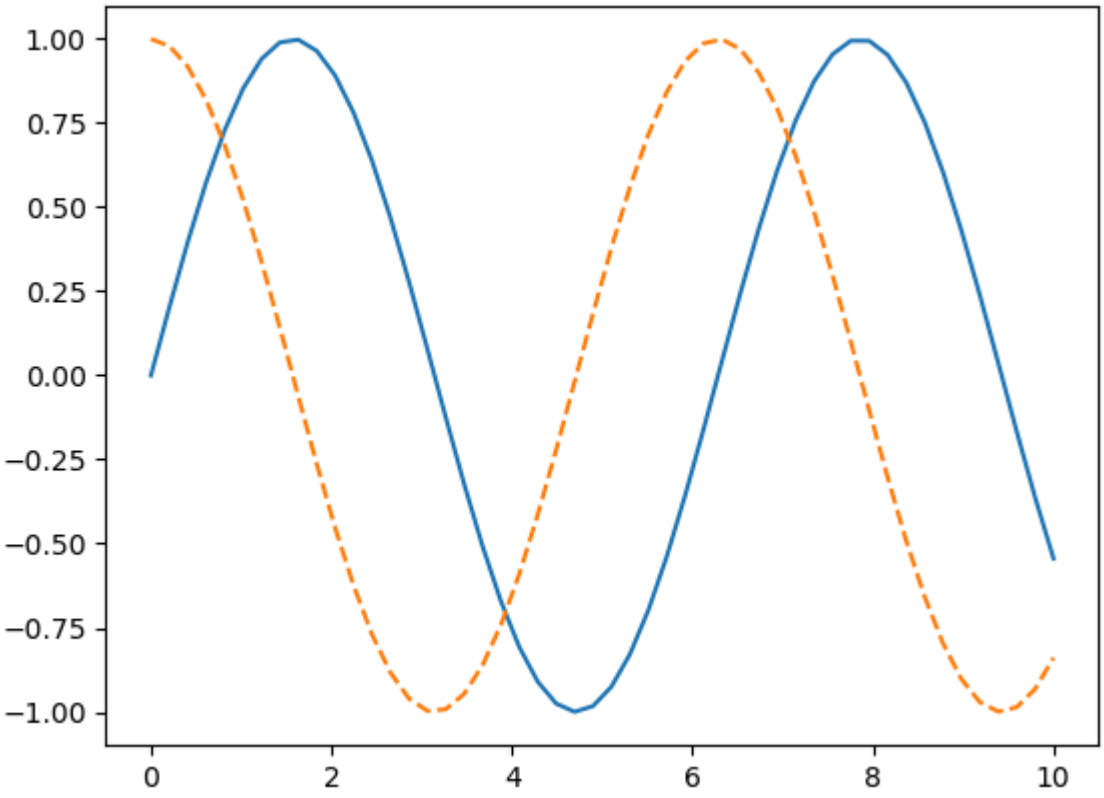
In [1]:
```
import warnings
warnings.filterwarnings('ignore')
```

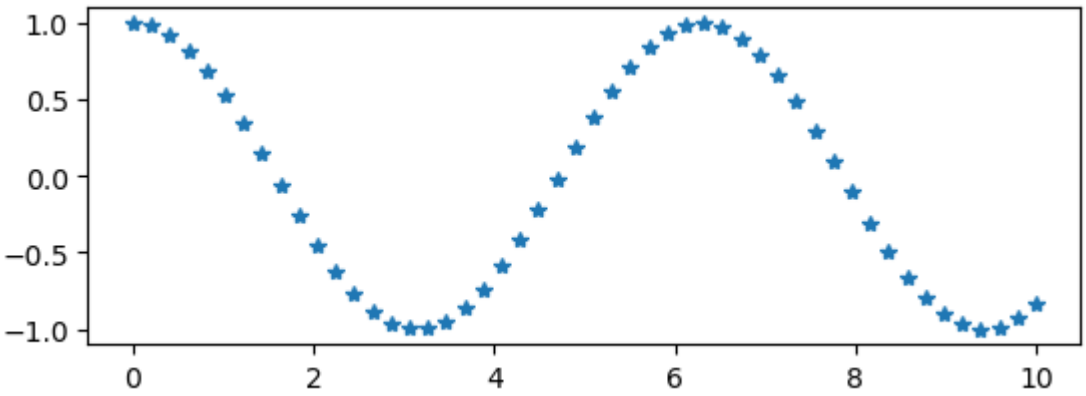
In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

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

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



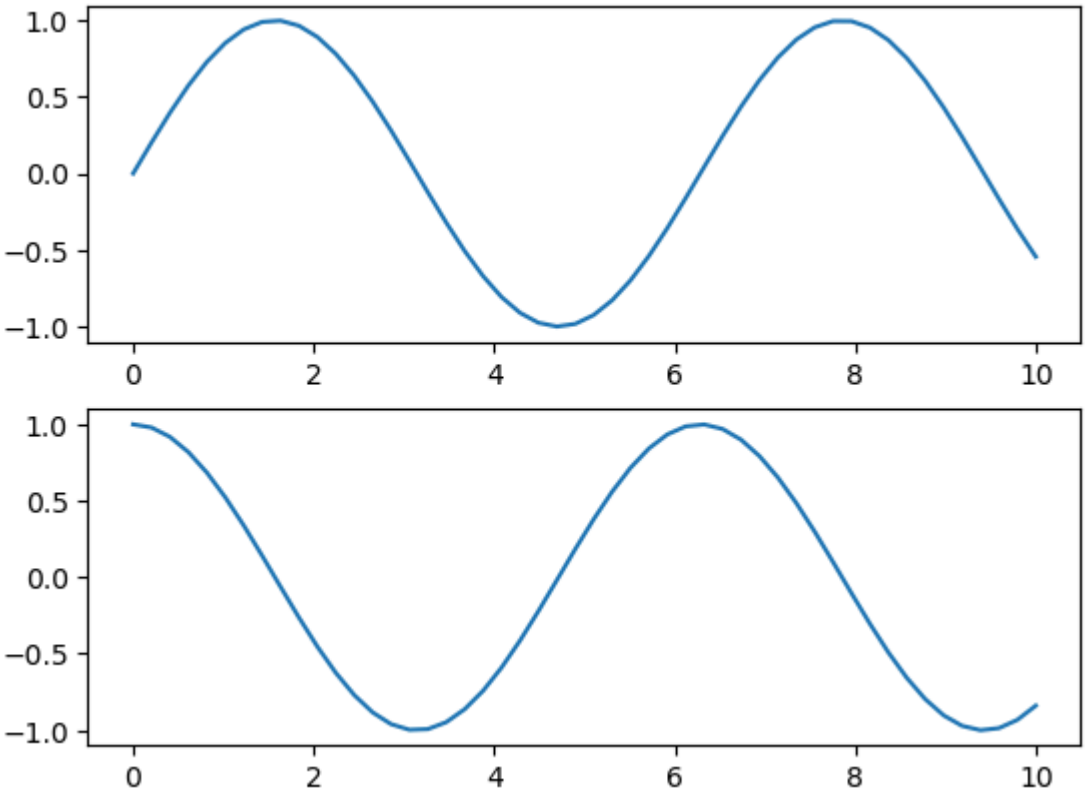
In [5]:
```
plt.subplot(2,1,1)
plt.plot(x1, np.cos(x1), '*')
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x20e62485f10>]
```



In [6]:
```
plt.figure()

plt.subplot(2,1,1)
plt.plot(x1,np.sin(x1))

plt.subplot(2,1,2)
plt.plot(x1,np.cos(x1))
```

Out[6]: [<matplotlib.lines.Line2D at 0x20e627ef790>]

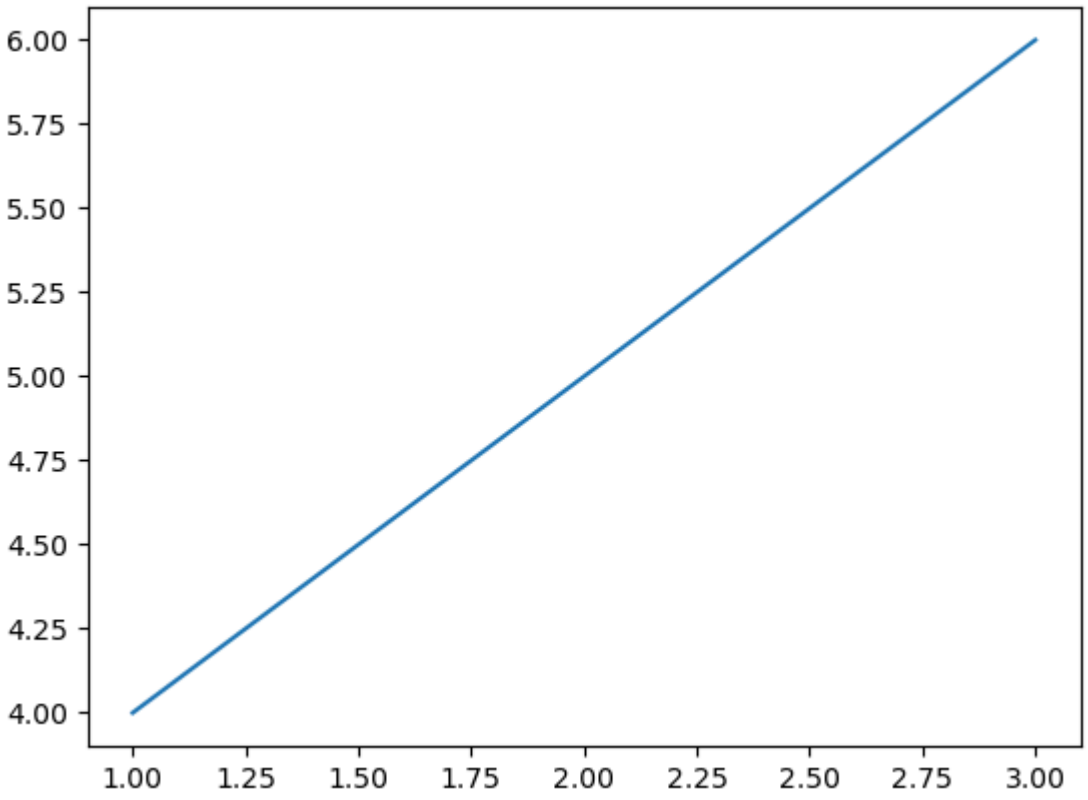


In [7]:
```
print(plt.gcf()) #Get Current Figure information fixed size 640x480 with 0 A
```

```
Figure(640x480)

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

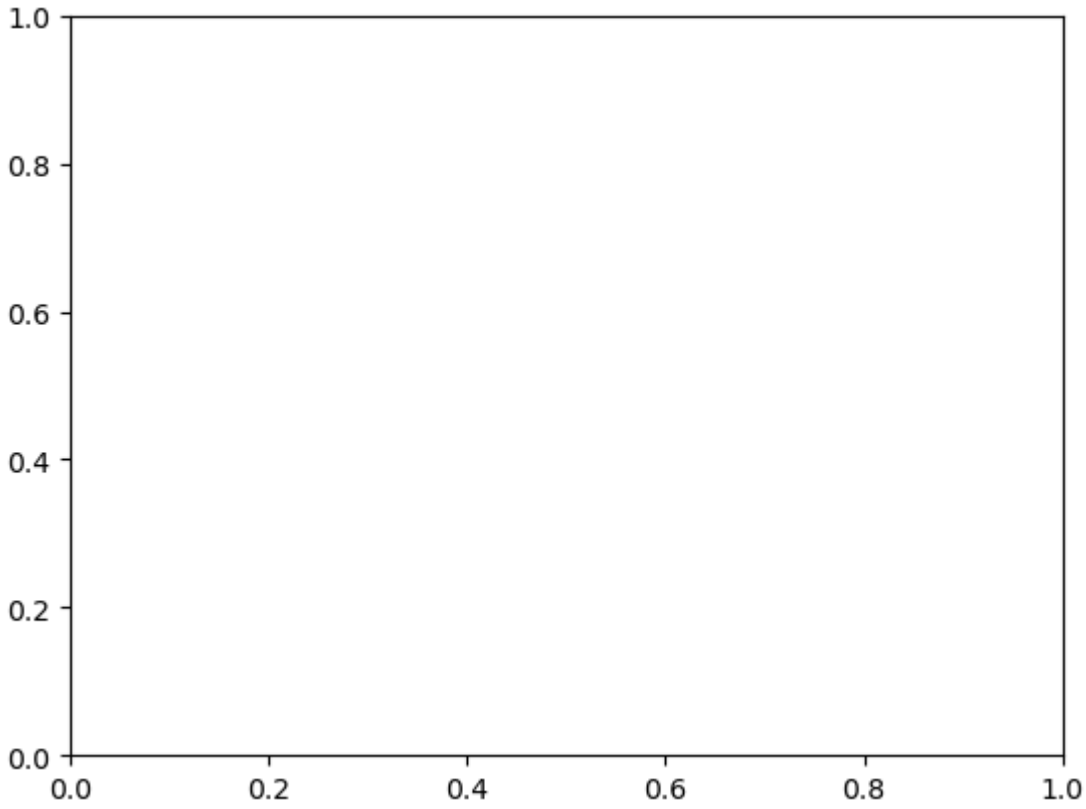
In [8]:
```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3], [4, 5, 6])
fig = plt.gcf()
print(fig)
```

