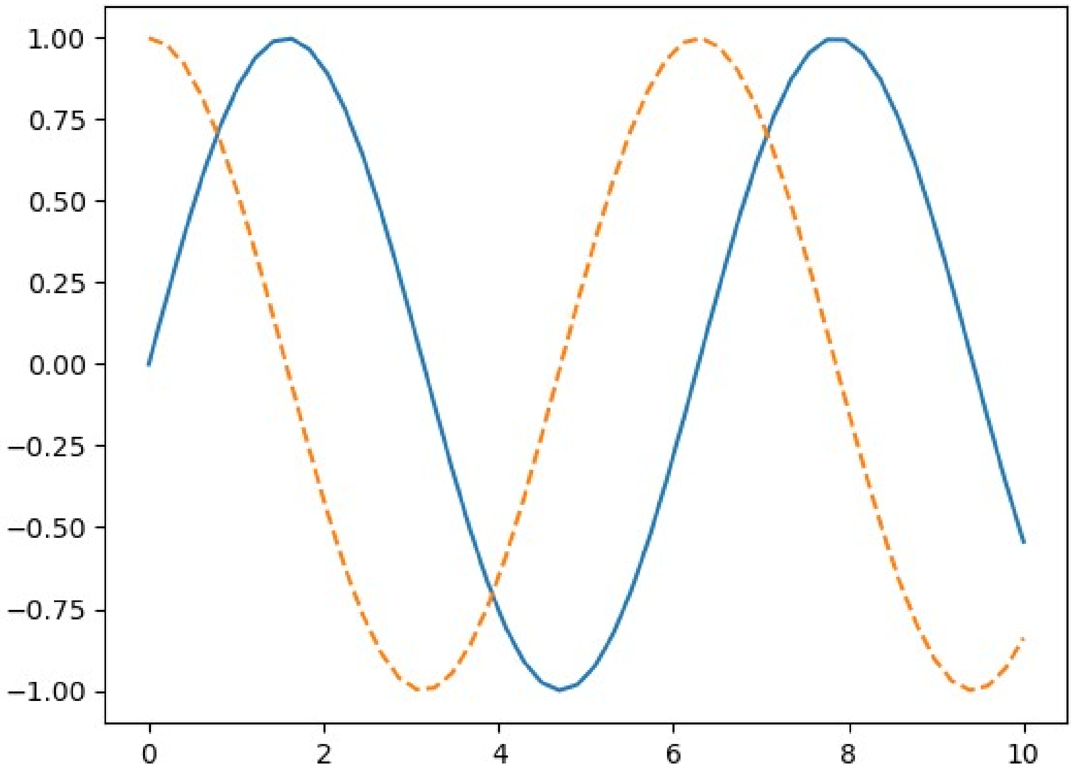
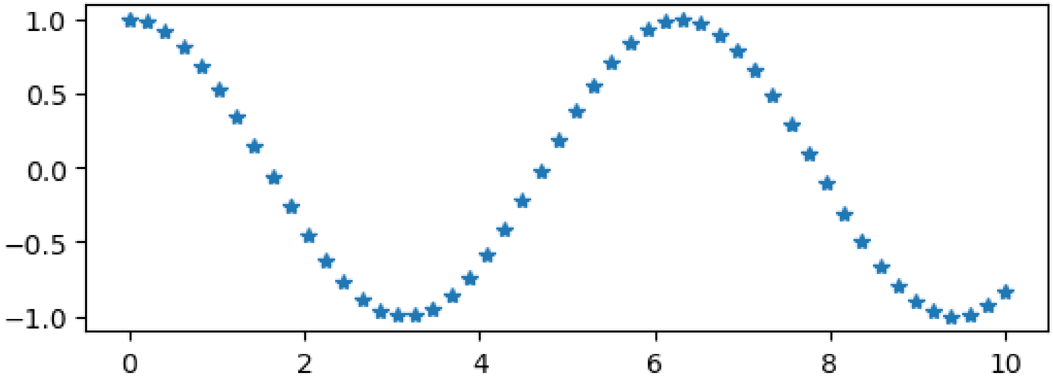
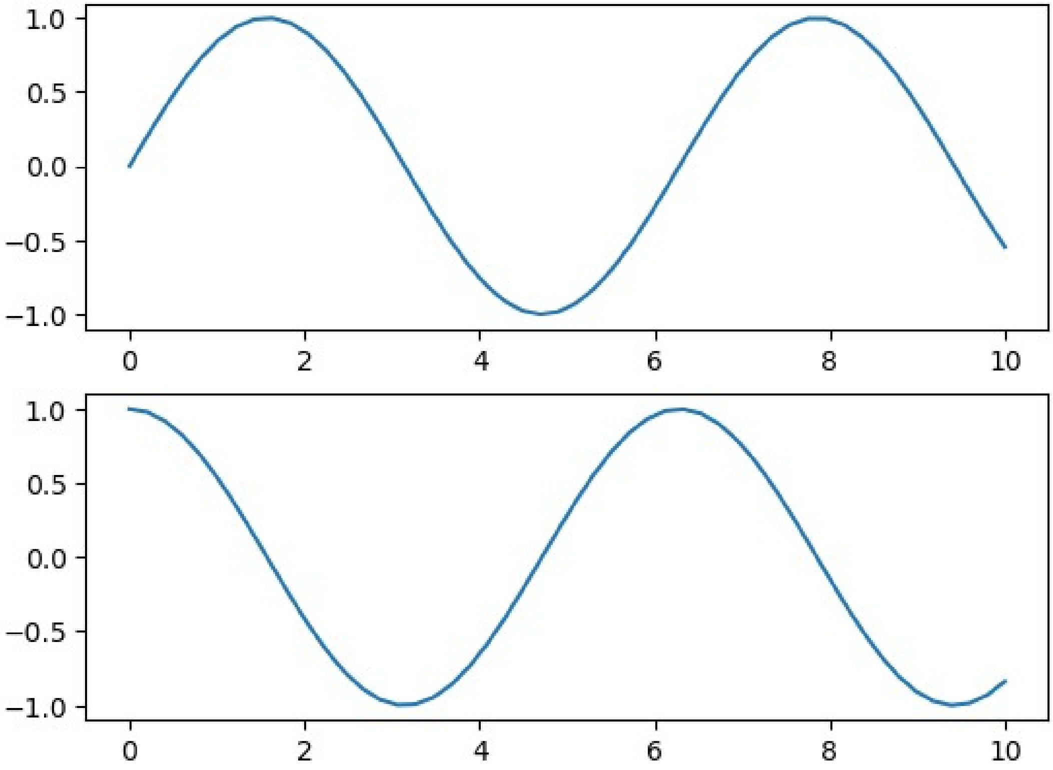
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

%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()
```



```
#create the first of two panels and set current axis
plt.subplot(2,1,1)
plt.plot(x1,np.cos(x1), '*')
plt.show()
```



```
#create a plot figure
plt.figure()

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

#create the first of two panels and set current axis
plt.subplot(2,1,1)
plt.plot(x1,np.sin(x1))
plt.show()
```



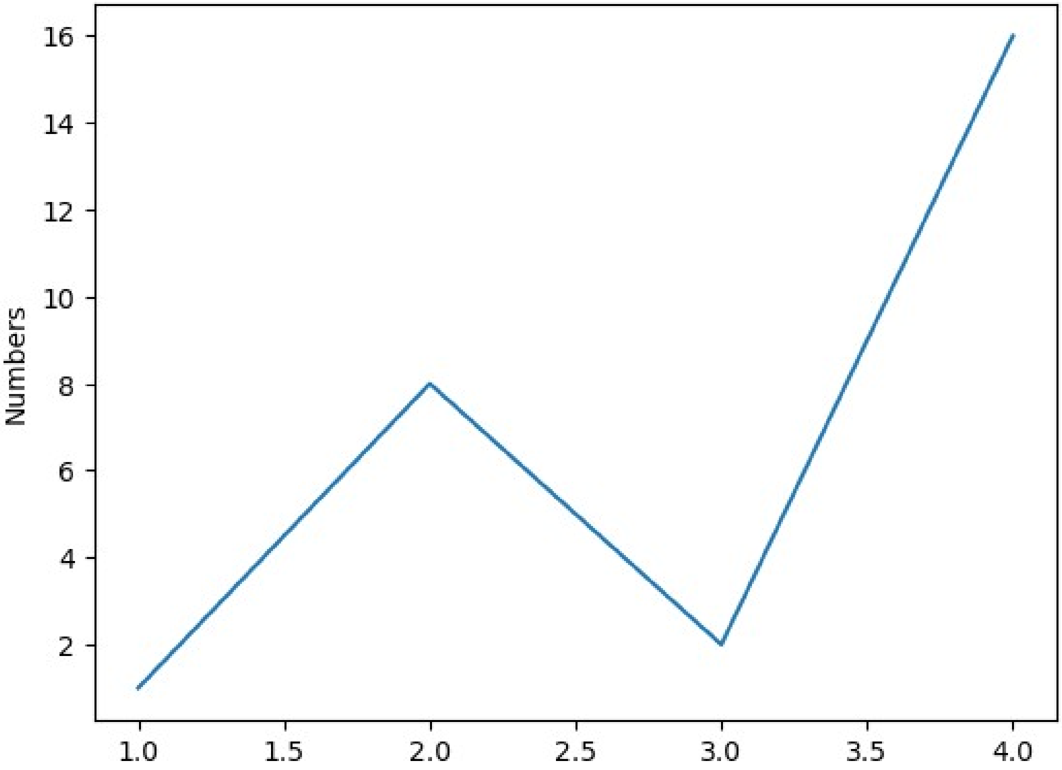
```
# get current figure information
print(plt.gcf())
```

