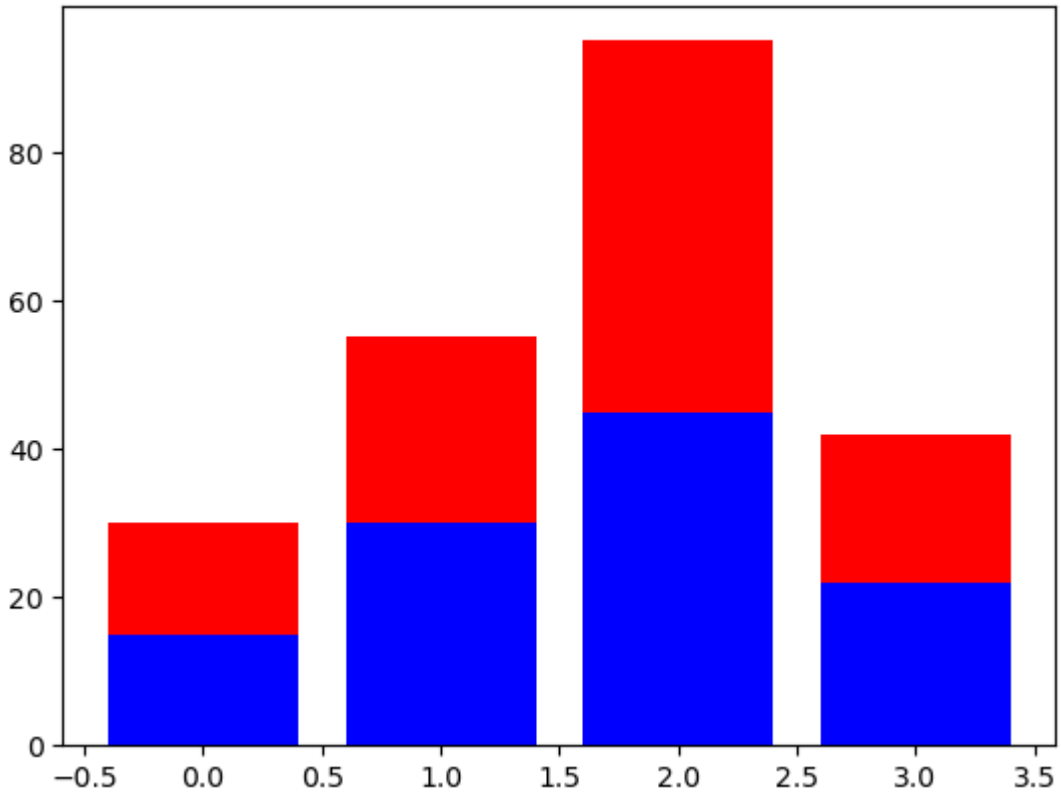
12.Error Bar Chart

In [38]:
```
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()
```



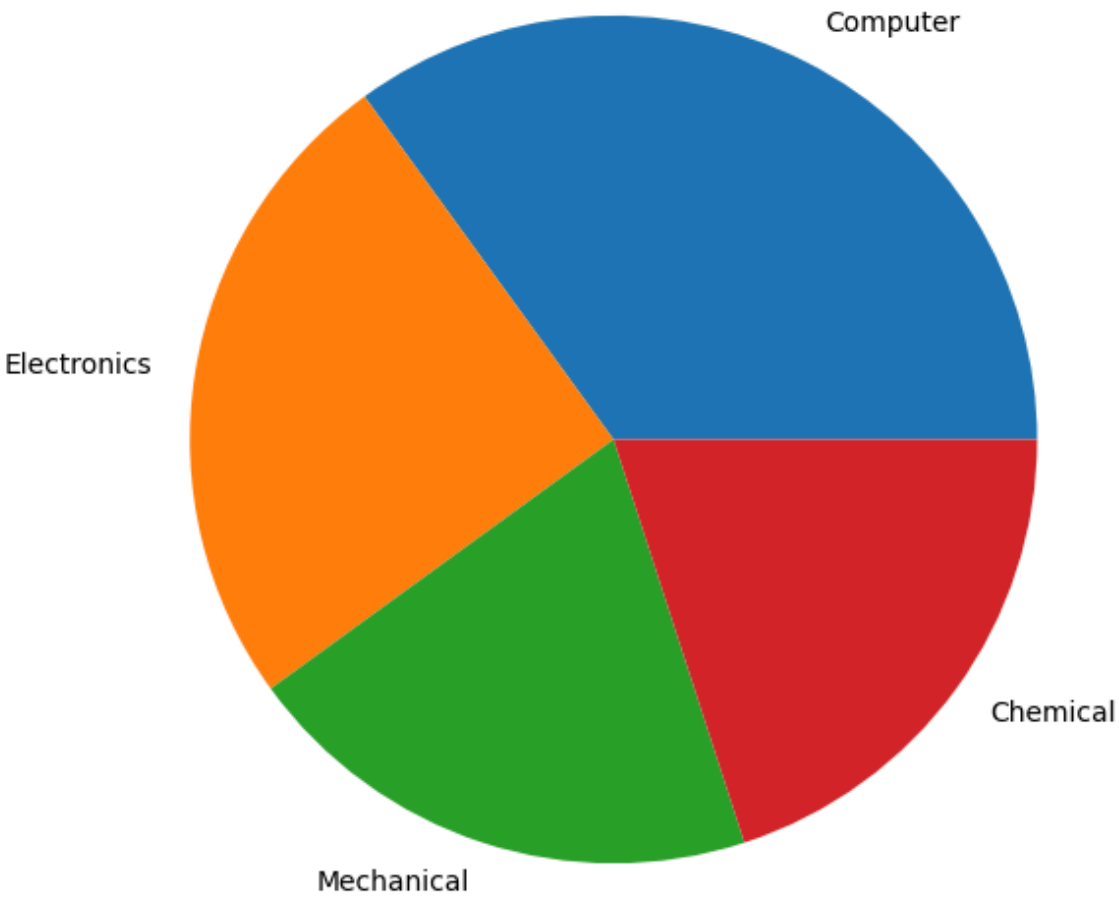
13.Stacked Bar Chart

In [39]:
```
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()
```



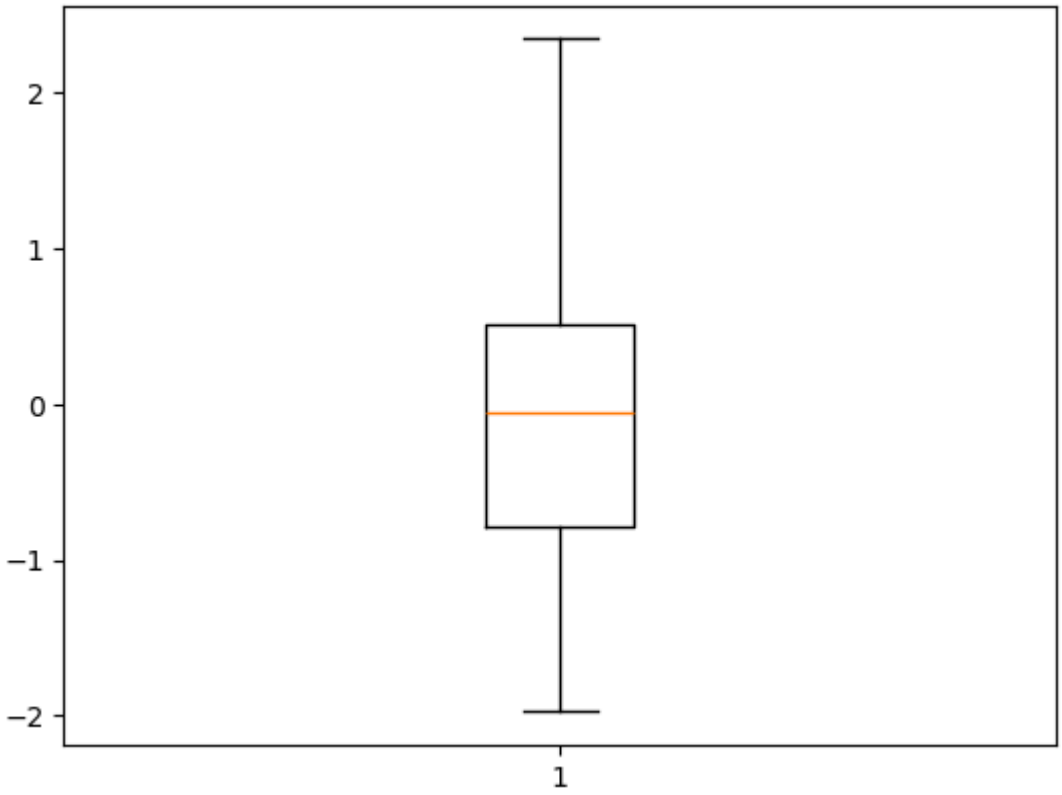
14.Pie Chart

In [40]:
```
plt.figure(figsize=(7,7))

x10 = [35, 25, 20, 20]

labels = ['Computer', 'Electronics', 'Mechanical', 'Chemical']

plt.pie(x10, labels=labels);

plt.show()
```