```
Figure(640x480)
```



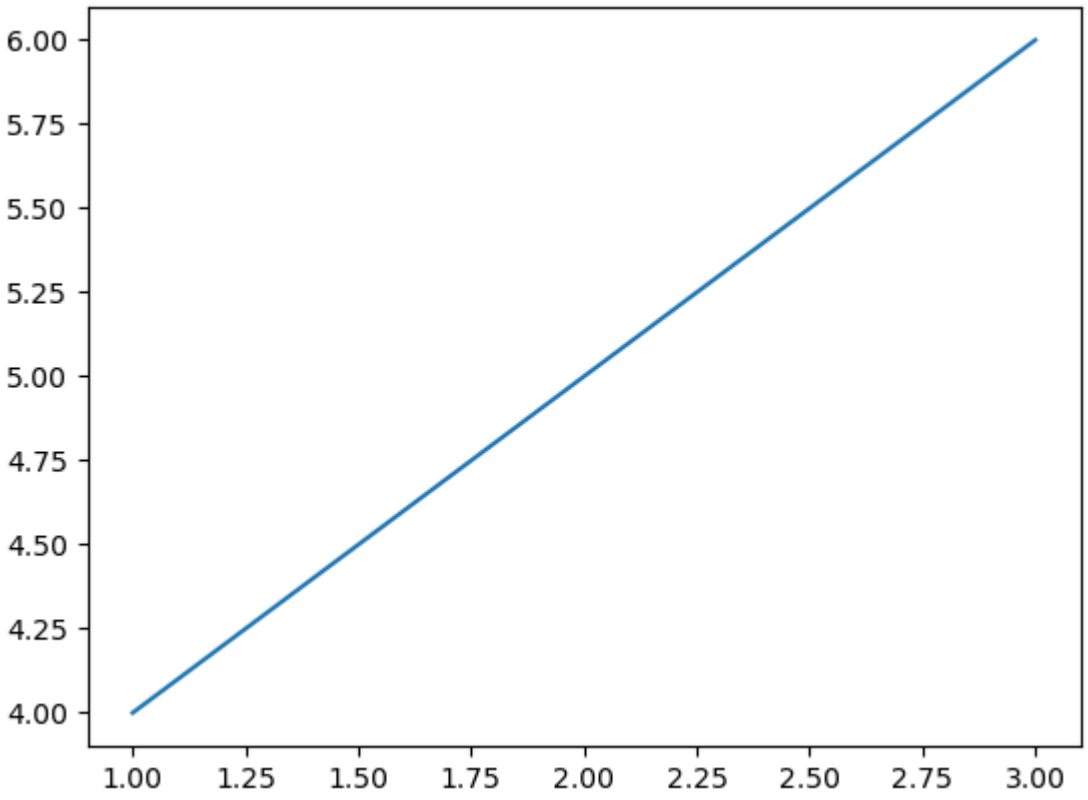
In [9]:
```
print(plt.gca()) #Get Current axis Information
```

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



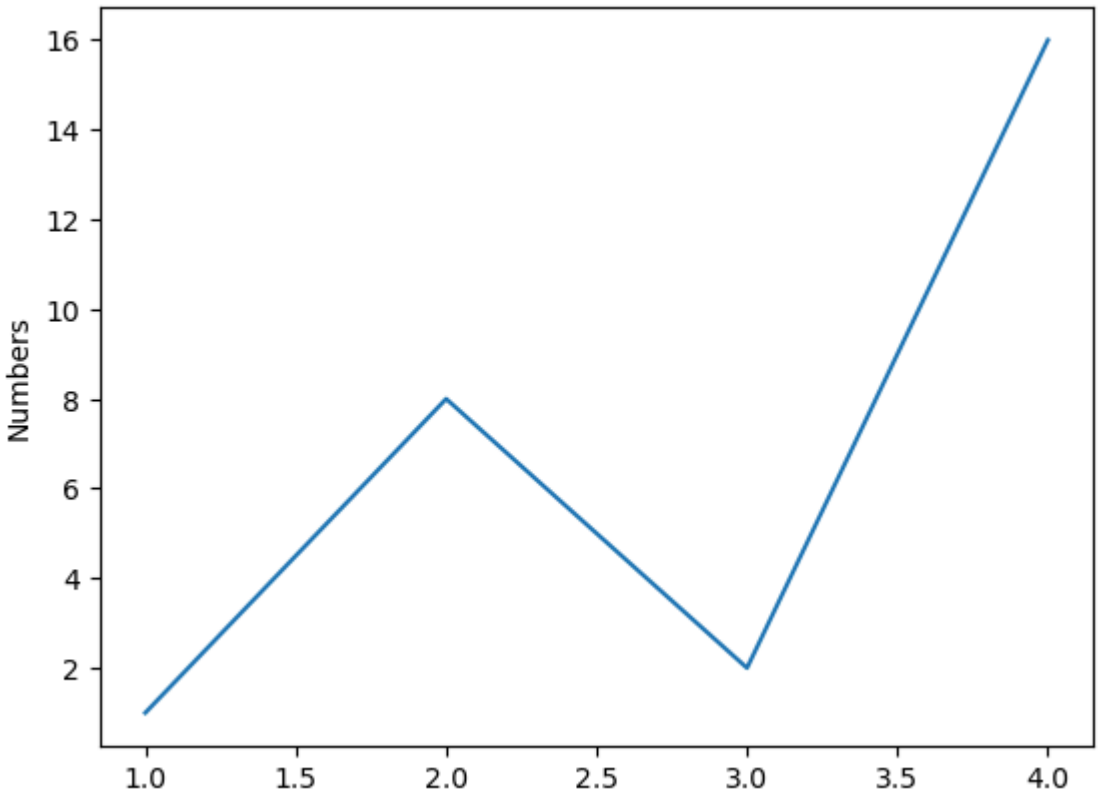
In [10]:
```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3], [4, 5, 6])
ax = plt.gca()
print(ax)
```

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



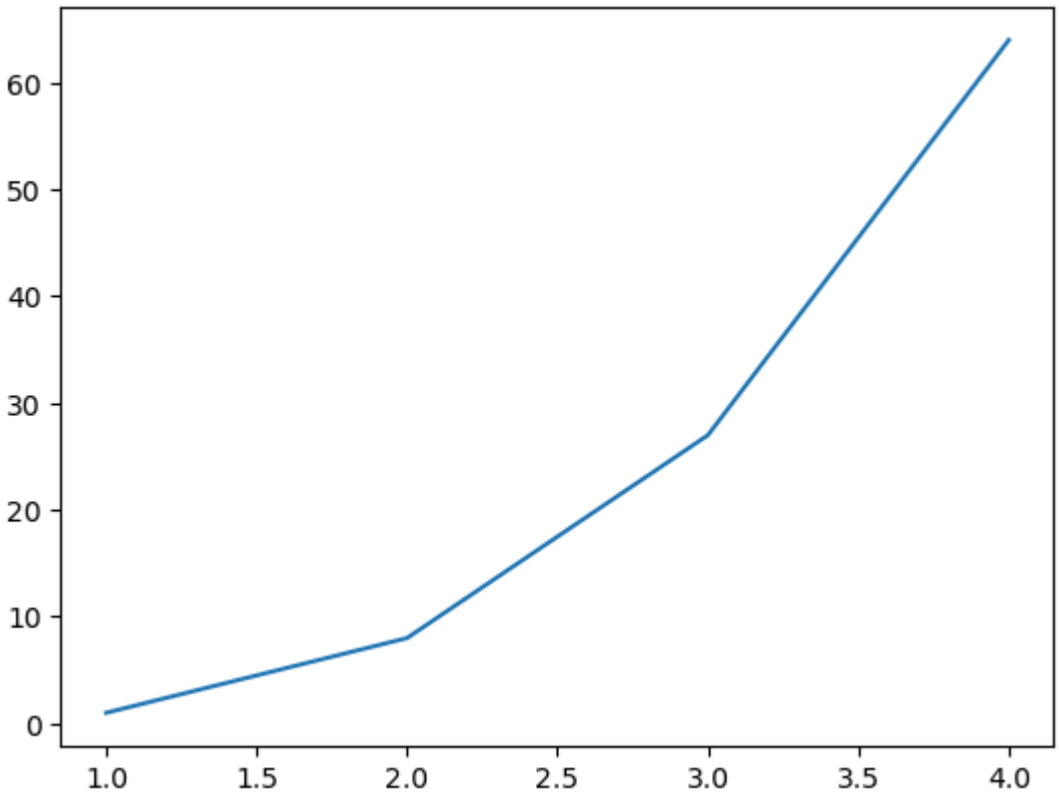
Visualization with pyplot

In [11]:
```
plt.plot([1,2,3,4],[1,8,2,16])
plt.ylabel('Numbers')
plt.show()
```



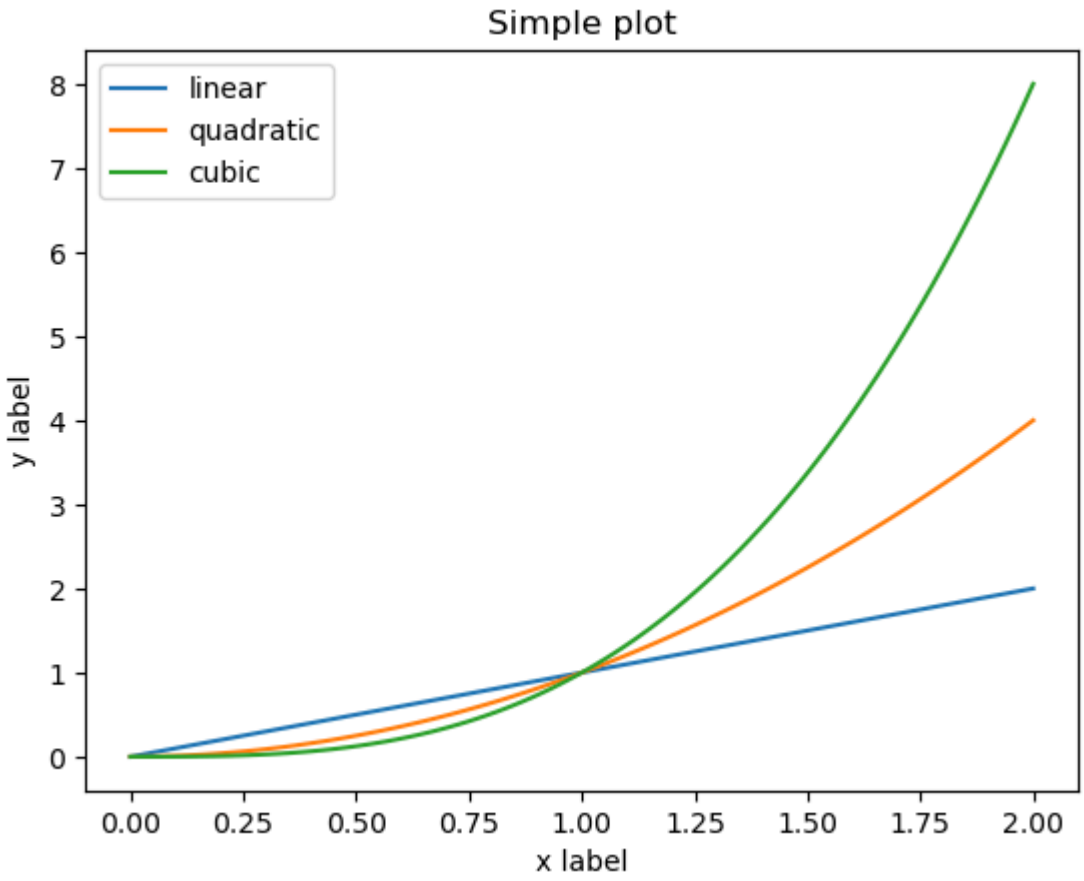
plot() Versatile Command

In [12]:
```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 8, 27, 64])
plt.show()
```



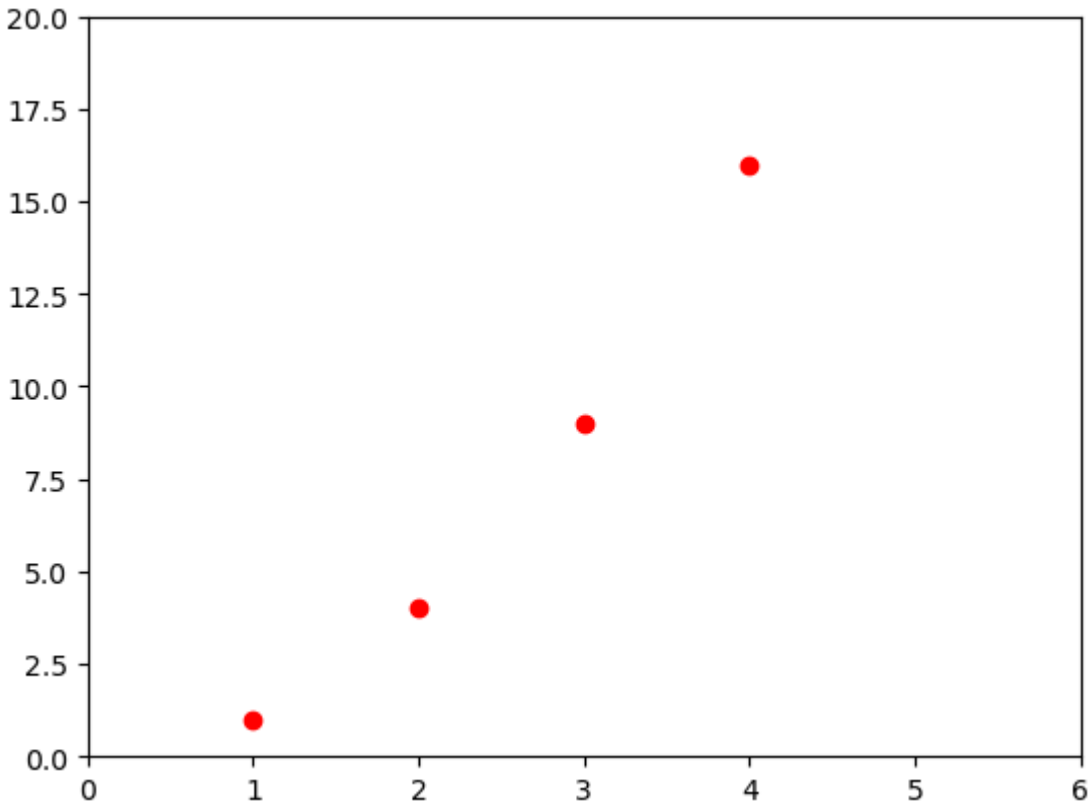
State-machine interface

In [13]:
```
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()
```

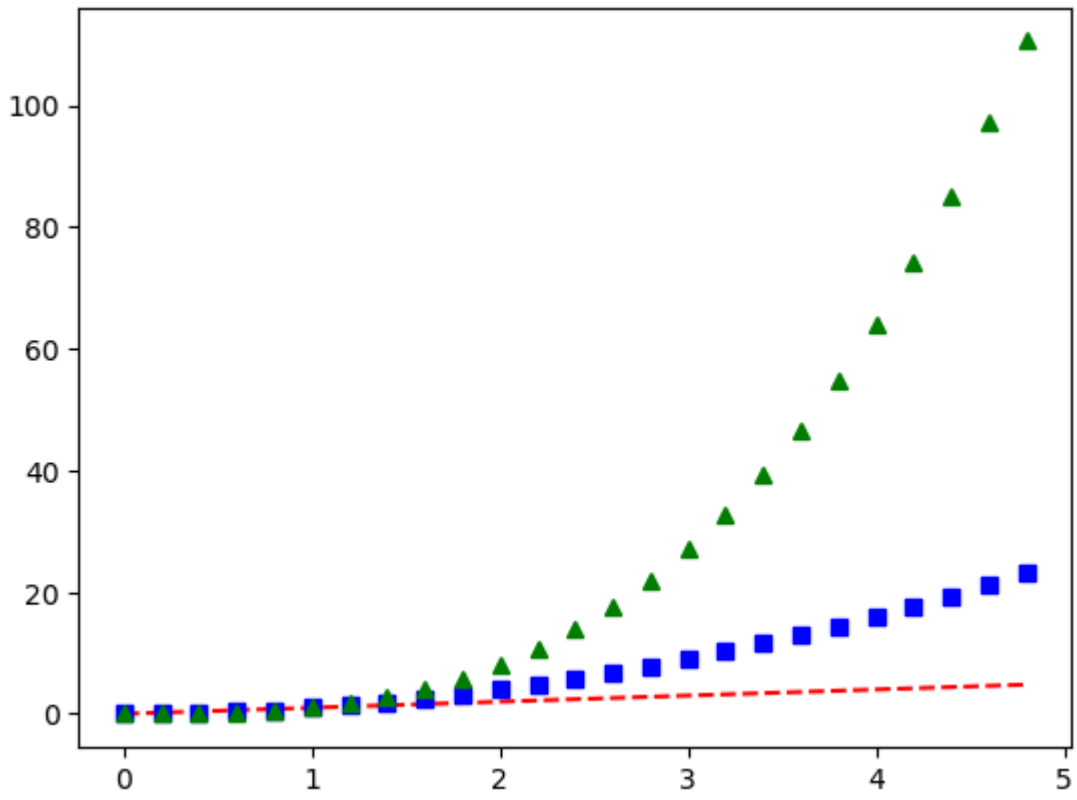


Formatting style plot

In [14]:
```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```