```
Figure(640x480)
```

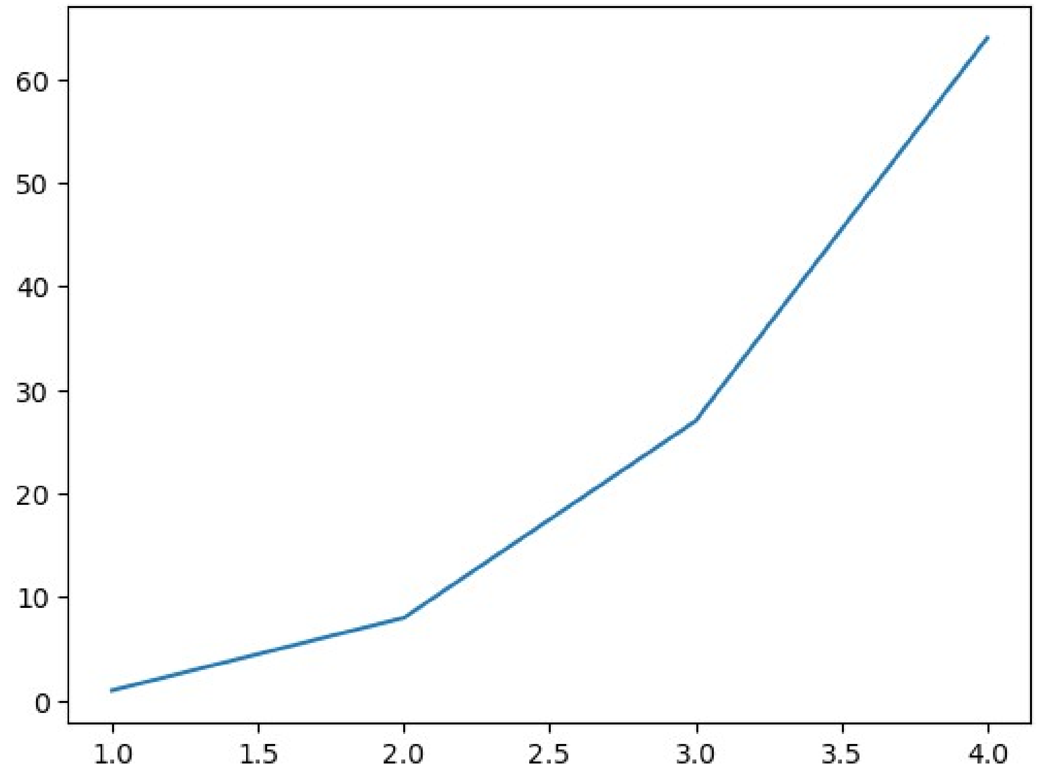
```
# get current figure information
print(plt.gca())
```

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

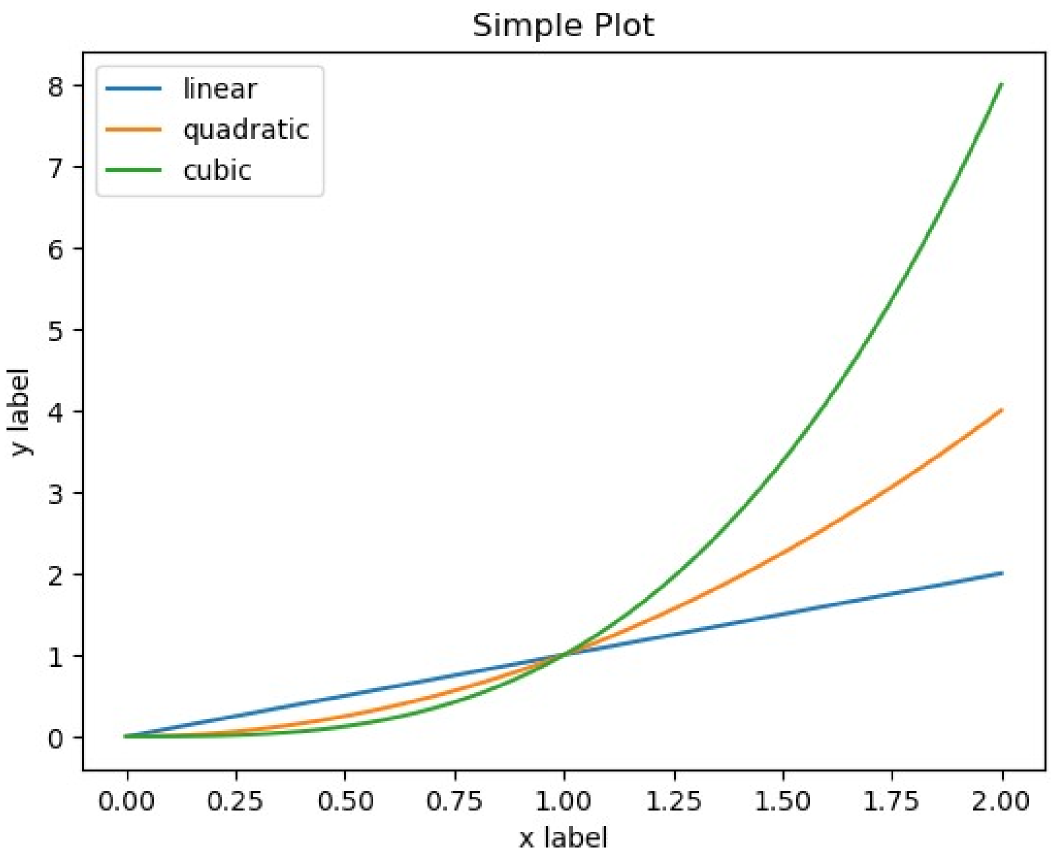
```
plt.plot([1,2,3,4], [1,8,2,16])
plt.ylabel('Numbers')
plt.show()
```



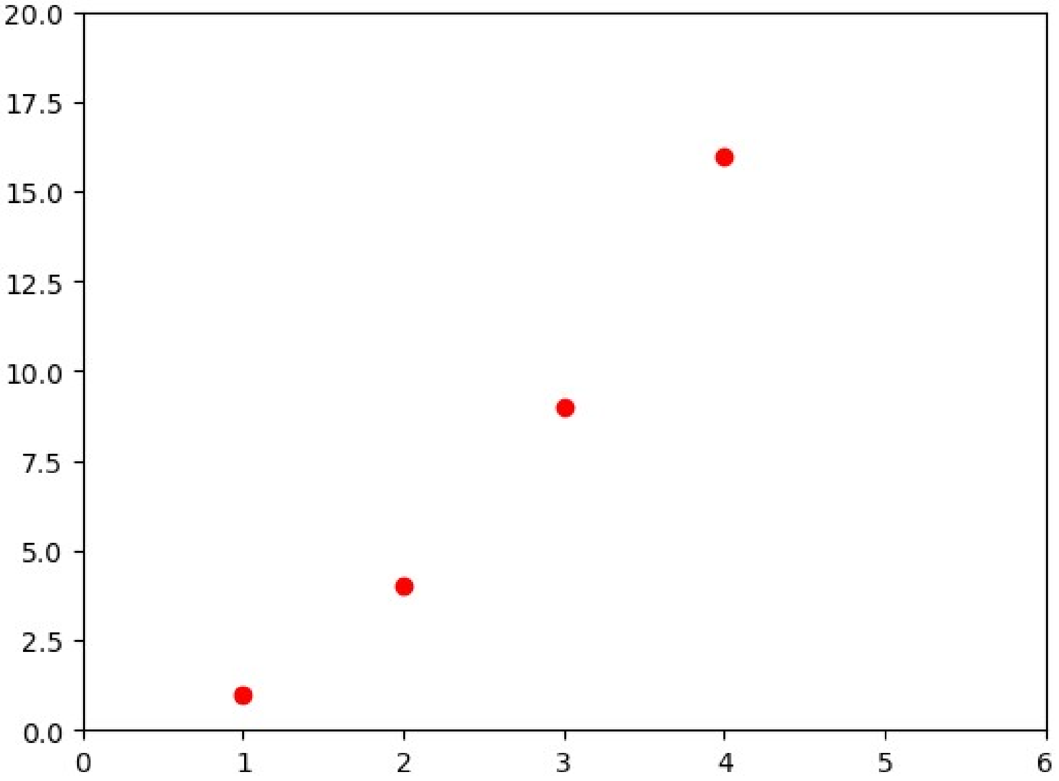
```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4],[1,8,27,64])
plt.show()
```



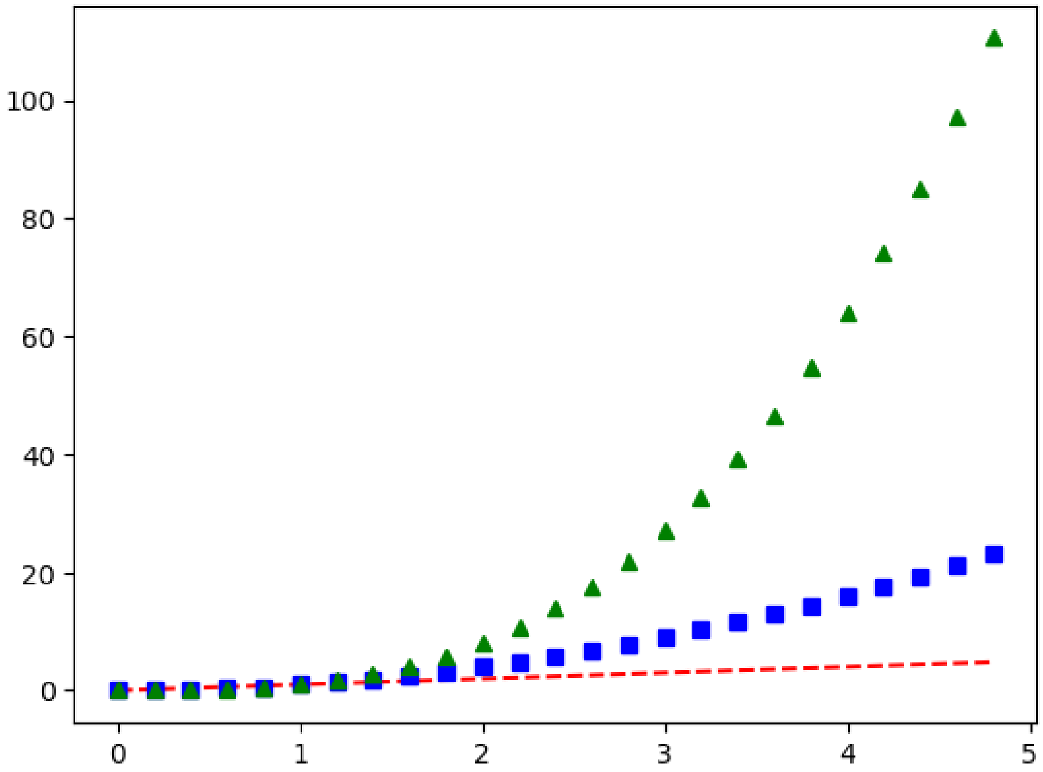
```
x = np.linspace(0, 2, 100)
plt.plot(x, x, label='linear')
plt.plot(x, x**2, label='quadratic')
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
plt.show()
```



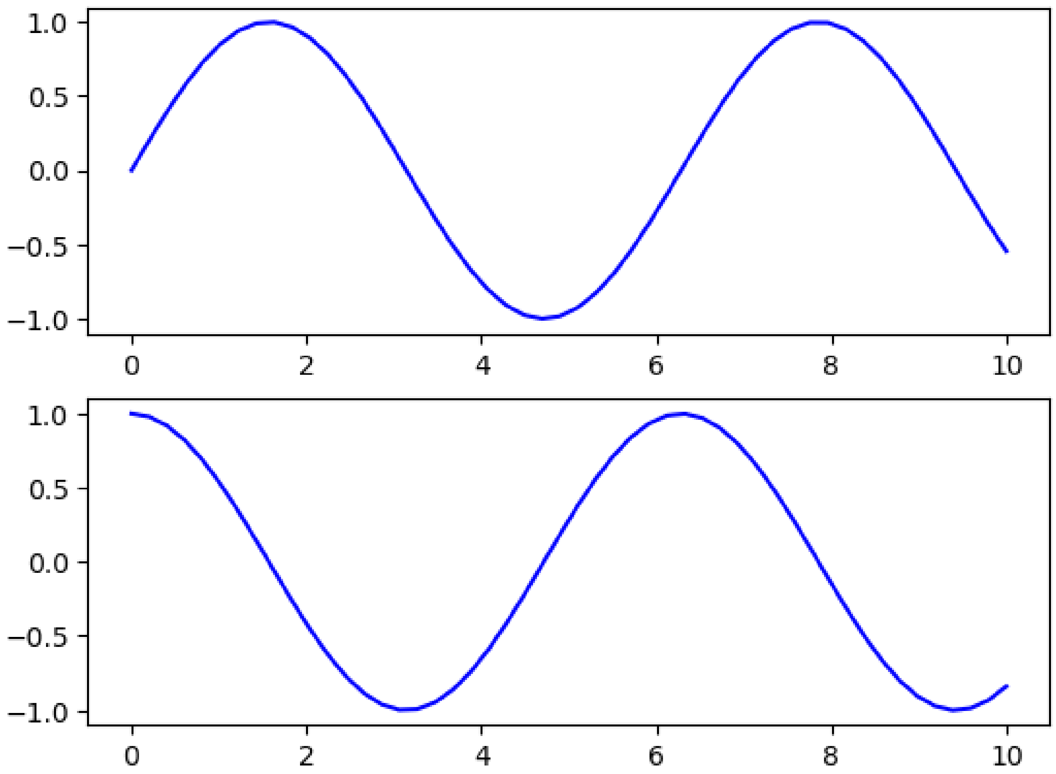
```
plt.plot([1,2,3,4], [1,4,9,16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```