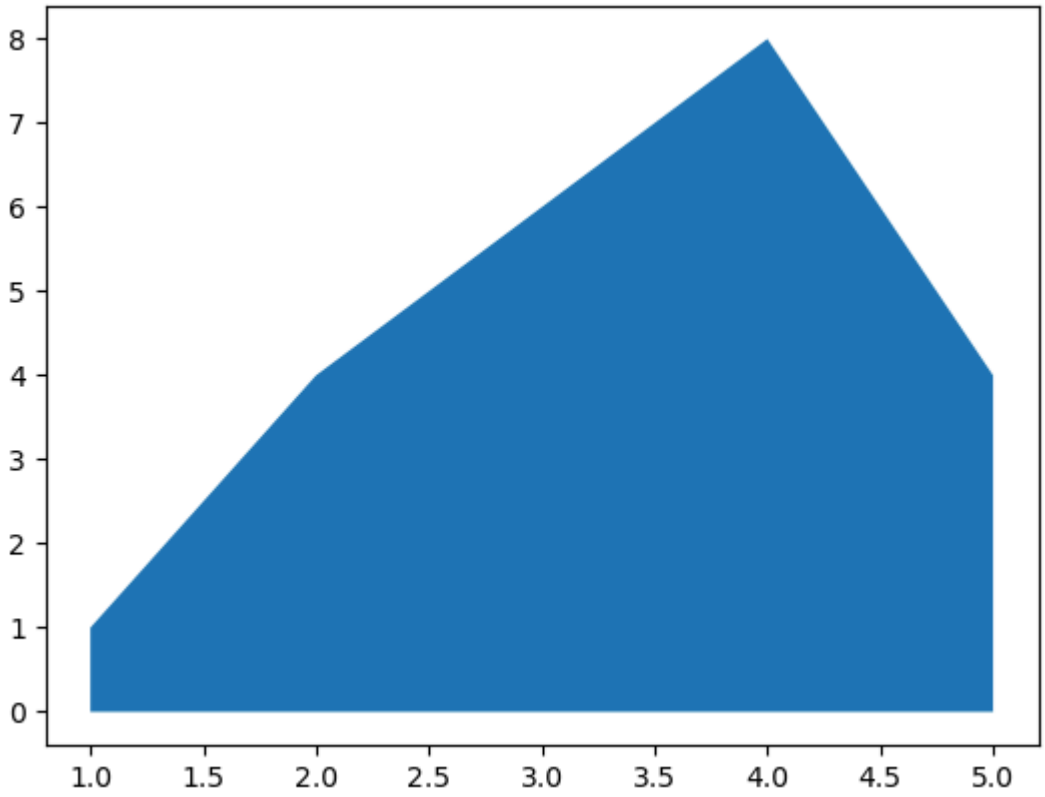
15.Boxplot

In [41]:
```
data3 = np.random.randn(100)

plt.boxplot(data3)

plt.show()
```



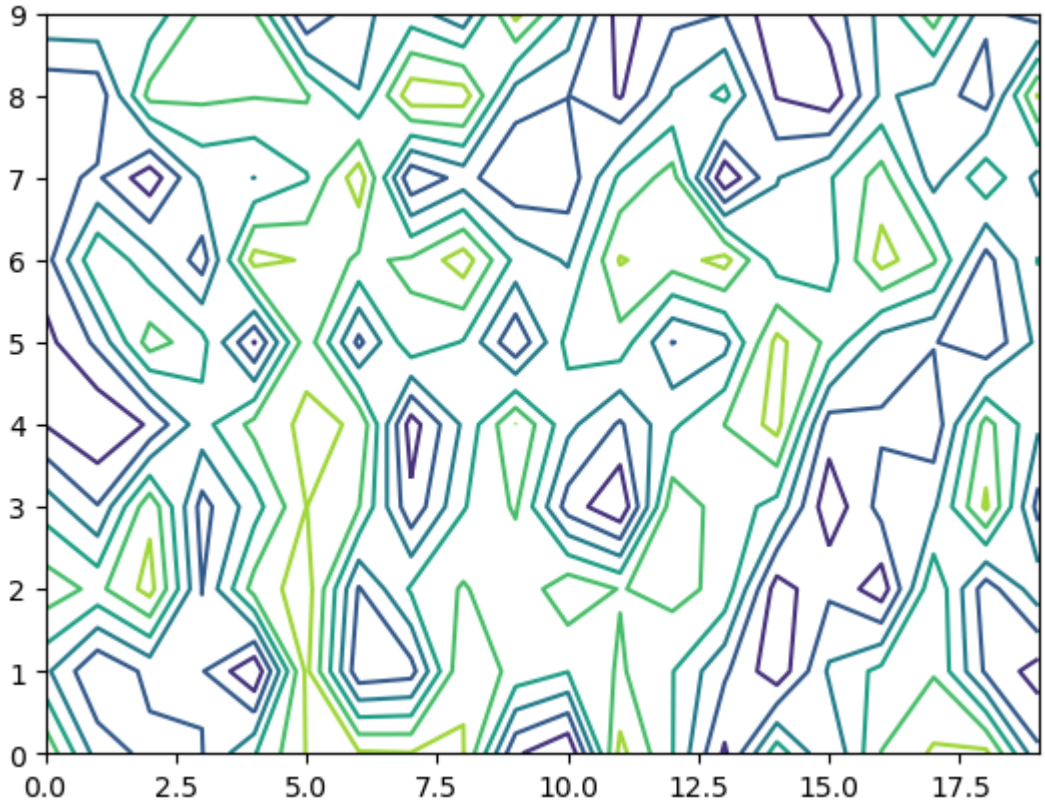
16.Area Chart

In [42]:
```
# Create some data
x12 = range(1, 6)
y12 = [1, 4, 6, 8, 4]

# Area plot
plt.fill_between(x12, y12)
plt.show()
```



17.Contour Plot

In [43]:
```
# Create a matrix
matrix1 = np.random.rand(10, 20)

cp = plt.contour(matrix1)

plt.show()
```



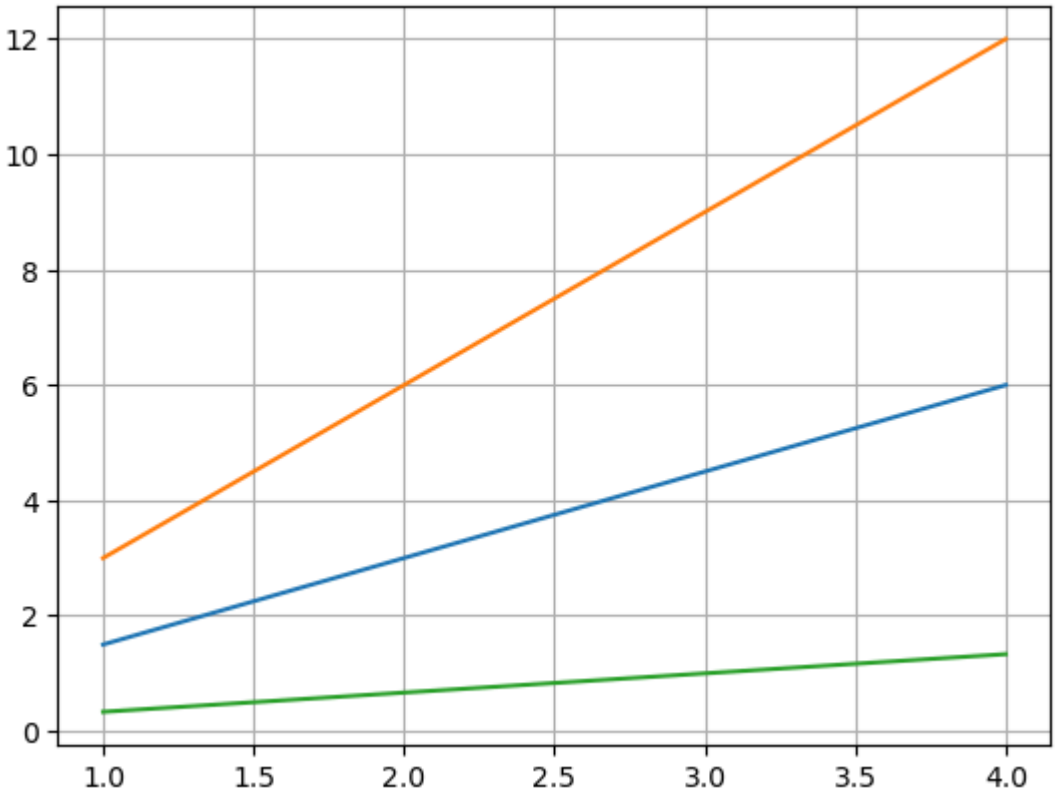
18.Styles with Matplotlib Plots

In [44]:
```
# 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', 'graysc
ale', 'petroff10', 'seaborn-v0_8', 'seaborn-v0_8-bright', 'seaborn-v0_8-colorblin
d', 'seaborn-v0_8-dark', 'seaborn-v0_8-dark-palette', 'seaborn-v0_8-darkgrid', 's
eaborn-v0_8-deep', 'seaborn-v0_8-muted', 'seaborn-v0_8-notebook', 'seaborn-v0_8-p
aper', 'seaborn-v0_8-pastel', 'seaborn-v0_8-poster', 'seaborn-v0_8-talk', 'seabor
n-v0_8-ticks', 'seaborn-v0_8-white', 'seaborn-v0_8-whitegrid', 'tableau-colorblin
d10']
```