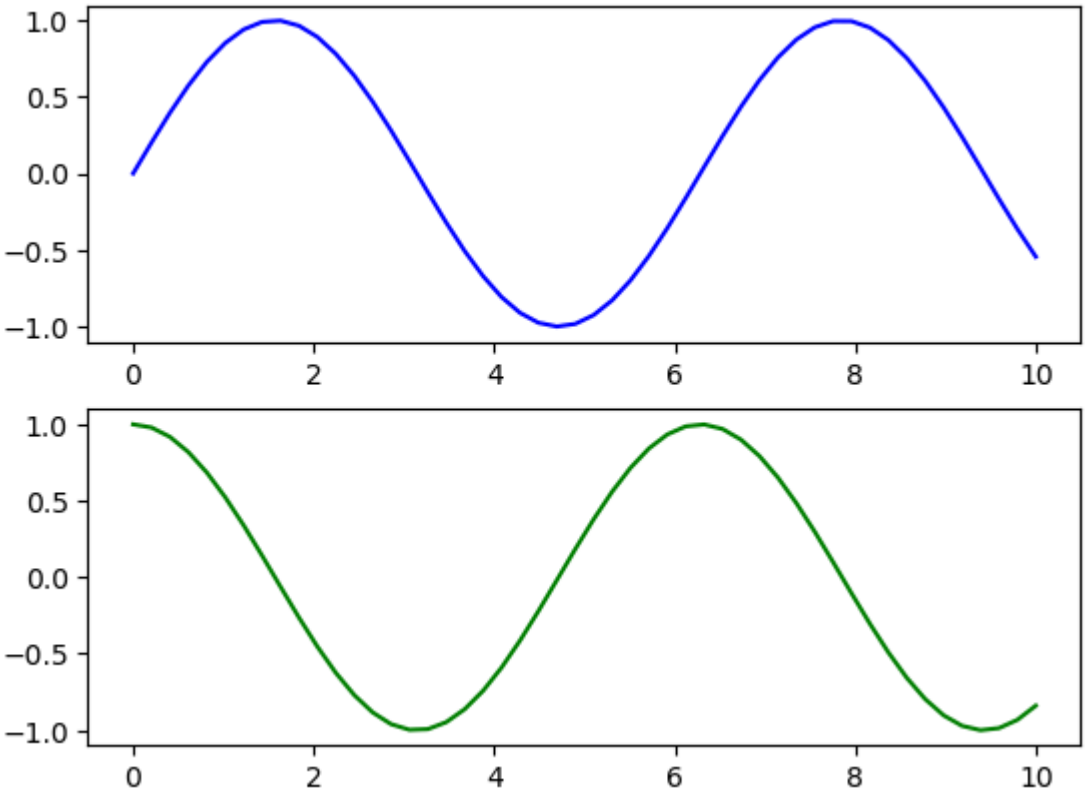
In [15]:
```
t = np.arange(0.,5.,0.2)

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



Object oriented API

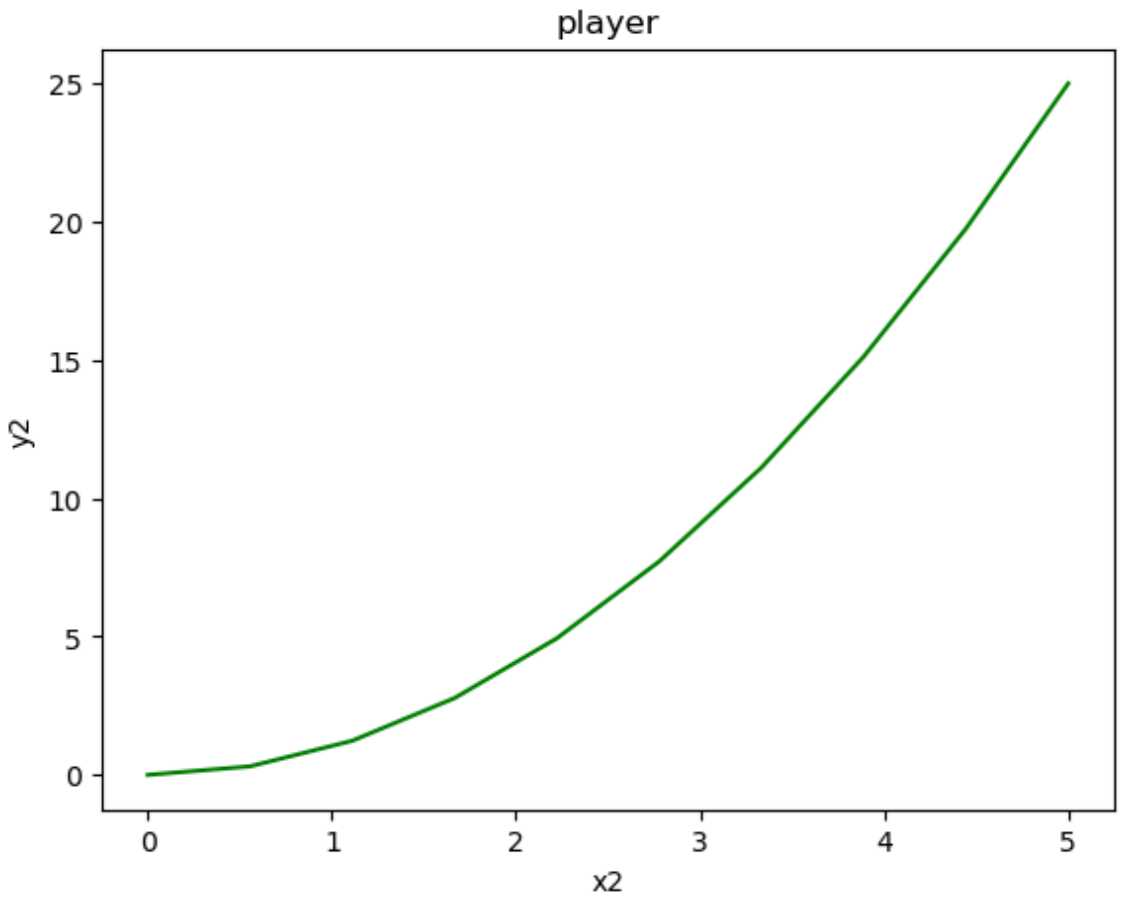
In [16]:
```
fig, ax = plt.subplots(2)
ax[0].plot(x1, np.sin(x1),'b-')
ax[1].plot(x1,np.cos(x1),'g-')
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x20e62677e90>]
```



In [17]:
```
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,'g')

axes.set_xlabel('x2')
axes.set_ylabel('y2')
axes.set_title('player')
```

Out[17]:
```
Text(0.5, 1.0, 'player')
```



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

x2 = np.linspace(0,2,4)
y2 = x2 ** 2

axes = fig.add_axes([0.2, 0.2, 0.8, 0.8])

axes.plot(x2,y2,'g')

axes.set_xlabel('x2')
axes.set_ylabel('y2')
axes.set_title('player')
```