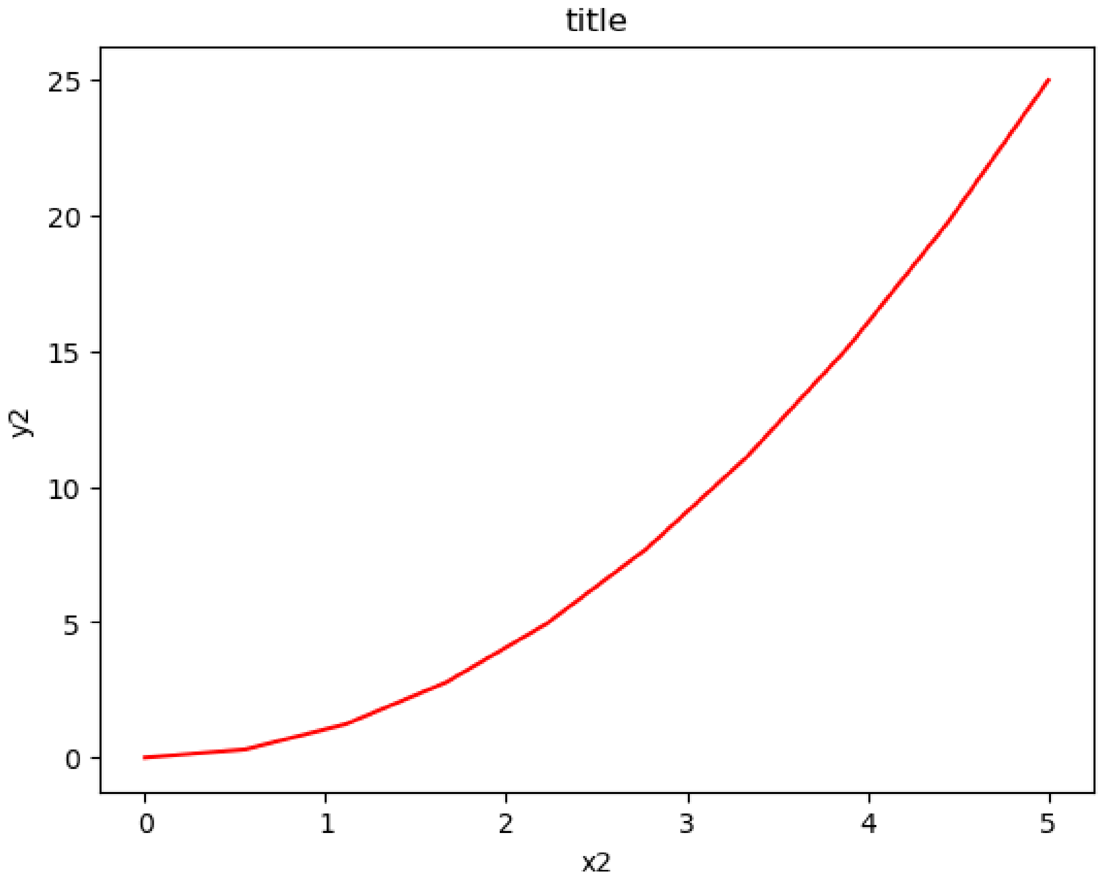
```
# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

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



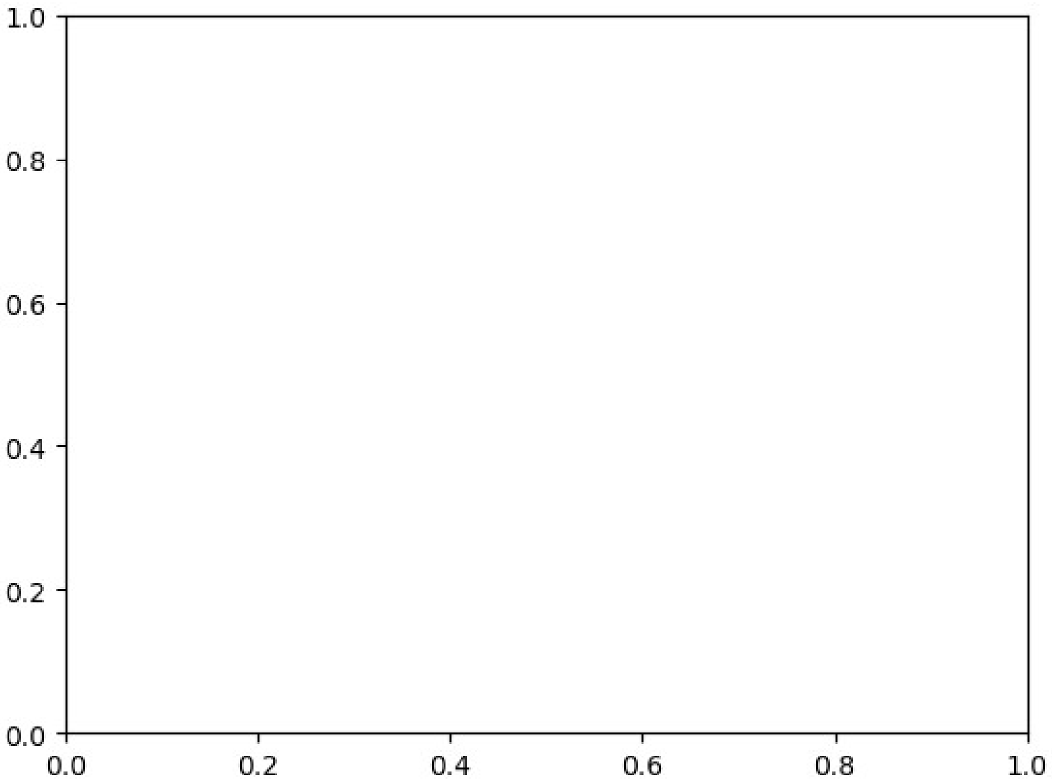
```
# First create a grid of plots
# ax will be an array of two Axes objects
fig, ax = plt.subplots(2)

#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()
```



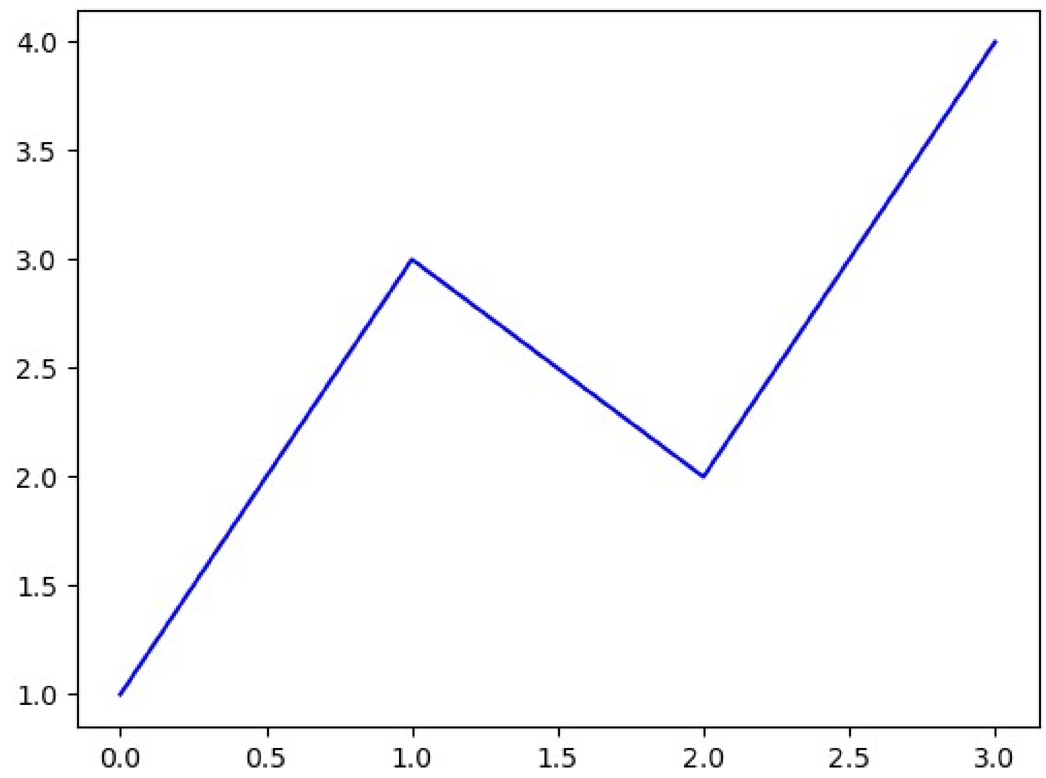
```
fig = plt.figure()
x2 = np.linspace(0,5,10)
y2= x2**2
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axes.plot(x2, y2, 'r')
axes.set_xlabel('x2')
axes.set_ylabel('y2')
axes.set_title('title');
plt.show()
```



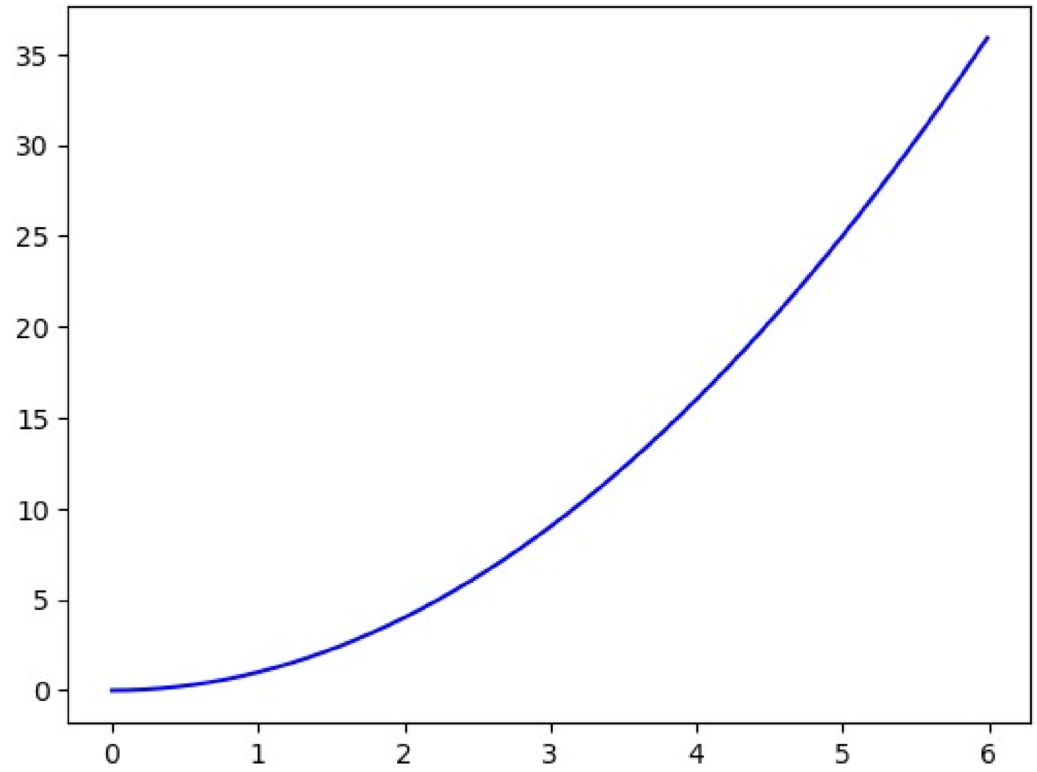
```
fig = plt.figure()

