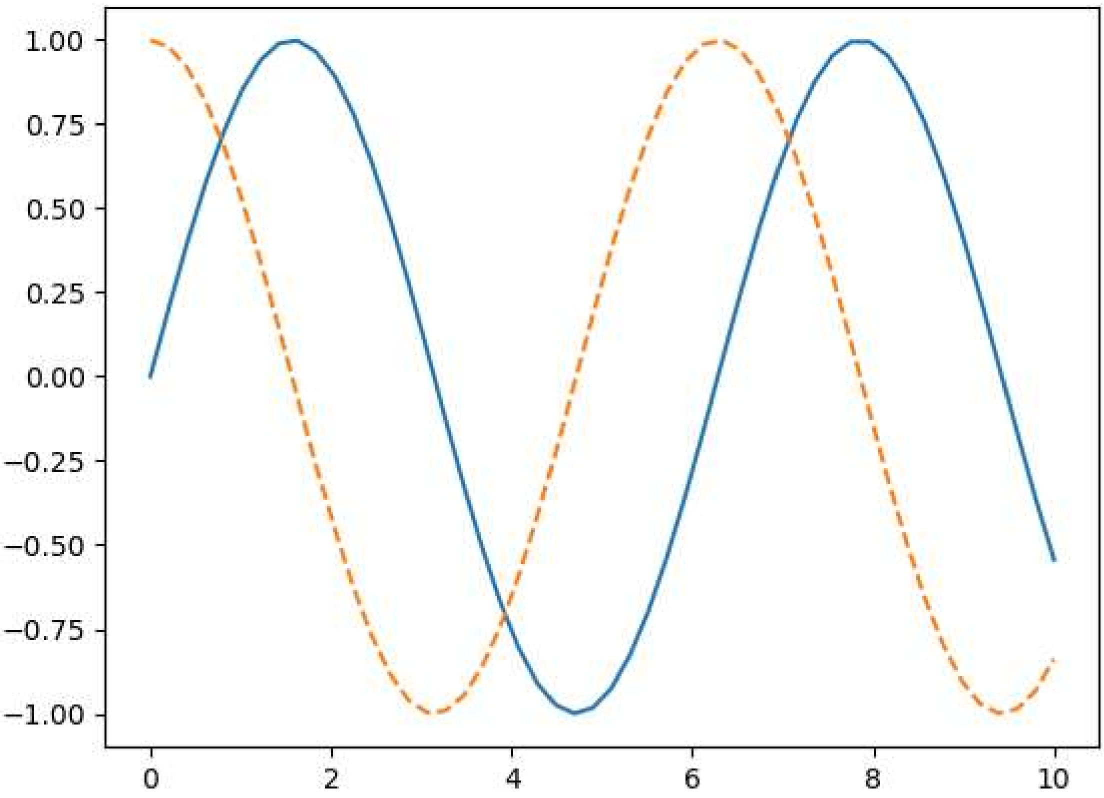
Matplotlib

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

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

In [5]:
```
 #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 [6]:
```
plt.gcf()     # get current figure
```

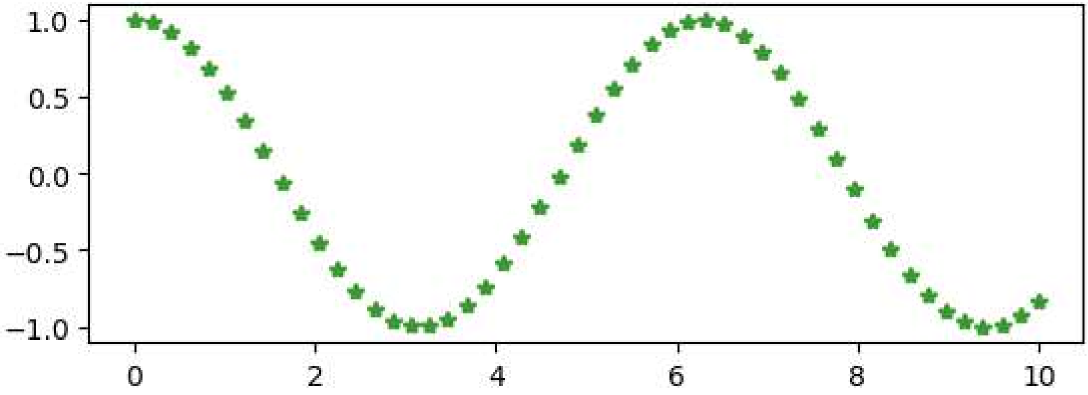
Out[6]:
```
<Figure size 640x480 with 0 Axes>
```

In [7]:
```
plt.gca()      # get current axes
```

```
Out[7]: <Axes: >
```

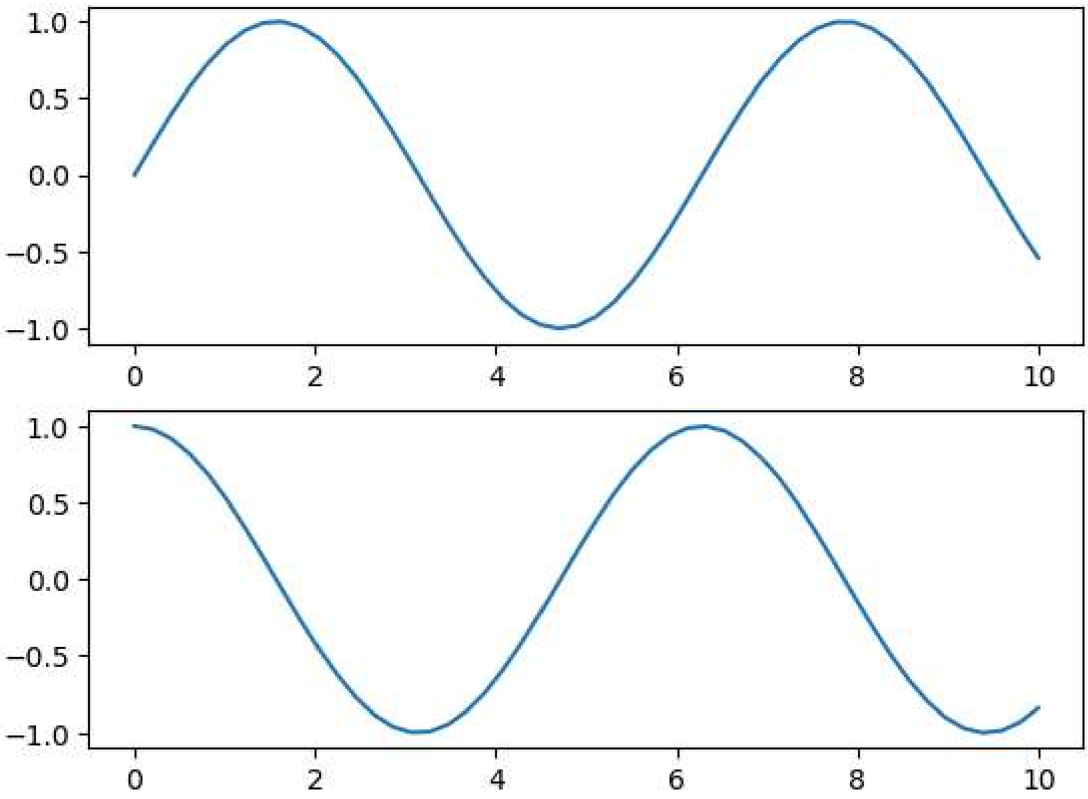
In [13]:

```
# 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 [18]:

```
# 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)     # (rows, columns, panel number)
plt.plot(x1,np.cos(x1));
plt.show()
```



In [26]:

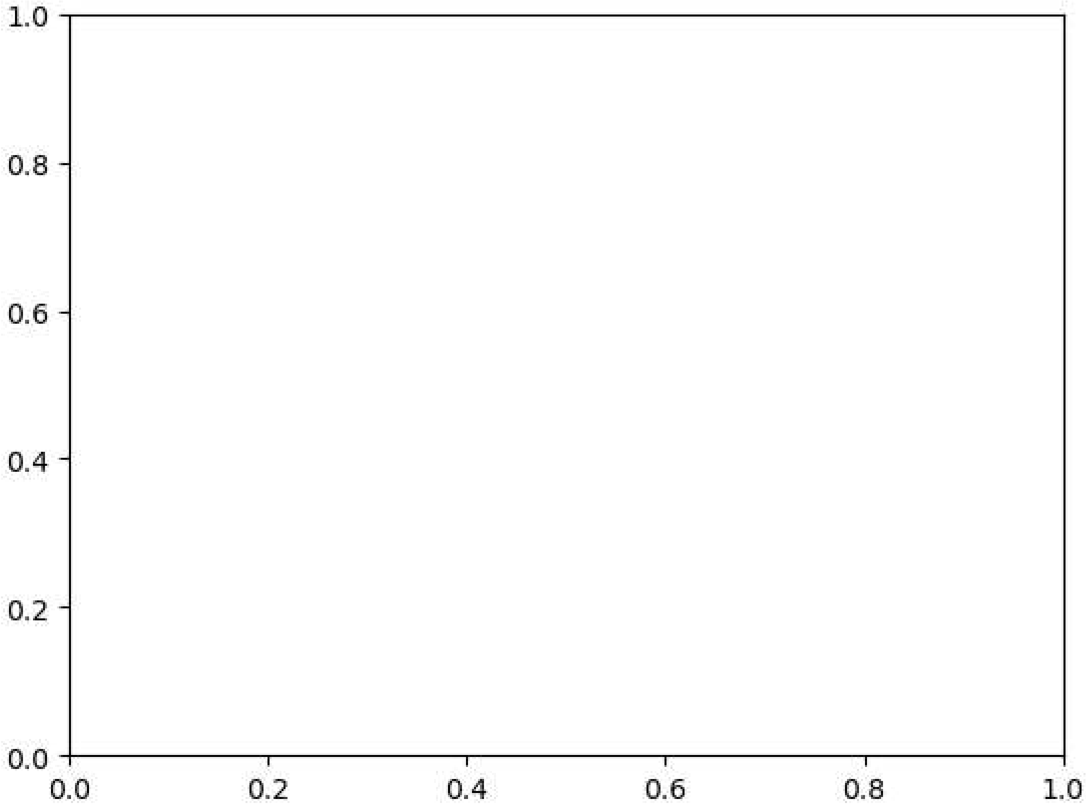
```
# get current figure information
print(plt.gcf())
plt.show()
```