Out[18]:
```
Text(0.5, 1.0, 'player')
```

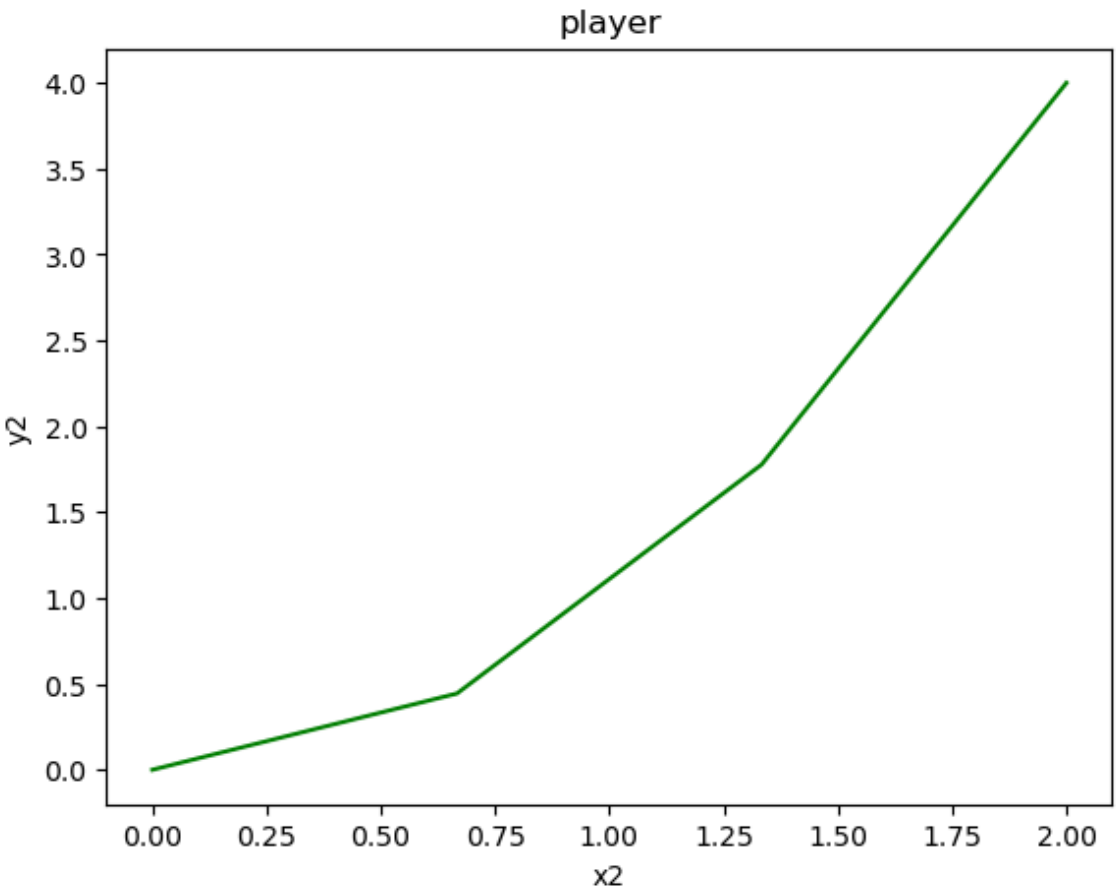


Figure and Axes

In [19]:
```
fig = plt.figure()
ax = plt.axes()
```

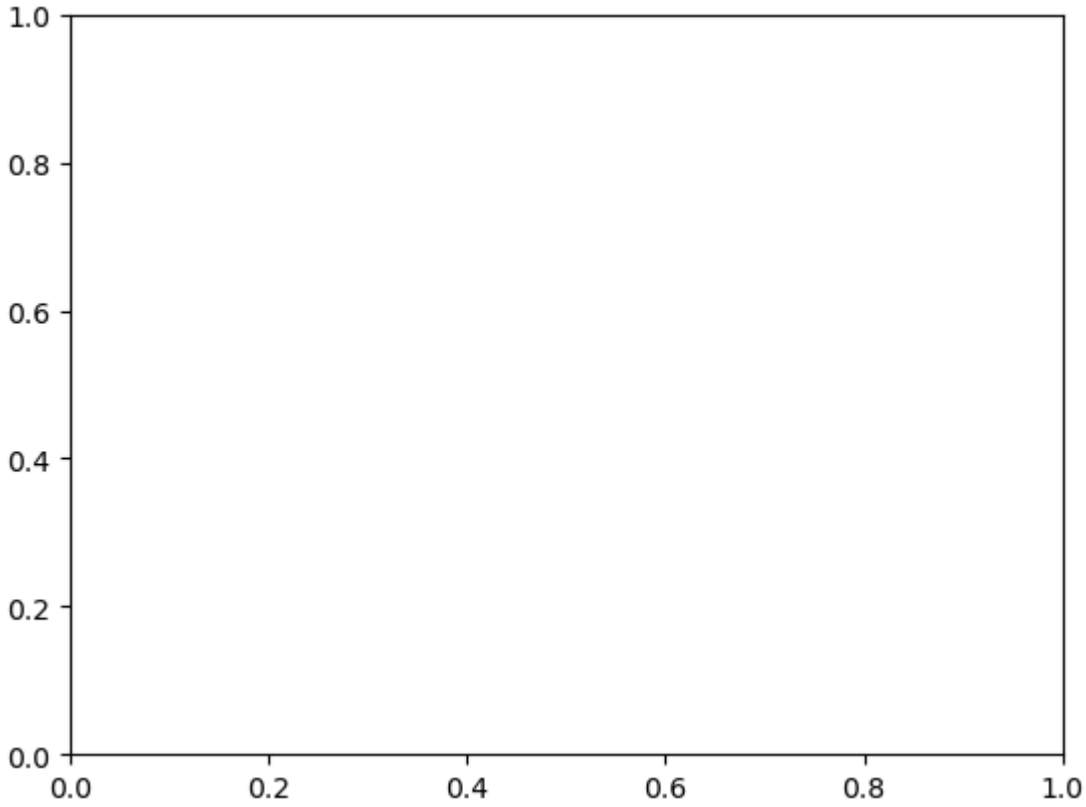
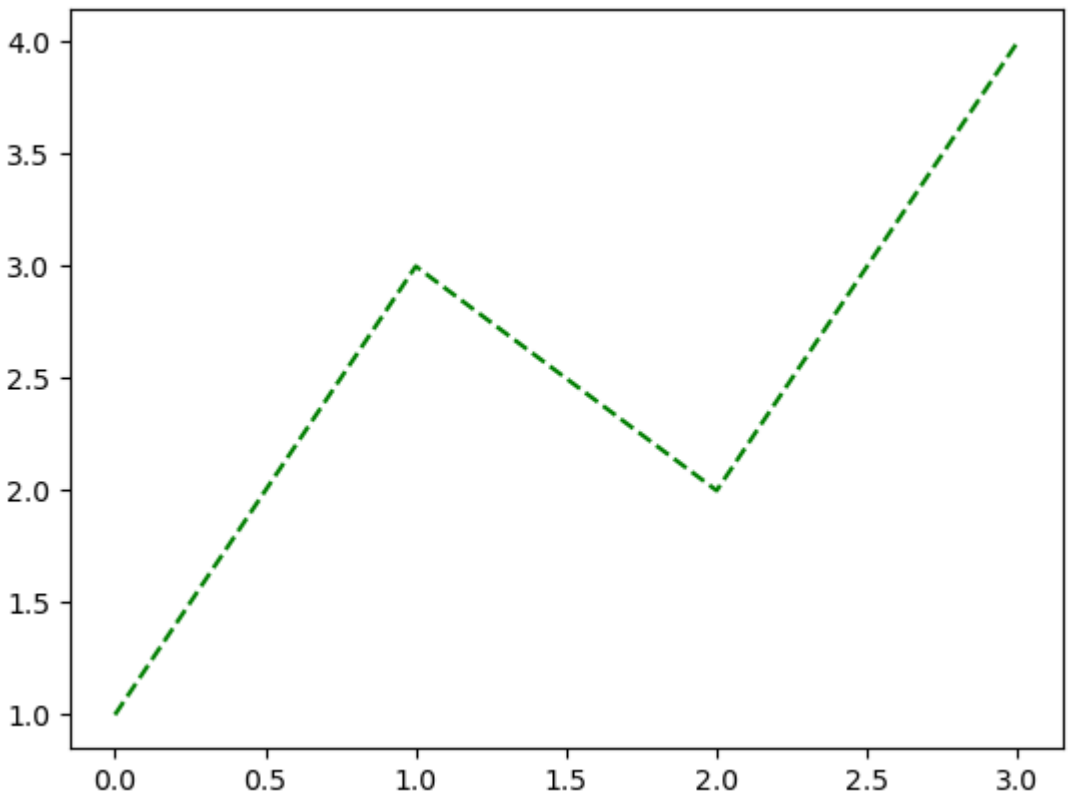
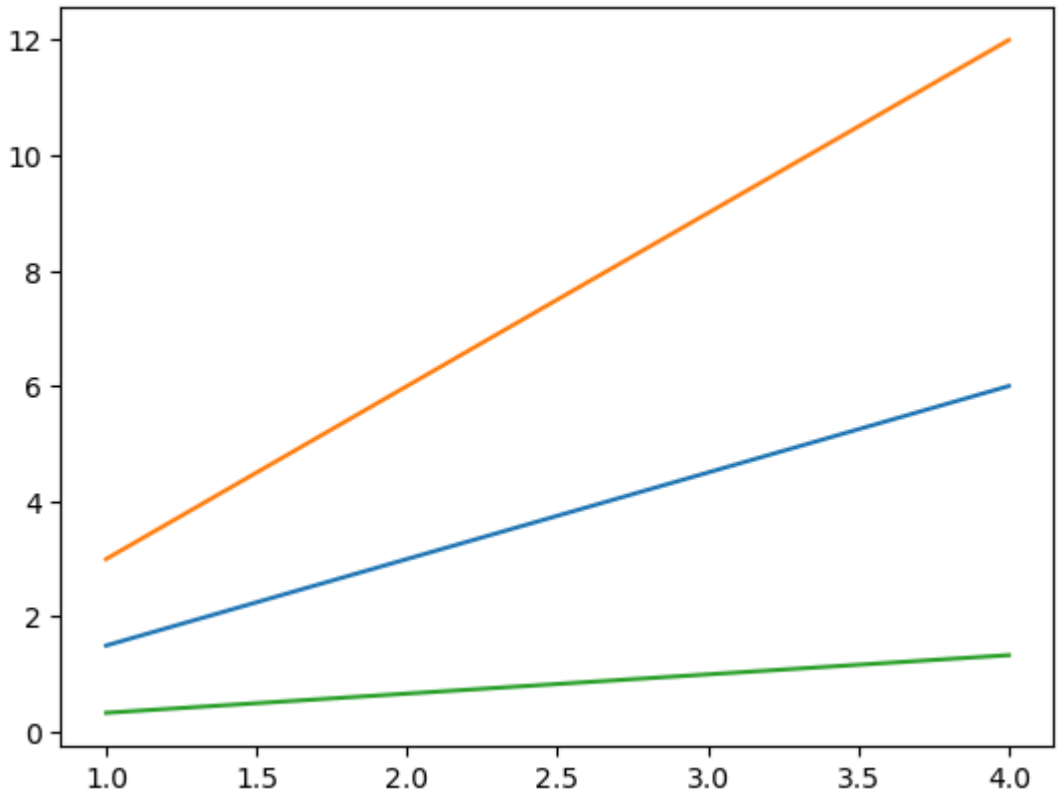


Figure Plot with Matplotlib

In [20]:
```
plt.plot([1,3,2,4],'g--')
plt.show()
```



In [21]:
```
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()
```

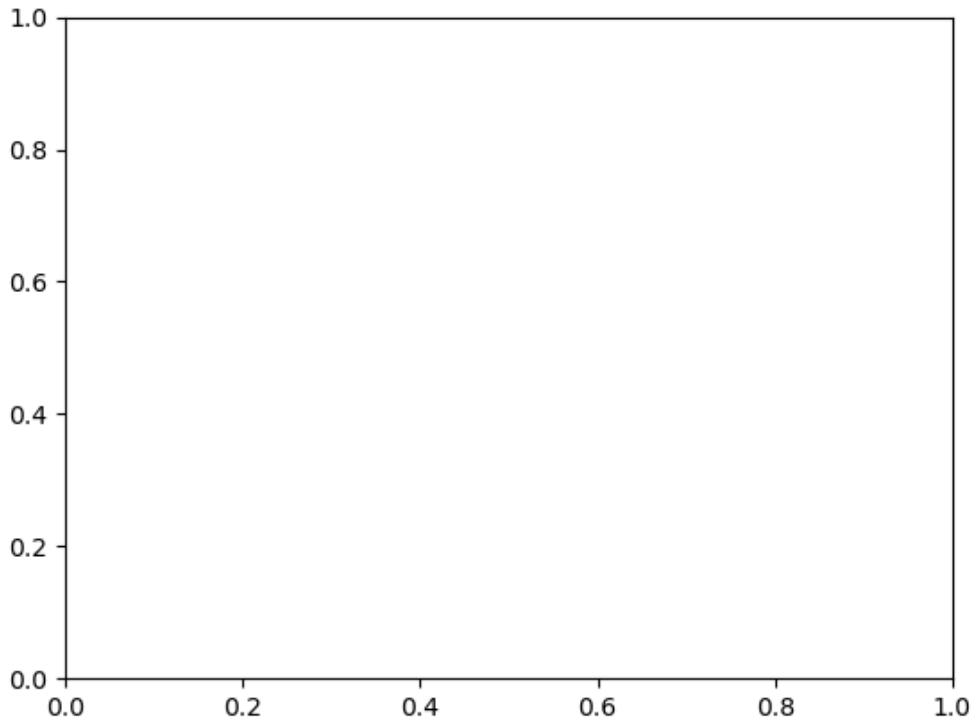


Saving the plot

In [22]:
```
fig.savefig('plot1.png')
```

In [23]:
```
# Explore the contents of figure

from IPython.display import Image

Image('plot1.png')
```

Out[23]:

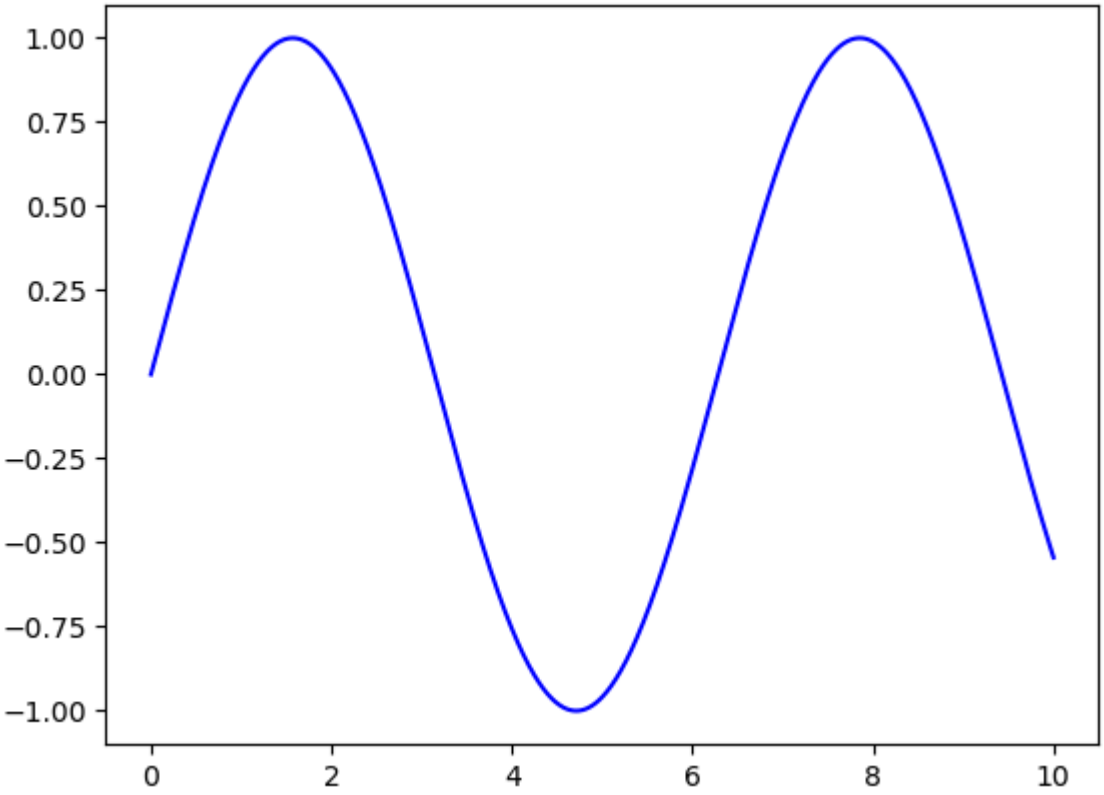


In [24]:
```
fig.canvas.get_supported_filetypes()
```

Out[24]:
```
{'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'}
```

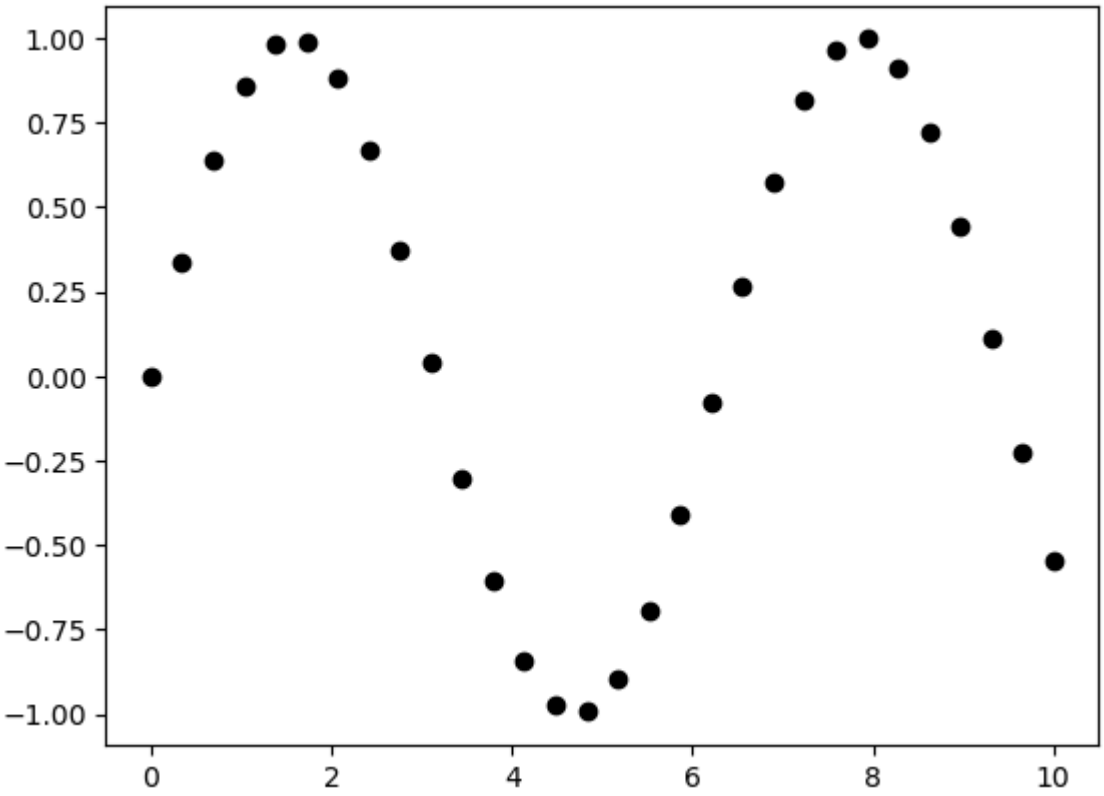
Line plot

In [25]:
```
# Create 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-');
```