ax = plt.axes()
plt.show()
```



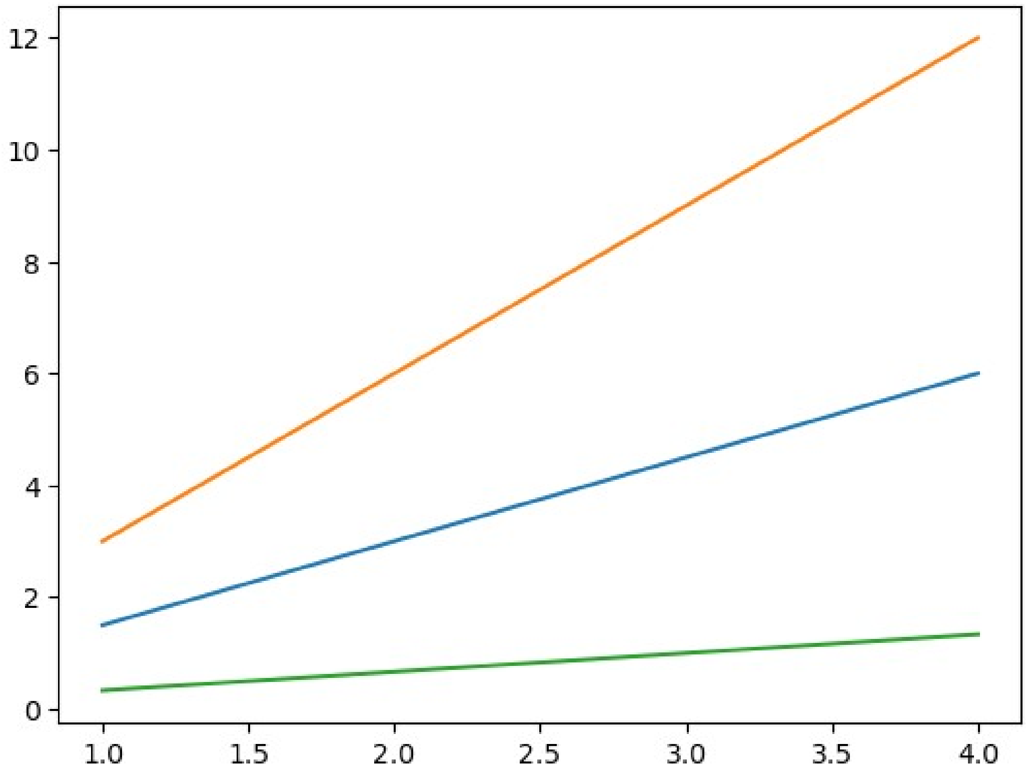
```
plt.plot([1, 3, 2, 4], 'b-')
plt.show()
```



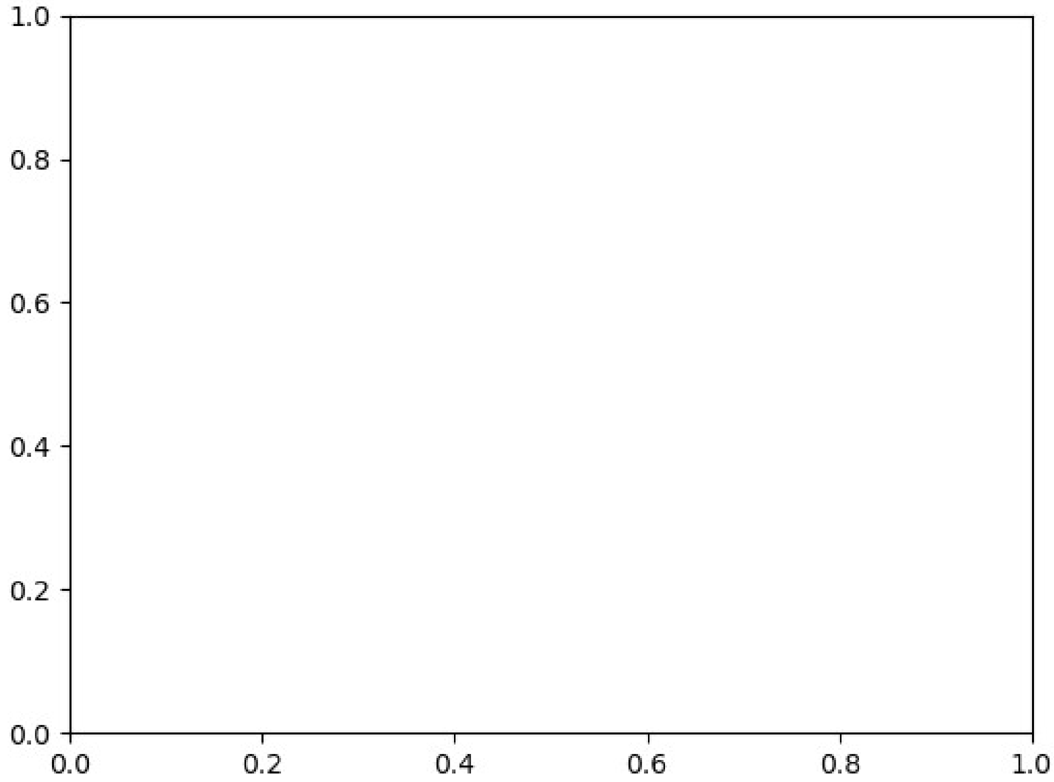
```
x3 = np.arange(0.0, 6.0, 0.01)

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

plt.show()
```



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

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

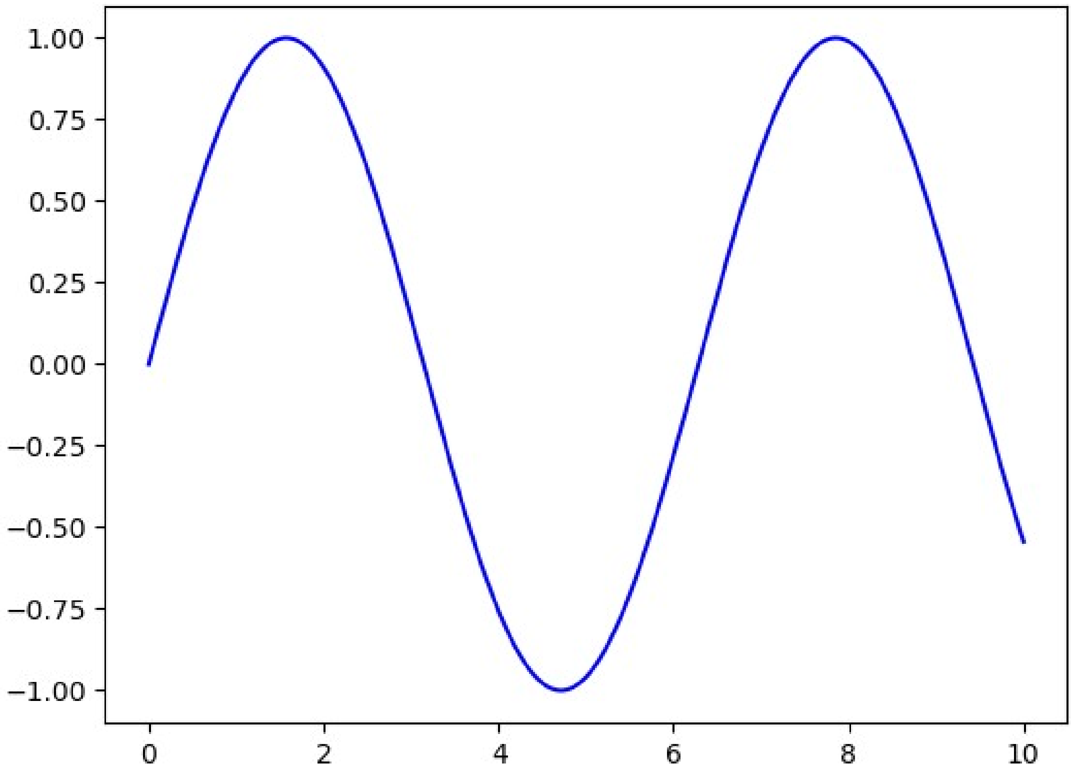


```
# Explore supported file formats

fig.canvas.get_supported_filetypes()
```

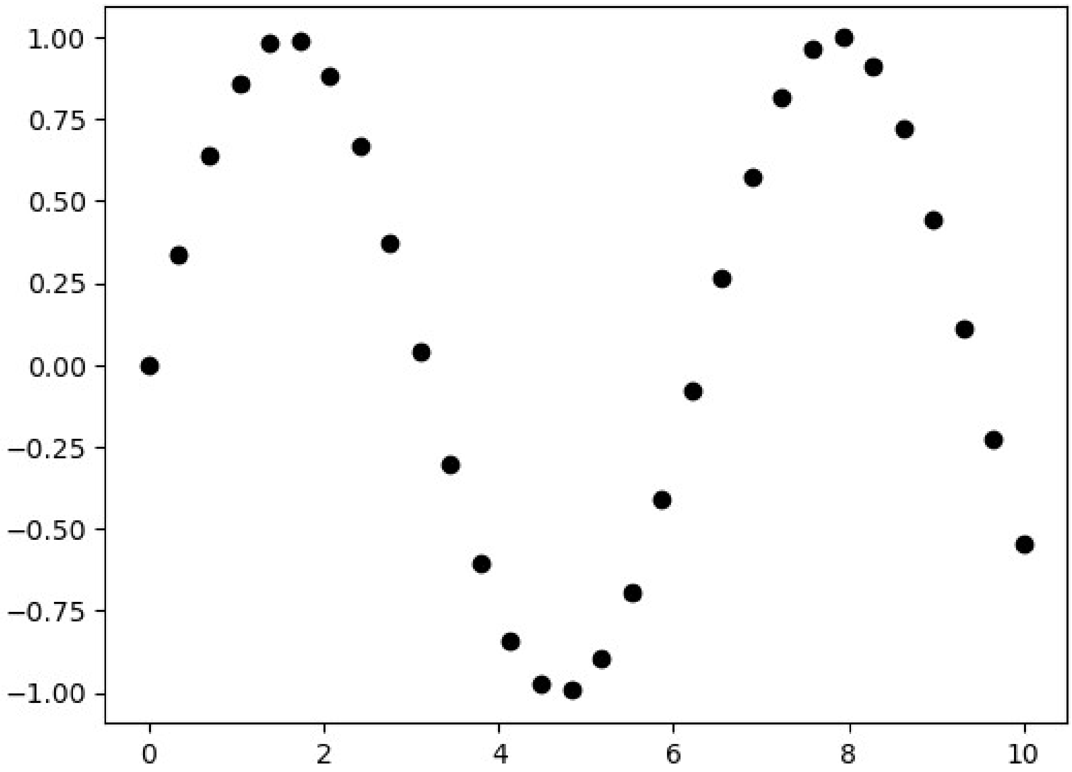
```
{'eps': 'Encapsulated Postscript',
 'jpg': 'Joint Photographic Experts Group',
 'jpeg': 'Joint Photographic Experts Group',
 'pdf': 'Portable Document Format',
 'pgf': 'PGF code for LaTeX',
 'png': 'Portable Network Graphics',
 'ps': 'Postscript',
 'raw': 'Raw RGBA bitmap',
 'rgba': 'Raw RGBA bitmap',
 'svg': 'Scalable Vector Graphics',
 'svgz': 'Scalable Vector Graphics',
 'tif': 'Tagged Image File Format',
 'tiff': 'Tagged Image File Format',
 'webp': 'WebP Image Format'}
```