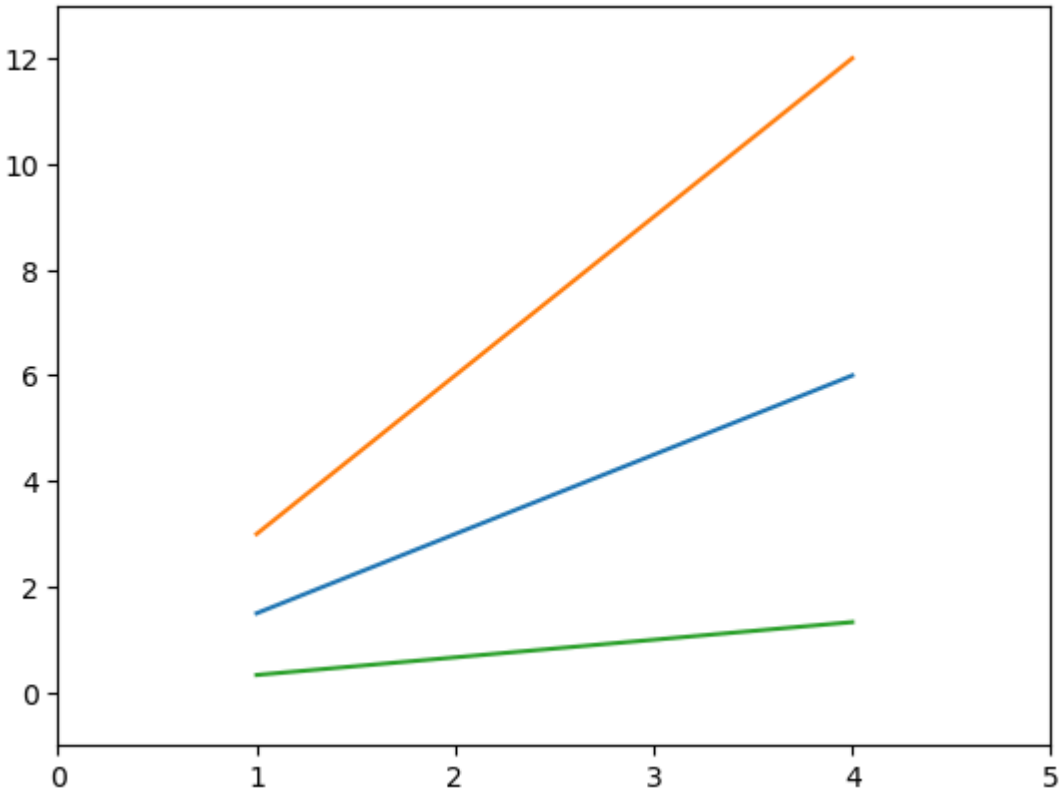
19.Adding a grid

In [46]:
```
x15 = np.arange(1, 5)

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

plt.grid(True)

plt.show()
```



20.Handling axes

In [47]:
```
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()
```

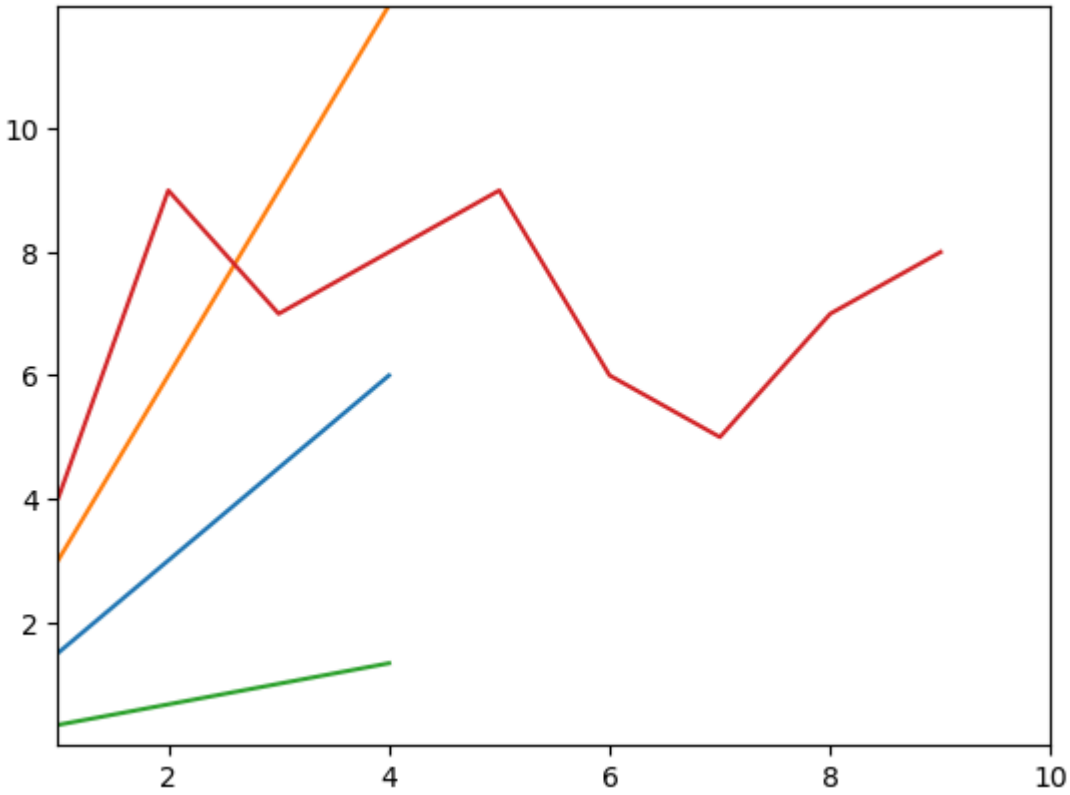


In [48]:
```
x15 = np.arange(1, 5)

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])
```