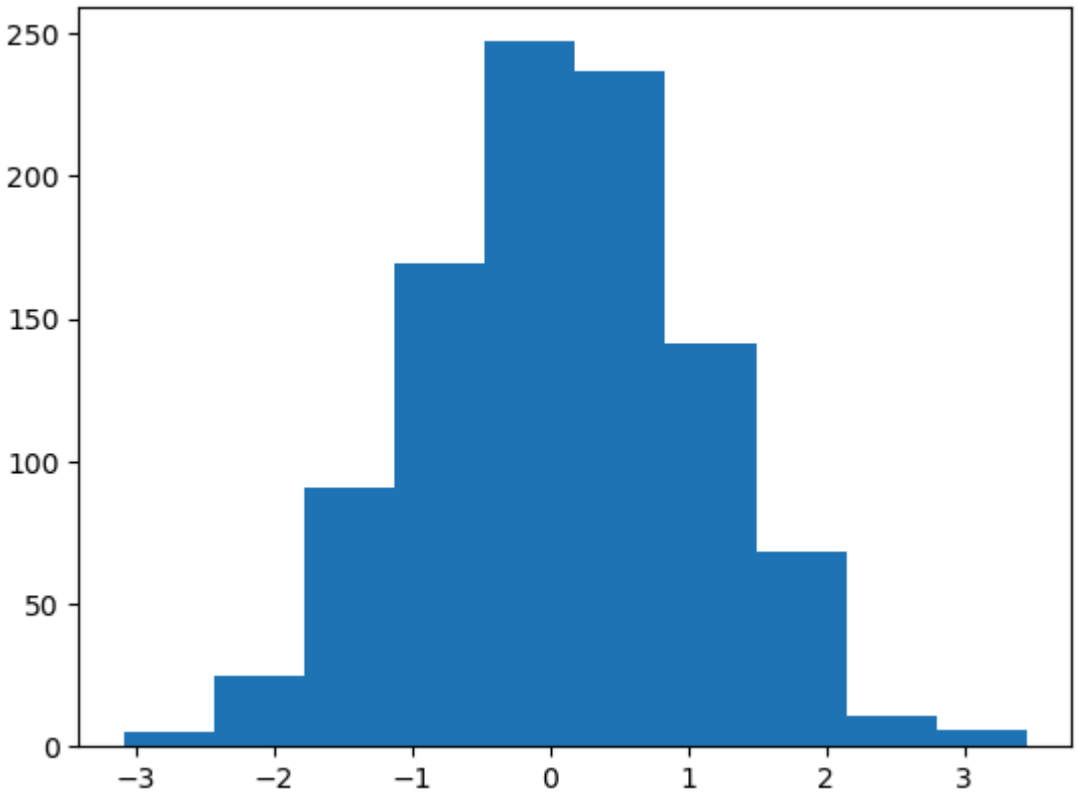
Scatter Plot

In [26]:
```
x = np.linspace(0, 10, 30)

y = np.sin(x)

plt.plot(x, y, 'o', color = 'black');
```