```
# Create figure and axes first
fig = plt.figure()
ax = plt.axes()
```

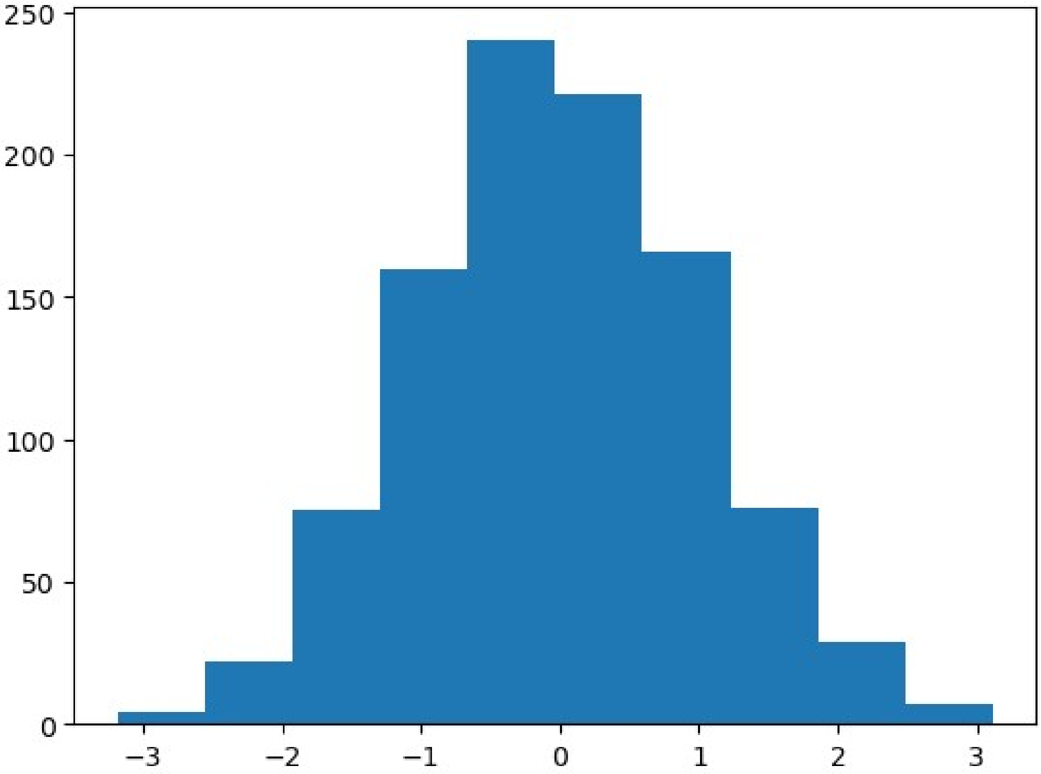
```
# Declare a variable x5
x5 = np.linspace(0, 10, 1000)
#plot the sinusoid functkon
ax.plot(x5, np.sin(x5), 'b-');
plt.show()
```



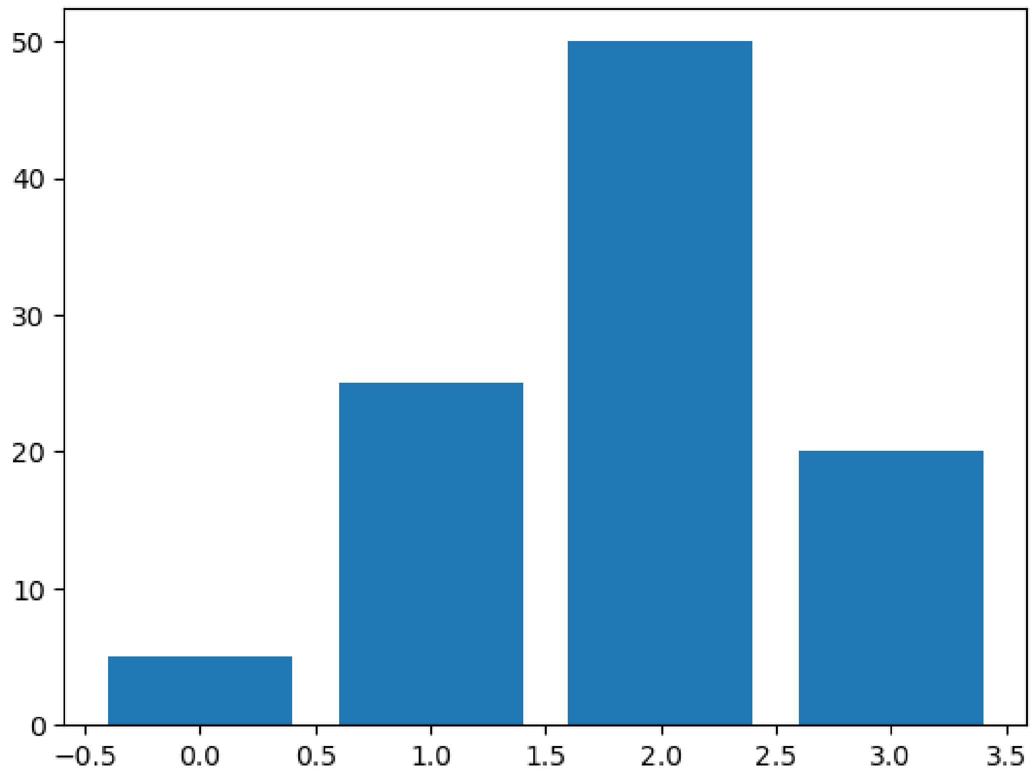
```
x7 = np.linspace(0, 10, 30)

y7 = np.sin(x7)

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



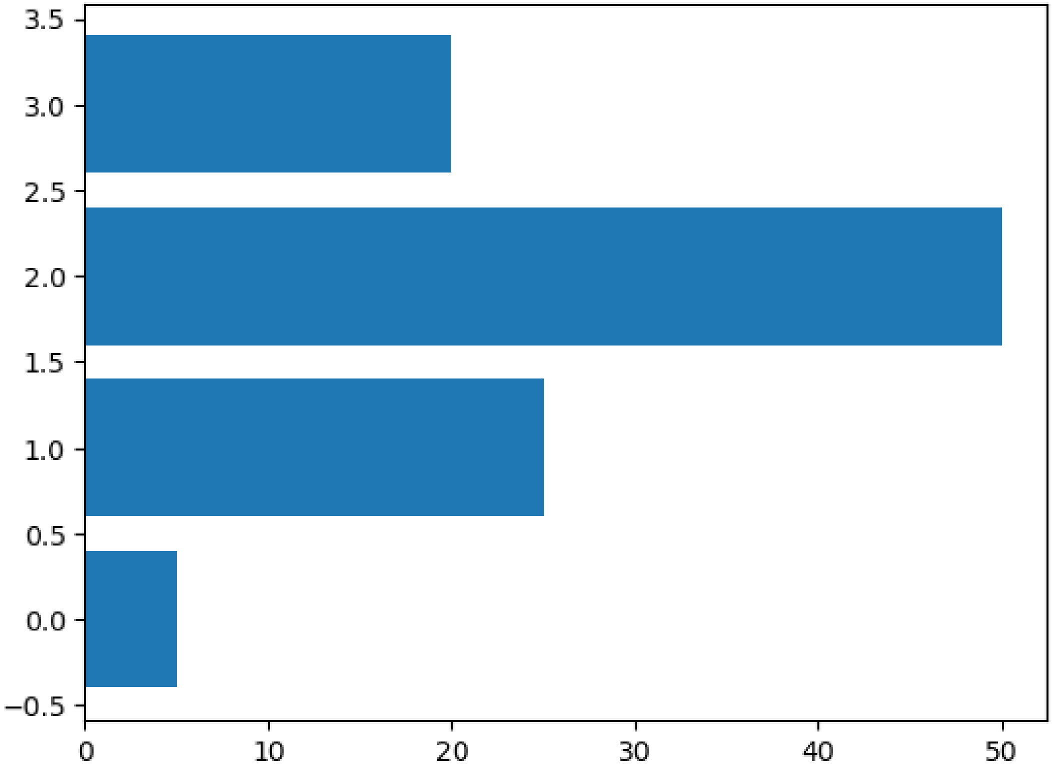
```
data1 = np.random.randn(1000)

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



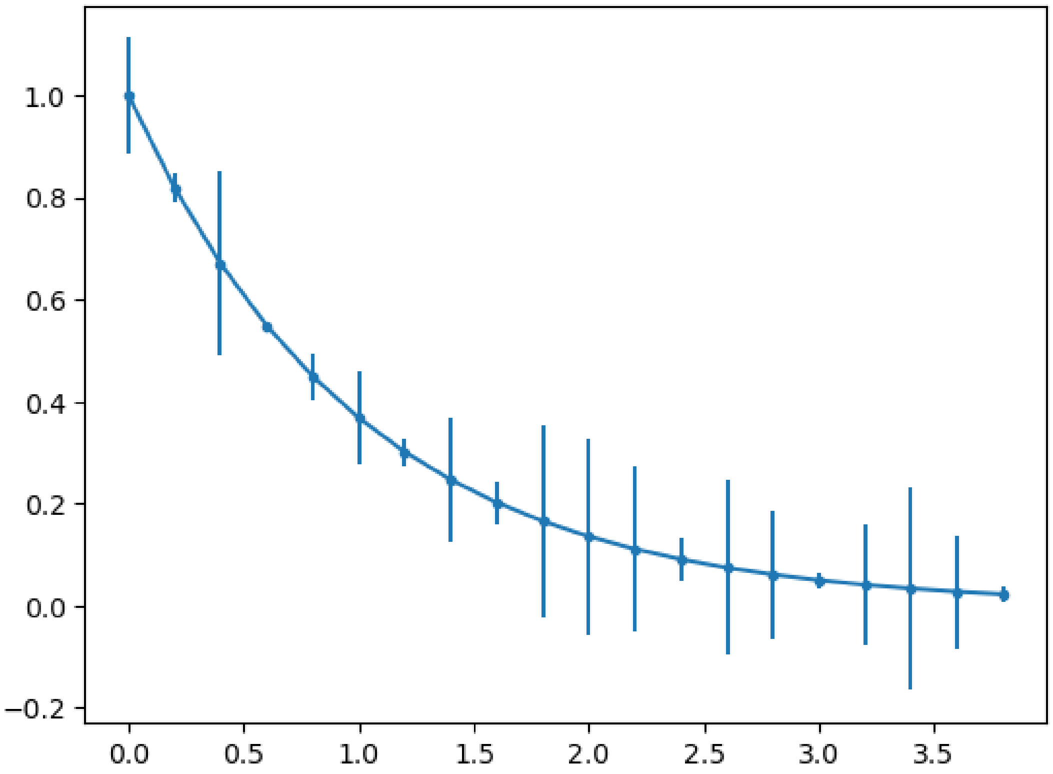
```
data2 = [5. ,25. ,50. , 20]
plt.bar(range(len(data2)),data2)
plt.show()
```



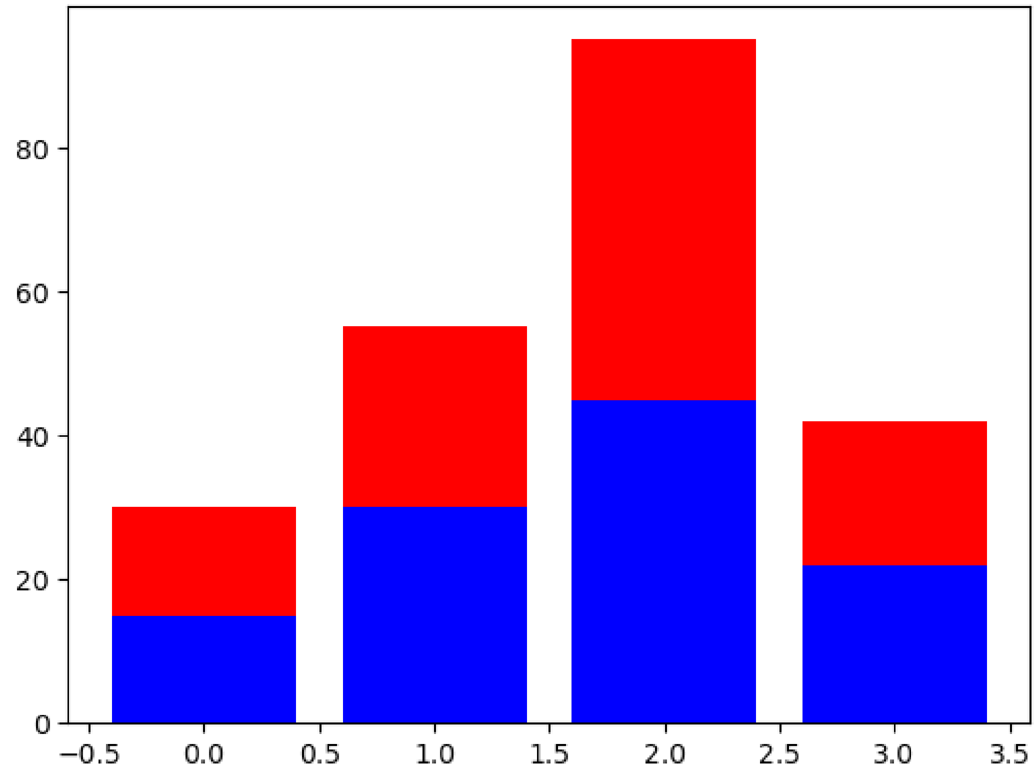
```
data2 = [5. , 25. ,50. , 20.]
plt.barh(range(len(data2)), data2)