Out[48]: (0.0, 12.0)

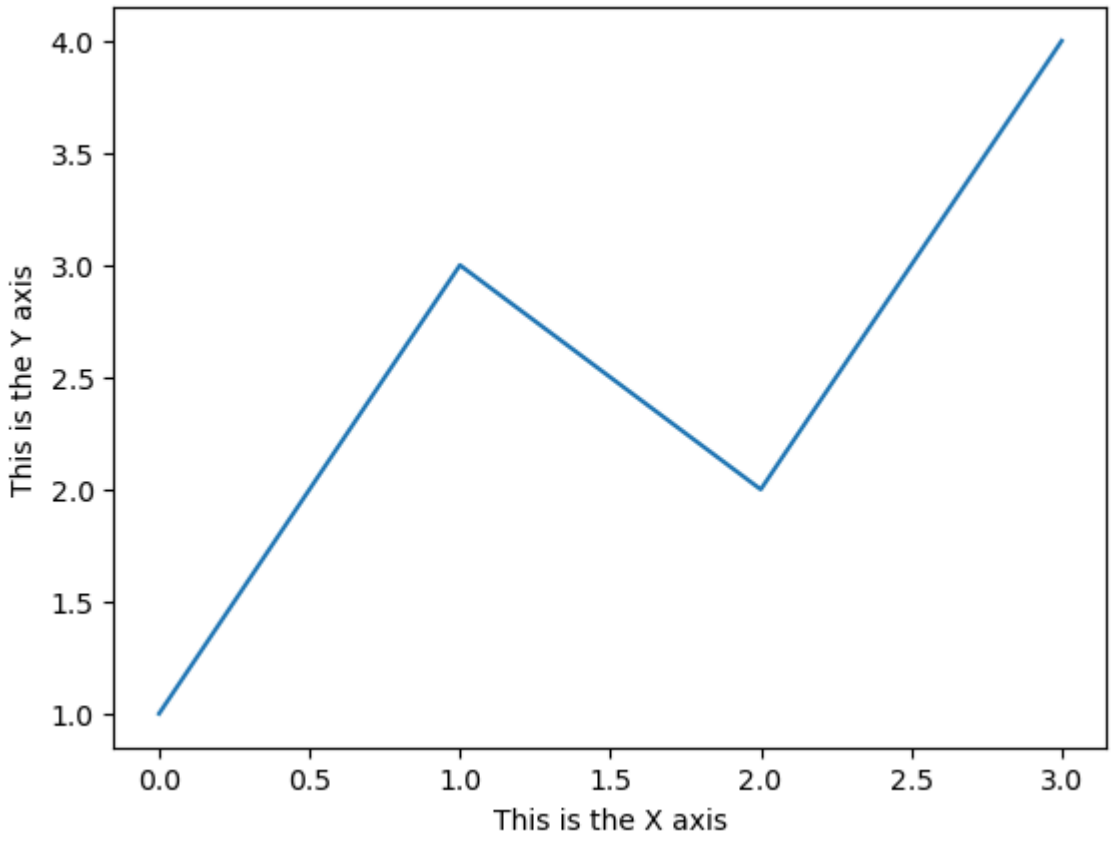
21.Handling X and Y ticks

In [49]:
```
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()
```



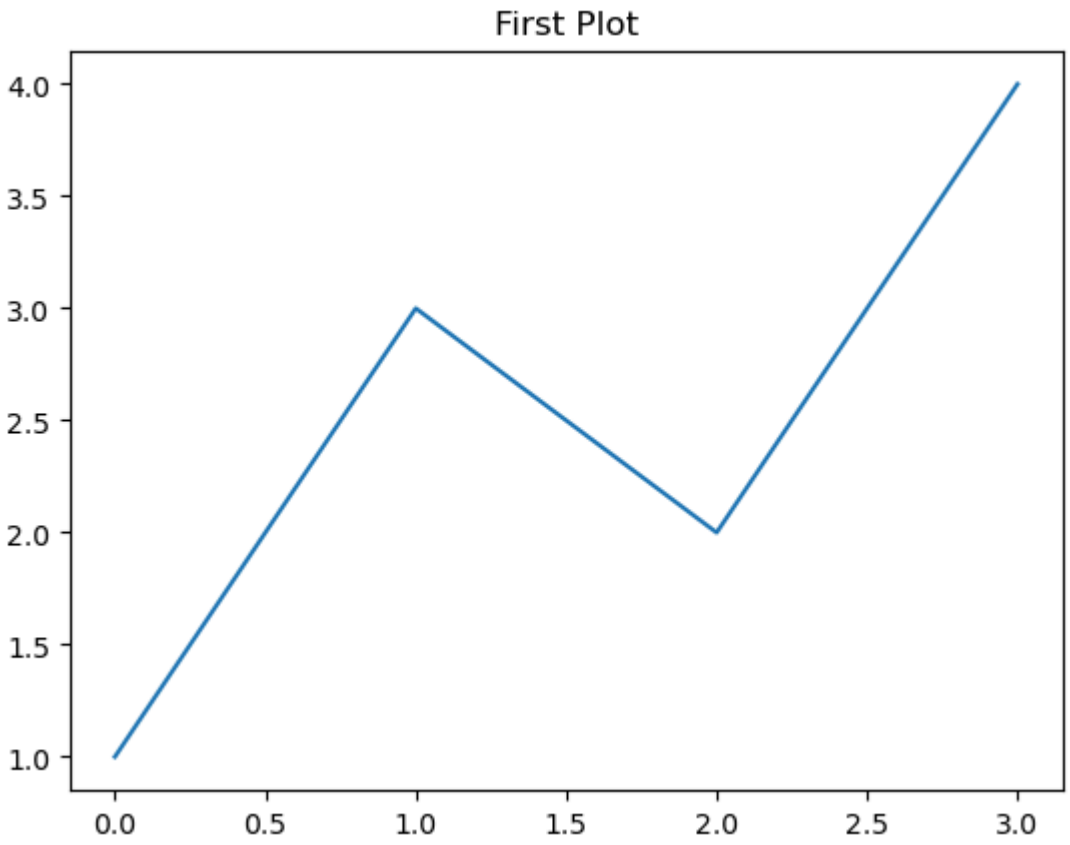
22.Adding labels

In [50]:
```
plt.plot([1, 3, 2, 4])