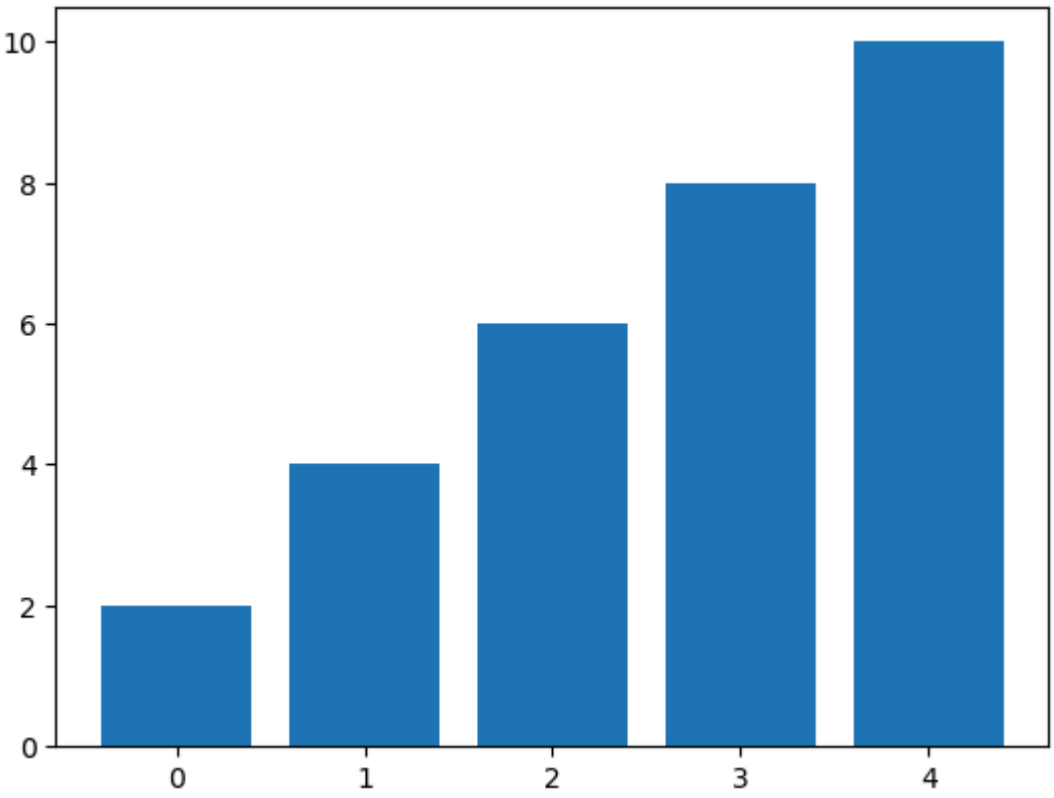
Histogram

In [27]:
```
d= np.random.randn(1000)

plt.hist(d);
```

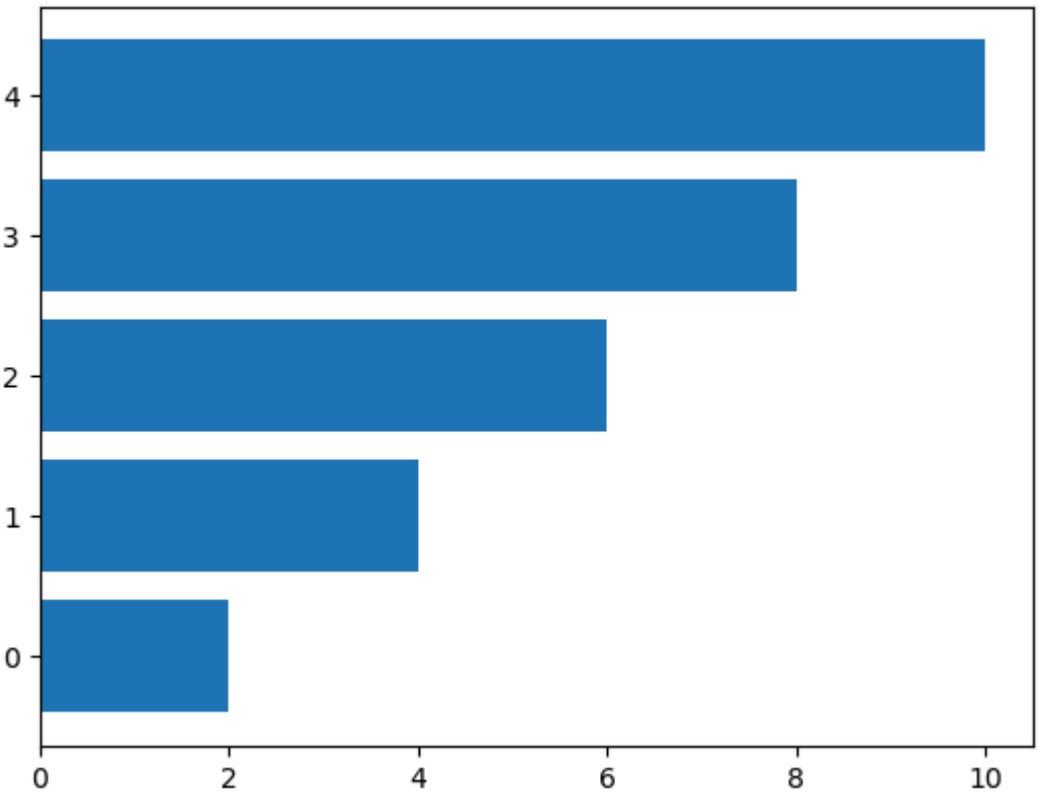


Bar Chart

In [28]:
```
d = [2,4,6,8,10]
plt.bar(range(len(d)),d)
plt.show()
```

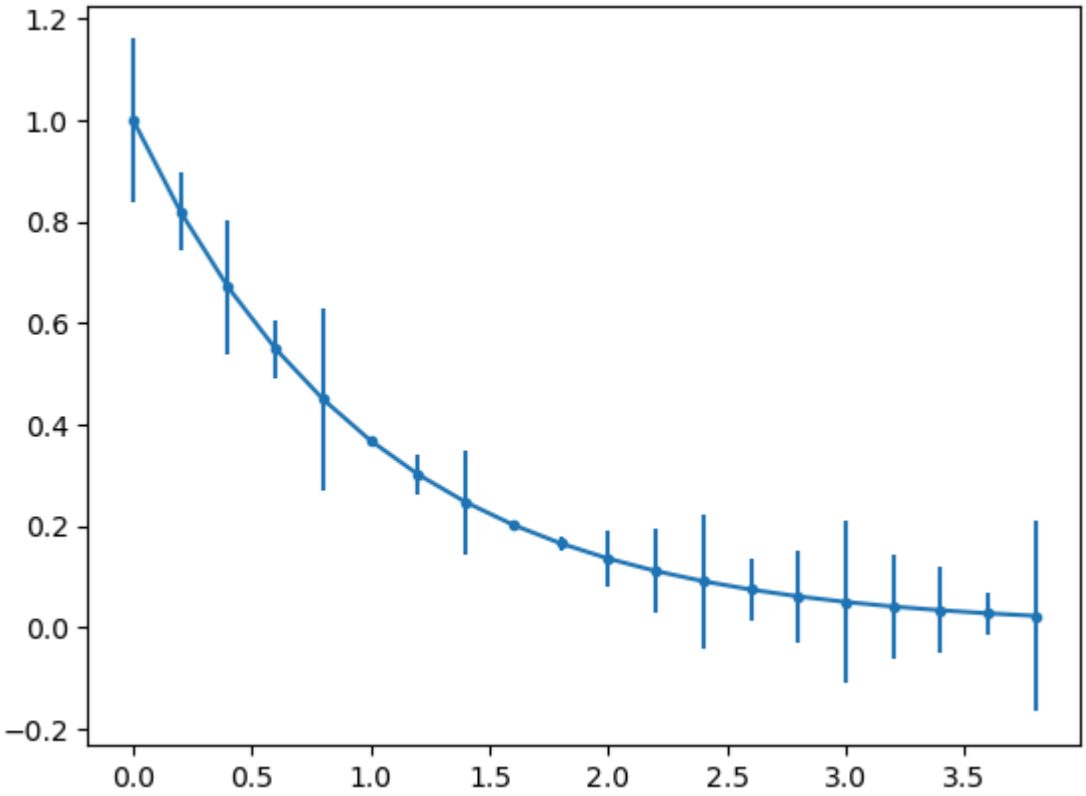


In [29]:
```
d = [2,4,6,8,10]
plt.barh(range(len(d)),d)
plt.show()
```



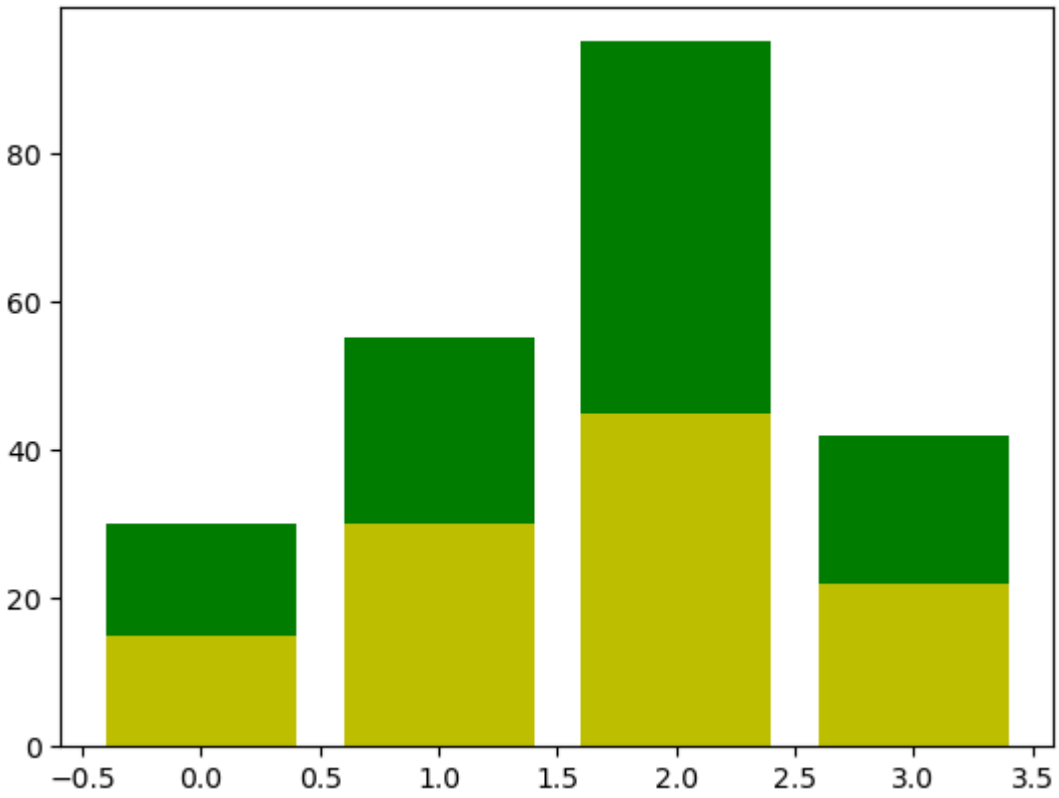
Error Bar Chart

In [30]:
```
x = np.arange(0, 4, 0.2)
y = np.exp(-x)

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

plt.errorbar(x,y, yerr = e1, fmt = '.-')
plt.show()
```



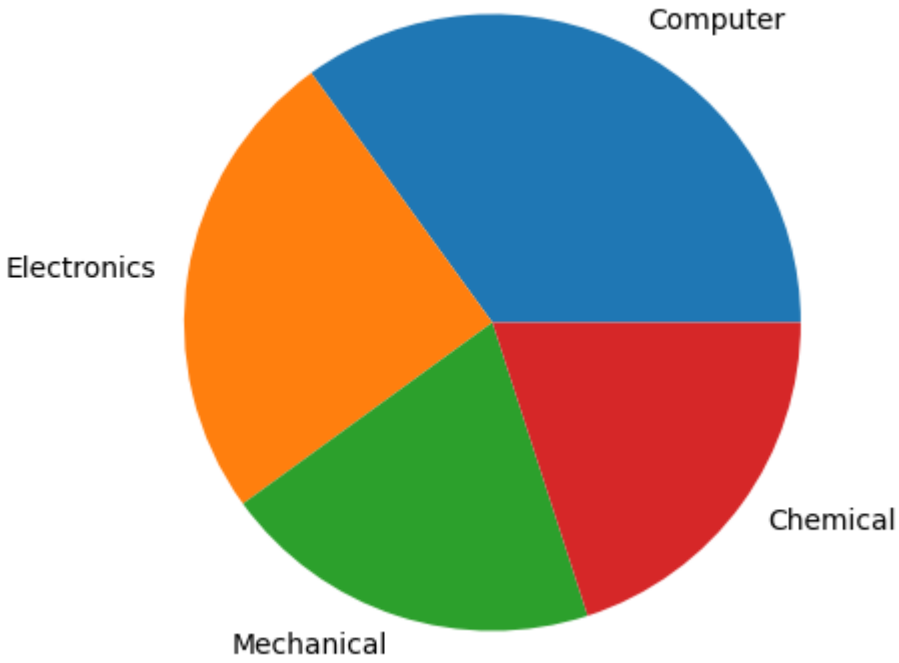
Stacked Bar Chart

In [31]:
```
A = [15., 30., 45., 22.]

B= [15.,25., 50., 20.]

z = range(4)

plt.bar(z, A, color = 'y')
plt.bar(z,B,color = 'g', bottom = A)
plt.show()
```

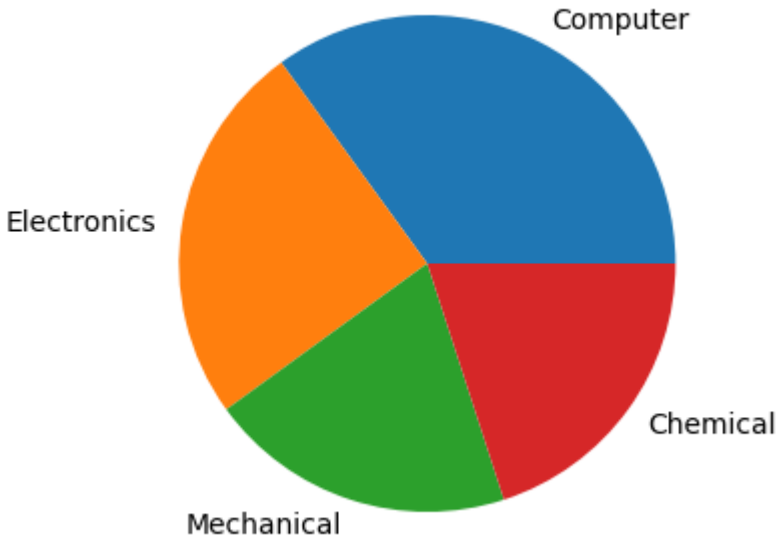


Pie Chart

In [32]:
```
plt.figure(figsize=(5,5))

x= [35,25,20,20]

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

plt.pie(x,labels=labels);

plt.show()
```

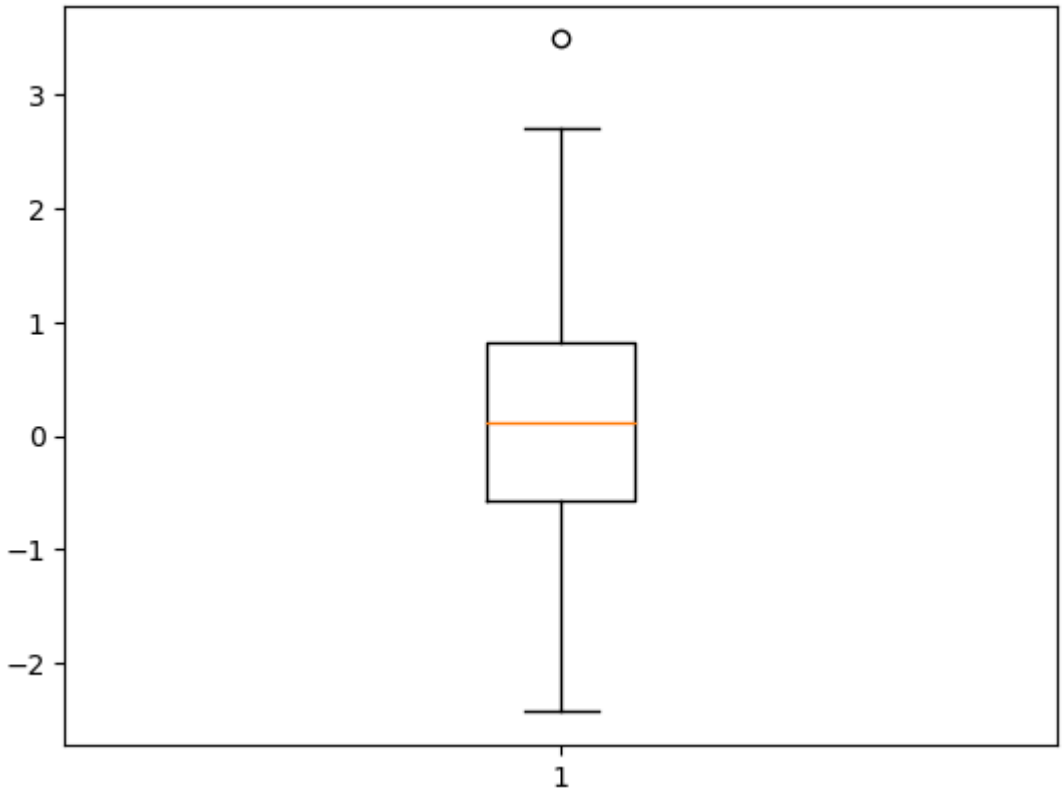


In [33]:
```
plt.figure(figsize=(4,9))

x= [35,25,20,20]

labels = ['Computer', 'Electronics', 'Mechanical', 'Chemical']
color = ['red','greeen','purple','blue']

plt.pie(x,labels=labels);

plt.show()
```



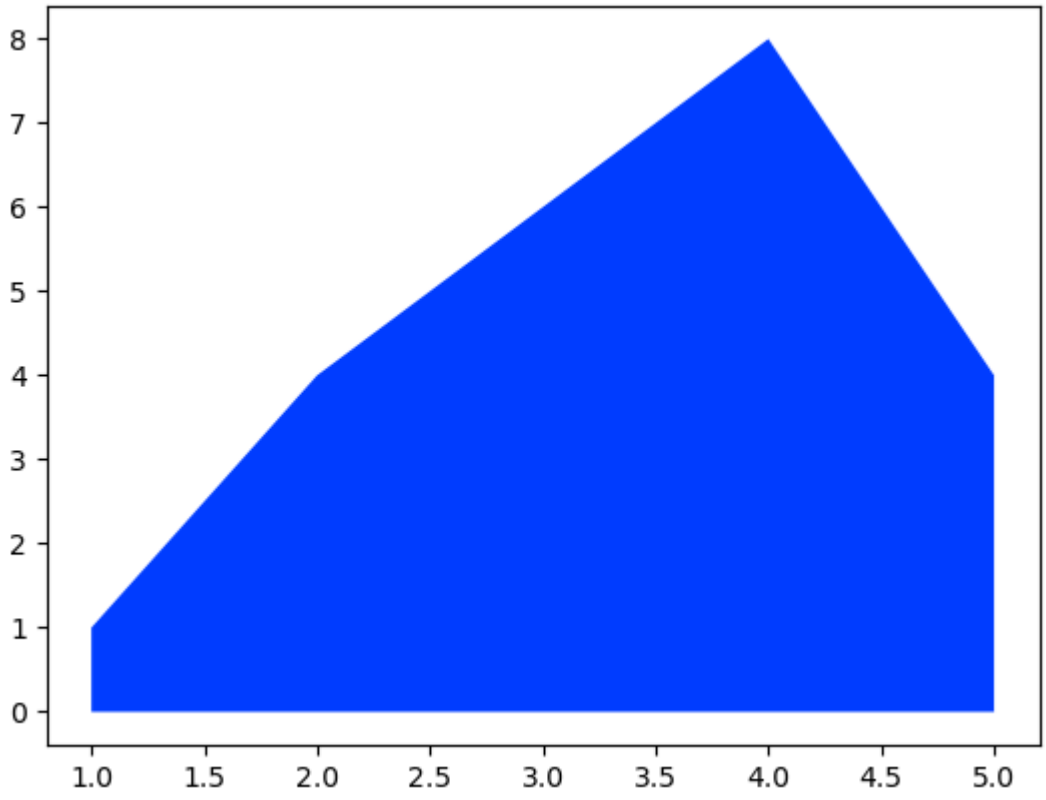
Box plot

In [34]:
```
data = np.random.randn(100)

plt.boxplot(data)

plt.show();
```



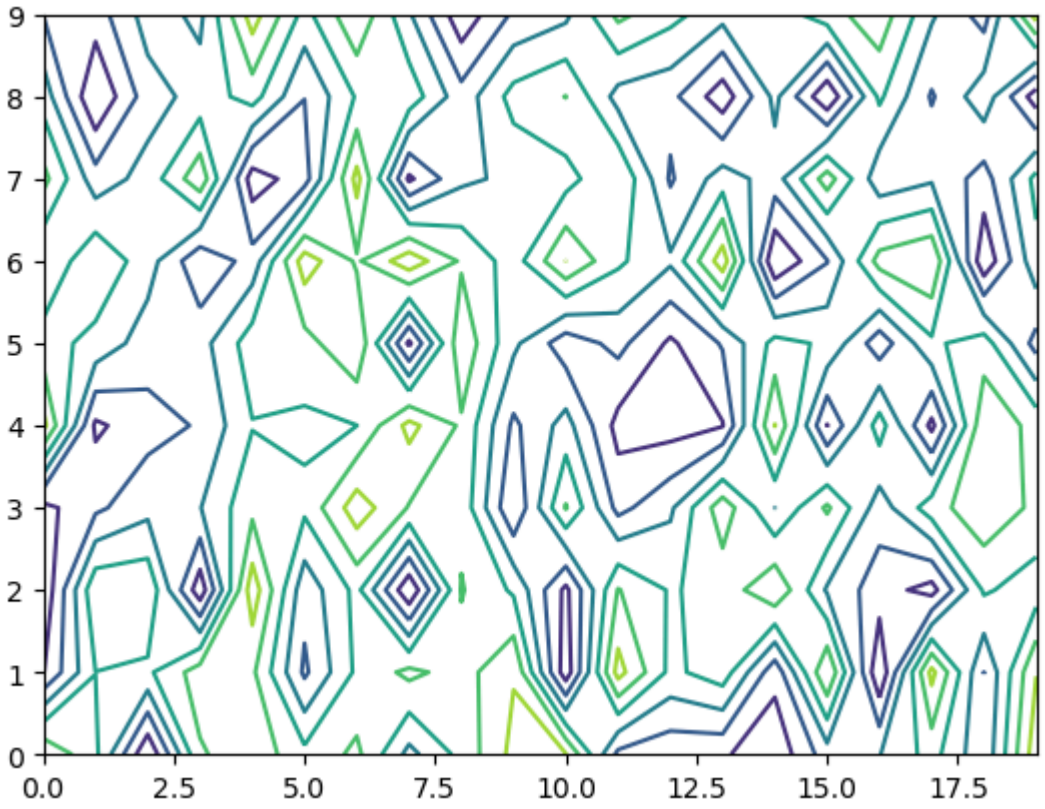
Area Chart

In [48]:
```
# Create some data
x = range(1, 6)
y = [1, 4, 6, 8, 4]

# Area plot
plt.fill_between(x, y)
plt.show()
```