plt.show()
```



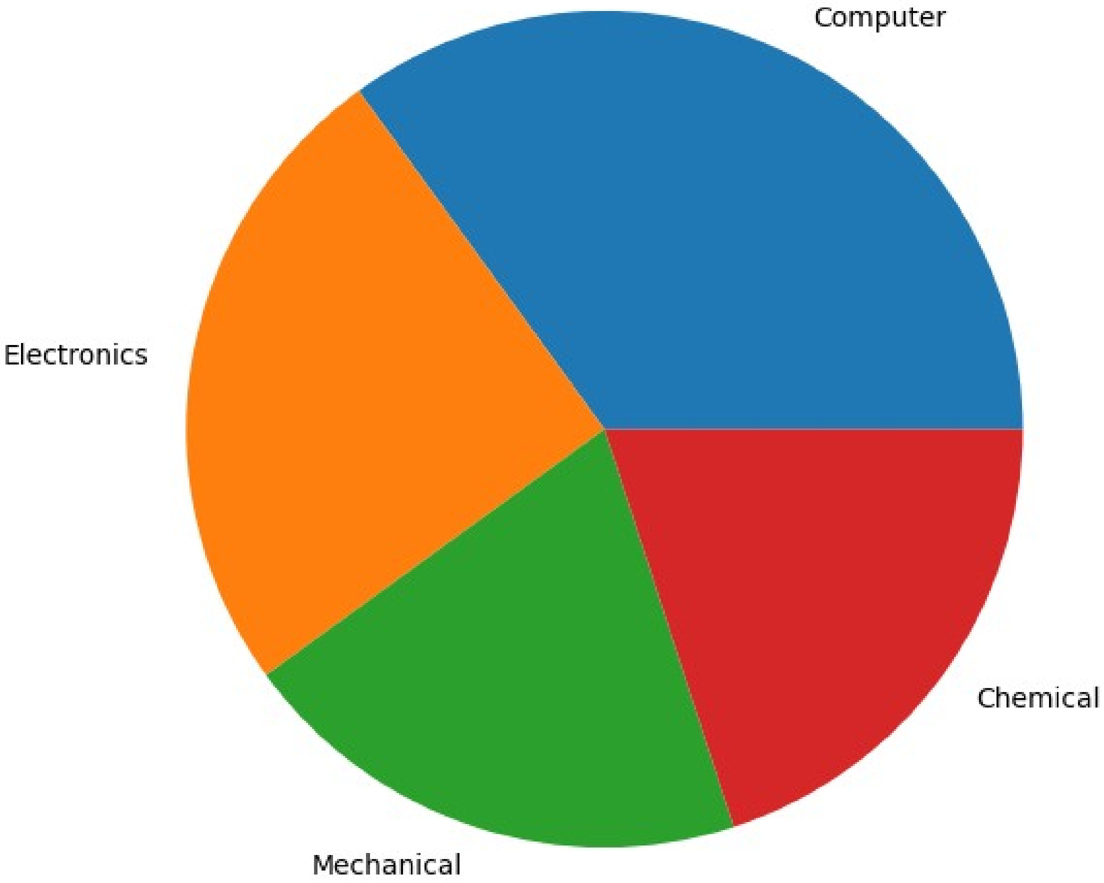
```
x9 = np.arange(0, 4, 0.2)
y9 = np.exp(-x9)
e1 = 0.1 * np.abs(np.random.randn(len(y9)))
plt.errorbar(x9,y9,yerr = e1, fmt = '.-')
plt.show();
```



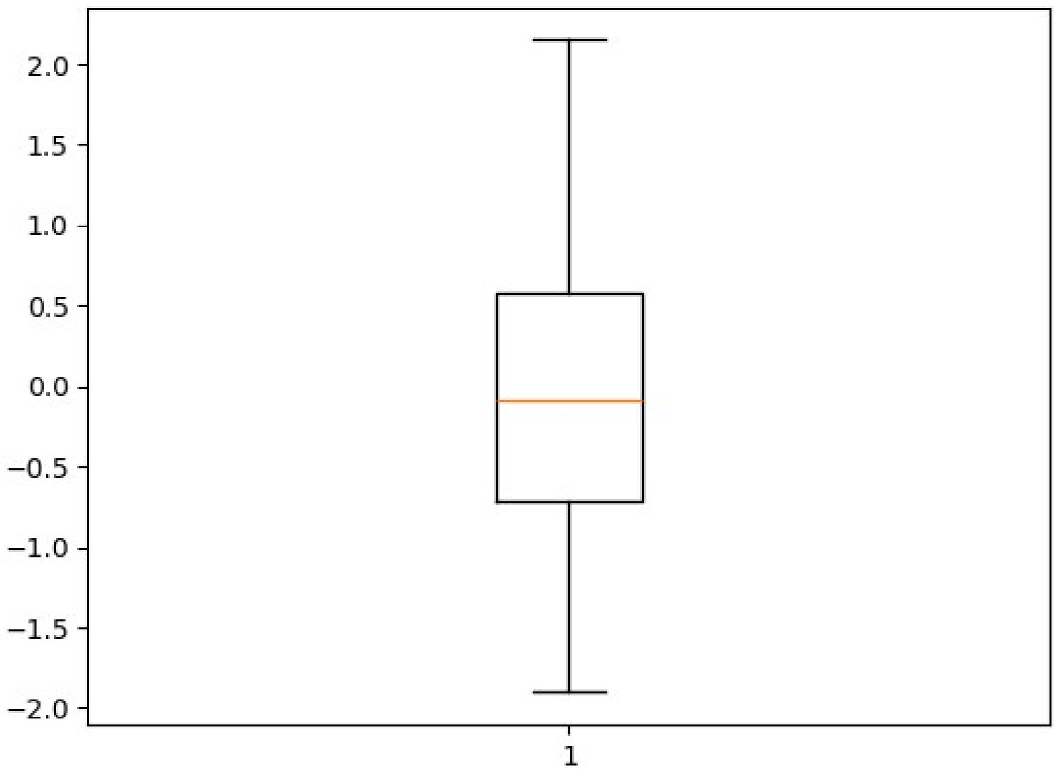
```
A = [15., 30., 45., 22.]
B = [15., 25., 50., 20.]
z2 = range(4)
plt.bar(z2, A, color = 'b')
plt.bar(z2, B, color = 'r', bottom = A)
plt.show()
```



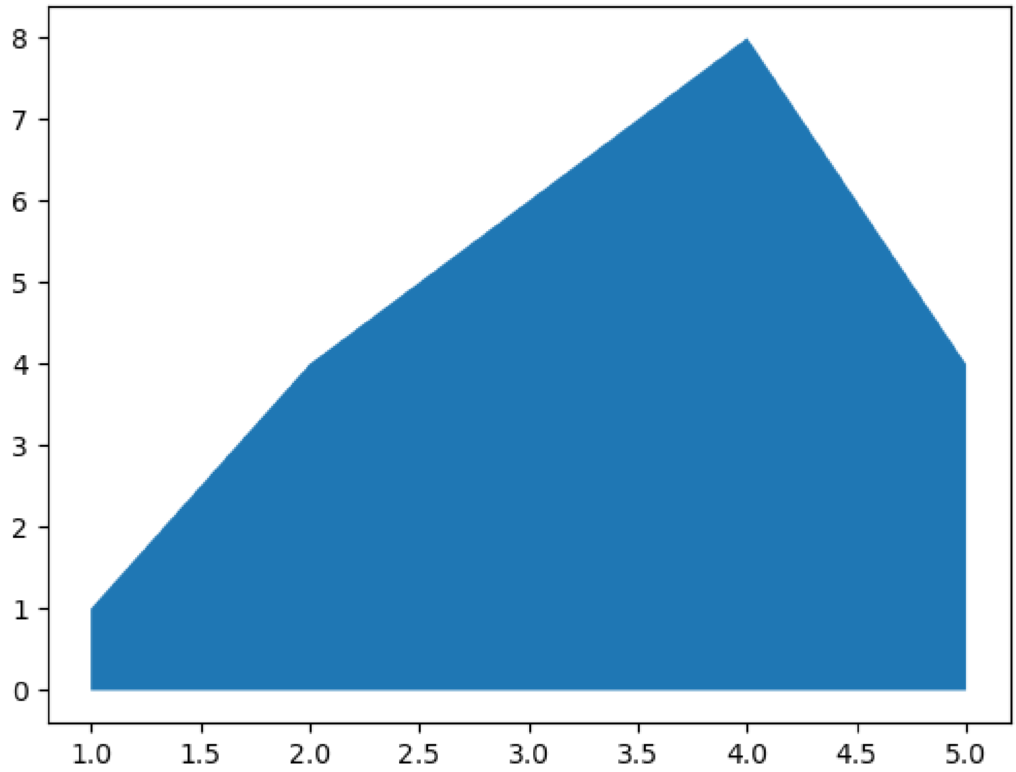
```
plt.figure(figsize=(7,7))
x10 = [35, 25, 20, 20]
labels = ['Computer', 'Electronics', 'Mechanical', 'Chemical']
plt.pie(x10, labels=labels);
plt.show()
```



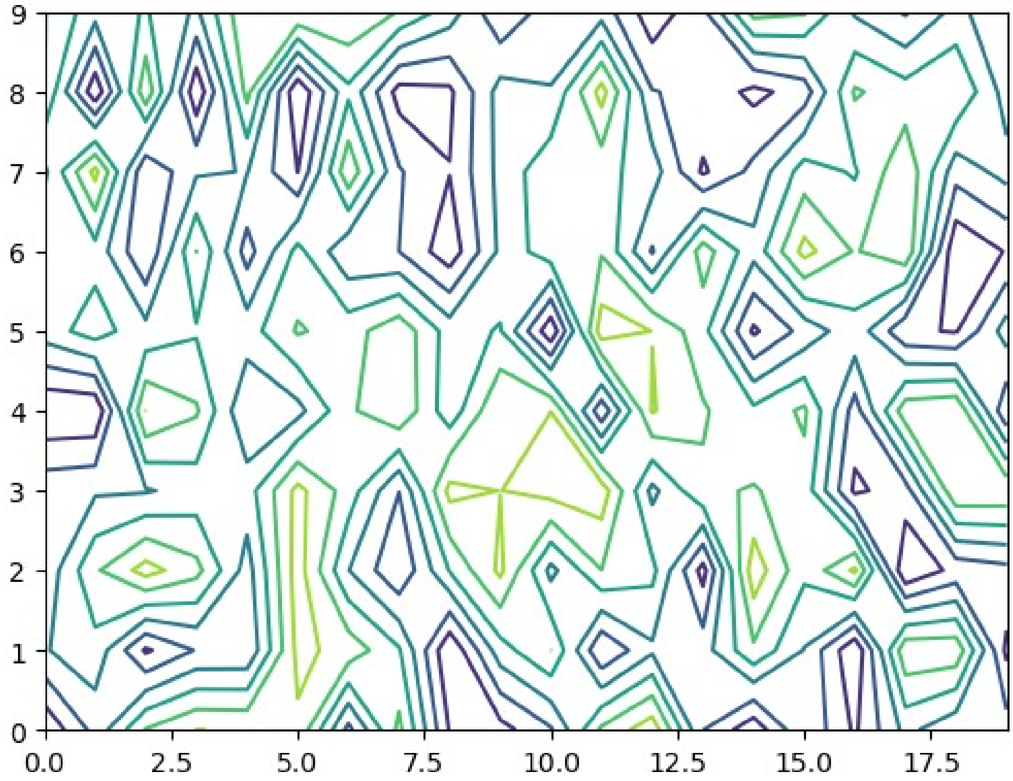
```
data3 = np.random.randn(100)
plt.boxplot(data3)
plt.show();
```



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



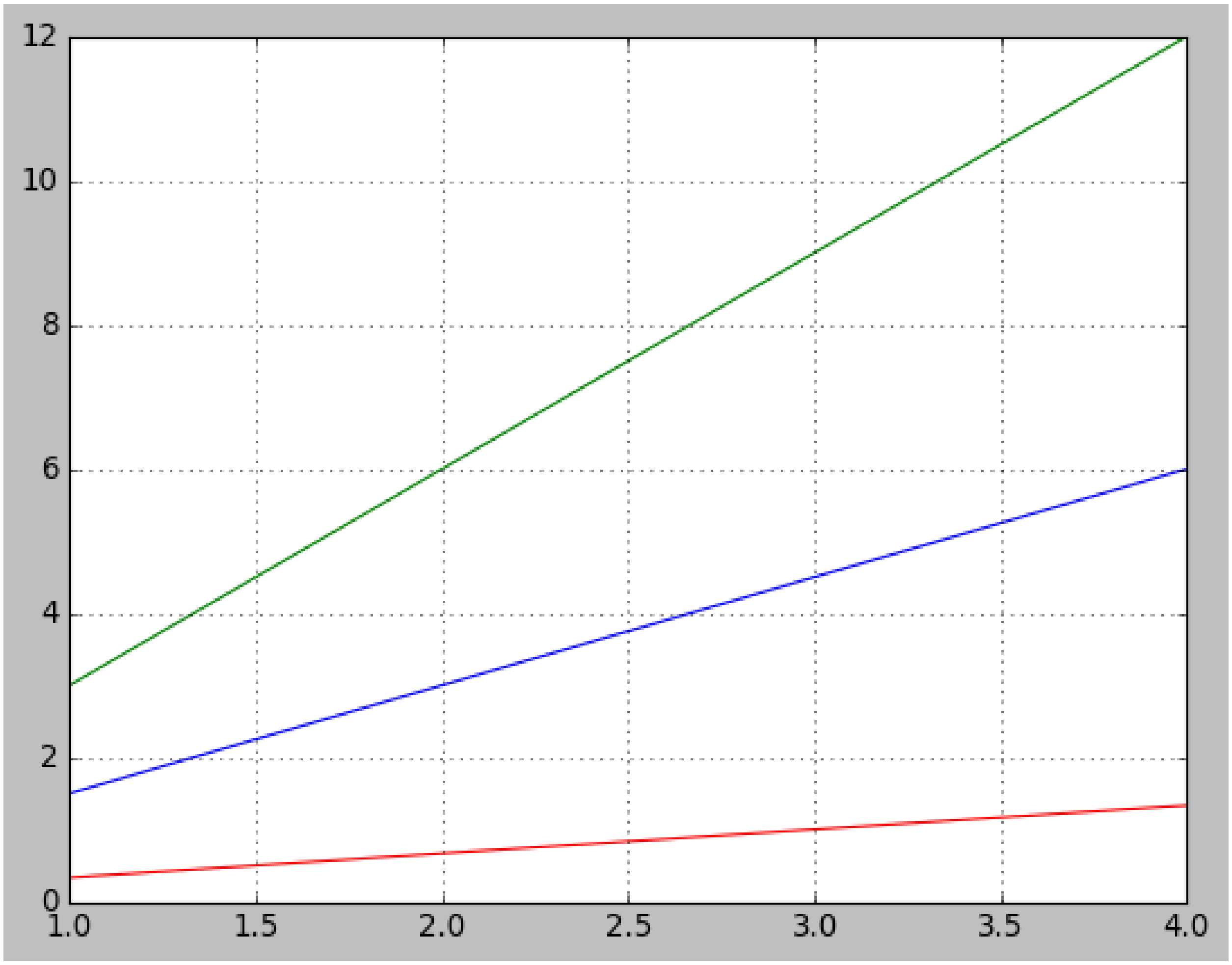
```
# Create a matrix
matrix1 = np.random.rand(10, 20)
cp = plt.contour(matrix1)
plt.show()
```



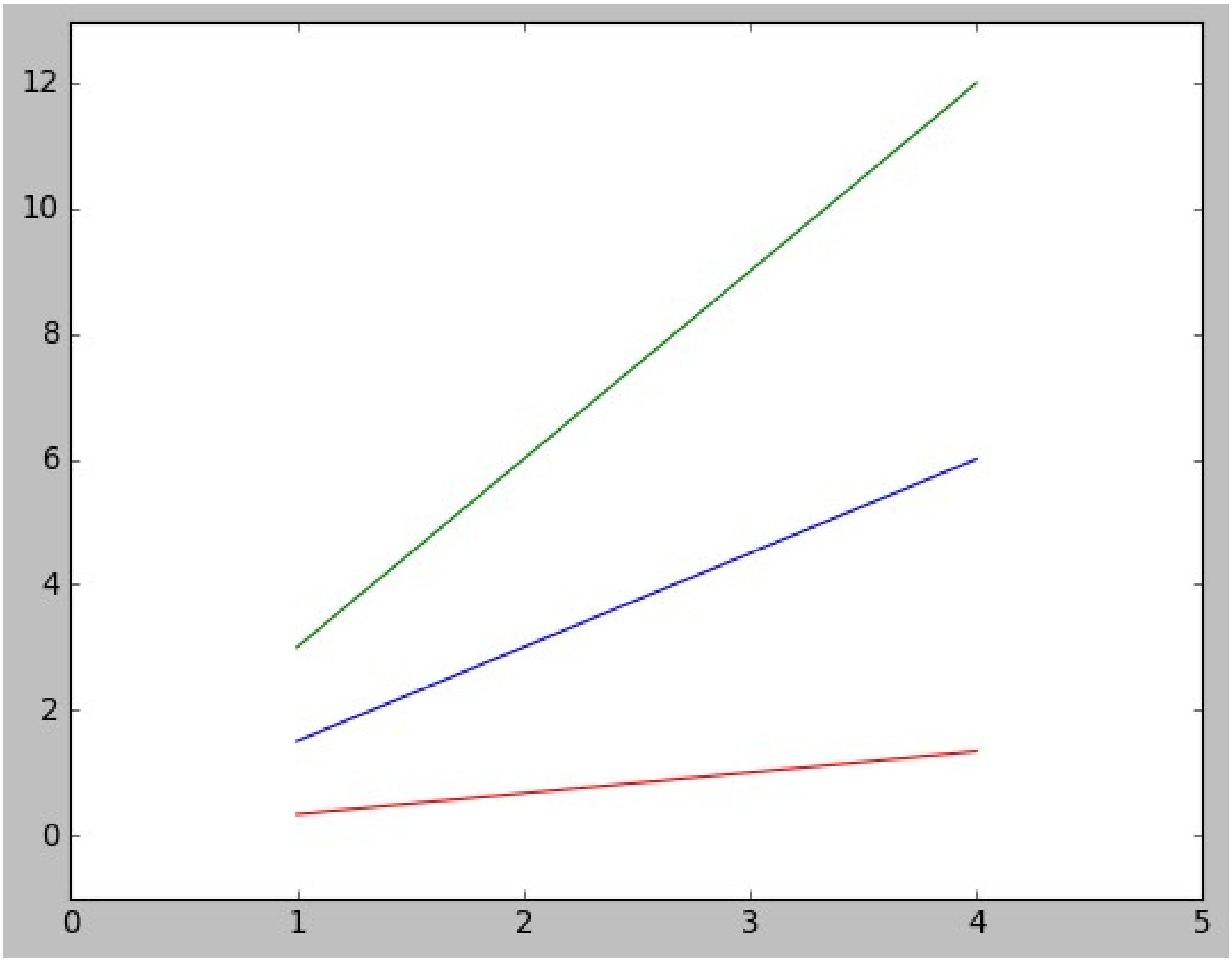
```python
# View list of all available styles
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', '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']
```