plt.xlabel('This is the X axis')

plt.ylabel('This is the Y axis')

plt.show()
```



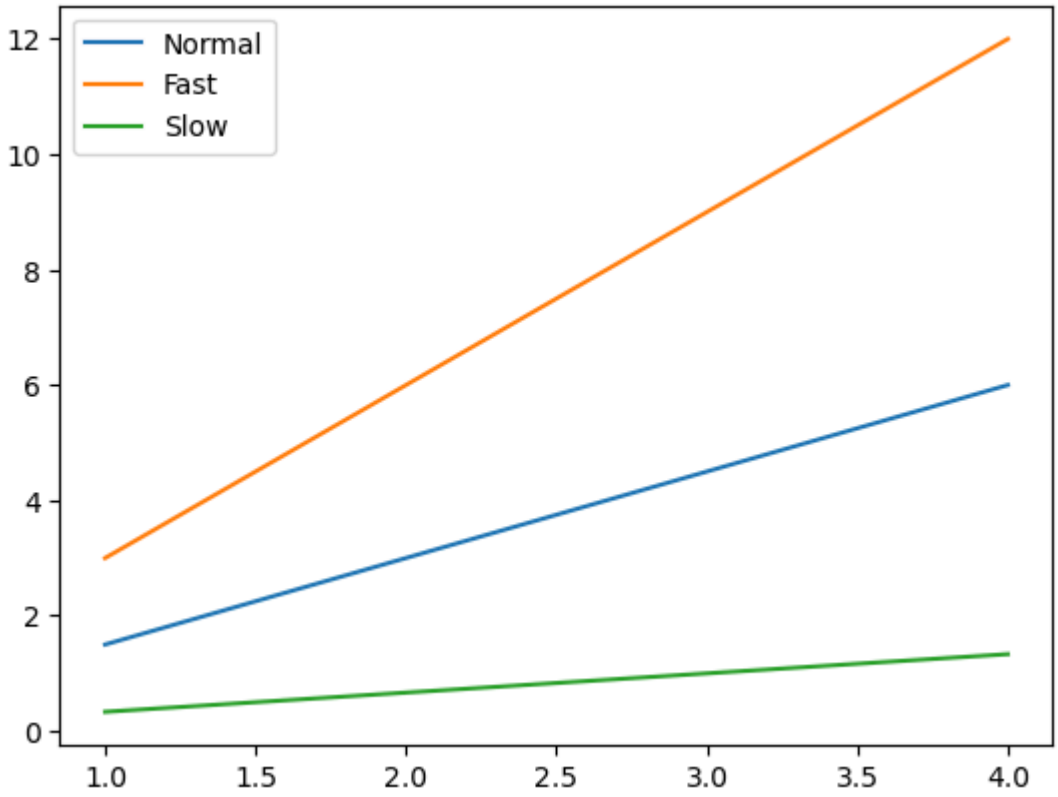
23.Adding a title

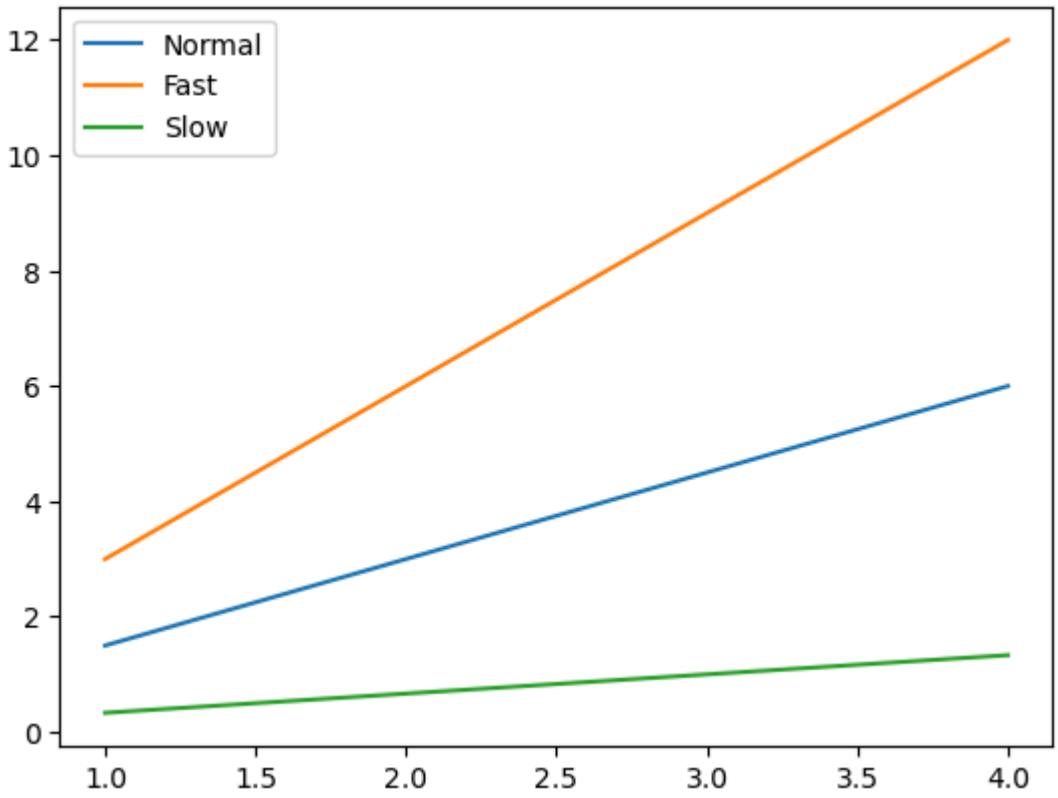
In [51]:
```
plt.plot([1, 3, 2, 4])

plt.title('First Plot')

plt.show()
```