Contour Plot

In [35]:
```
matrix1 = np.random.rand(10, 20)
cp = plt.contour(matrix1)
plt.show()
```



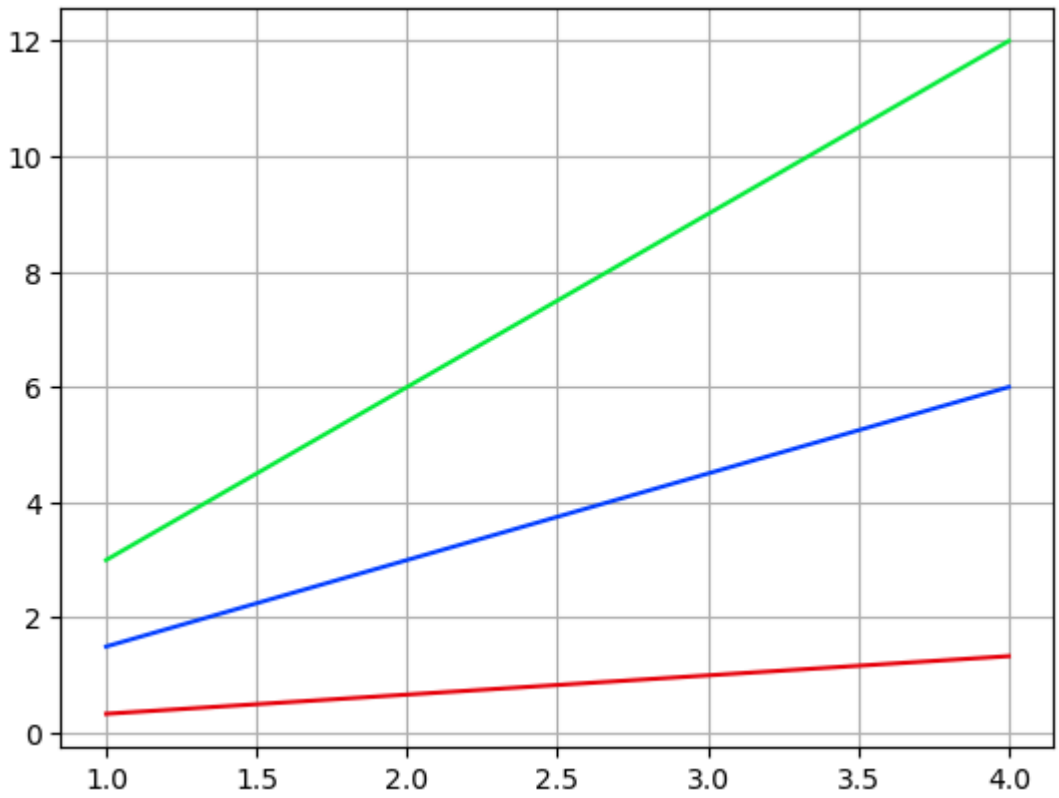
In [36]:
```
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-n
ogrid', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'g
gplot', '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-not
ebook', 'seaborn-v0_8-paper', 'seaborn-v0_8-pastel', 'seaborn-v0_8-poste
r', 'seaborn-v0_8-talk', 'seaborn-v0_8-ticks', 'seaborn-v0_8-white', 'seab
orn-v0_8-whitegrid', 'tableau-colorblind10']
```

In [37]:
```
plt.style.use('seaborn-bright')
```

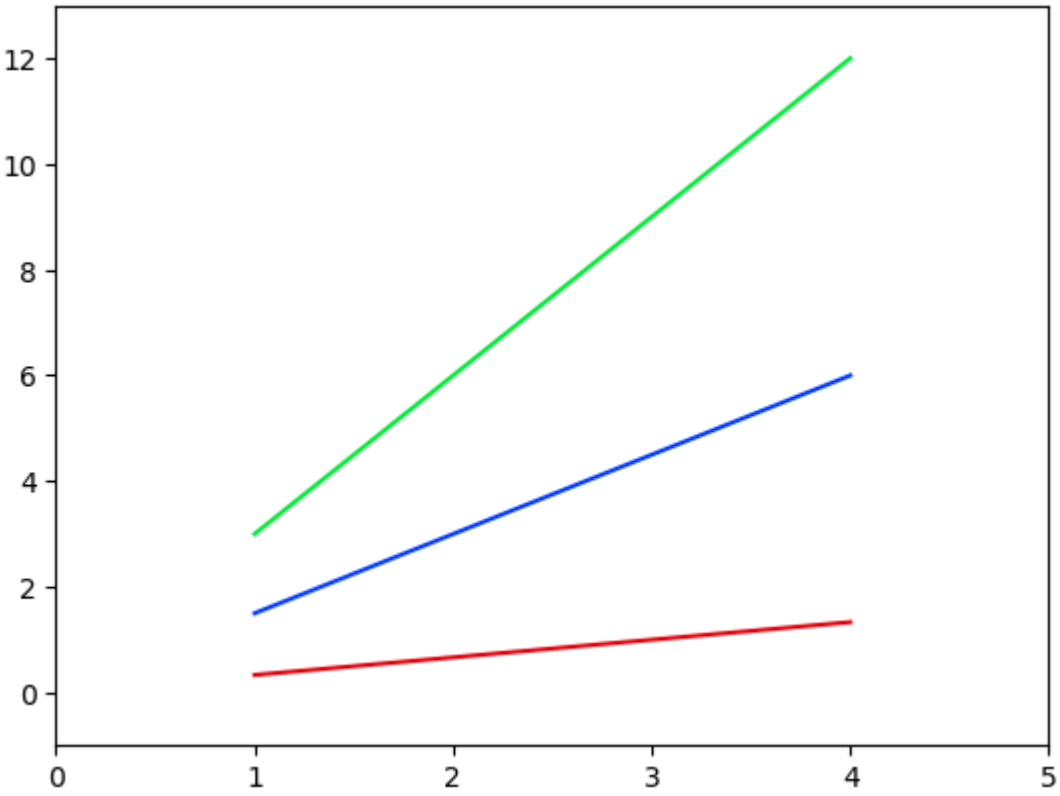
Grid

In [38]:
```
x = np.arange(1,5)
plt.plot(x,x*1.5,x,x*3.0,x,x/3.0)
plt.grid(True)
plt.show()
```



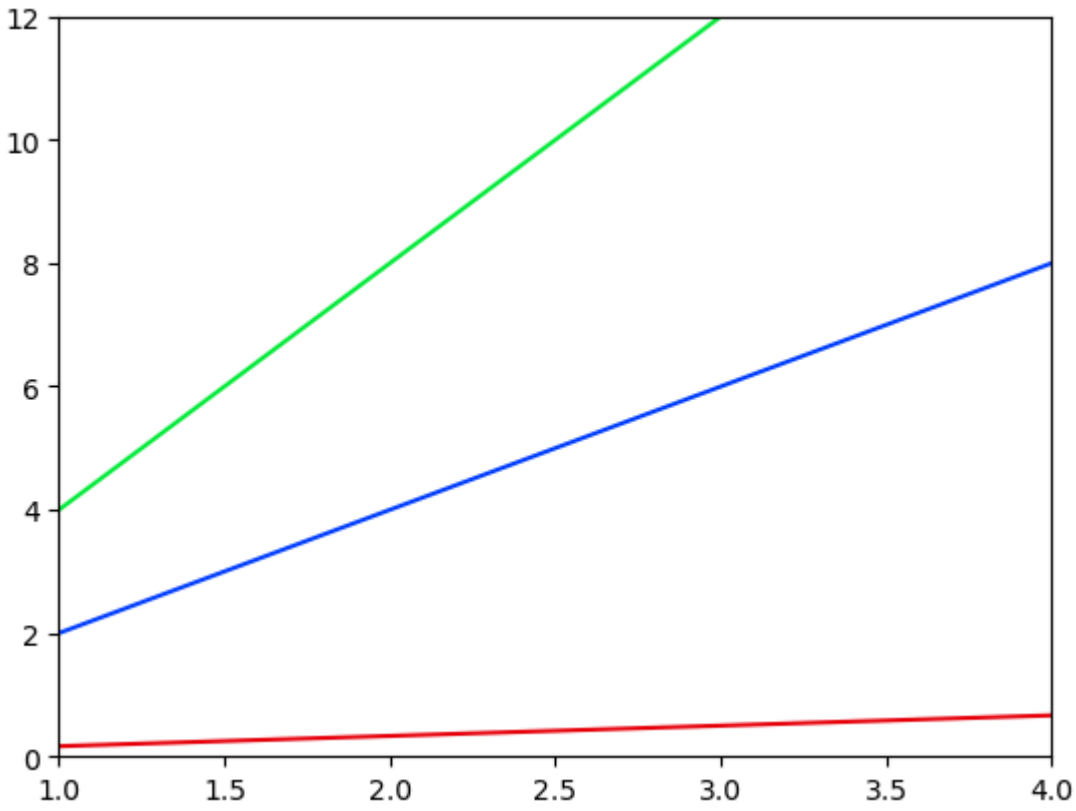
Handling Axes

In [39]:
```
x = np.arange(1,5)
plt.plot(x,x*1.5,x,x*3.0,x,x/3.0)
plt.axis()
plt.axis([0, 5, -1, 13])
plt.show()
```



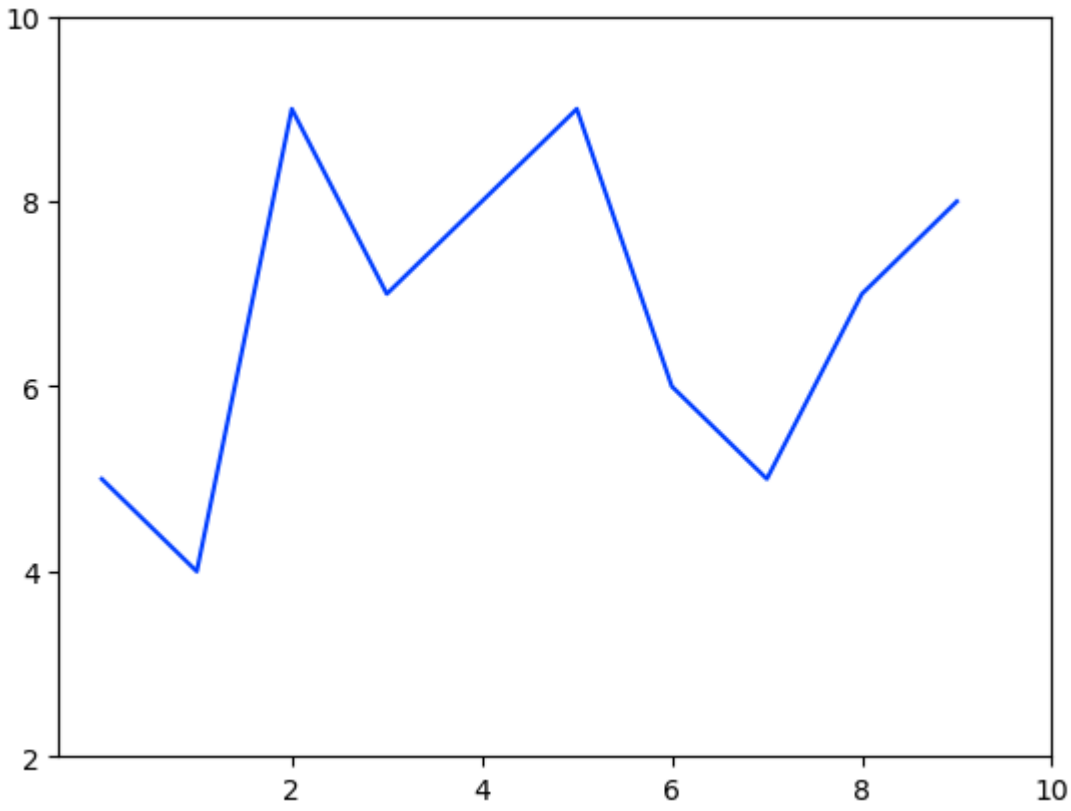
In [40]:
```
x = np.arange(1,5)
plt.plot(x,x*2,x,x*4,x,x/6)
plt.xlim([1.0, 4.0])
plt.ylim([0.0, 12.0])
```

Out[40]: (0.0, 12.0)



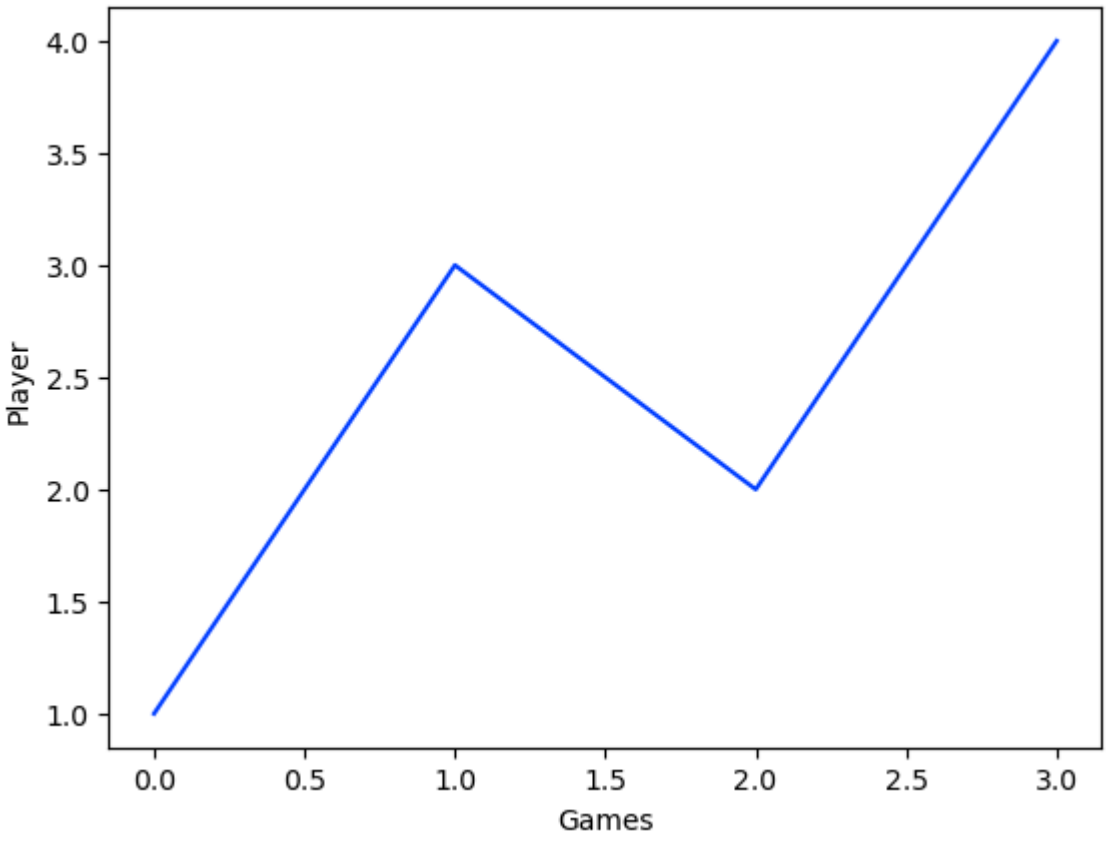
Handling X and Y axis

In [41]:
```
x = [5, 4, 9, 7, 8, 9, 6, 5, 7, 8]

plt.plot(x)

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

plt.show()
```



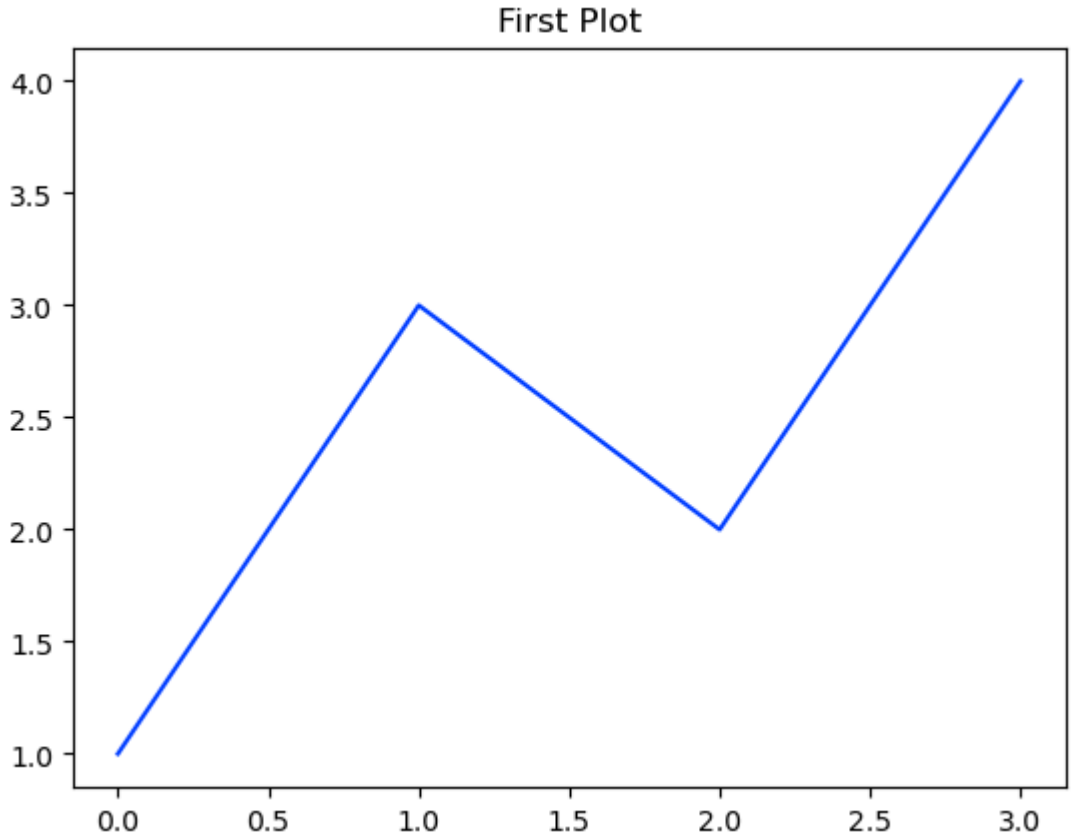
Adding Lables

In [42]:
```
plt.plot([1, 3, 2, 4])

plt.xlabel('Games')

plt.ylabel('Player')

plt.show()
```



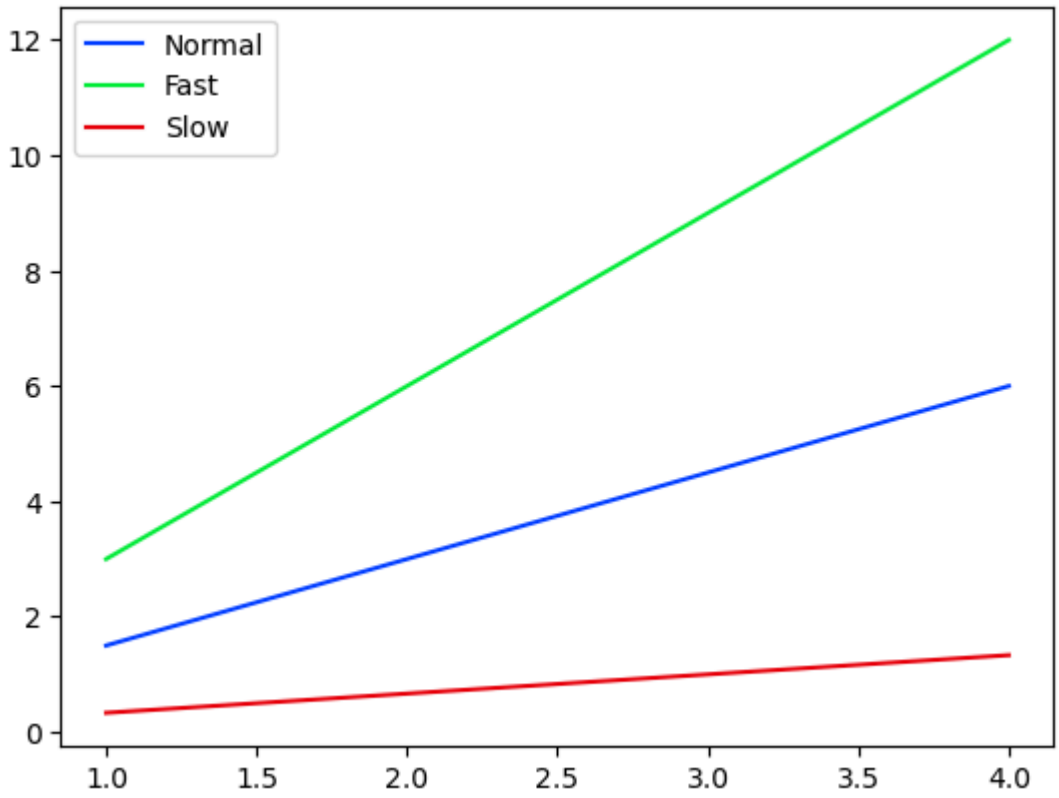
Adding Title

In [43]:
```
plt.plot([1, 3, 2, 4])

plt.title('First Plot')

plt.show()
```

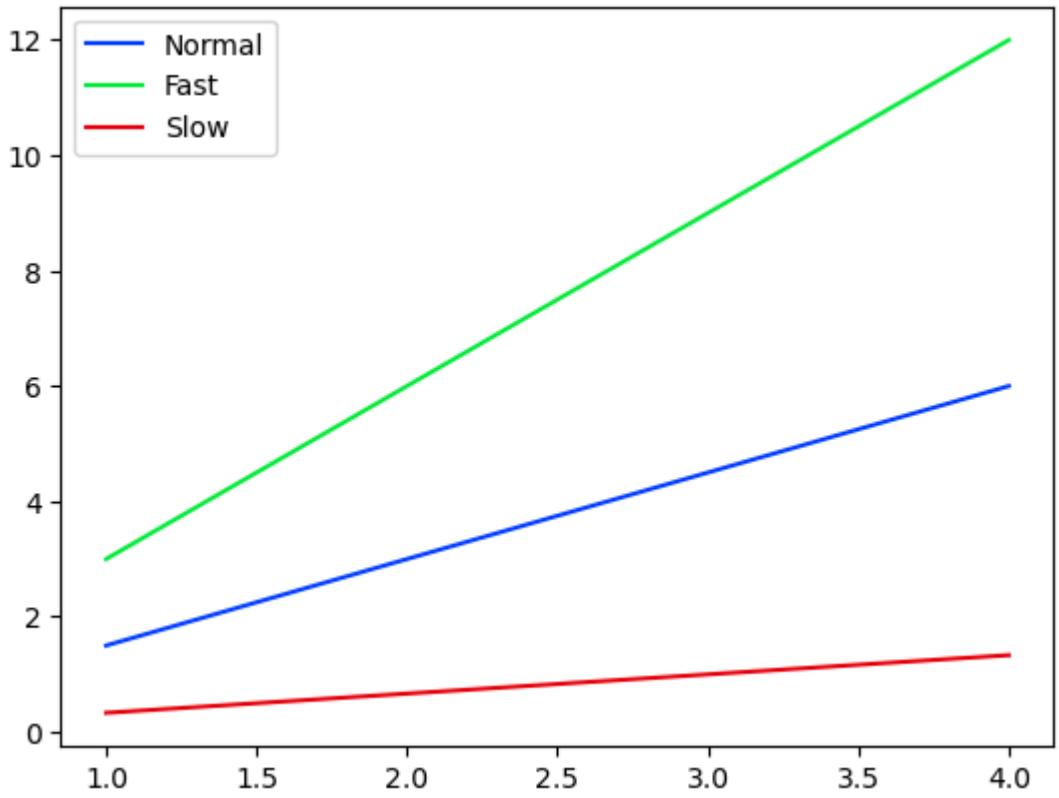


Adding Legend

In [44]:
```
x = np.arange(1, 5)

fig, ax = plt.subplots()

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

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

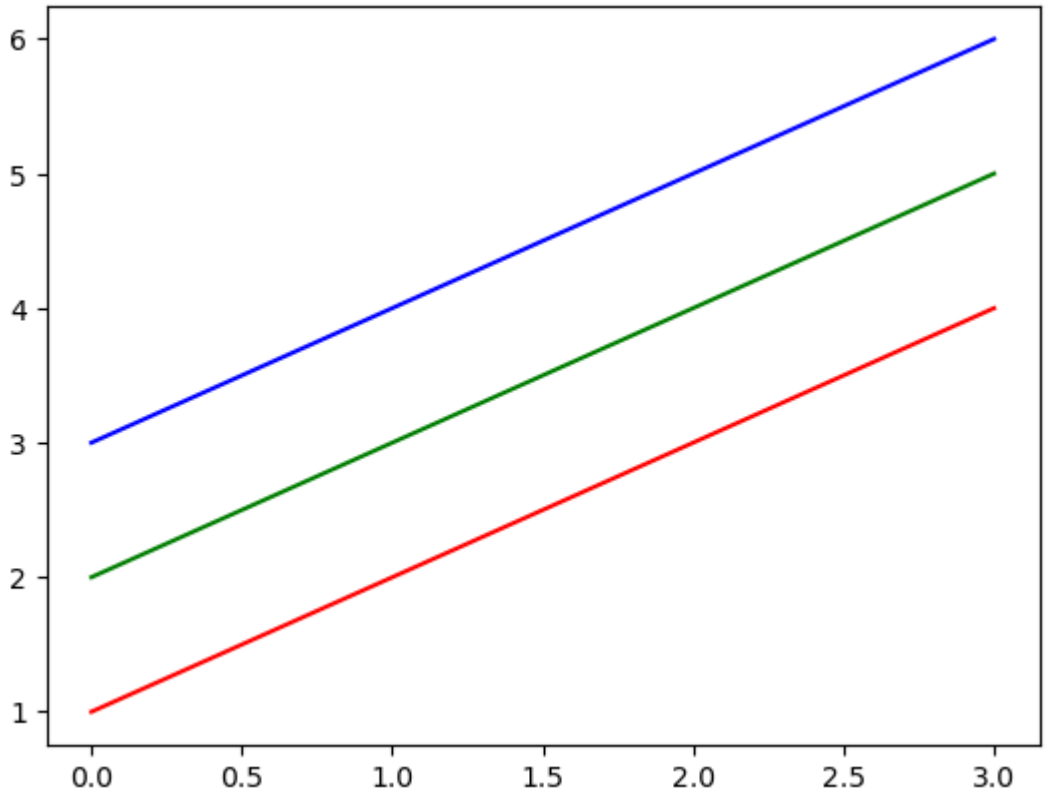


In [45]:
```
x = np.arange(1, 5)

fig, ax = plt.subplots()

ax.plot(x, x*1.5, label='Normal')
ax.plot(x, x*3.0, label='Fast')
ax.plot(x, x/3.0, label='Slow')

ax.legend();
```



Control Colours

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

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

plt.show()
```



Control line styles

In [47]:
```
x = np.arange(1, 5)

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

plt.show()
```