```python
# Set styles for plots
plt.style.use('classic')

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



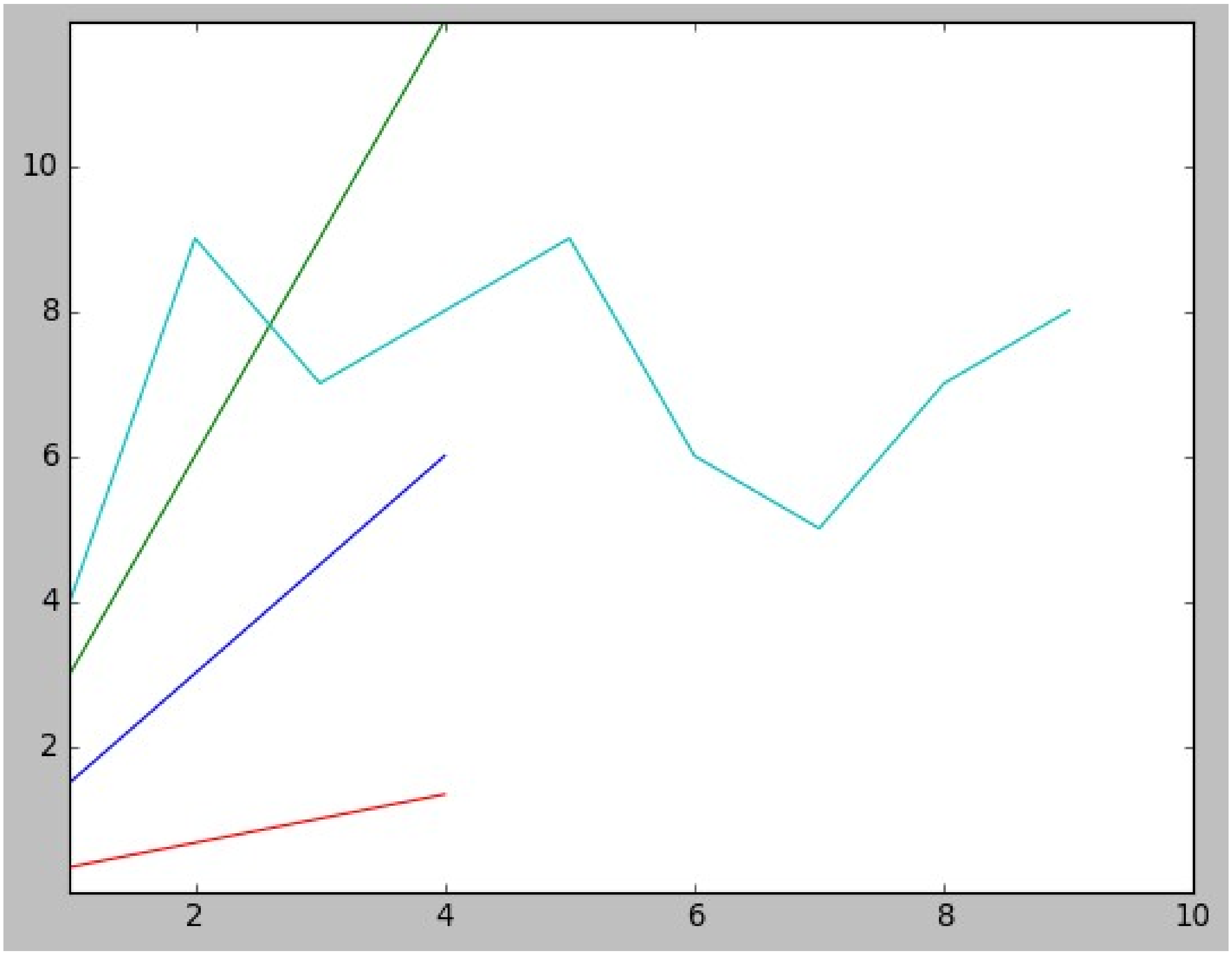
```
x15 = np.arange(1, 5)
plt.plot(x15, x15*1.5, x15, x15*3.0, x15, x15/3.0)
plt.axis() # shows the current axis limits values
plt.axis([0, 5, -1, 13])
plt.show()
```



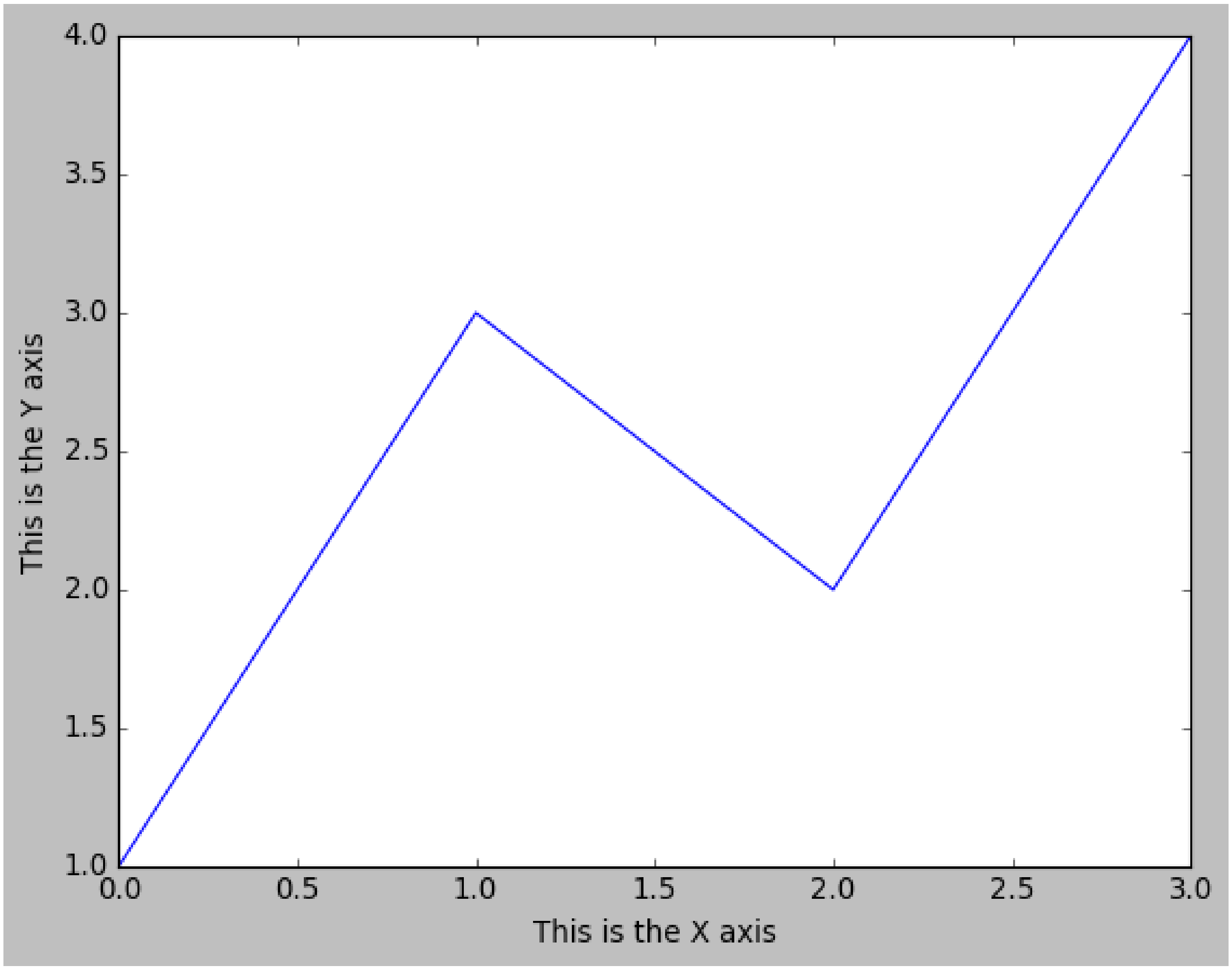
```
x15 = np.arange(1, 5)
plt.plot(x15, x15*1.5, x15, x15*3.0, x15, x15/3.0)
plt.xlim([1.0, 4.0])
plt.ylim([0.0, 12.0])
```

```
(0.0, 12.0)
```

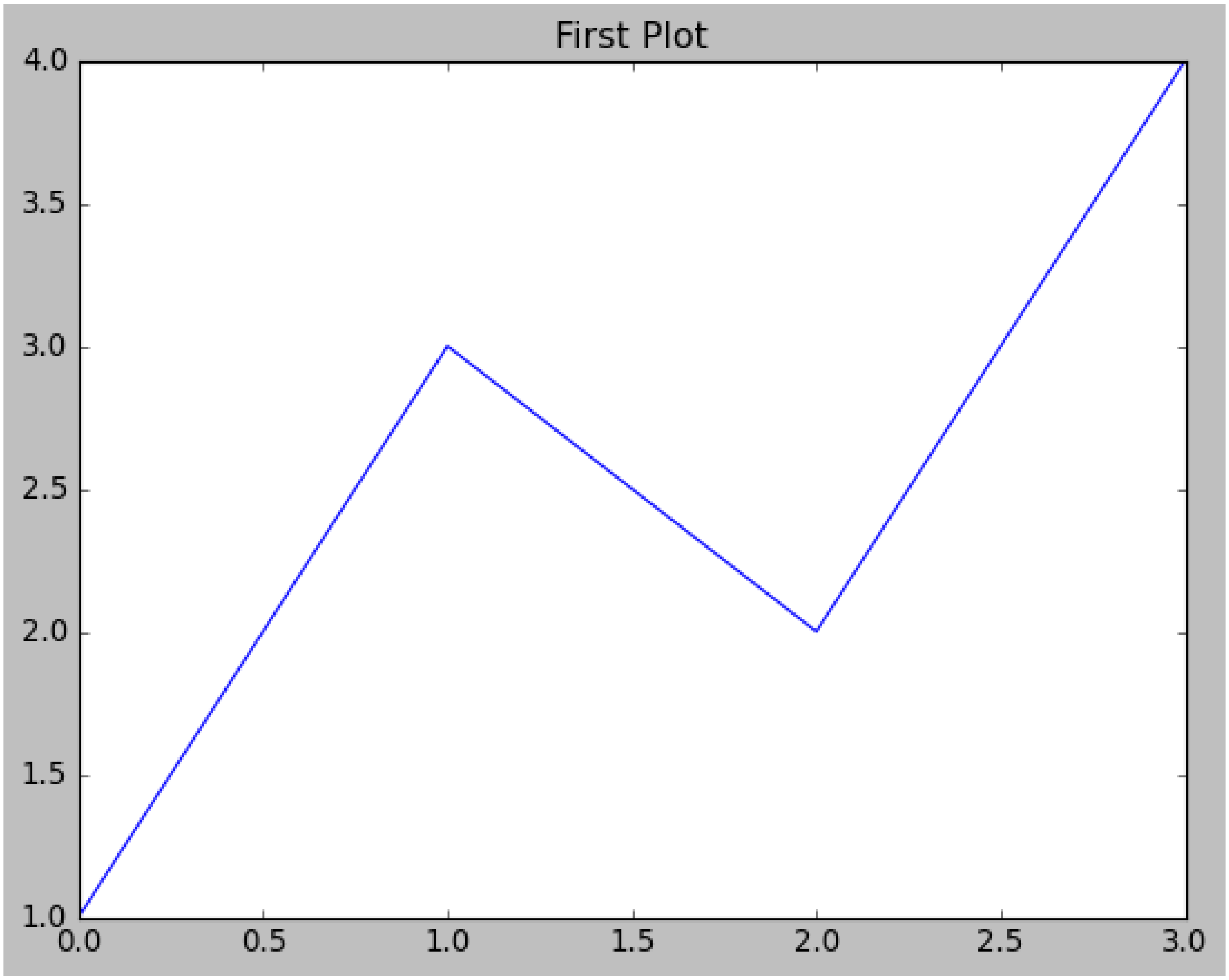
```
u = [5, 4, 9, 7, 8, 9, 6, 5, 7, 8]
plt.plot(u)
plt.xticks([2, 4, 6, 8, 10])
plt.yticks([2, 4, 6, 8, 10])
plt.show()
```



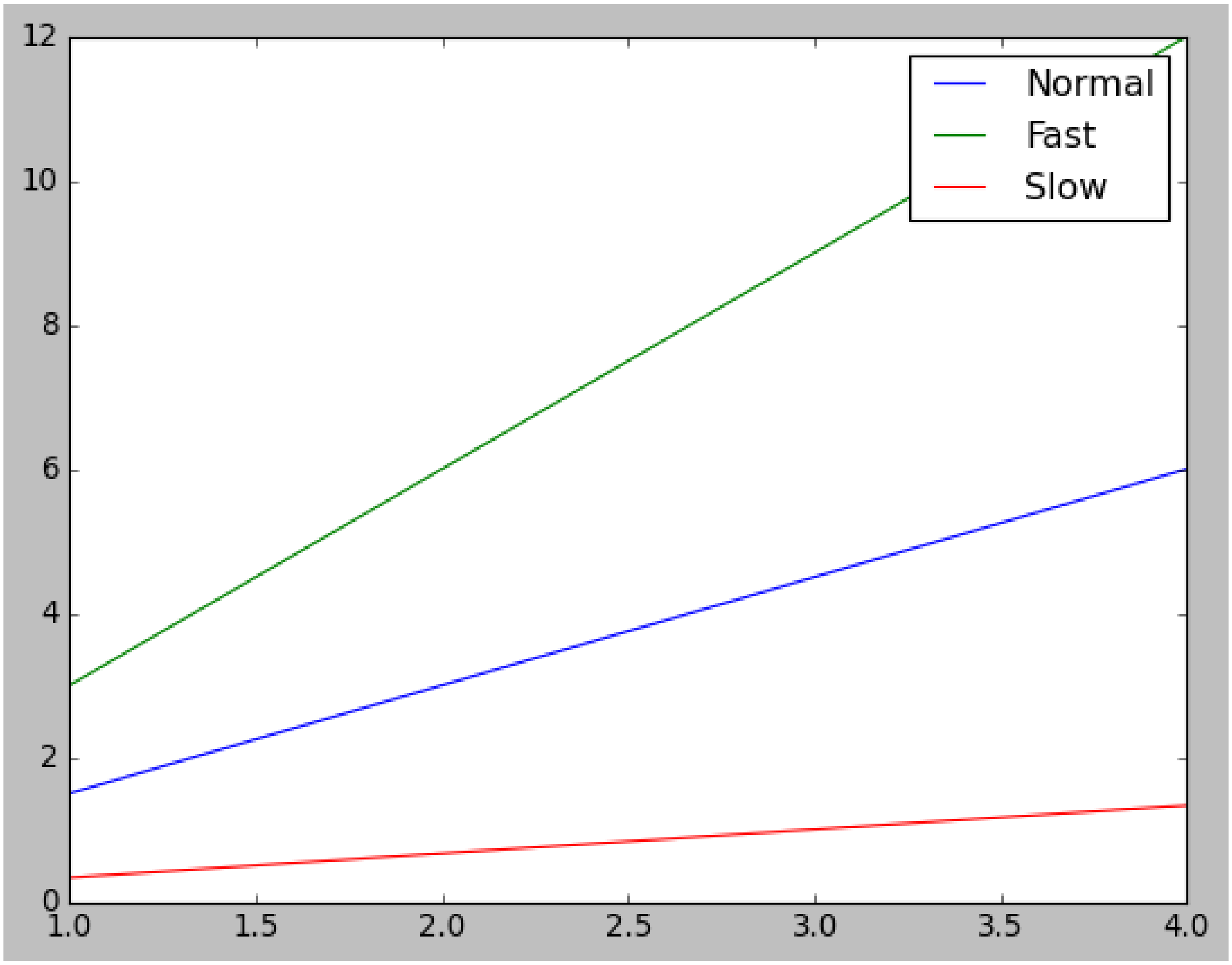
```
plt.plot([1, 3, 2, 4])
plt.xlabel('This is the X axis')
plt.ylabel('This is the Y axis')
plt.show()
```



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



```
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()
```



```
x15 = np.arange(1, 5)
fig, ax = plt.subplots()
ax.plot(x15, x15*1.5, label='Normal')
```