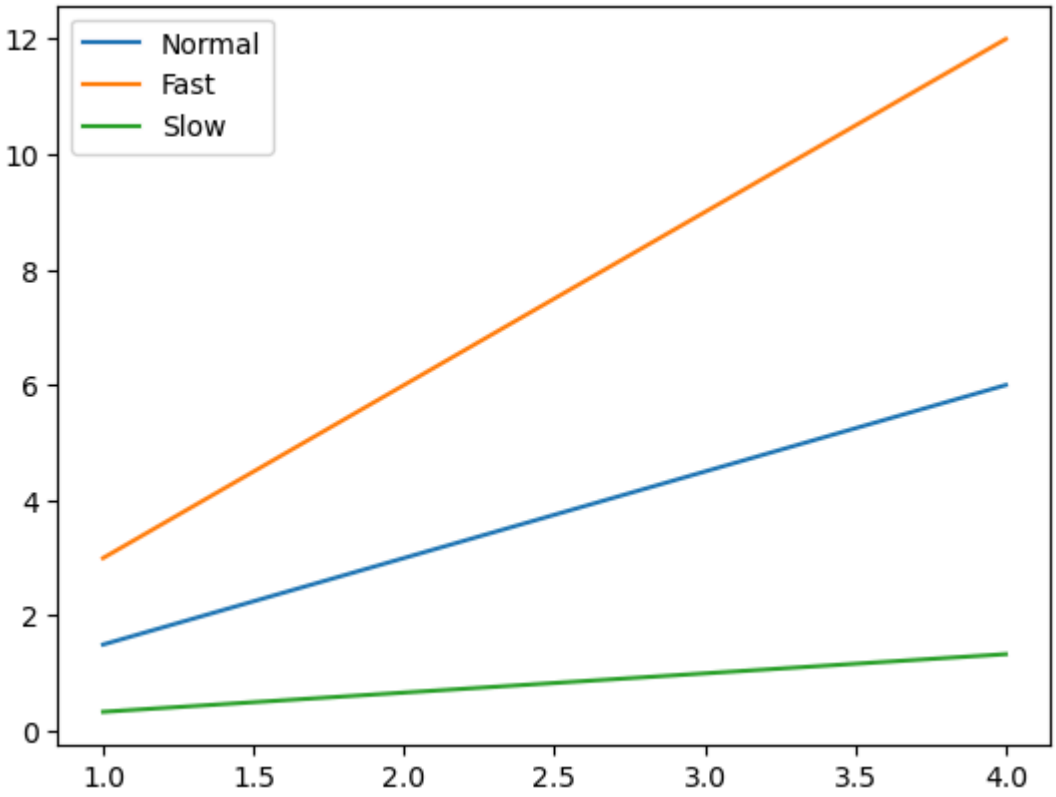
24.Adding a legend

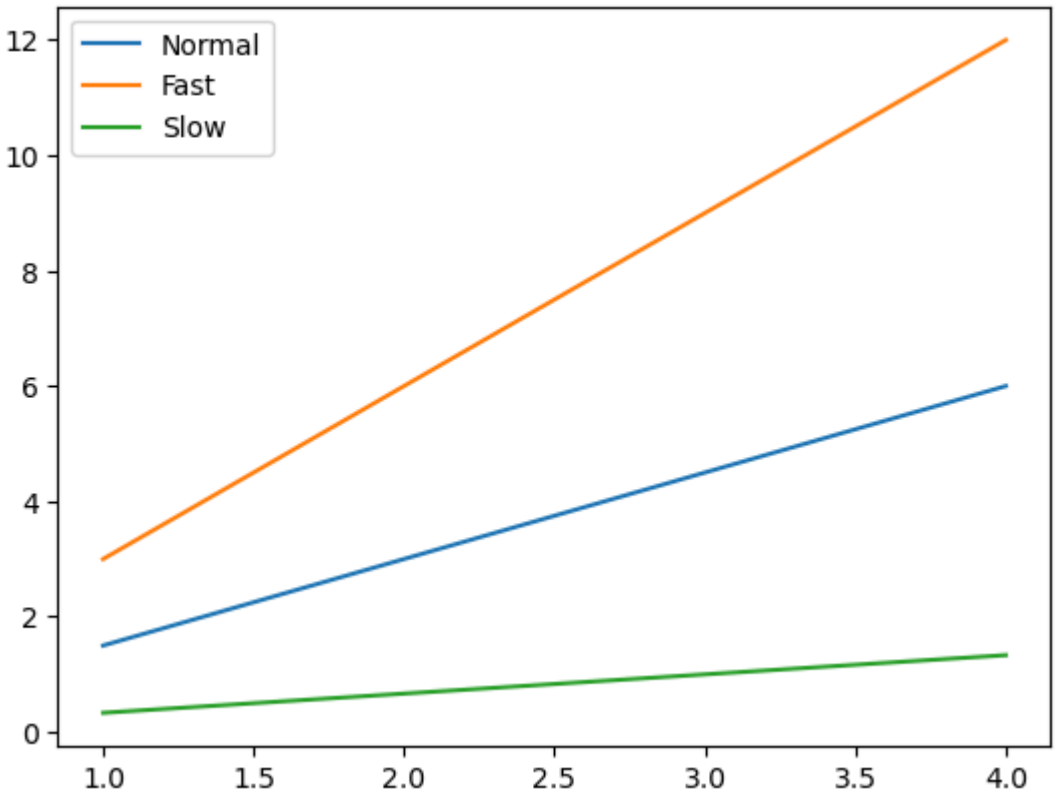
In [53]:
```
x15 = np.arange(1, 5)

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 [55]:
```
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()
```

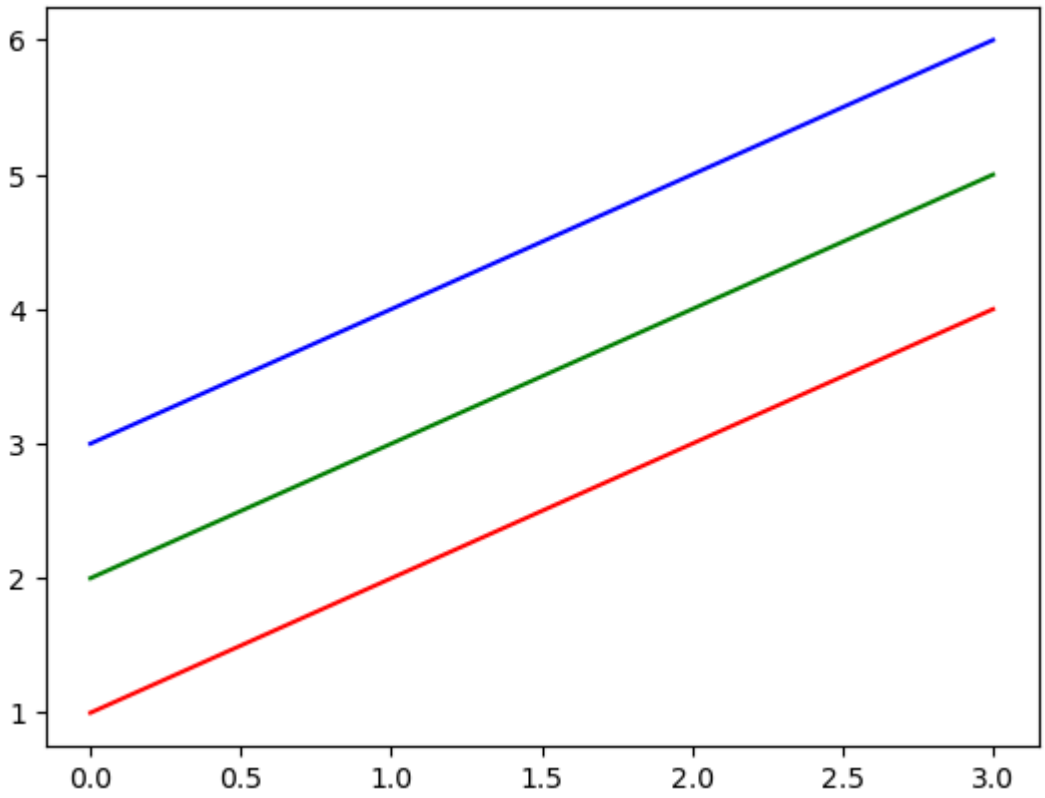