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

In [27]:

```
# get current axis information

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

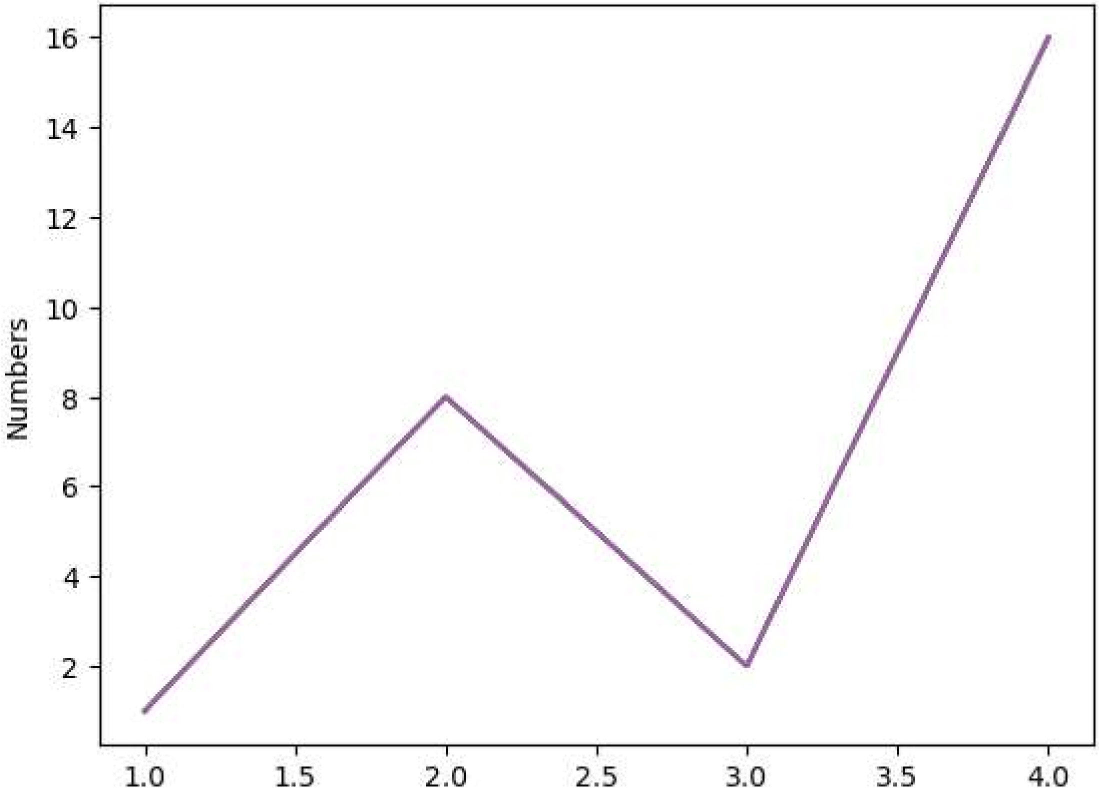
```
Axes(0.125,0.11;0.775x0.77)
```



Visualization with Pyplot

In [32]:

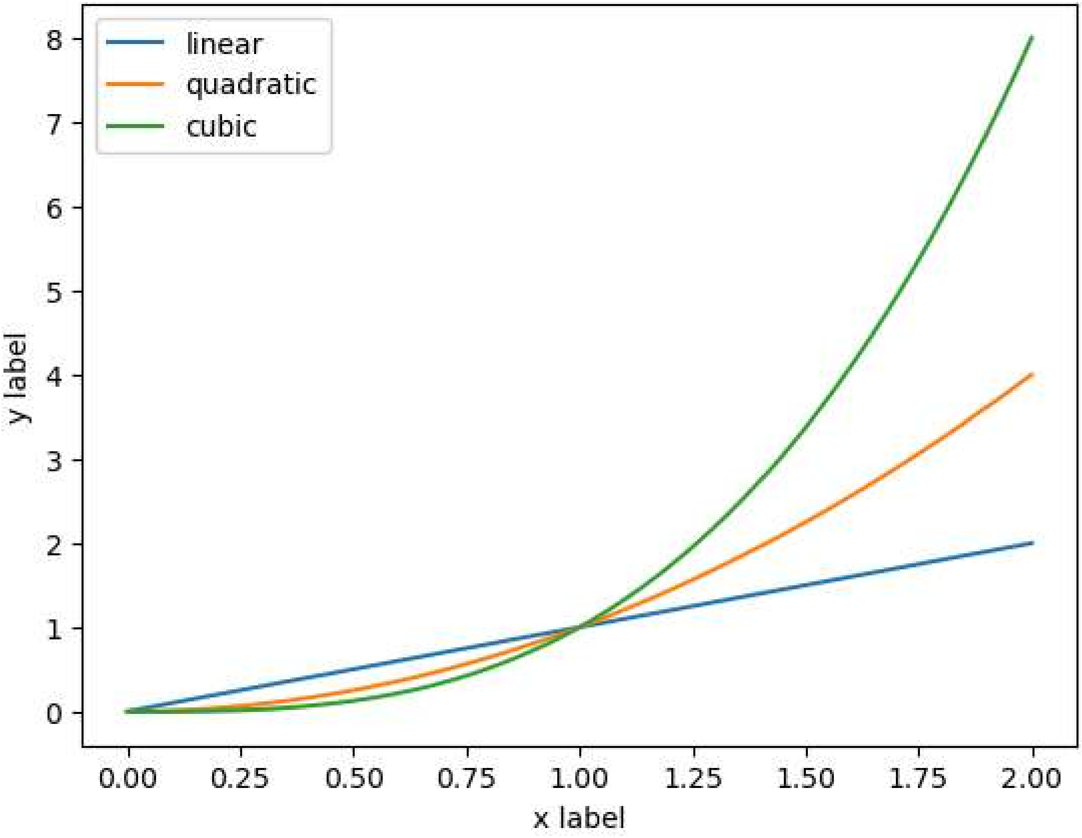
```
plt.plot([1,2,3,4],[1,8,2,16])
plt.ylabel('Numbers')
plt.show()
```



State-machine interface

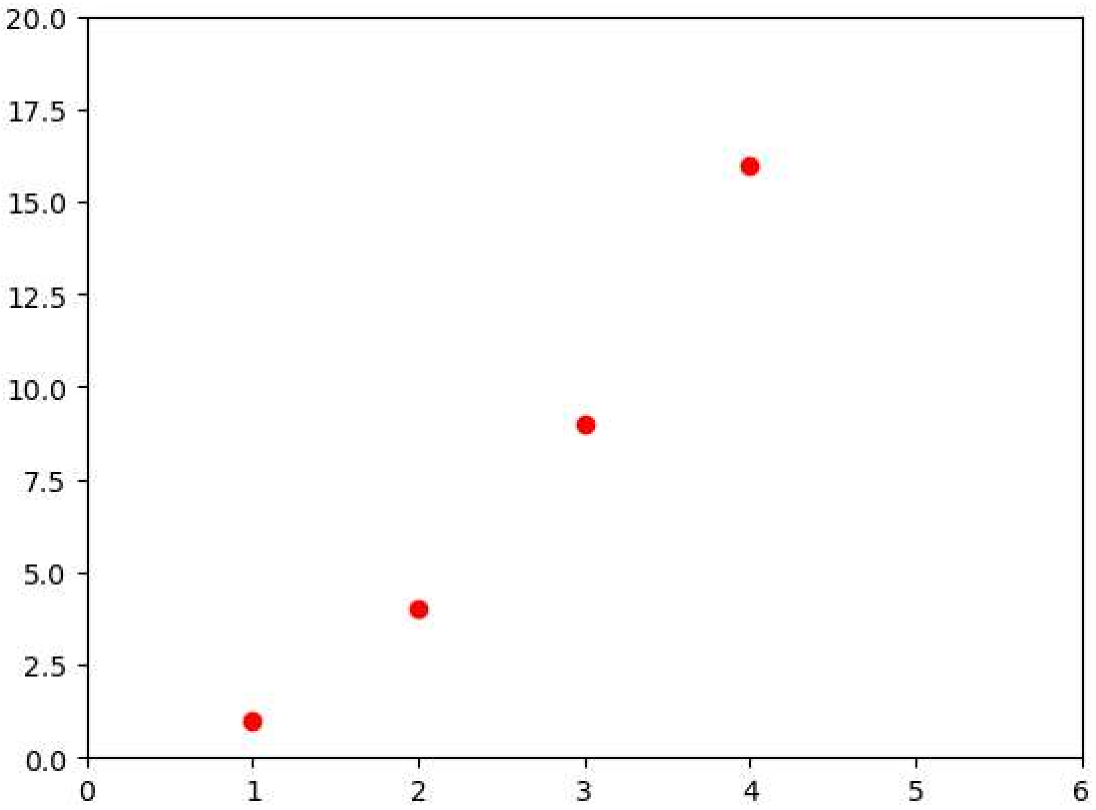
In [37]:

```
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.legend()
plt.show()
```



Formatting the style of plot

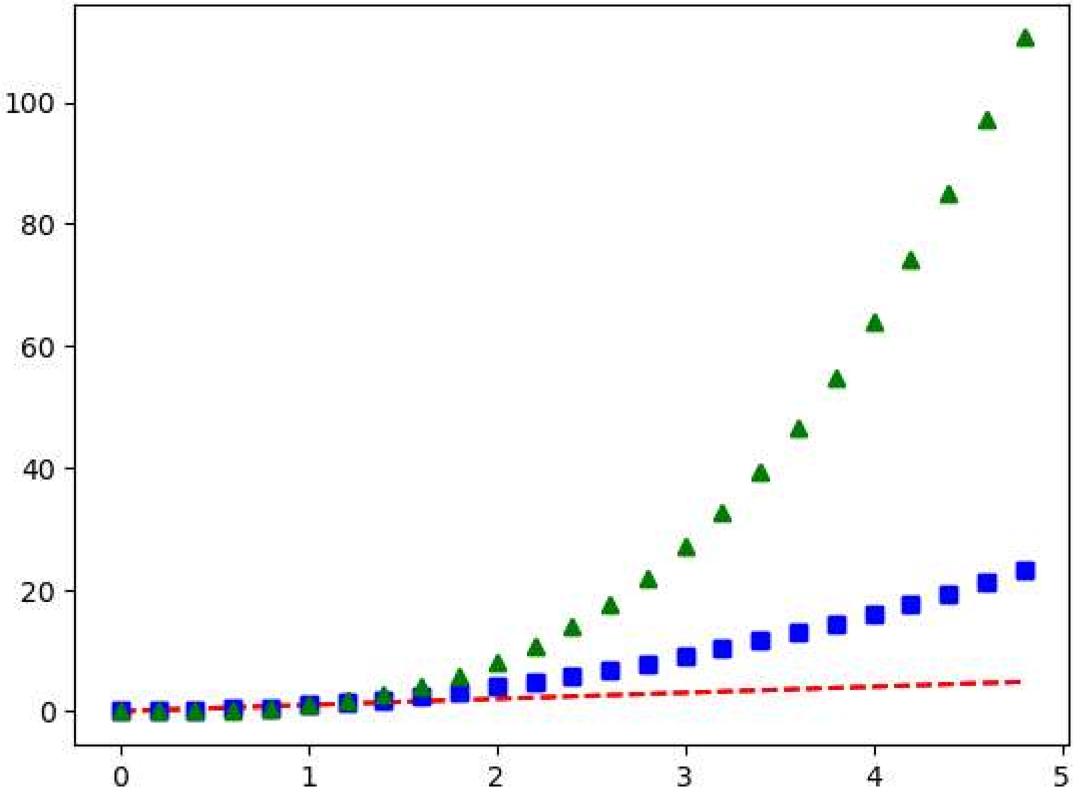
In [38]:
```
plt.plot([1,2,3,4],[1,4,9,16],'ro')
plt.axis([0,6,0,20])
plt.show()
```



Working with NumPy arrays

In [40]:

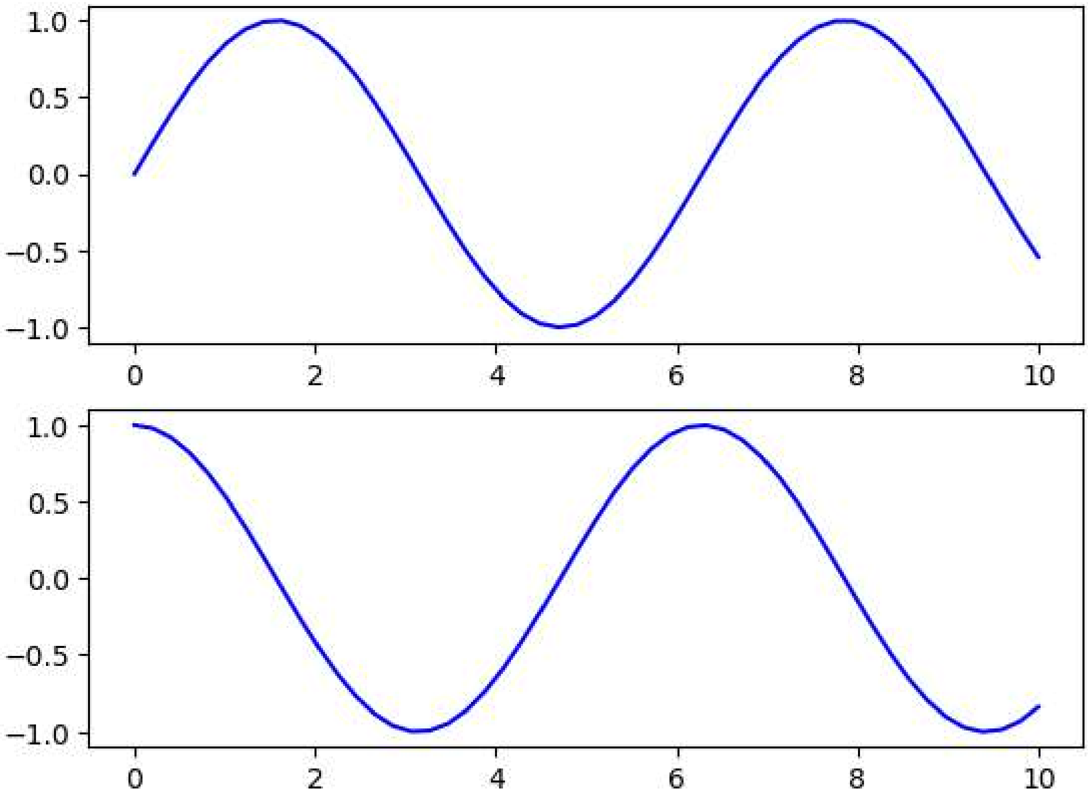
```
# evenly sampled time at 200ms intervals
t = np.arange(0.,5.,0.2)
# red dashes, blue squares and green triangles
plt.plot(t,t,'r--',t,t**2,'bs',t,t**3,'g^')
plt.show()
```



Object-Oriented API

In [43]:

```
# 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 [47]:

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

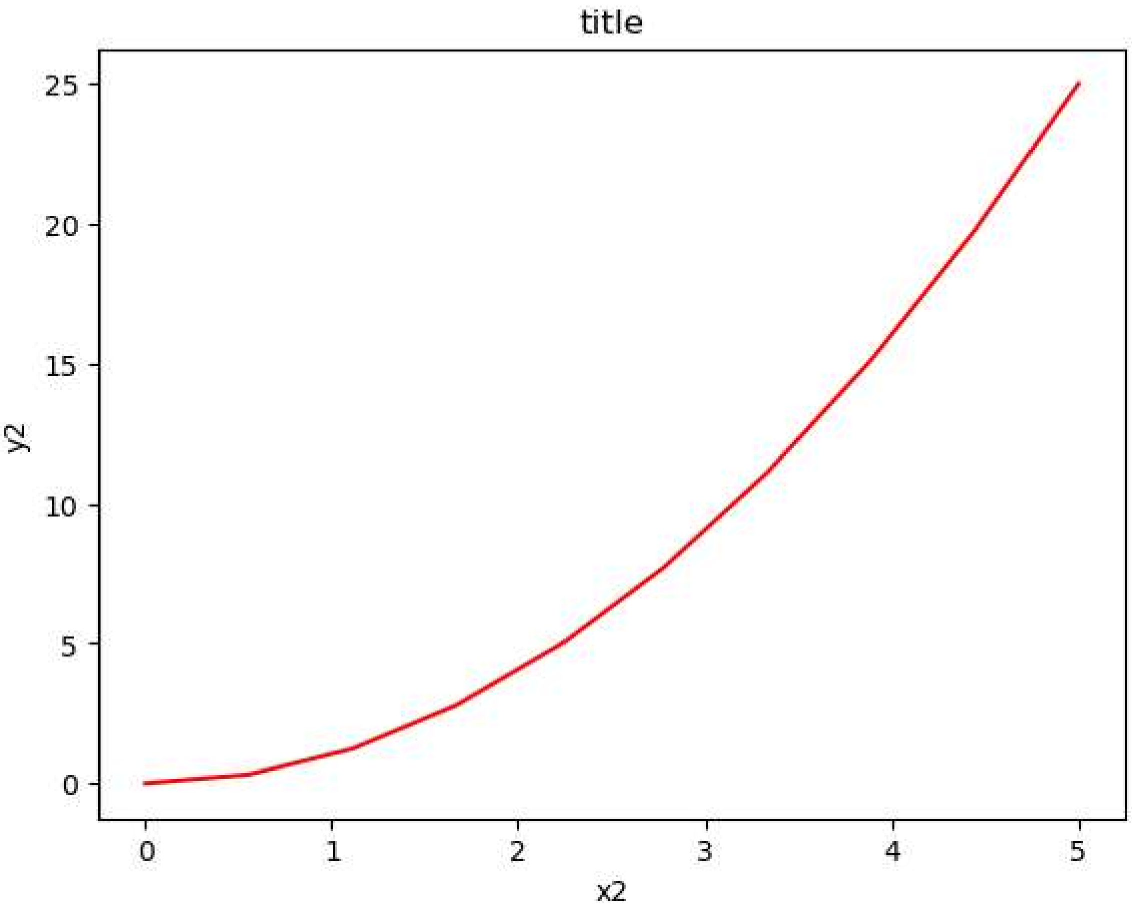


Figure and Axes

In [51]:
```
fig = plt.figure()
ax = plt.axes()
plt.show()
```

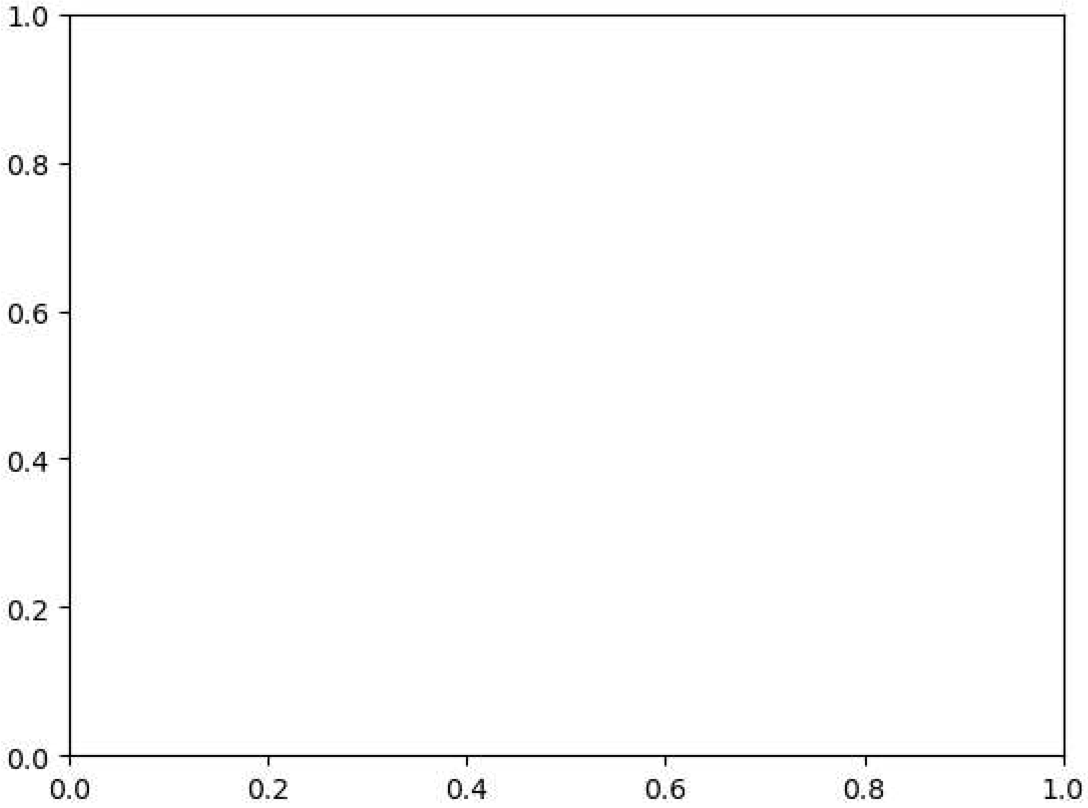
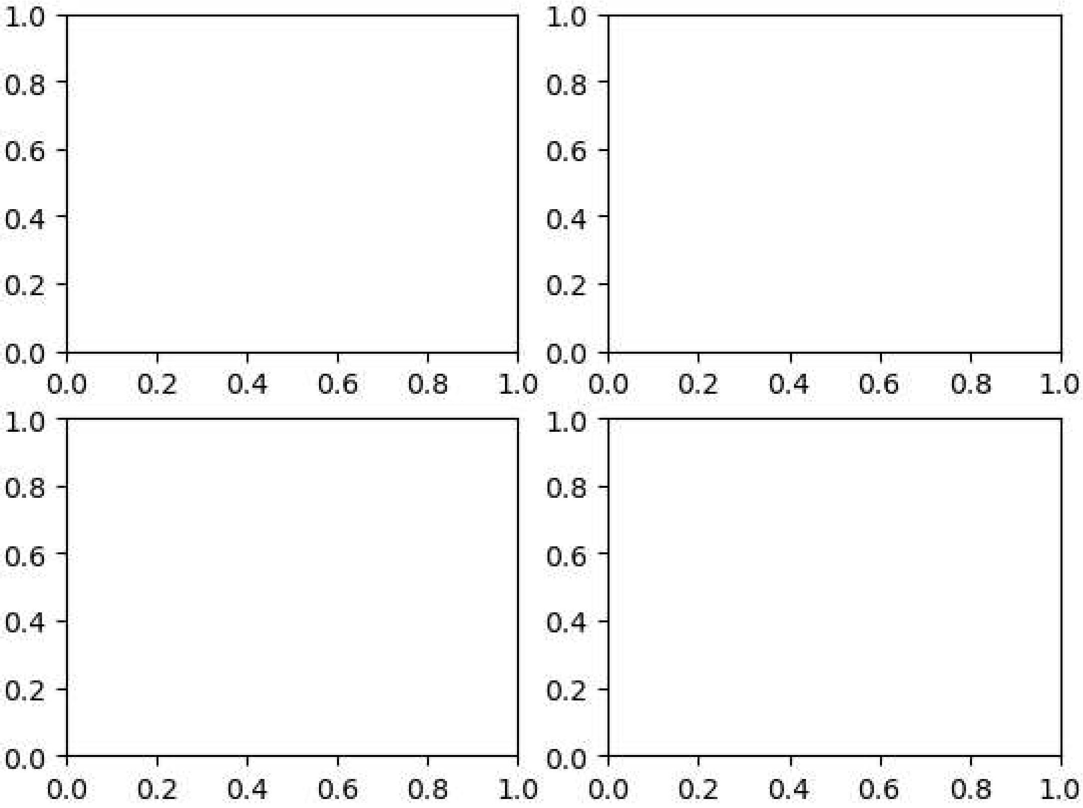


Figure and Subplots

In [52]:

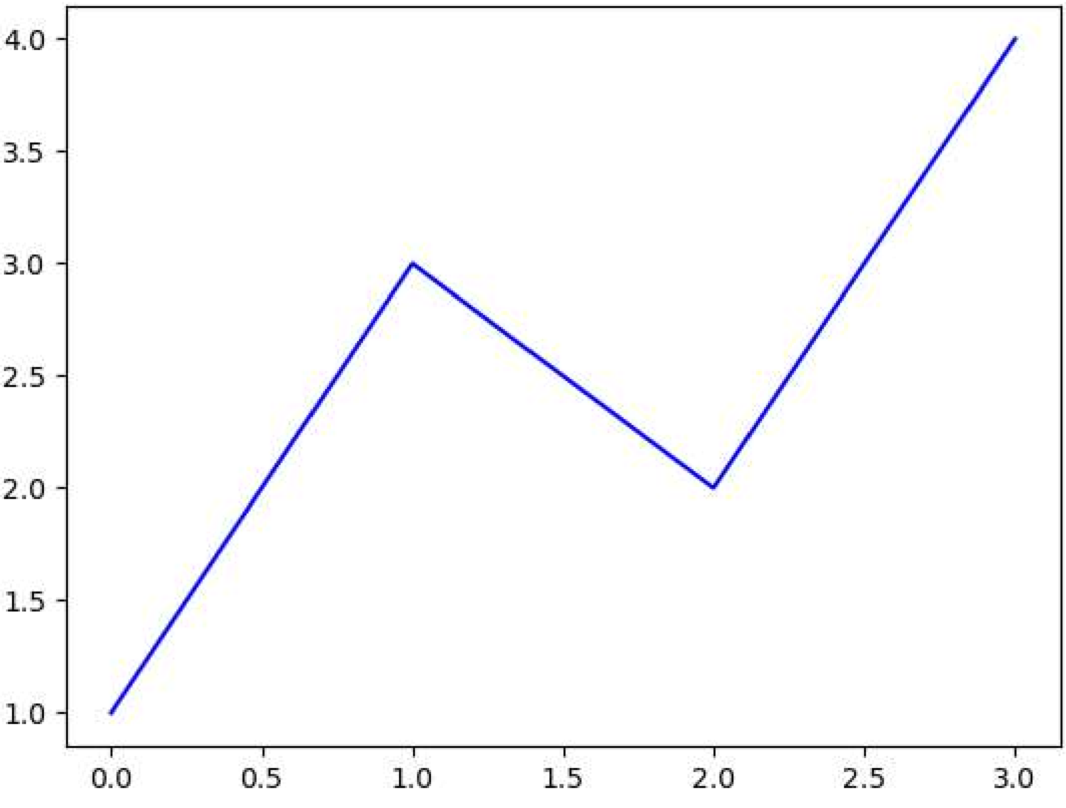
```
fig = plt.figure()

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 [66]:

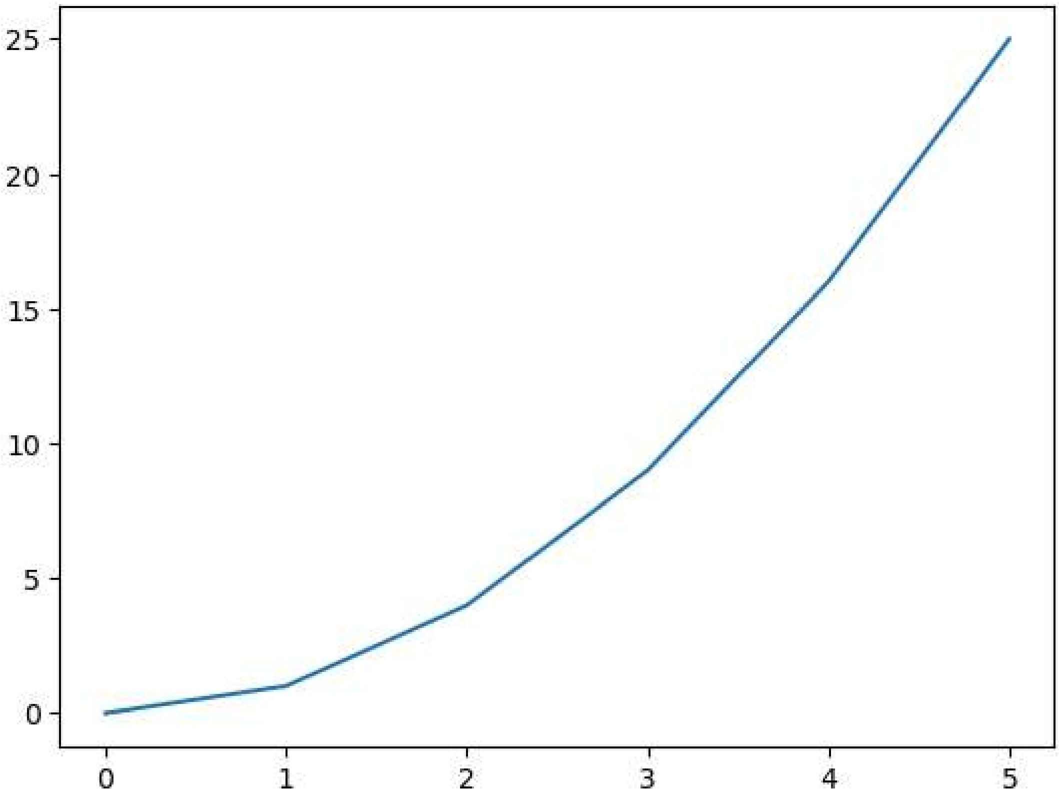
```
plt.plot([1,3,2,4],'b-')
plt.show()
```



Specify both Lists

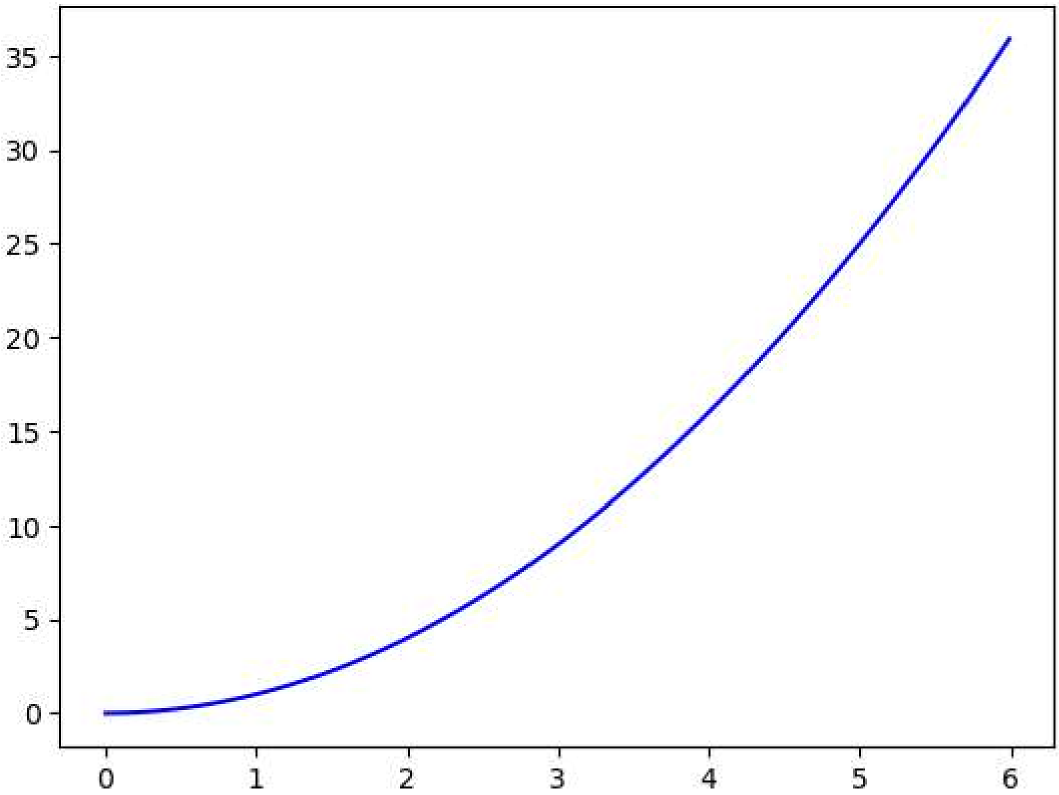
In [72]:

```
x3 = range(6)
plt.plot(x3, [x1**2 for x1 in x3])
plt.show()
```



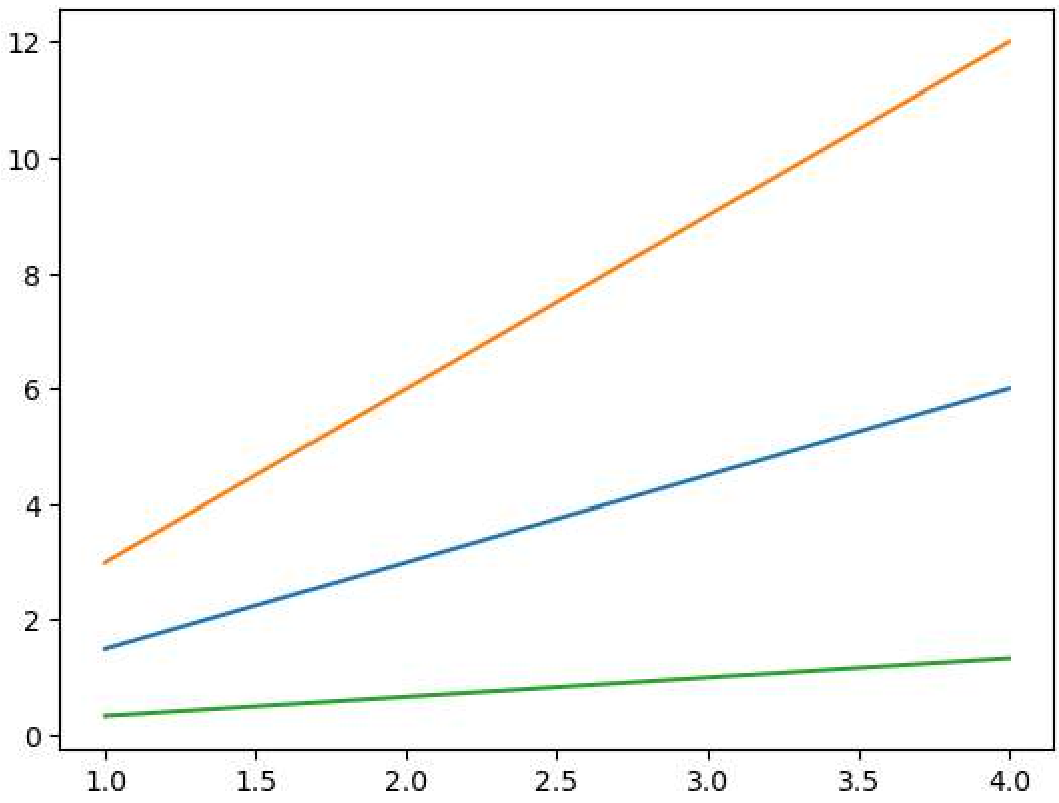
In [71]:

```
x3 = np.arange(0.0,6.0,0.01)
plt.plot(x3,[x1**2 for x1 in x3], 'b-')
plt.show()
```



Multiline Plots

```
In [73]: x4 = range(1,5)
         plt.plot(x4, [x1*1.5 for x1 in x4])
         plt.plot(x4, [x1*3 for x1 in x4])
         plt.plot(x4, [x1/3.0 for x1 in x4])
         plt.show()
```



Saving the Plot

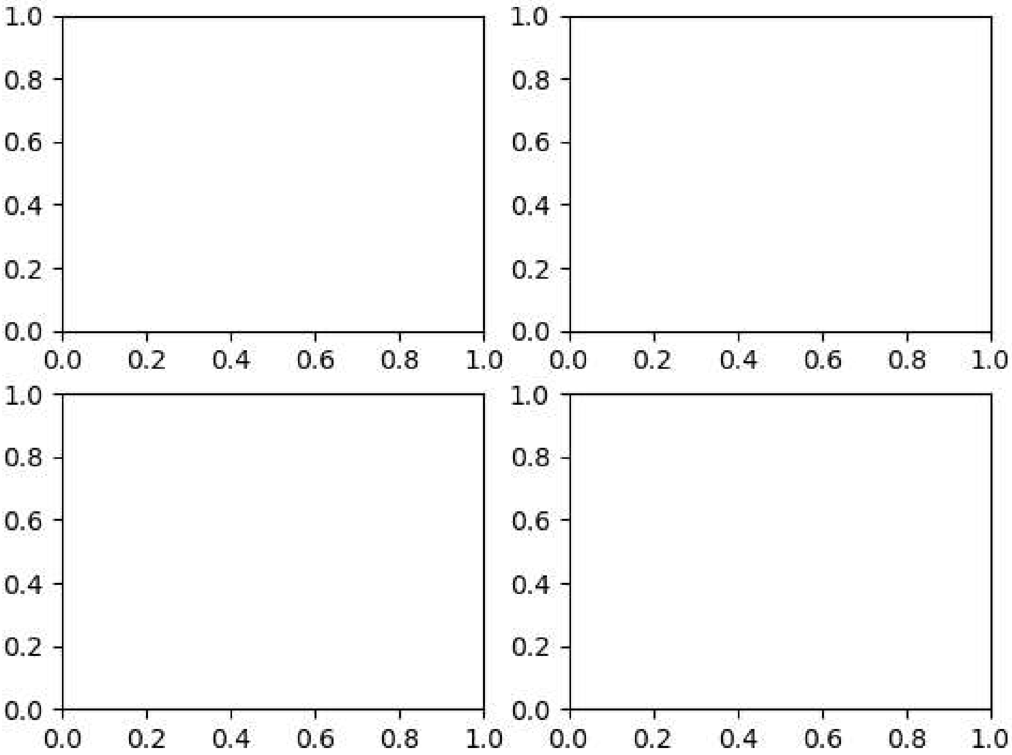
In [74]:

```
# Saving the figure
fig.savefig('plot1.png')
```

In [81]:

```
# Explore the contents of figure
from IPython.display import Image
Image('plot1.png')
```

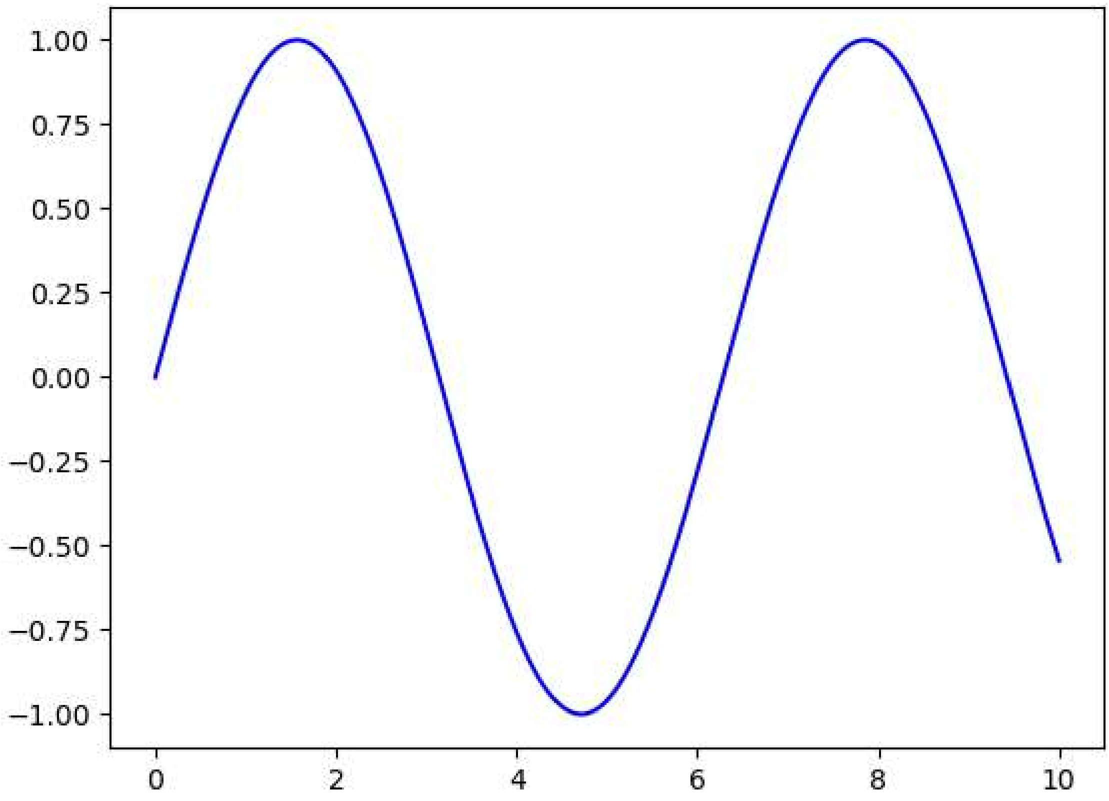
Out[81]:



Line Plot

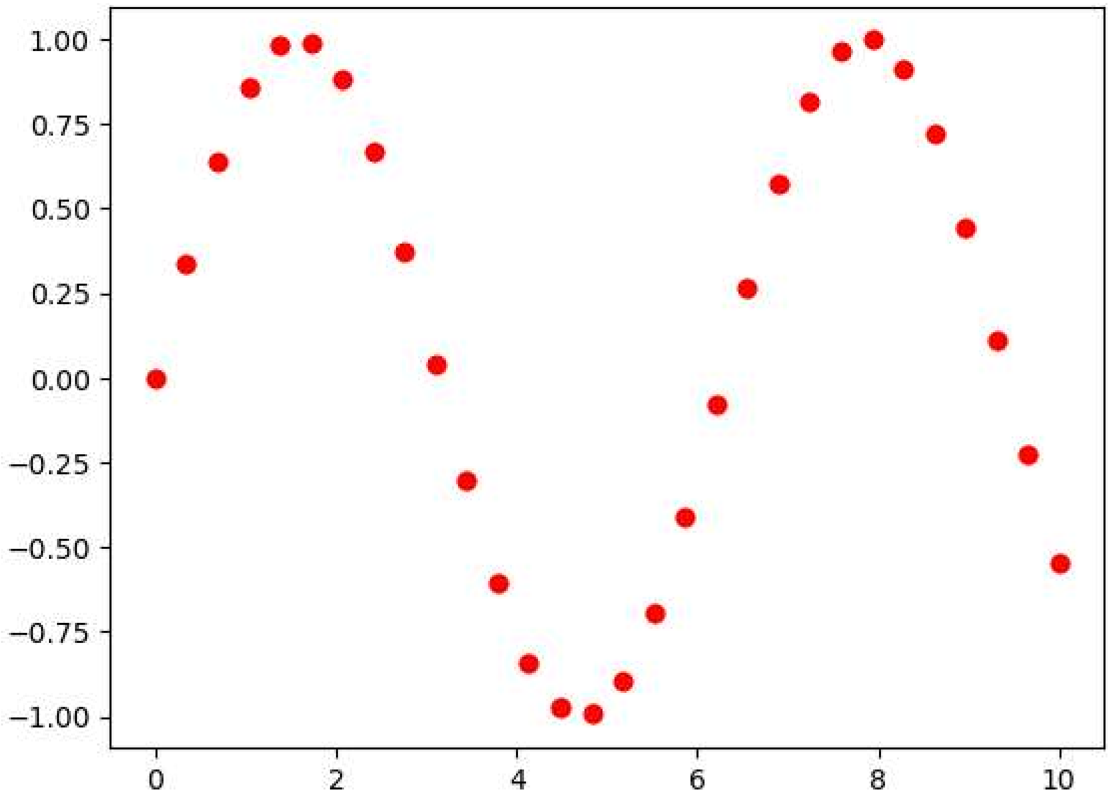
In [86]:

```
# 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-')
plt.show()
```



Scatter Plot

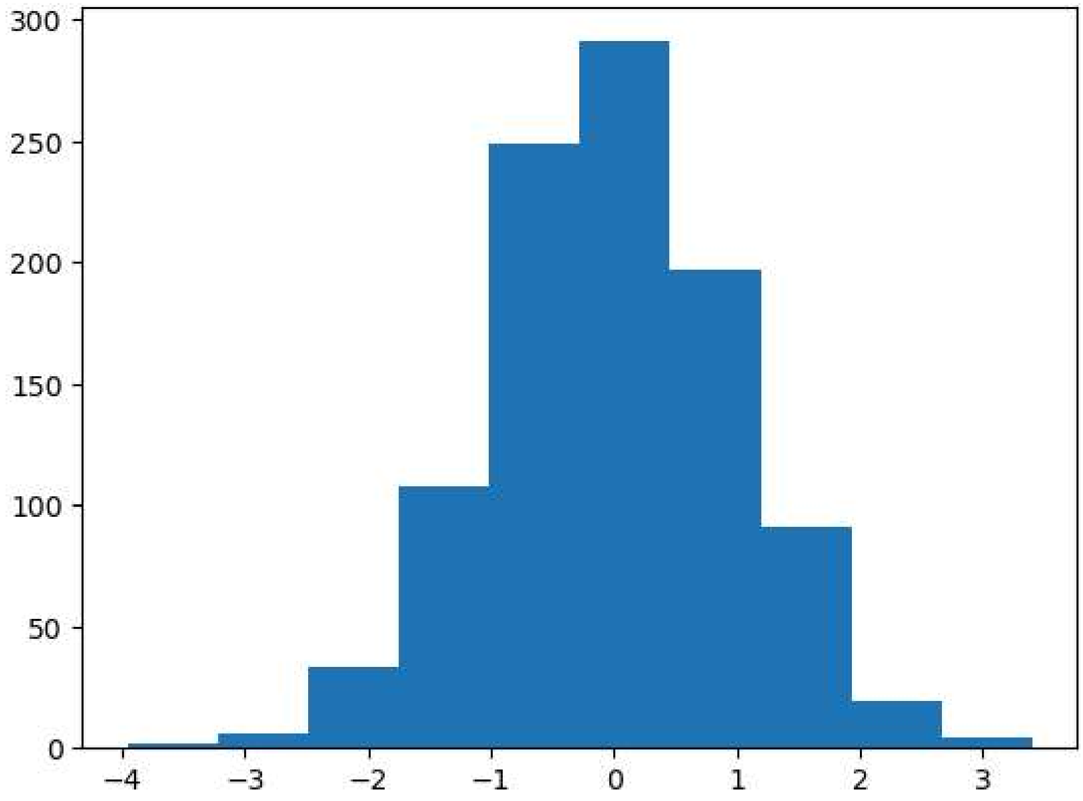
```
In [88]: x7 = np.linspace(0,10,30)

         y7 = np.sin(x7)

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



Histogram

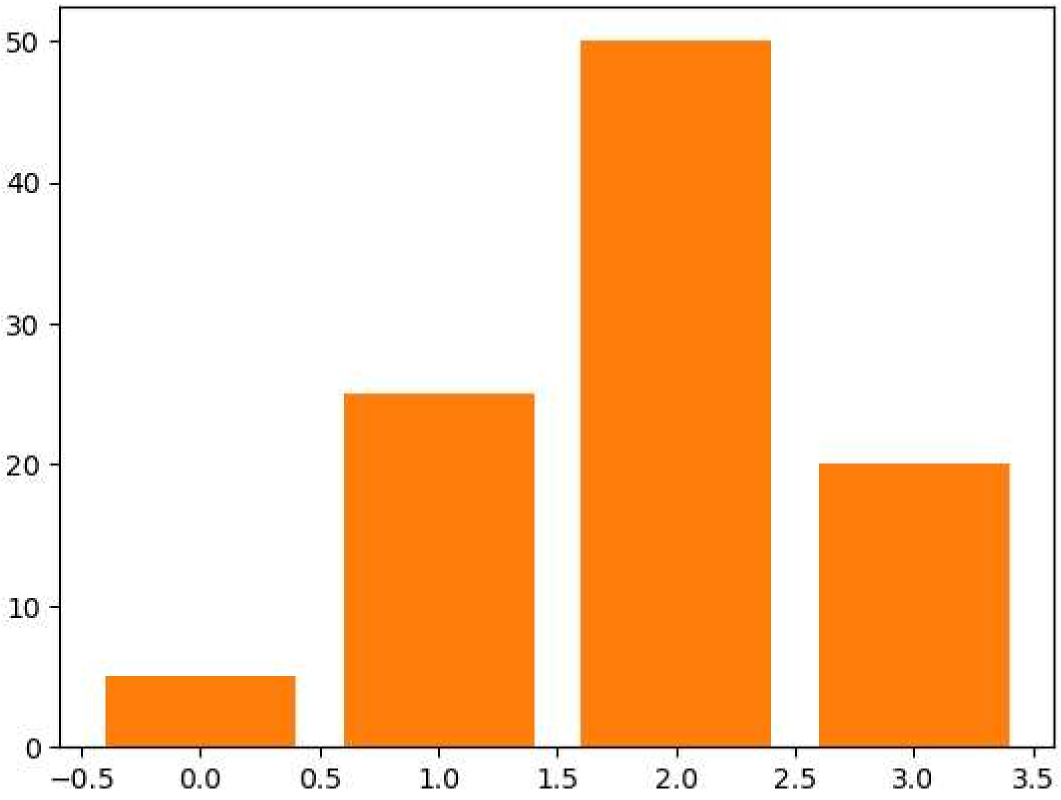
In [91]:

```
data1 = np.random.randn(1000)
plt.hist(data1);
plt.show()
```



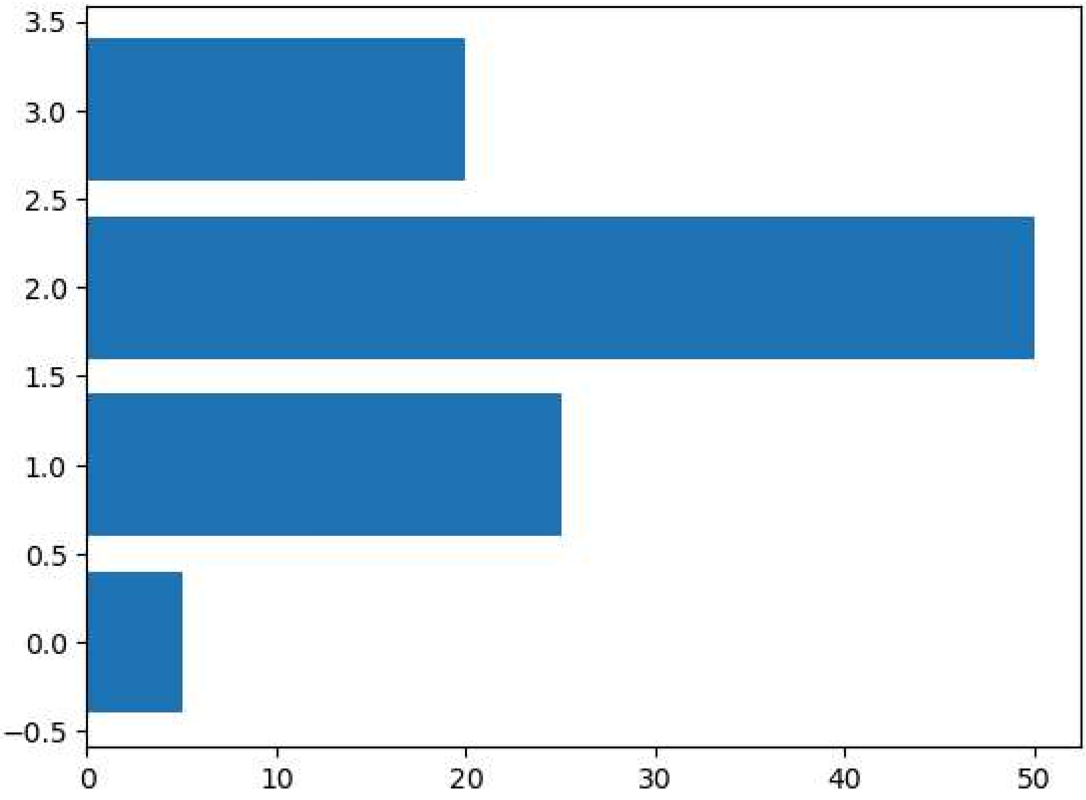
Bar Chart

```
In [93]: data2 = [5., 25., 50., 20.]
         plt.bar(range(len(data2)),data2)
         plt.show()
```



Horizontal Bar Chart

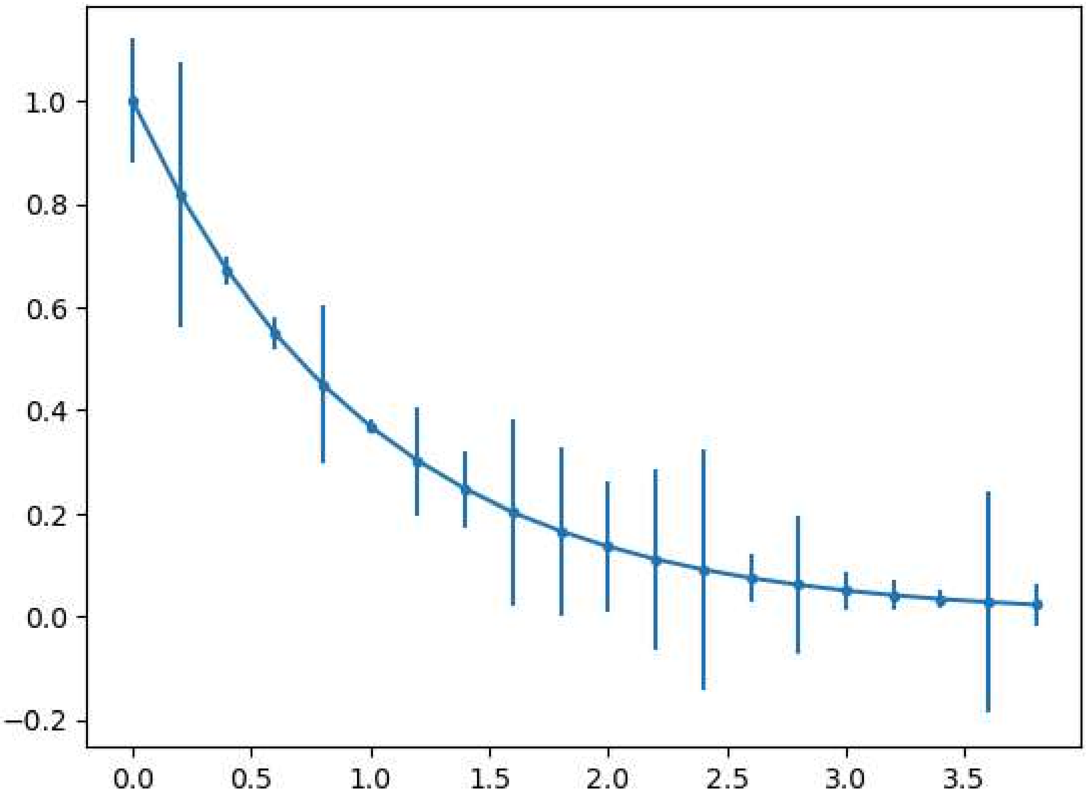
```
In [94]: data2 = [5., 25., 50., 20.]
         plt.barh(range(len(data2)),data2)
         plt.show()
```



Error Bar Chart

In [97]:

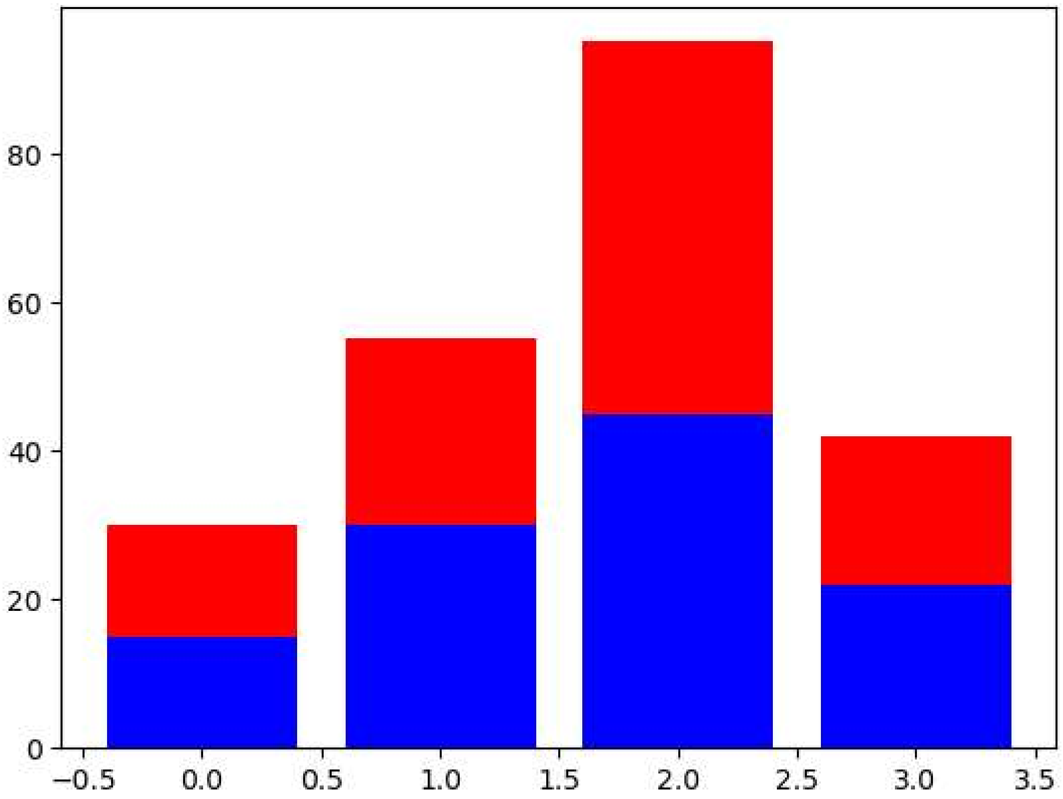
```
x9 = np.arange(0,4,0.2)

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 [101…

```python
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()
```

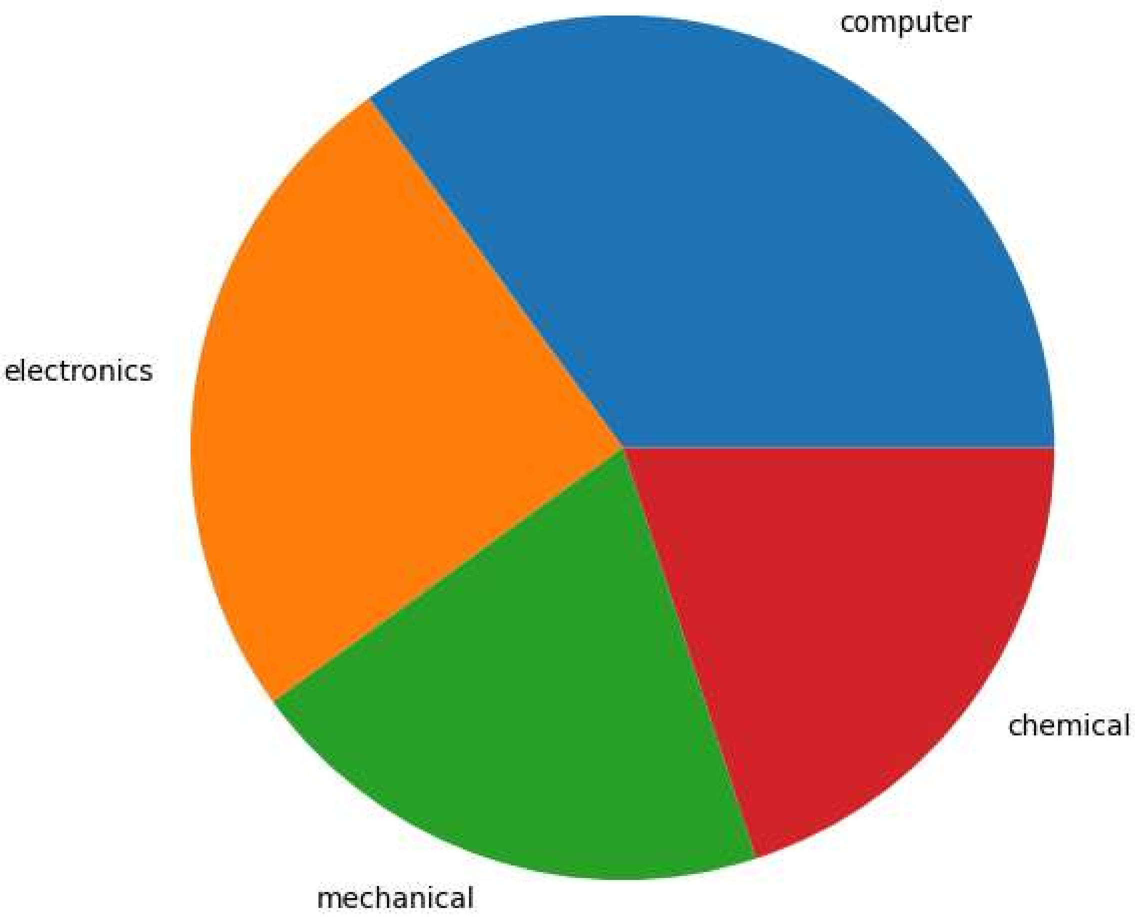


Pie Chart

In [104…

```
plt.figure(figsize=(7,7))
x10=[35,25,20,20]

labels=['computer','electronics','mechanical','chemical']

plt.pie(x10, labels=labels);
plt.show()
```

```
<Figure size 700x700 with 0 Axes>
<Figure size 700x700 with 0 Axes>
```



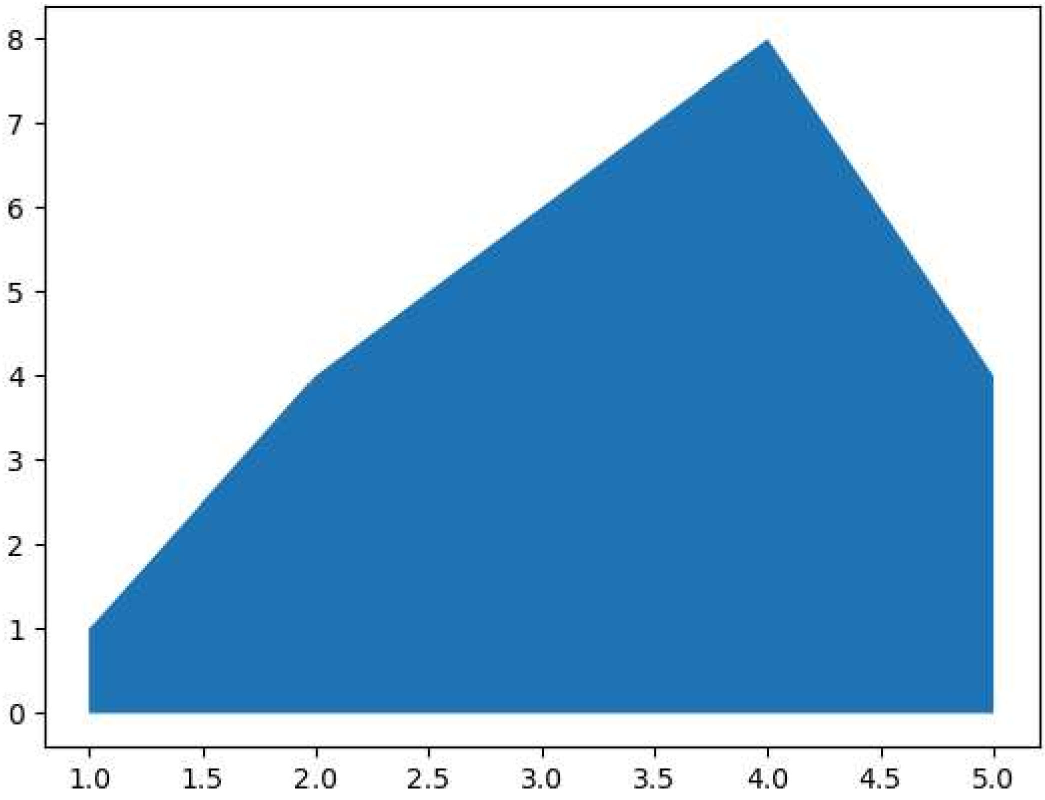
Boxplot

data3 = np.random.randn(100) plt.boxplot(data3) plt.show();

Area Chart

In [108…

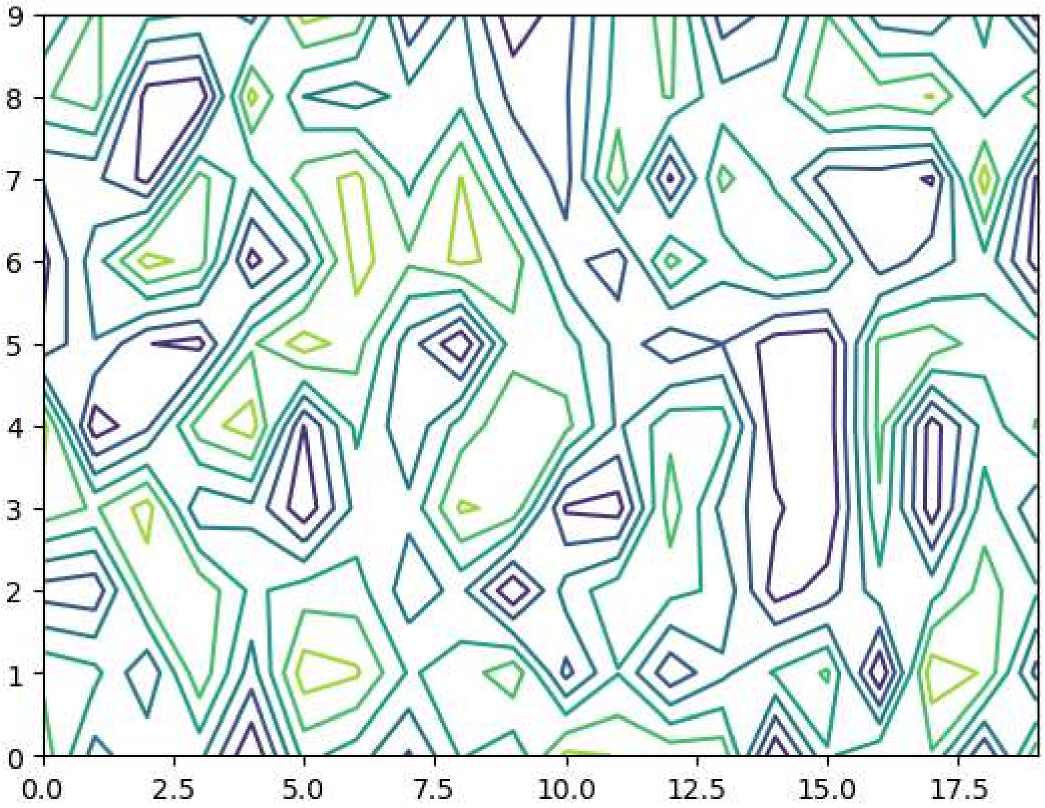
```
# Create some data
x12 = range(1,6)

y12 = [1,4,6,8,4]

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



Contour Plot

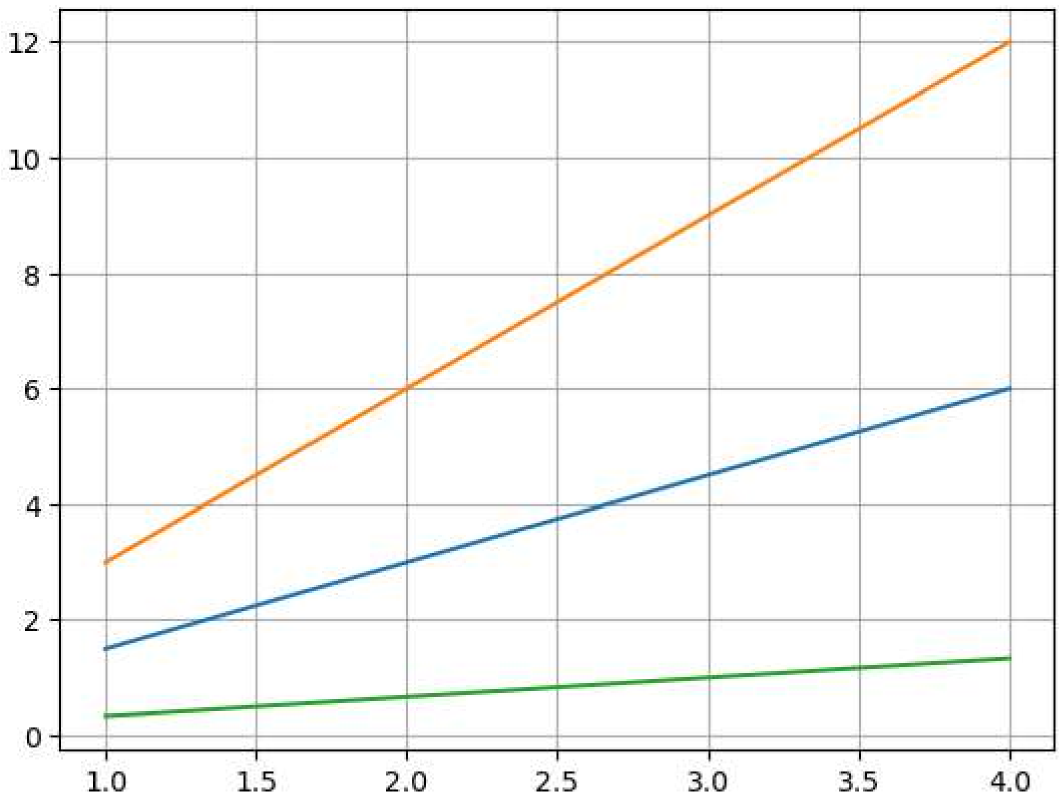
In [109…

```
# Create a matrix
matrix1 = np.random.rand(10, 20)

cp = plt.contour(matrix1)

plt.show()
```



Adding a grid

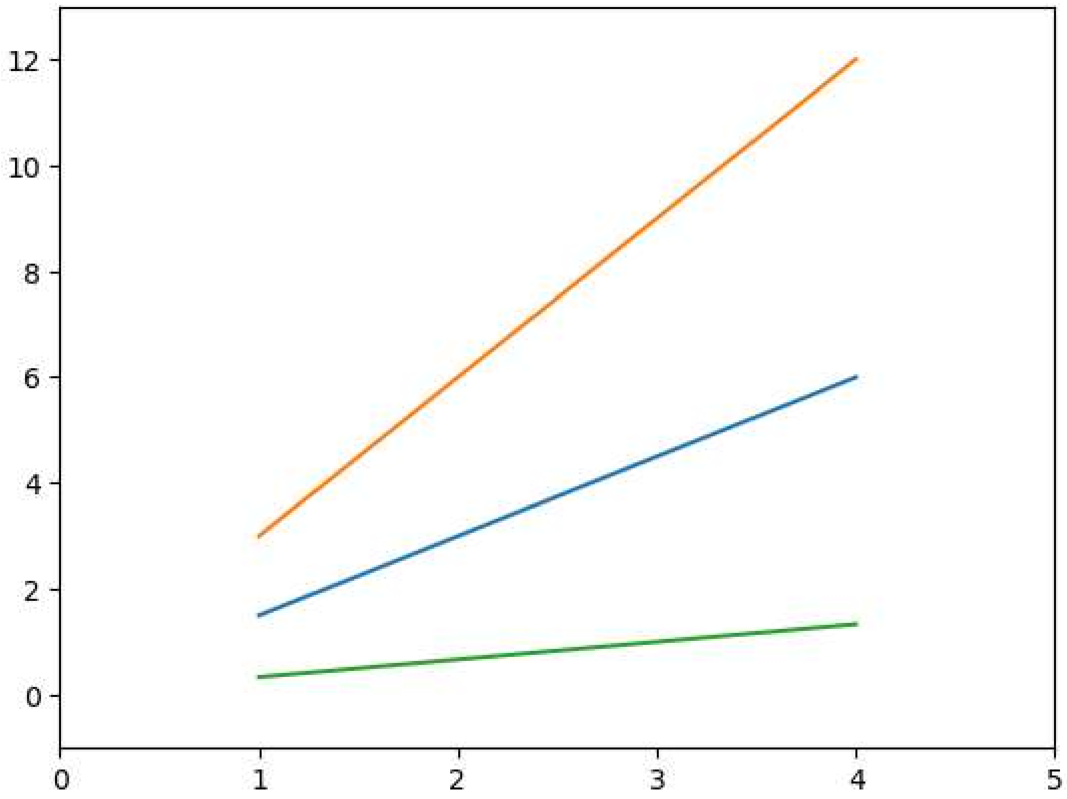
```
In [115…
x15 = np.arange(1, 5)
plt.plot(x15, x15*1.5,x15, x15*3.0, x15, x15/3.0)
plt.grid(True)
plt.show()
```



Handling axes

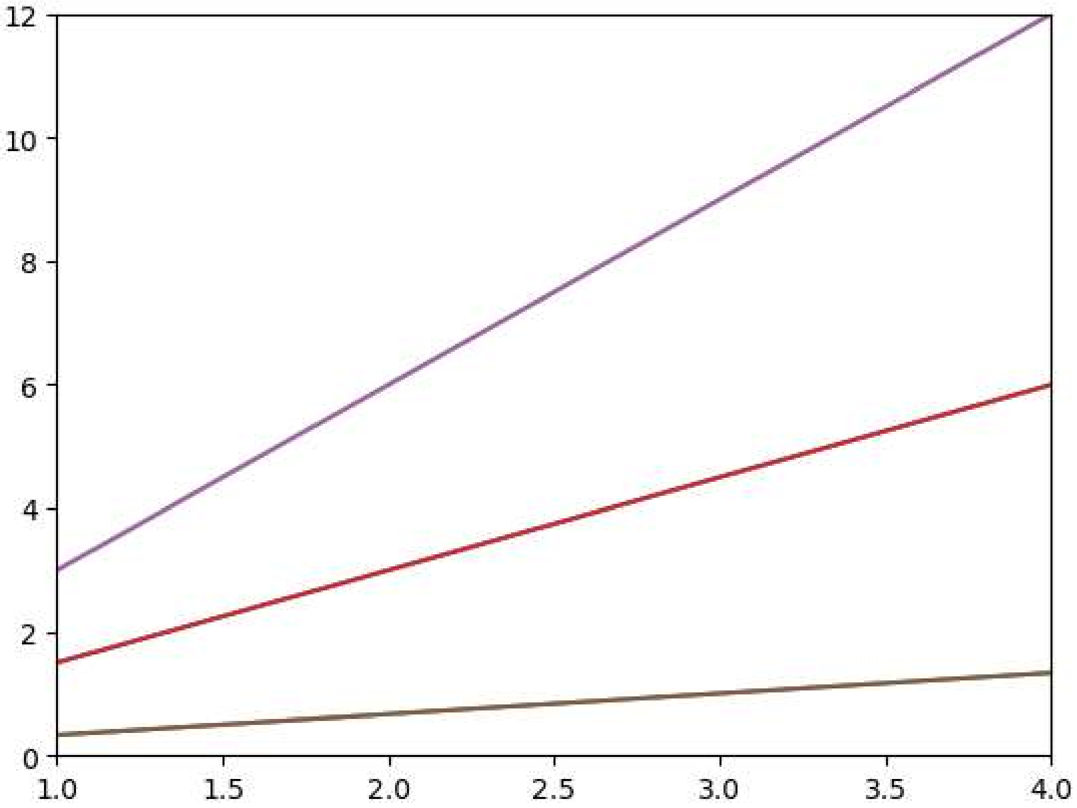
In [116…

```
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 [119…

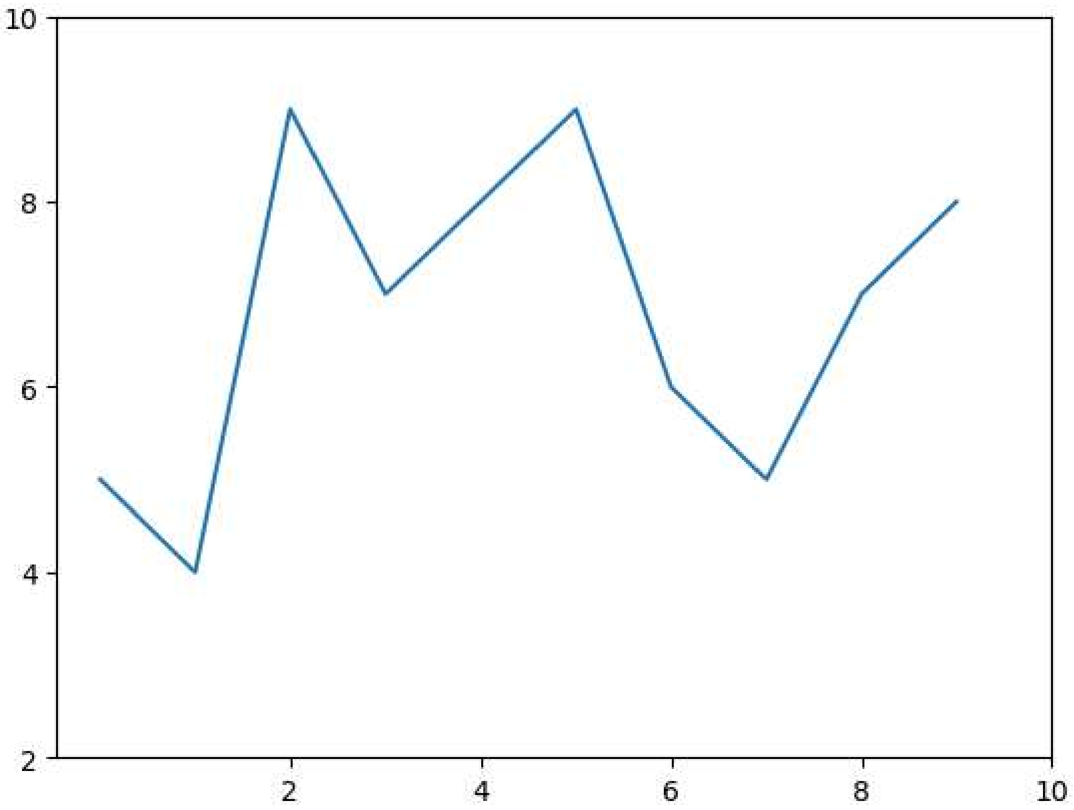
```
x15 = np.arange(1, 5)

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 [120…

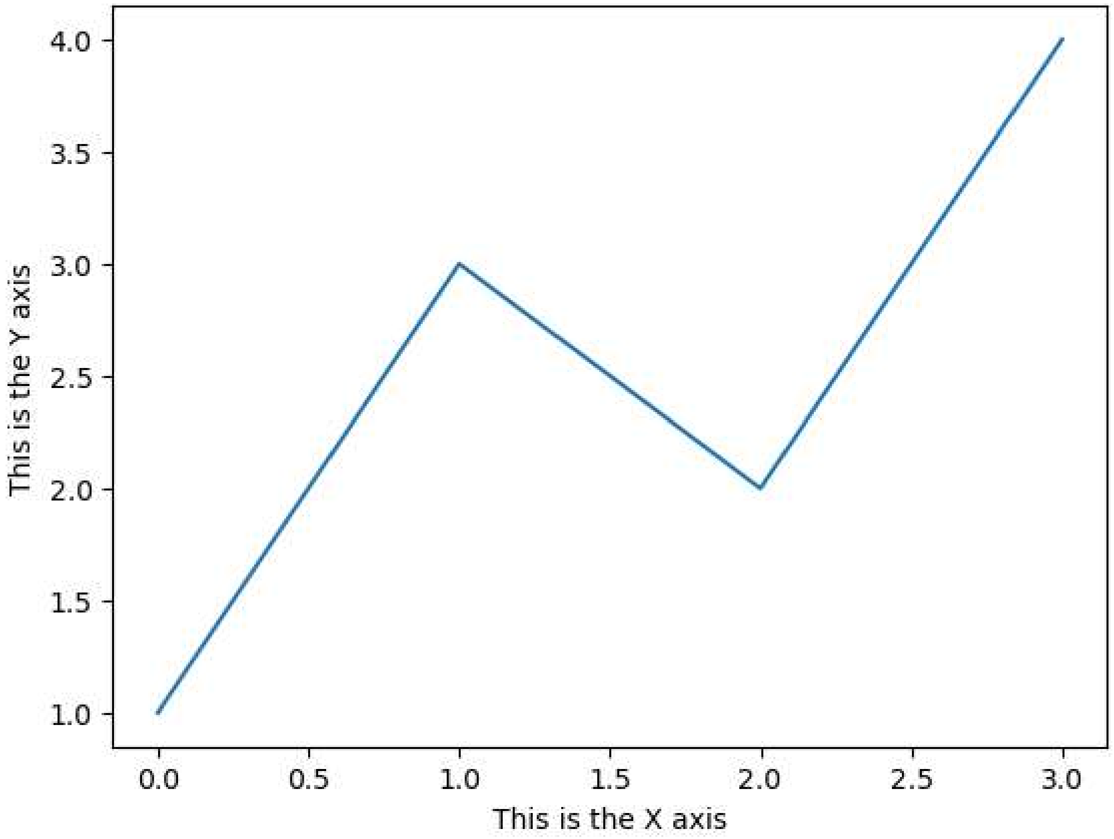
```
u = [5,4,9,7,8,9,6,5,7,8]
plt.plot(u)

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



Adding labels

In [121…

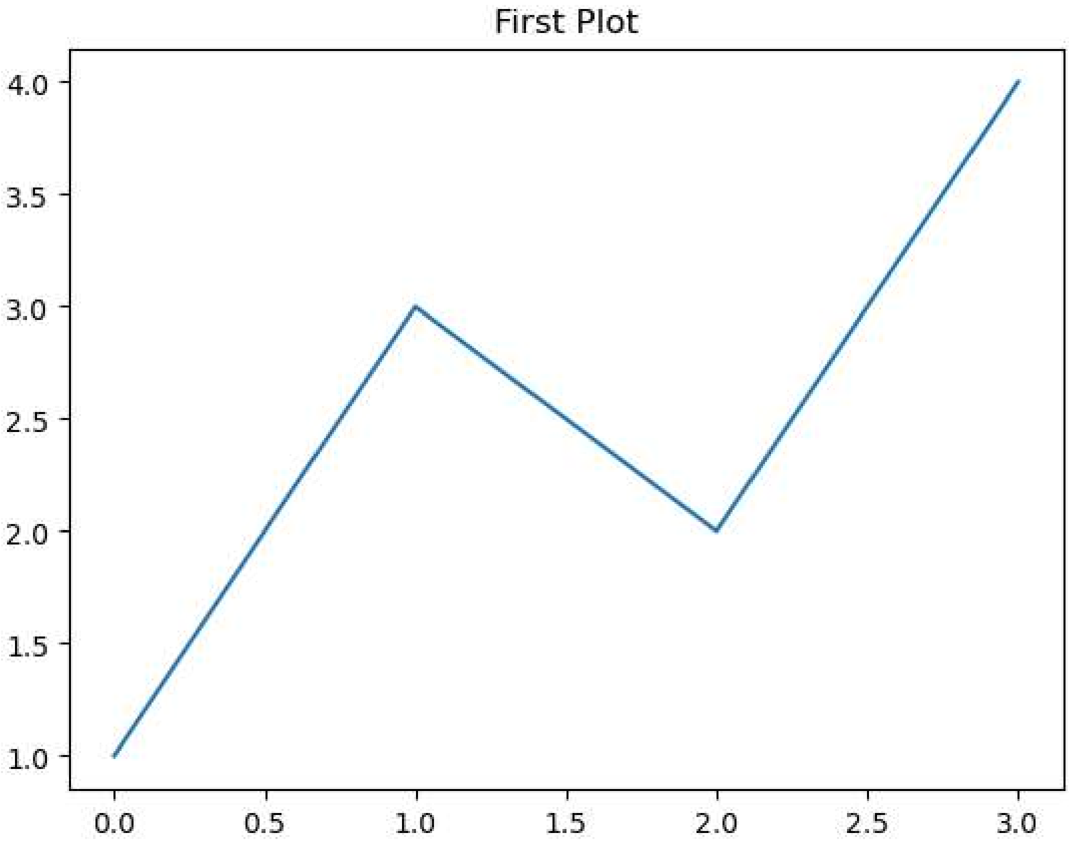
```
plt.plot([1, 3, 2, 4])

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

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

plt.show()
```



Adding a title

In [122…

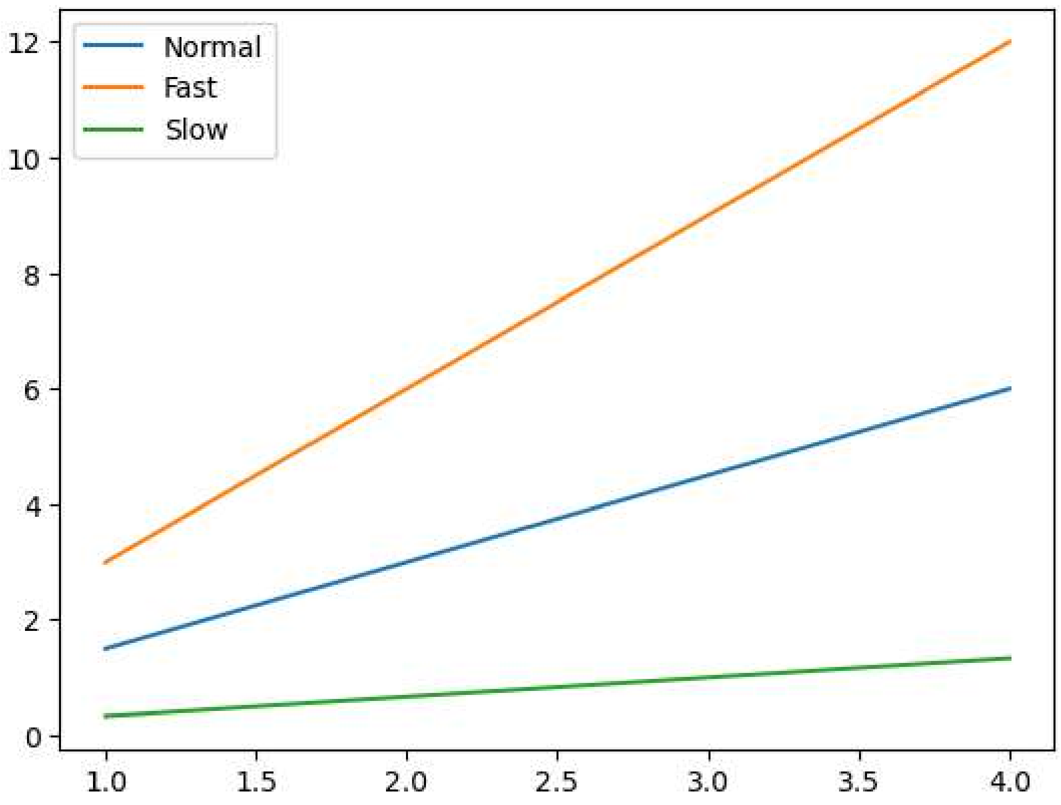
```
plt.plot([1, 3, 2, 4])

plt.title('First Plot')

plt.show()
```



Adding a legend

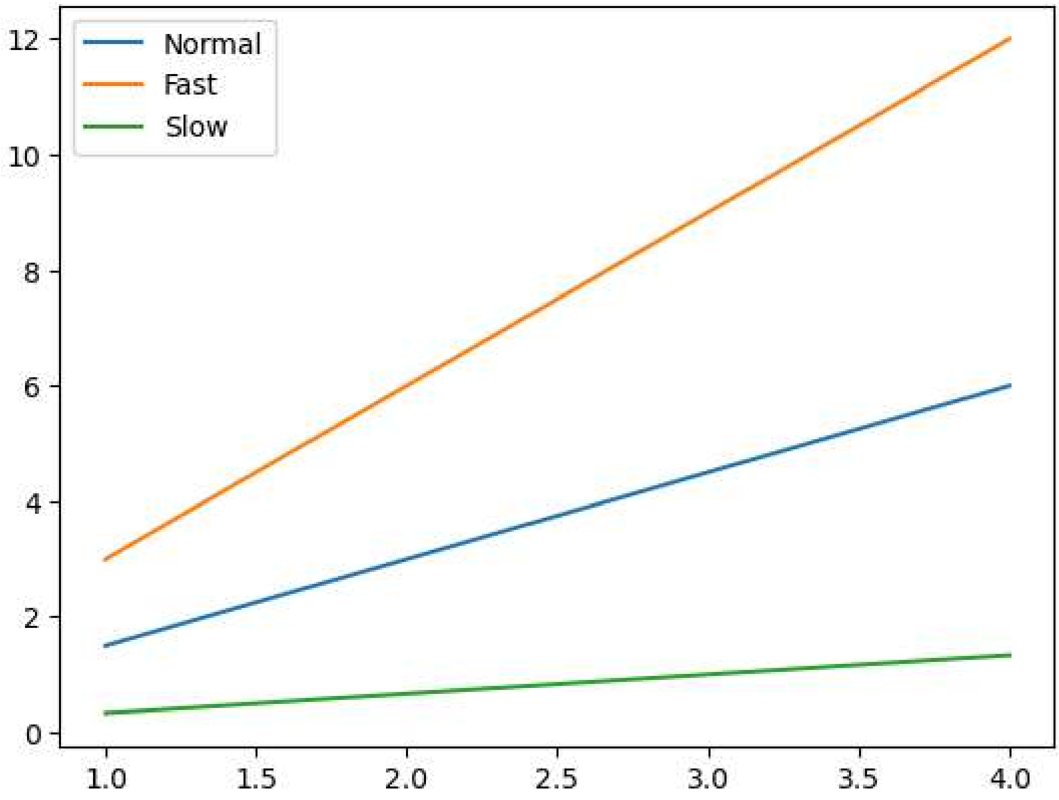
In [126…

```
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 [129…

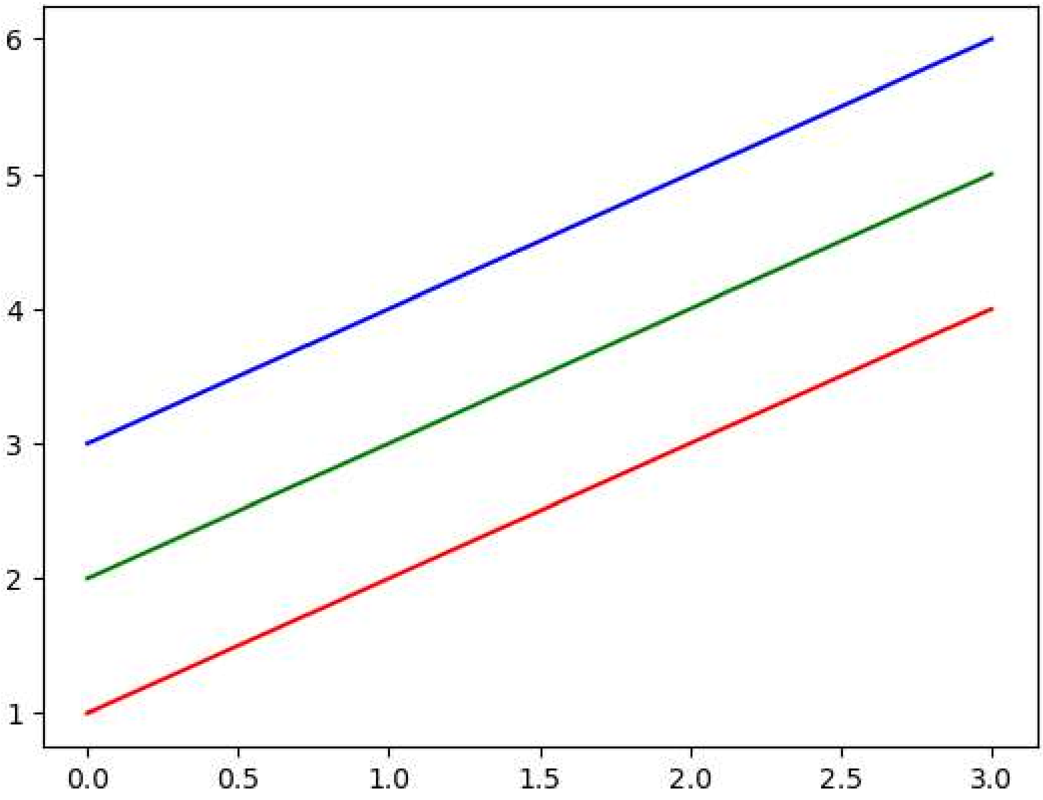
```
x15 = np.arange(1, 5)

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 [130…

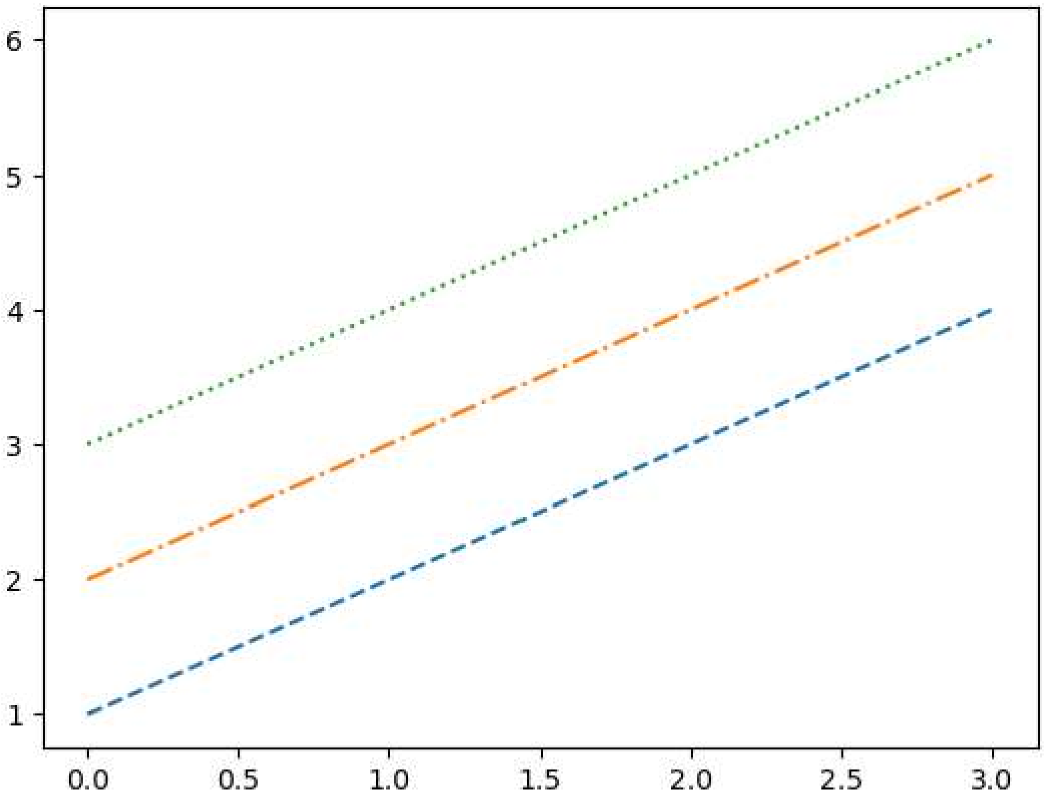
```
x16 = np.arange(1, 5)

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

plt.show()
```



Control line styles

In [131…

```
x16 = np.arange(1, 5)

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

plt.show()
```


6
5
4
3
2
1
0.0
0.5
1.0
1.5
2.0
2.5
3.0