25.Control colours

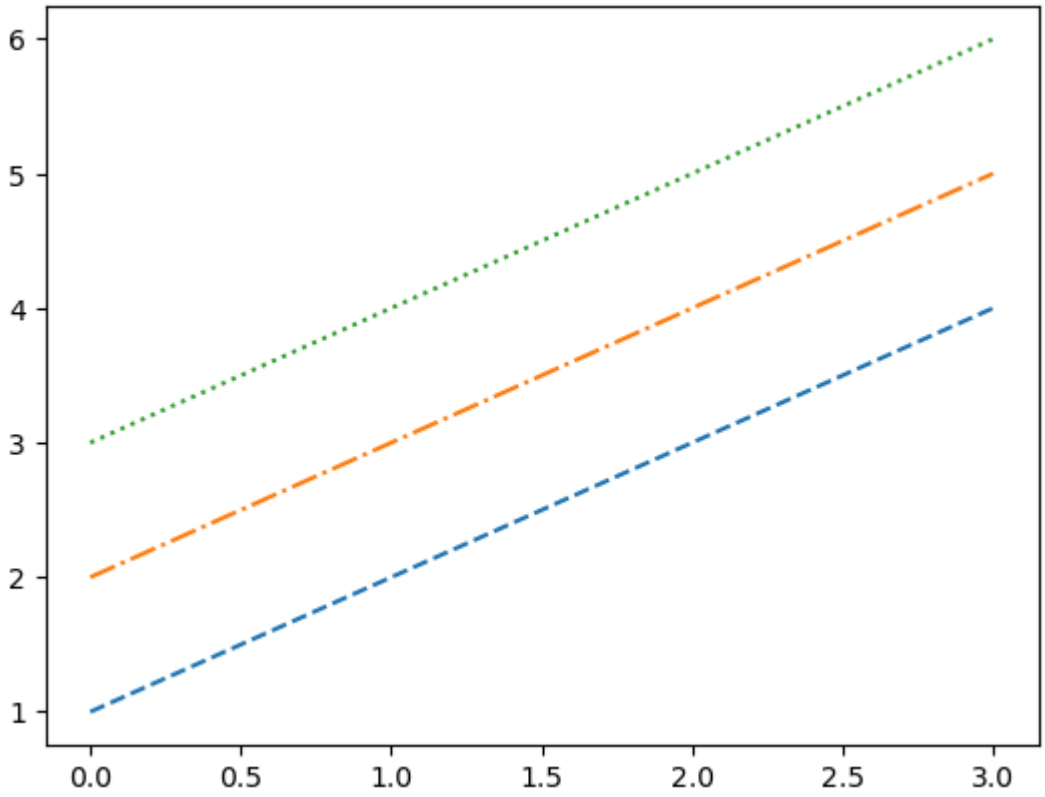
In [56]:
```
x16 = np.arange(1, 5)

plt.plot(x16, 'r')
plt.plot(x16+1, 'g')
plt.plot(x16+2, 'b')
```

```
plt.show()
```



26.Control line styles

In [57]:
```
x16 = np.arange(1, 5)

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

plt.show()
```



In [ ]: