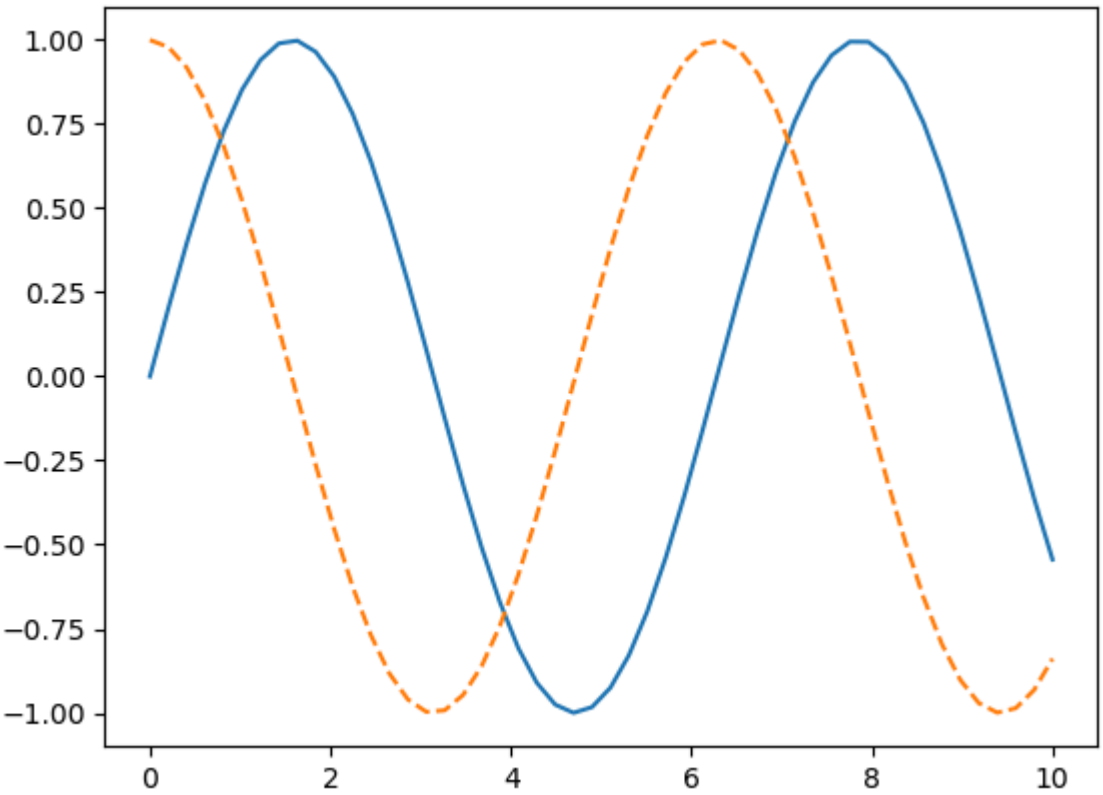
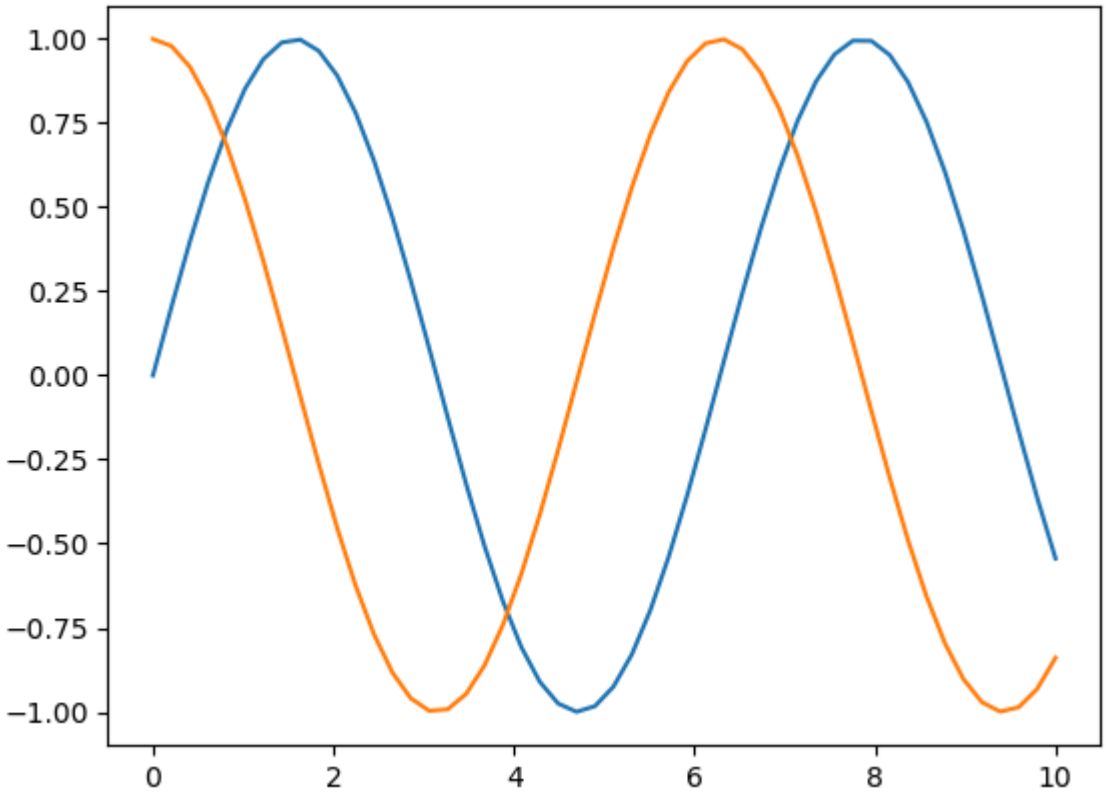
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
%matplotlib inline
x1 = np.linspace(0, 10, 50)
# create a plot figure
plt.plot(x1, np.sin(x1), '-')
plt.plot(x1, np.cos(x1), '--')
plt.show()
```



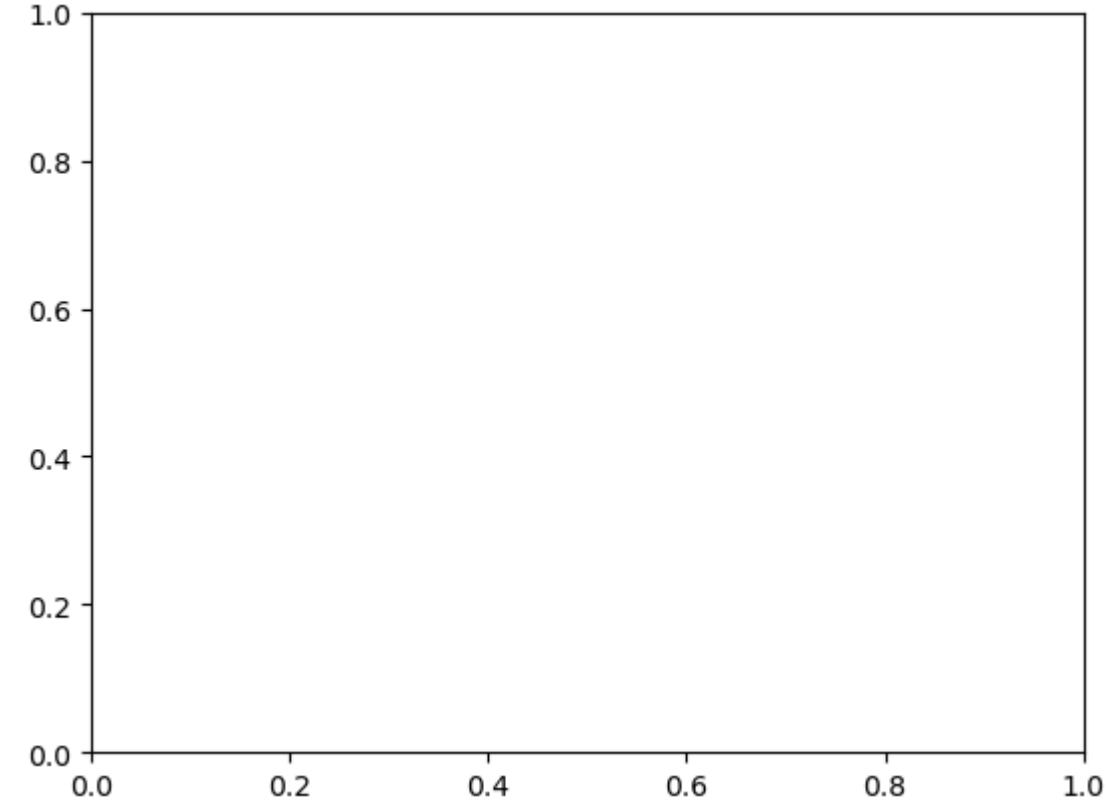
In [10]:
```
%matplotlib inline
x1 = np.linspace(0,10,50)
plt.plot(x1,np.sin(x1))
plt.plot(x1,np.cos(x1))
plt.show()
```



Out[10]: <Axes: >

In [16]:
```
plt.gcf() #get current figure
```

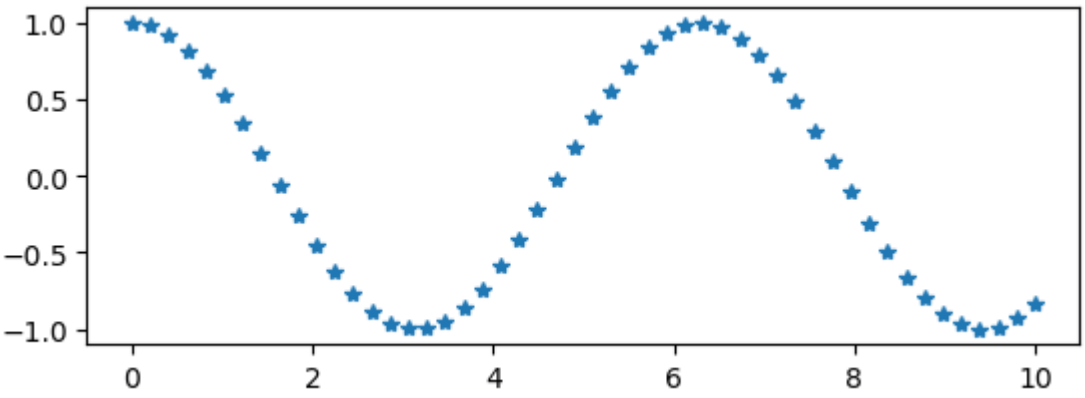
Out[16]:



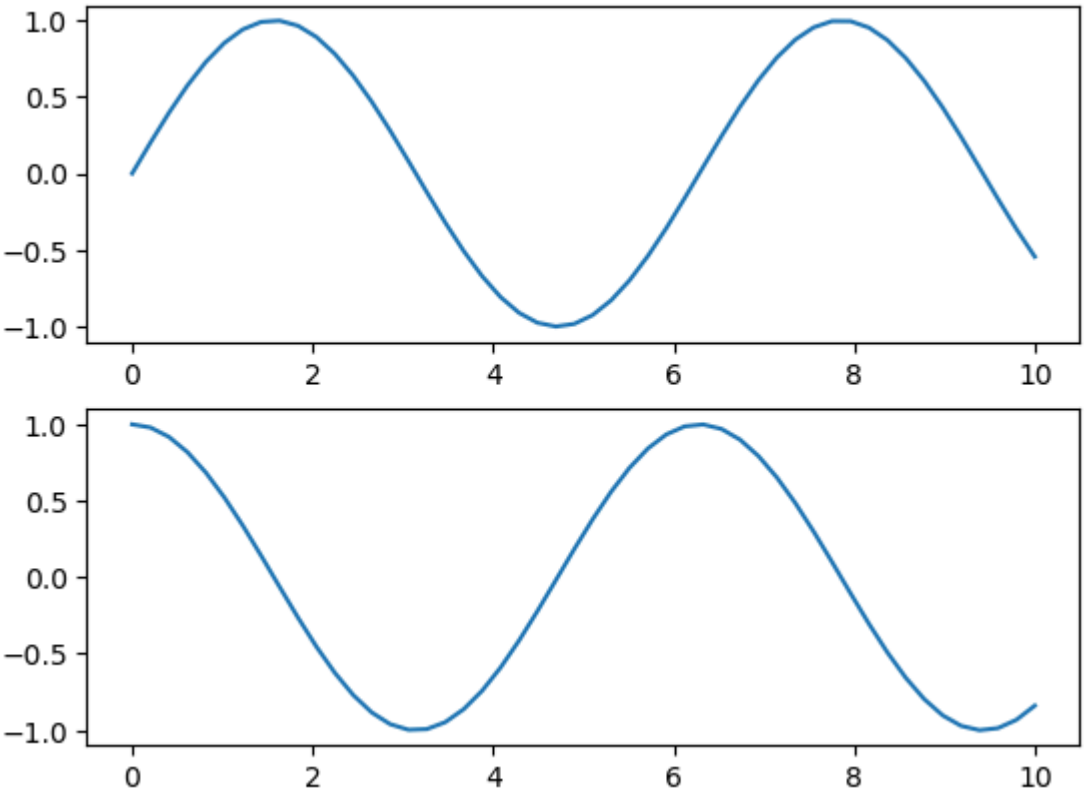
In [15]:
```
plt.gca() #get current axes
```

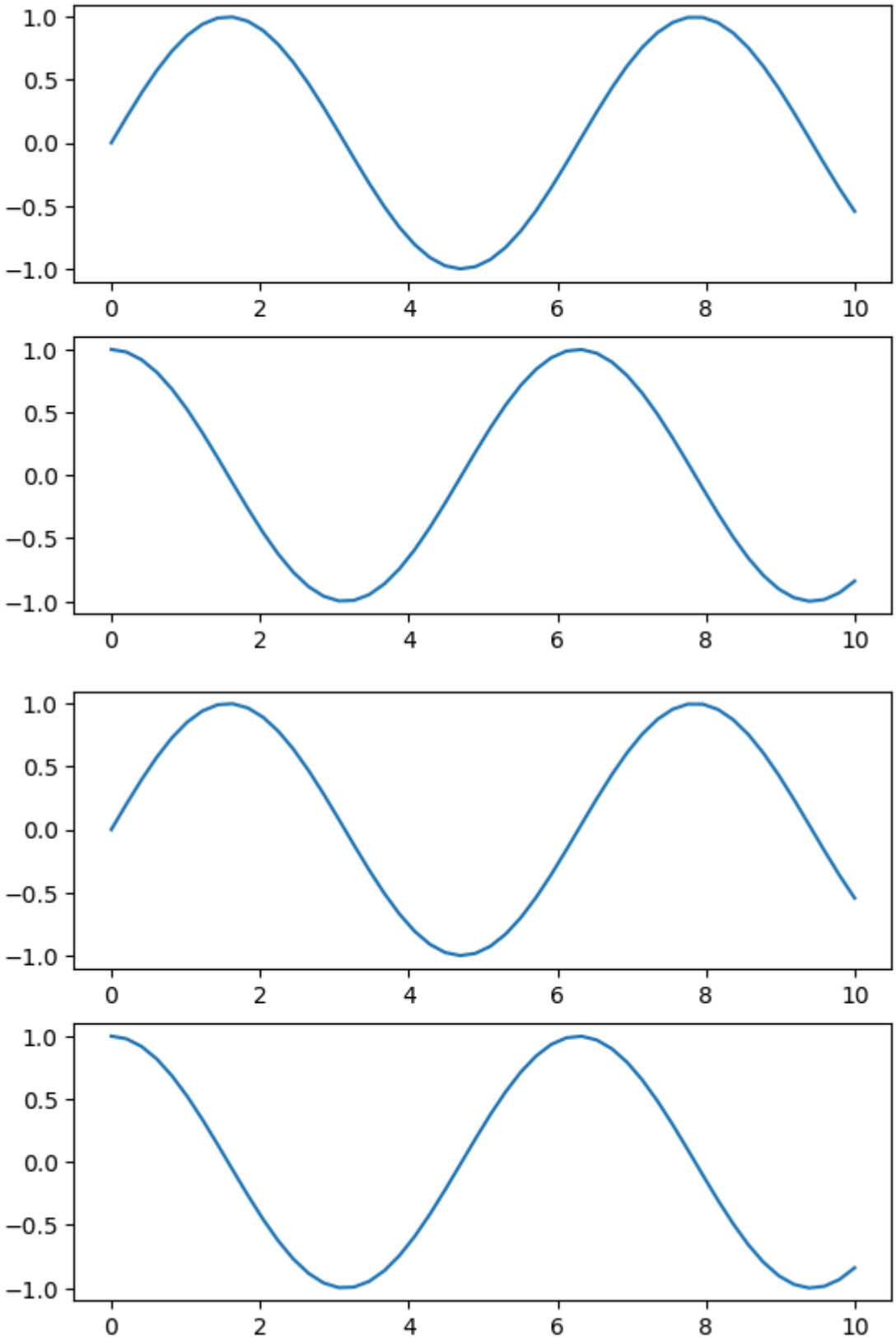
Out[15]: <Axes: >

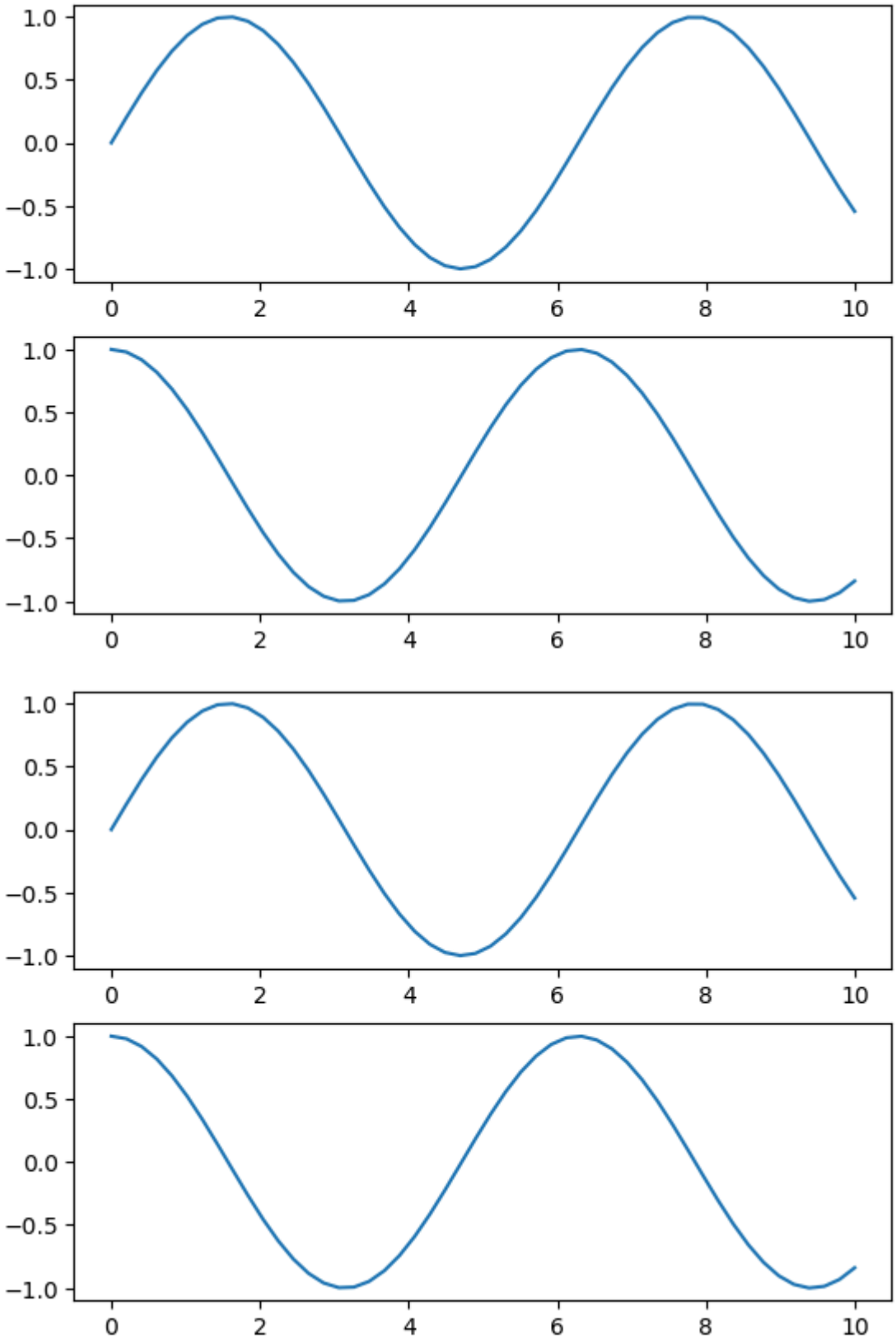
In [19]:
```
# create the first of two panels and set current axis
plt.subplot(2, 1, 1)   #(rows, columns, panel no)
plt.plot(x1, np.cos(x1), '*')
plt.show()
```

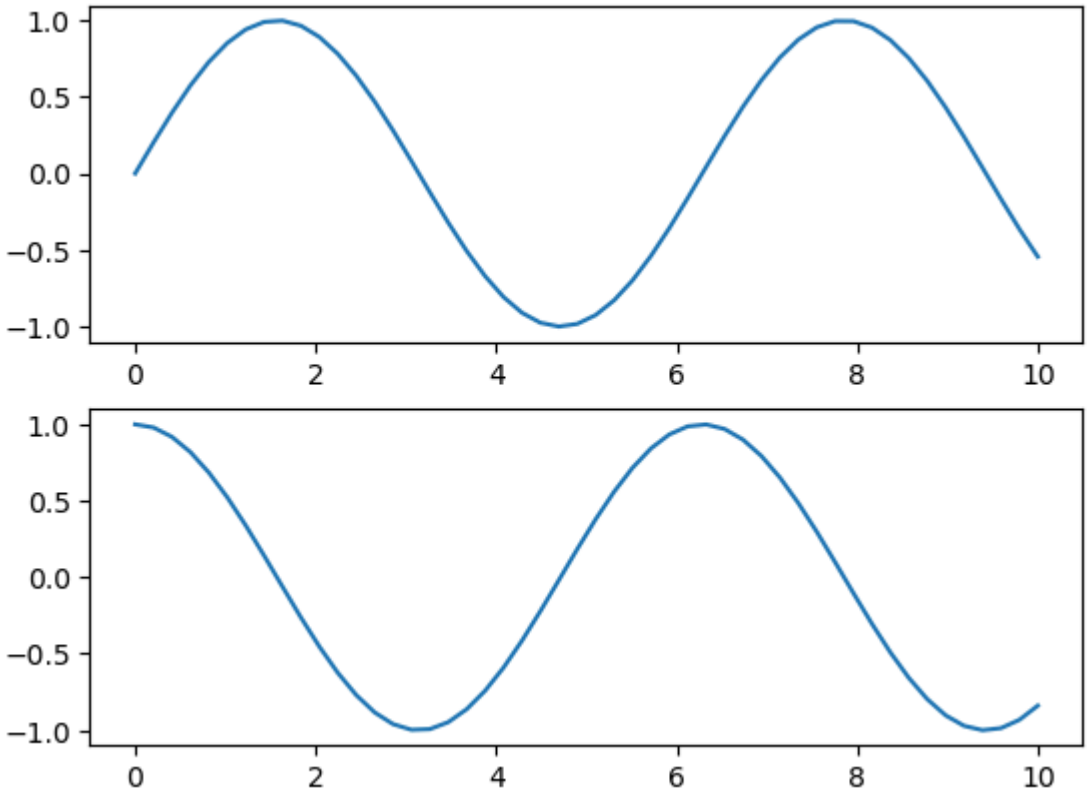


In [25]:
```
# create a plot figure
plt.figure()

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

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

plt.show()
```




1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10


1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10



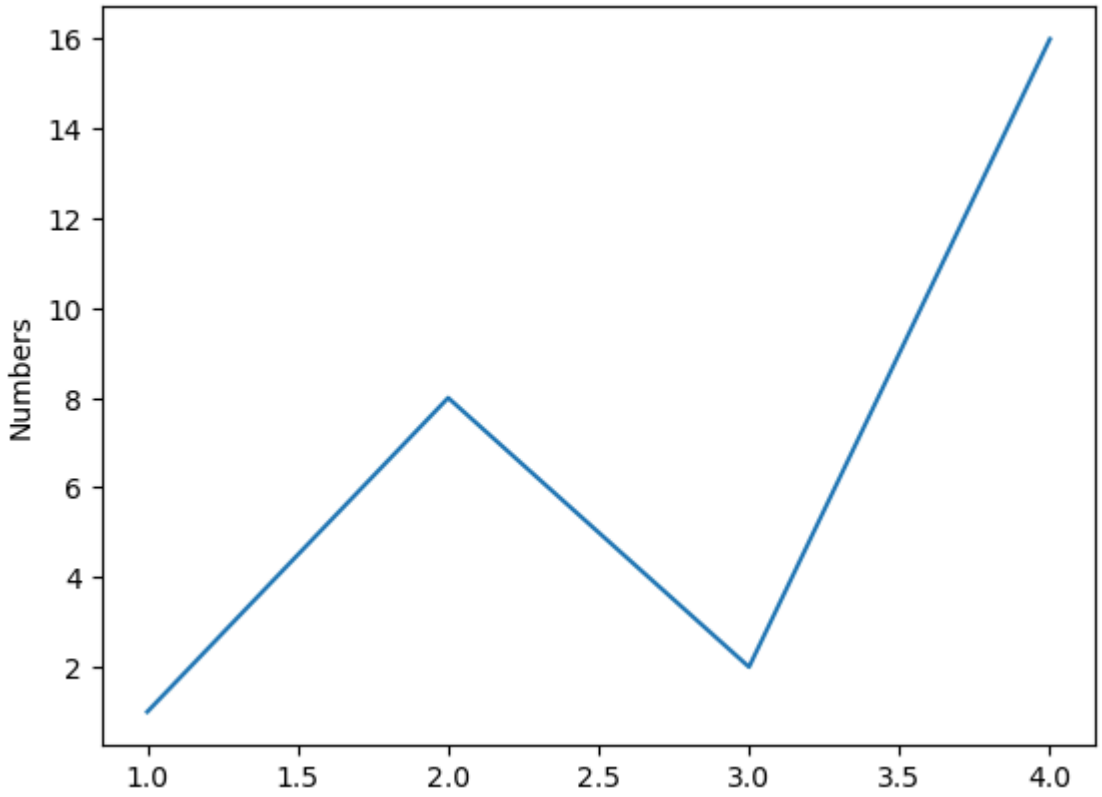
In [29]:
```
print(plt.gcf())
```

```
Figure(640x480)
```

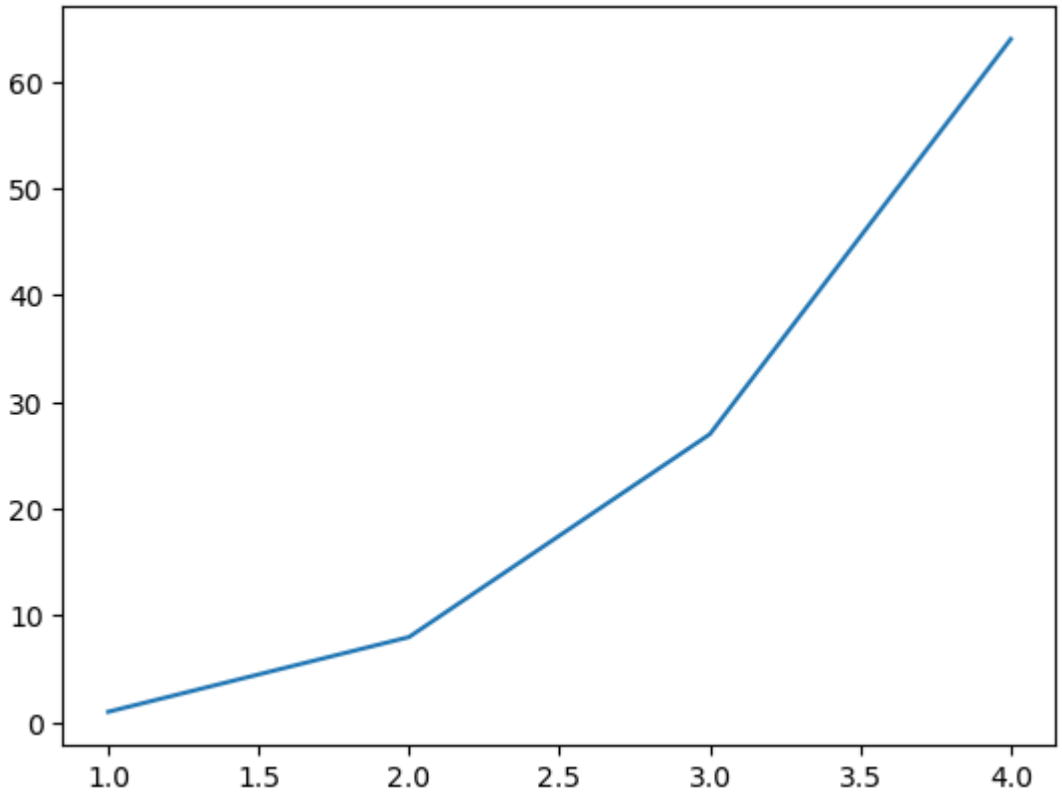
In [30]:
```
print(plt.gca())
```

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

In [33]:
```
plt.plot([1,2,3,4],[1,8,2,16])
plt.ylabel("Numbers")
plt.show()
```



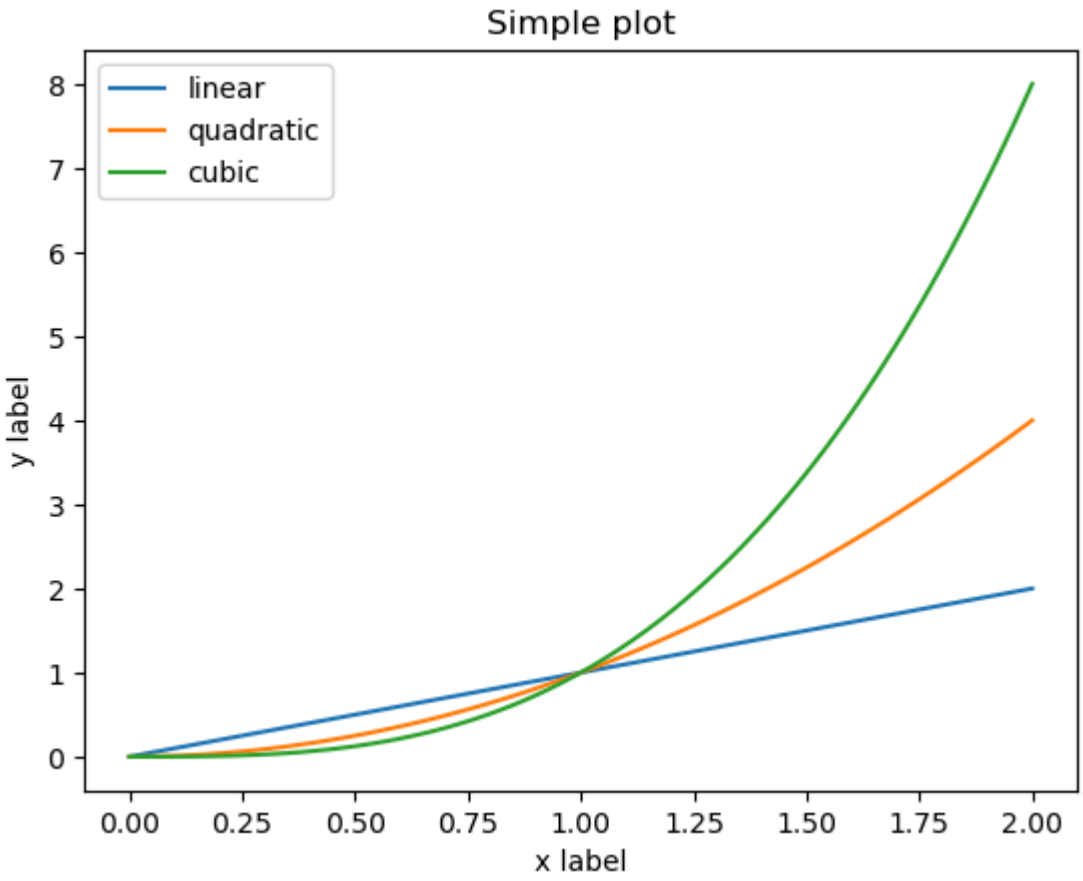
In [34]:
```
plt.plot([1,2,3,4],[1,8,27,64])
plt.show()
```



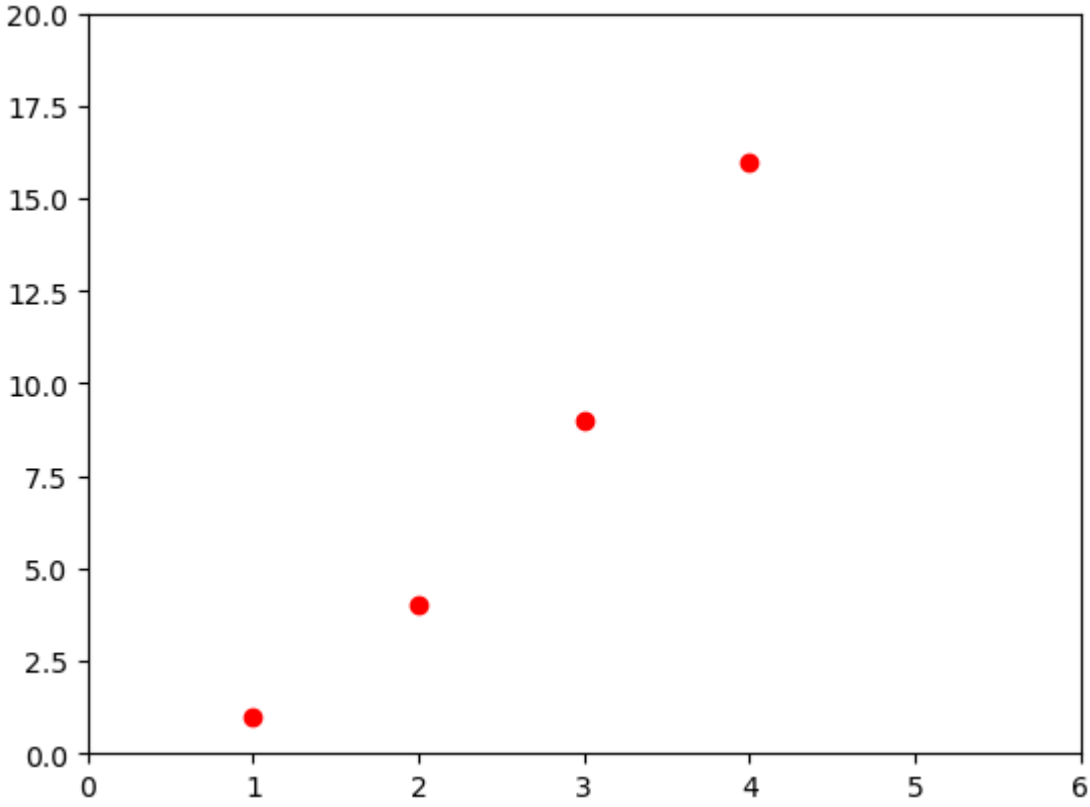
State-machine interface

Pyplot provides the state-machine interface to the underlying object-oriented plotting library. The state-machine implicitly and automatically creates figures and axes to achieve the desired plot. For example:

In [36]:
```
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()
```

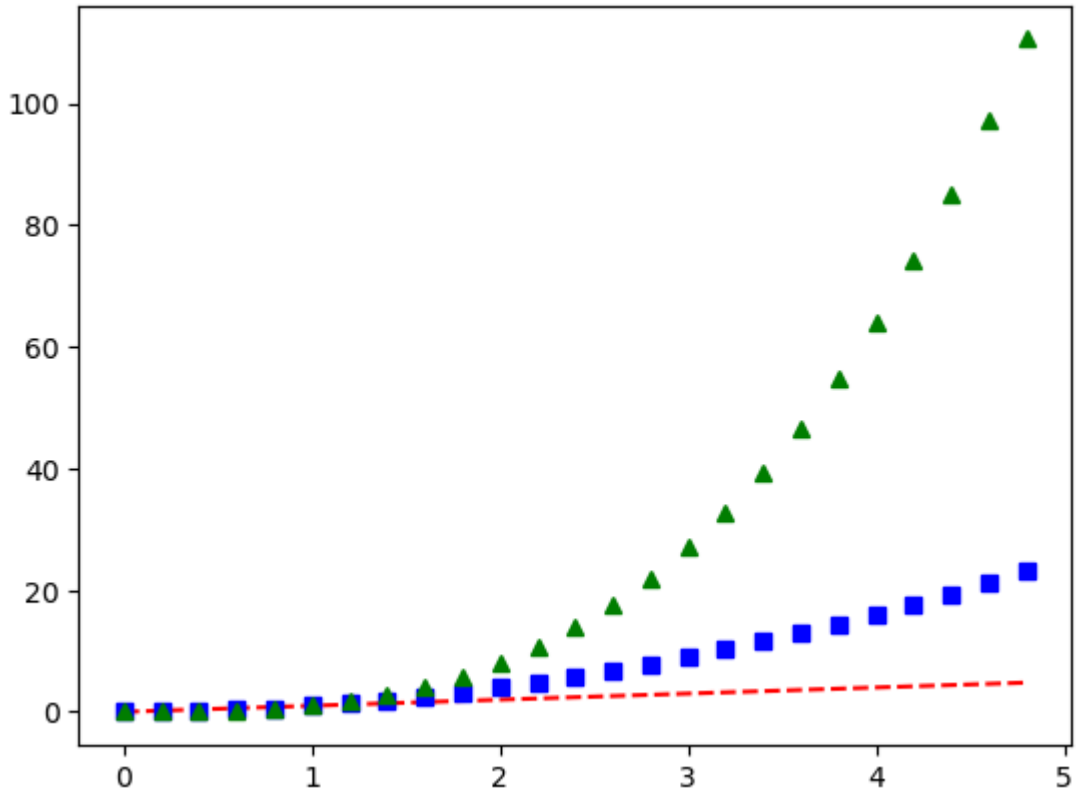


In [42]:
```
plt.plot([1,2,3,4],[1,4,9,16],'ro') #if u dont need full lines just dots
plt.axis([0,6,0,20])
plt.show()
```

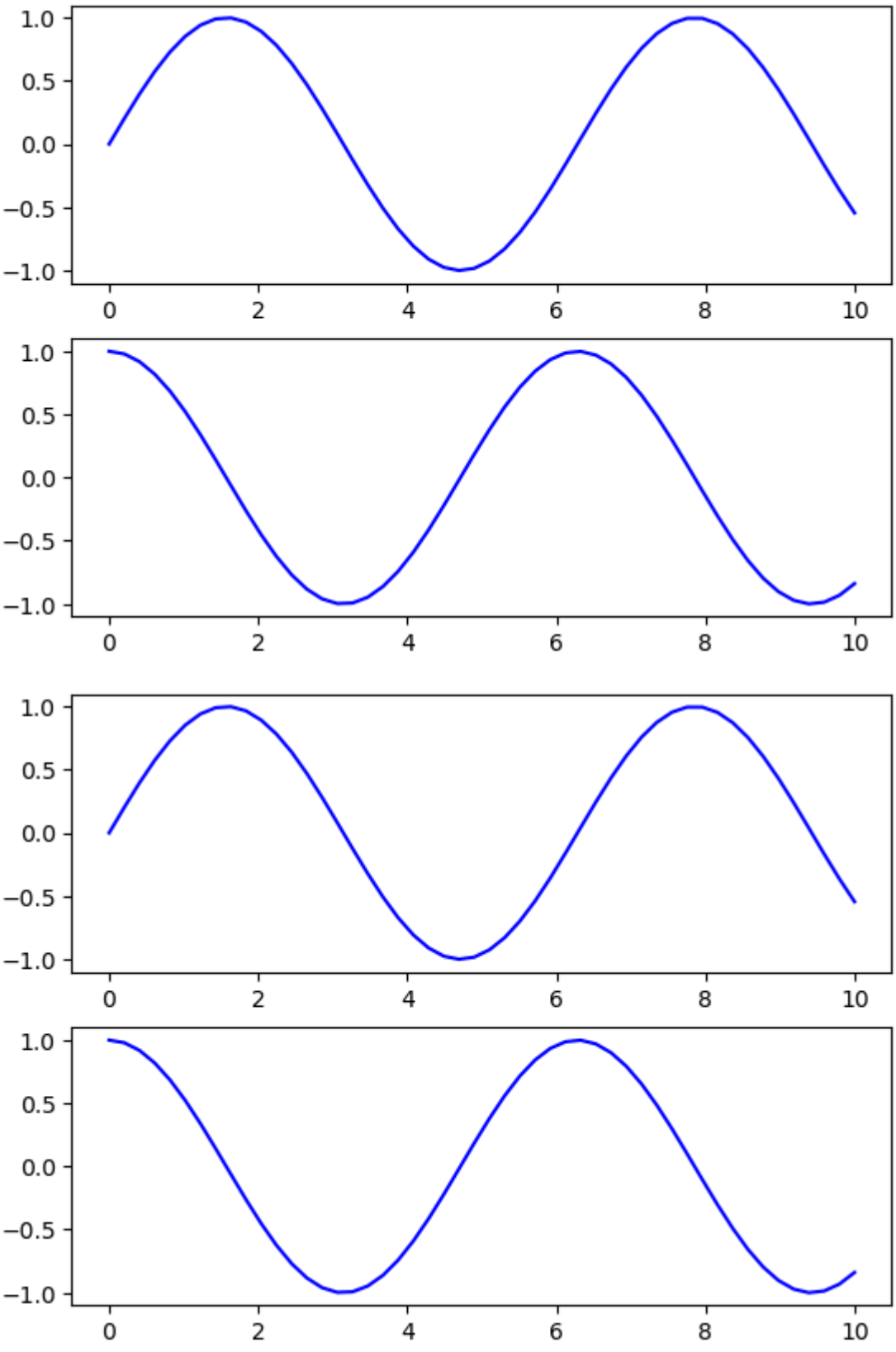


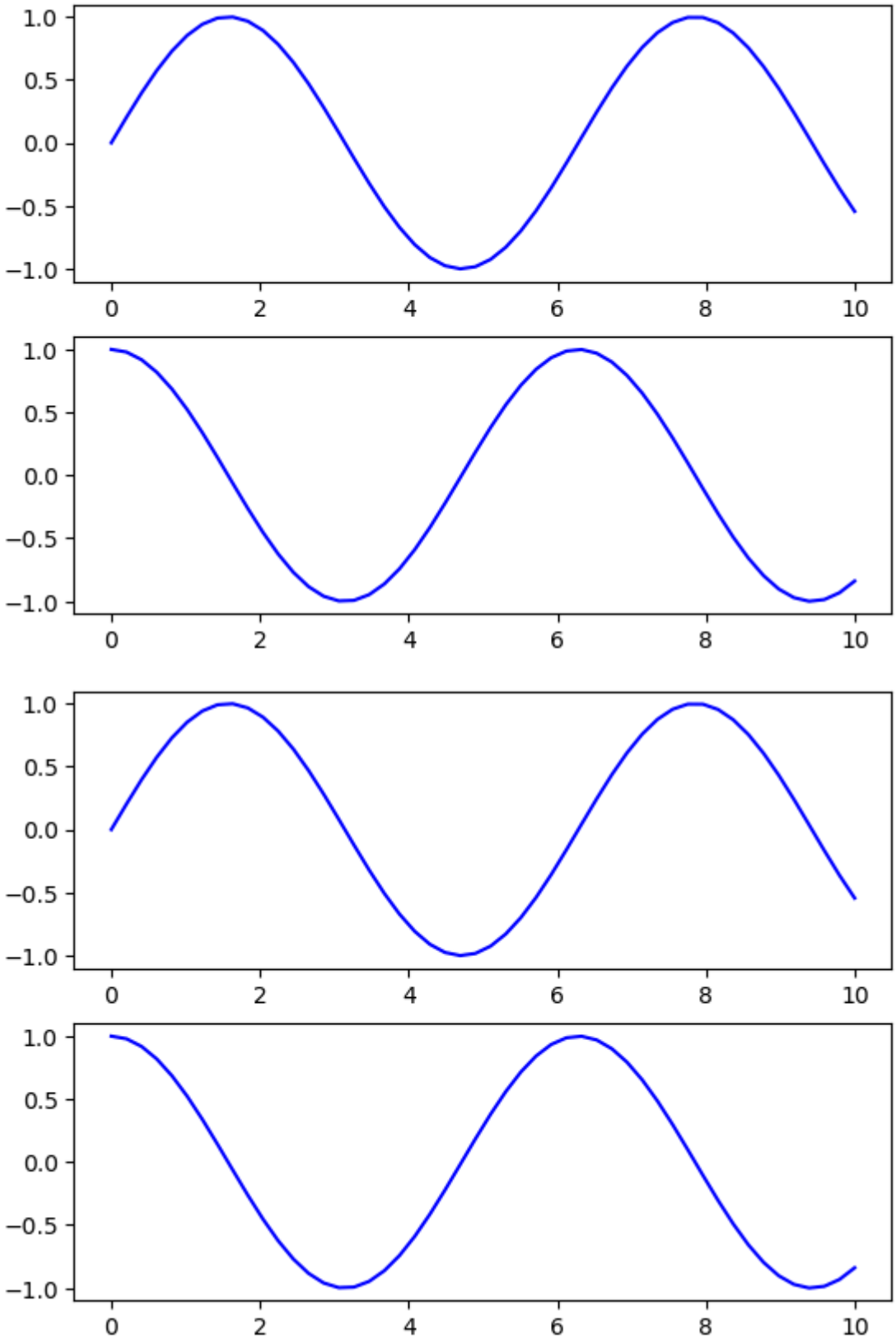
working with numpy arrays

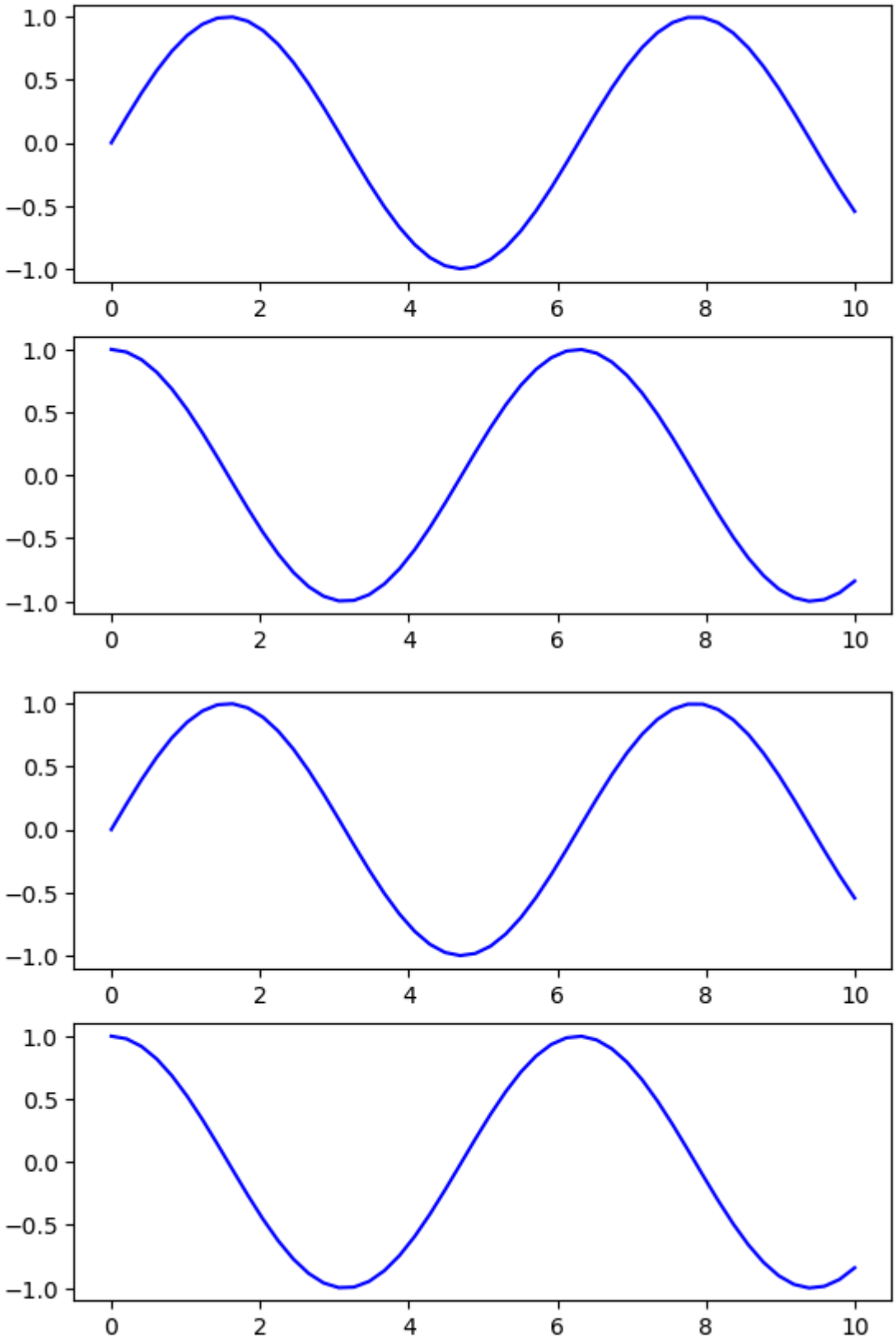
In [47]:
```
t = np.arange(0.,5.,0.2) #(start,stop,step)
plt.plot(t,t,'r--',t,t**2,'bs',t,t**3,'g^')
plt.show()
```



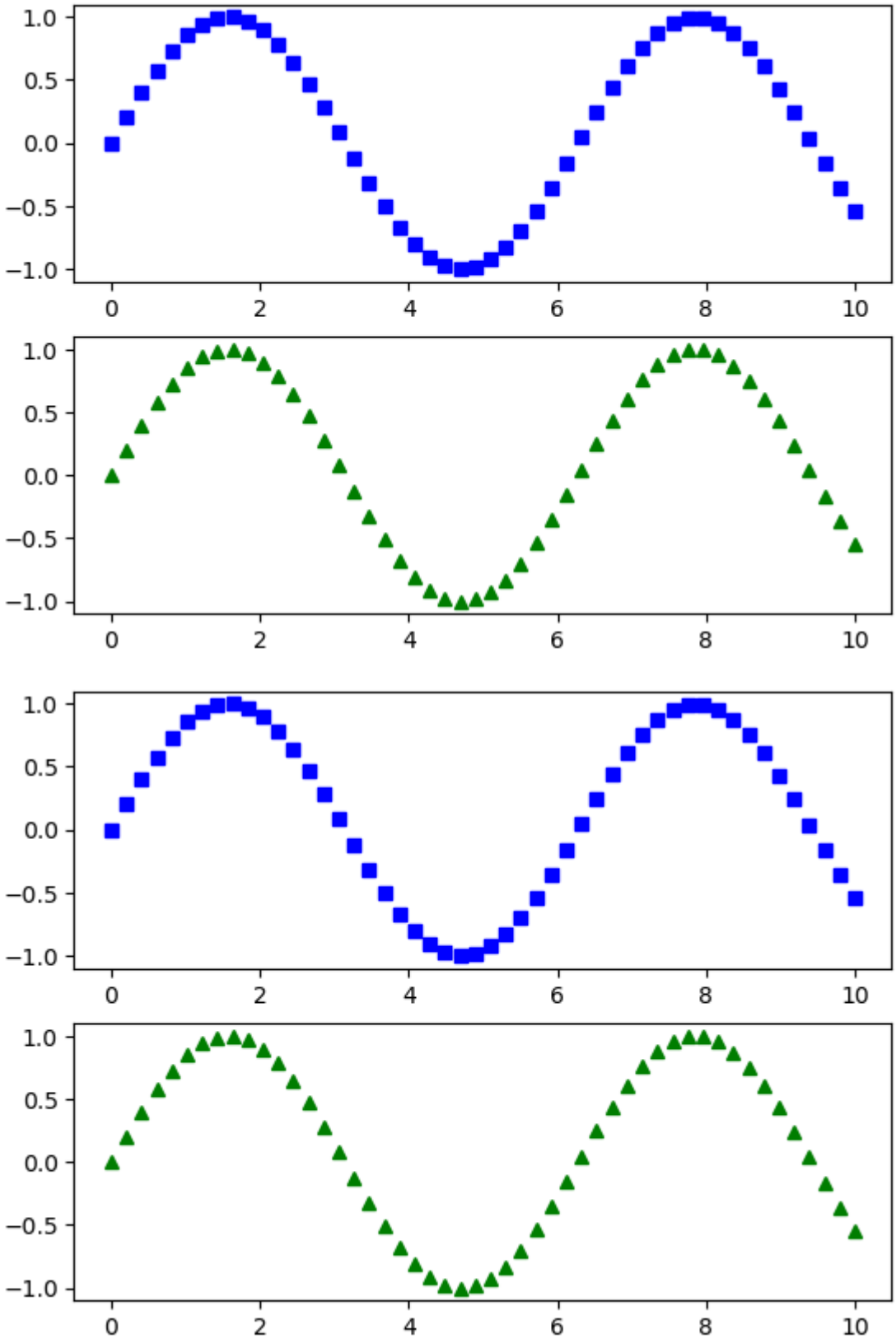
In [53]:
```
# First create a grid of plots
# ax will be an array of two Axes objects
fig, ax = plt.subplots(2)
ax[0].plot(x1, np.sin(x1), 'b-')
ax[1].plot(x1, np.cos(x1), 'b-');
plt.show()
```




1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10

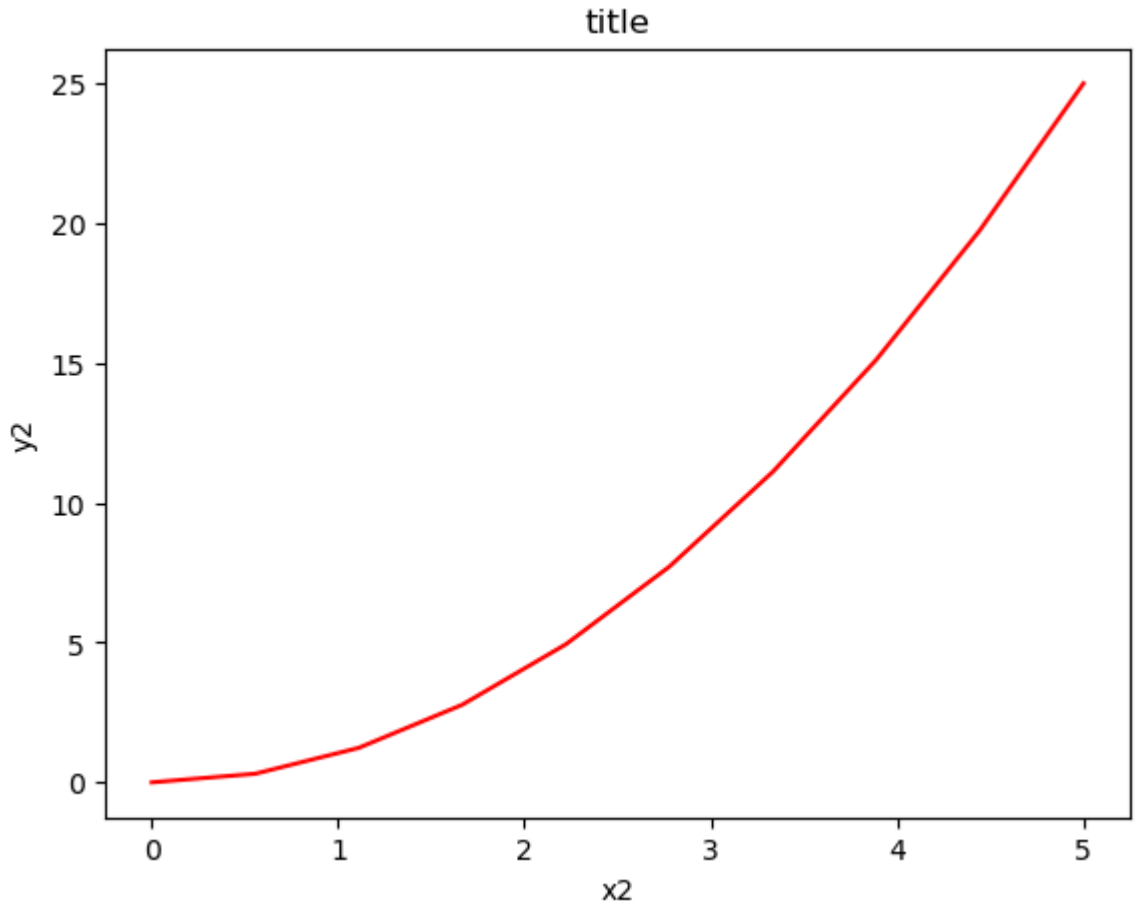


In [55]:
```
fig,ax = plt.subplots(2)

ax[0].plot(x1,np.sin(x1),'bs')
ax[1].plot(x1,np.sin(x1),'g^')
plt.show()
```

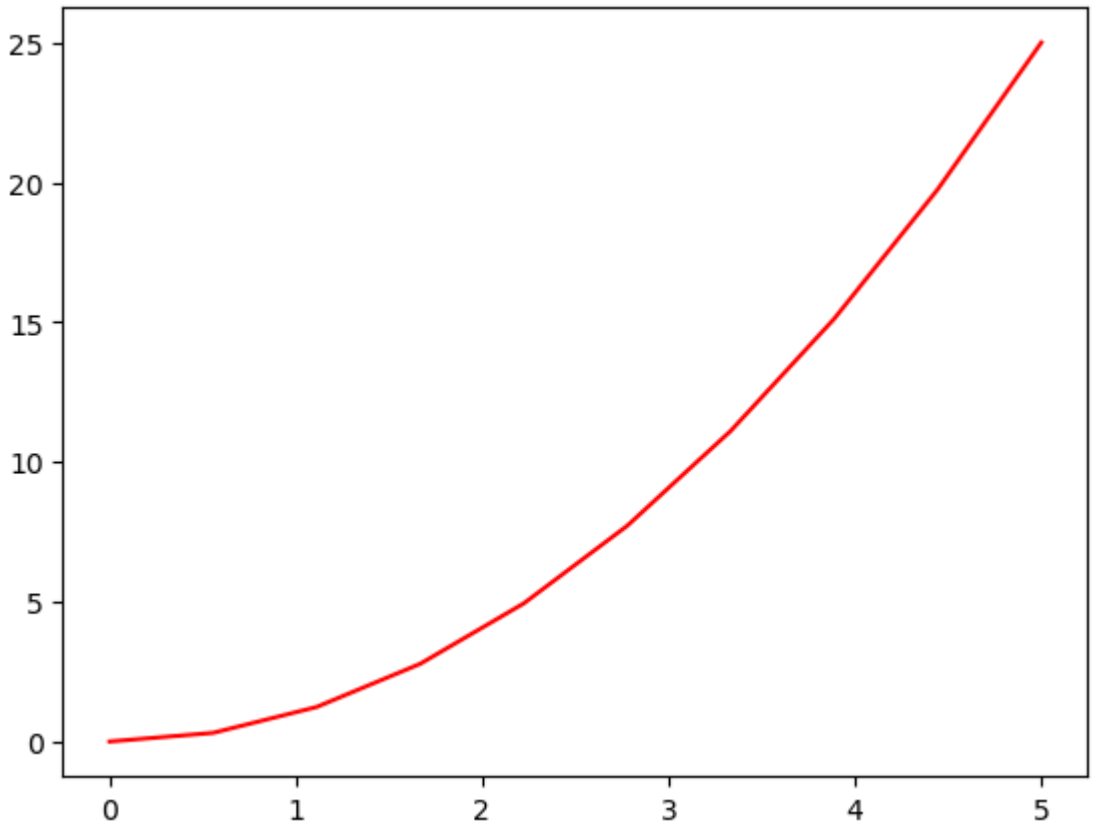


In [58]:
```
fig = plt.figure()

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')
```

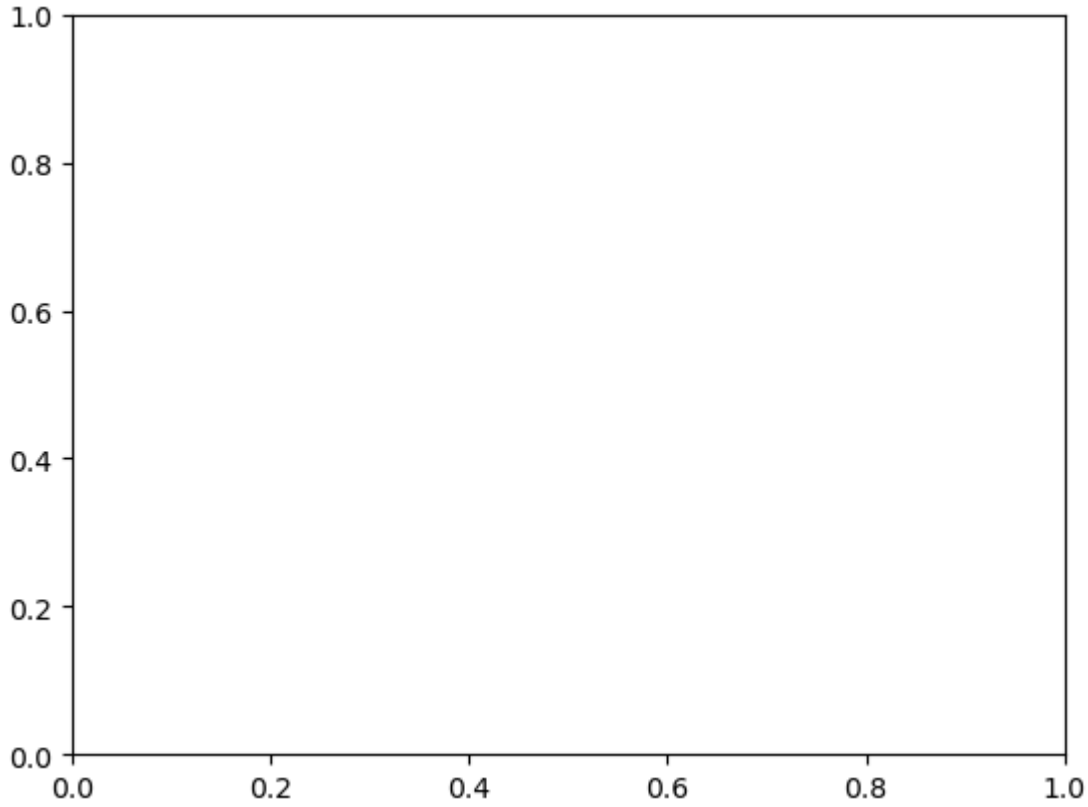
```
axes.set_ylabel('y2')
axes.set_title('title');
plt.show()
```



In [64]:
```
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')
plt.show()
```



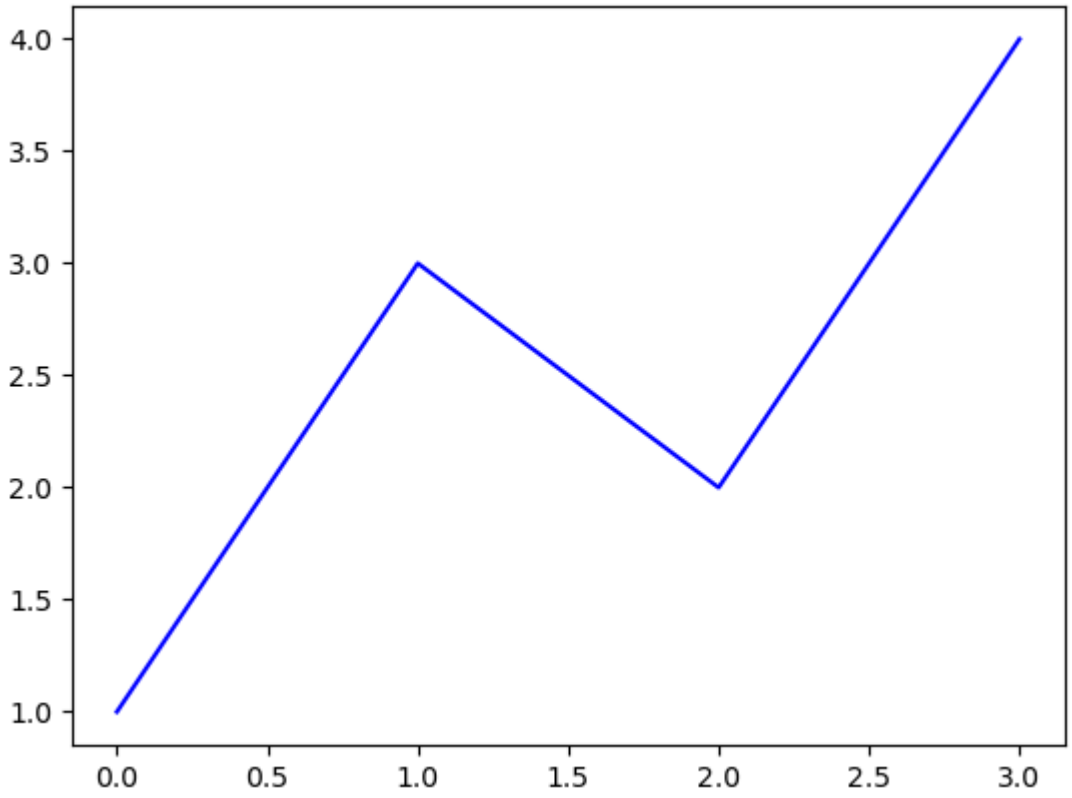
In [70]:
```
axes = plt.axes()
fig = plt.figure()
plt.show()
```



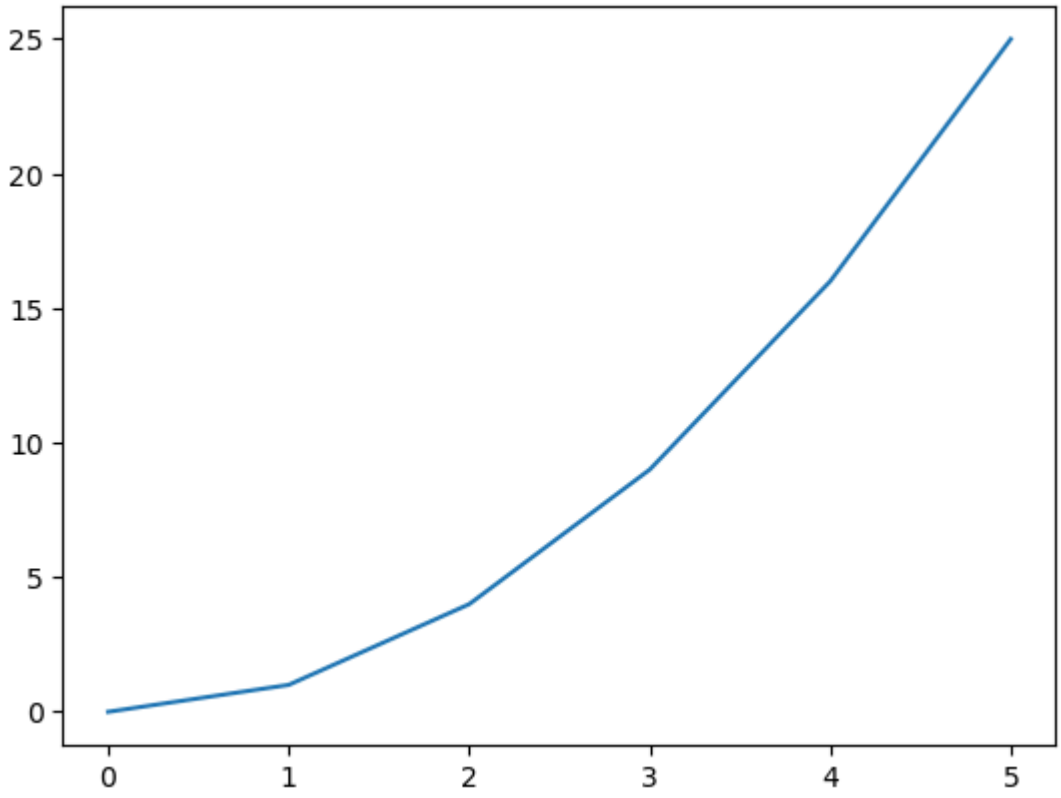

```
<Figure size 640x480 with 0 Axes>
```

start plotting with matplotlib

In [69]:
```
plt.plot([1,3,2,4],'b-')
plt.show()
```

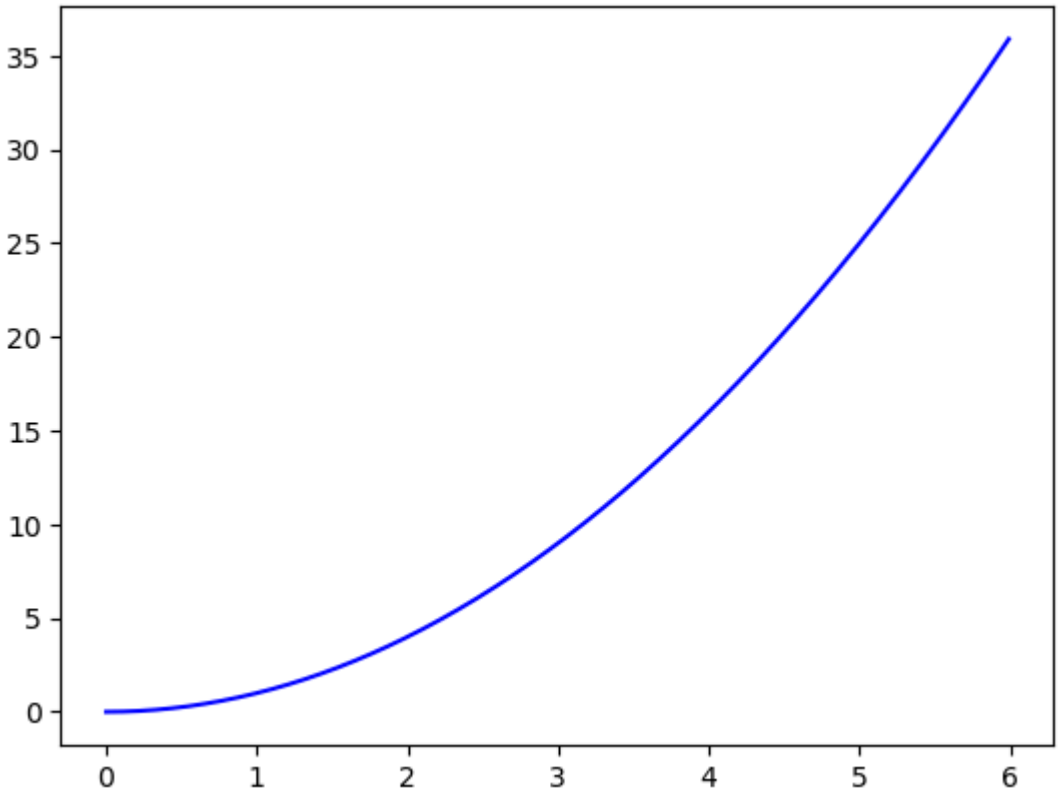


In [72]:
```
x3 = range(6)
plt.plot(x3, [xi**2 for xi in x3])
plt.show()
```



In [77]:
```
x3 = np.arange(0.0,6.0,0.01)
plt.plot(x3, [xi**2 for xi in x3],'b-')
```

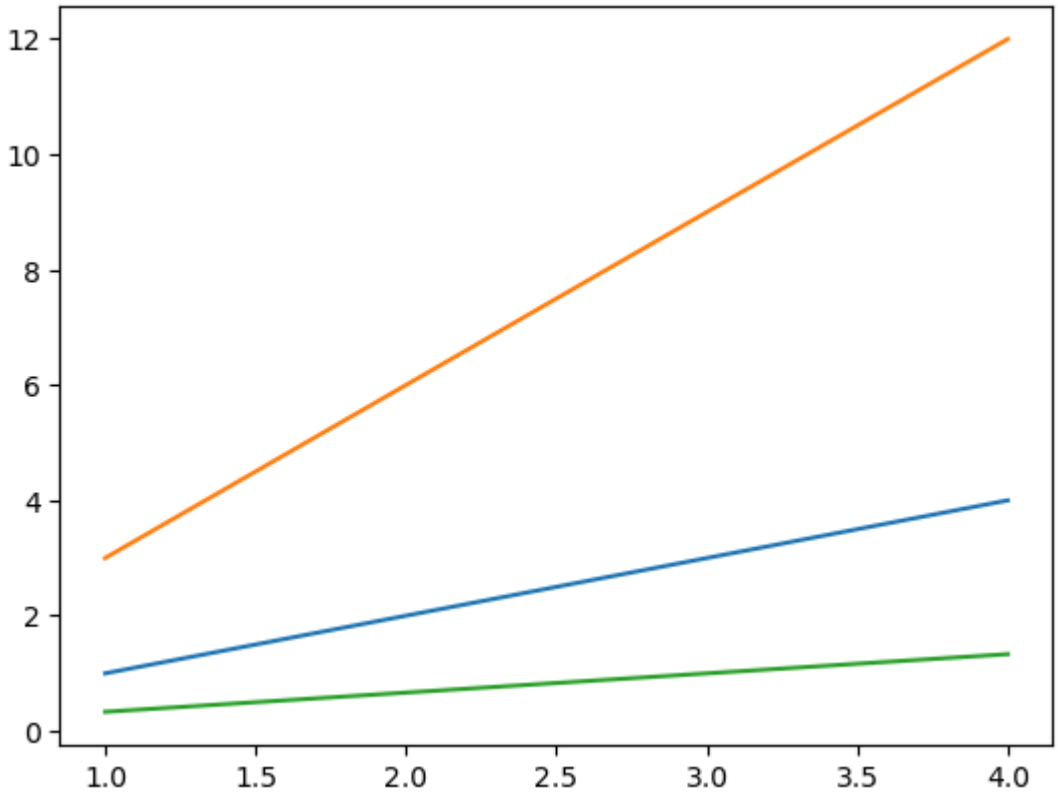
```
plt.show()
```



multiline plots

Multiline Plots mean plotting more than one plot on the same figure. We can plot more than one plot on the same figure. It can be achieved by plotting all the lines before calling show(). It can be done as follows:-

In [78]:
```
x4 = range(1,5)

plt.plot(x4, [xi*1 for xi in x4])
plt.plot(x4, [xi*3 for xi in x4])
plt.plot(x4, [xi/3 for xi in x4])
plt.show()
```



In [80]:
```
fig.savefig('plot1.png')
```

In [85]:
```
from IPython.display import Image

Image('plot1.png')
```

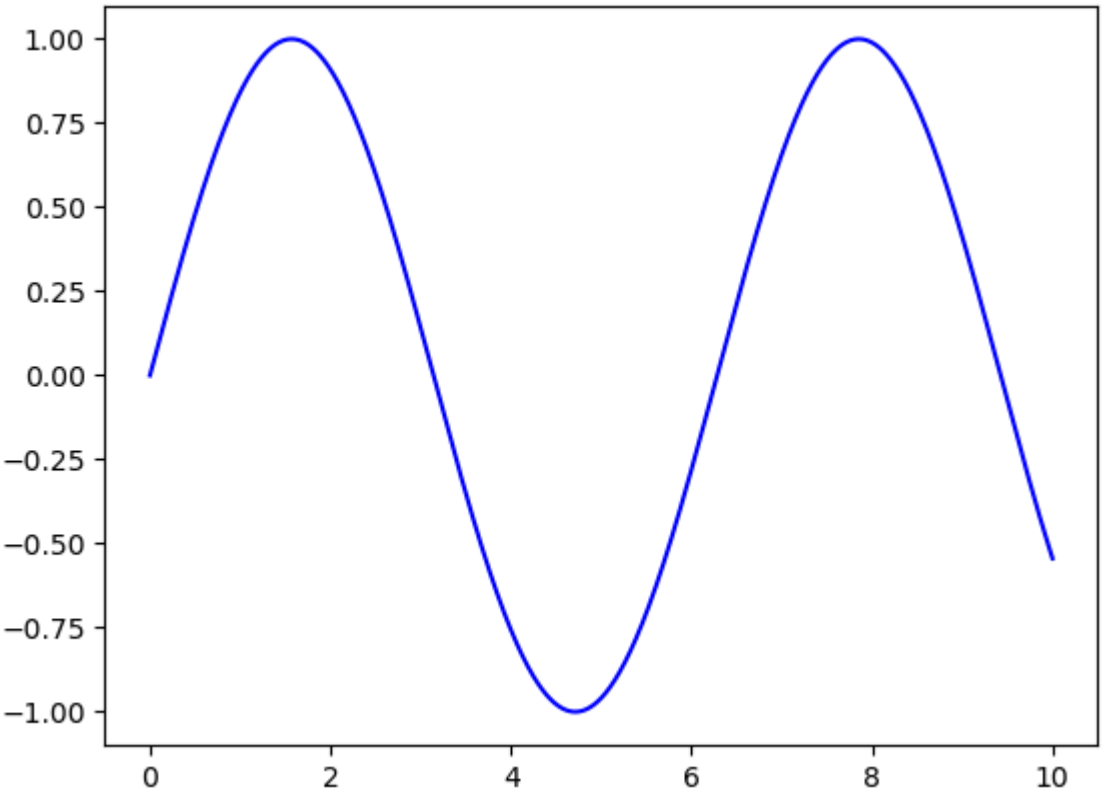
Out[85]:

In [86]:
```
fig.canvas.get_supported_filetypes()
```

Out[86]:
```
{'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'}
```

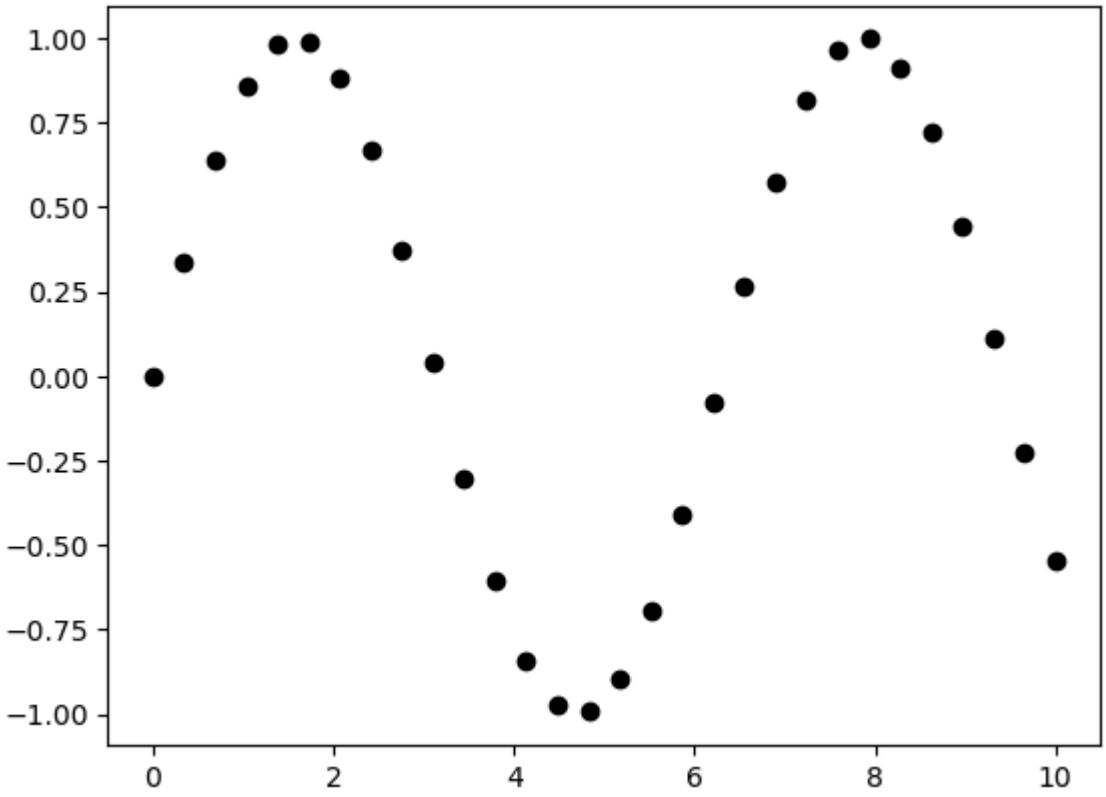
line plot

In [90]:
```
#creating figure and axes first
fig = plt.figure()
ax = plt.axes()

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



scatter plot

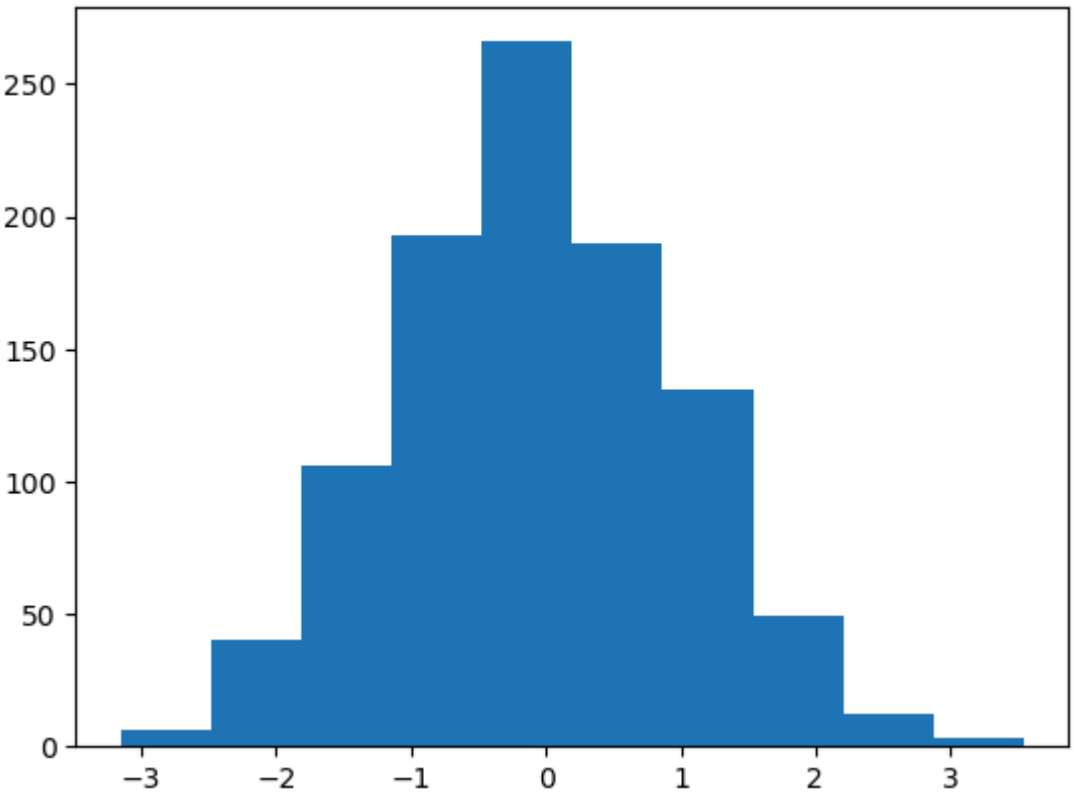
In [99]:
```
x6 = np.linspace(0,10,30)
plt.plot(x6,np.sin(x6),'o',color='black',)
plt.show()
```



histogram

In [101…

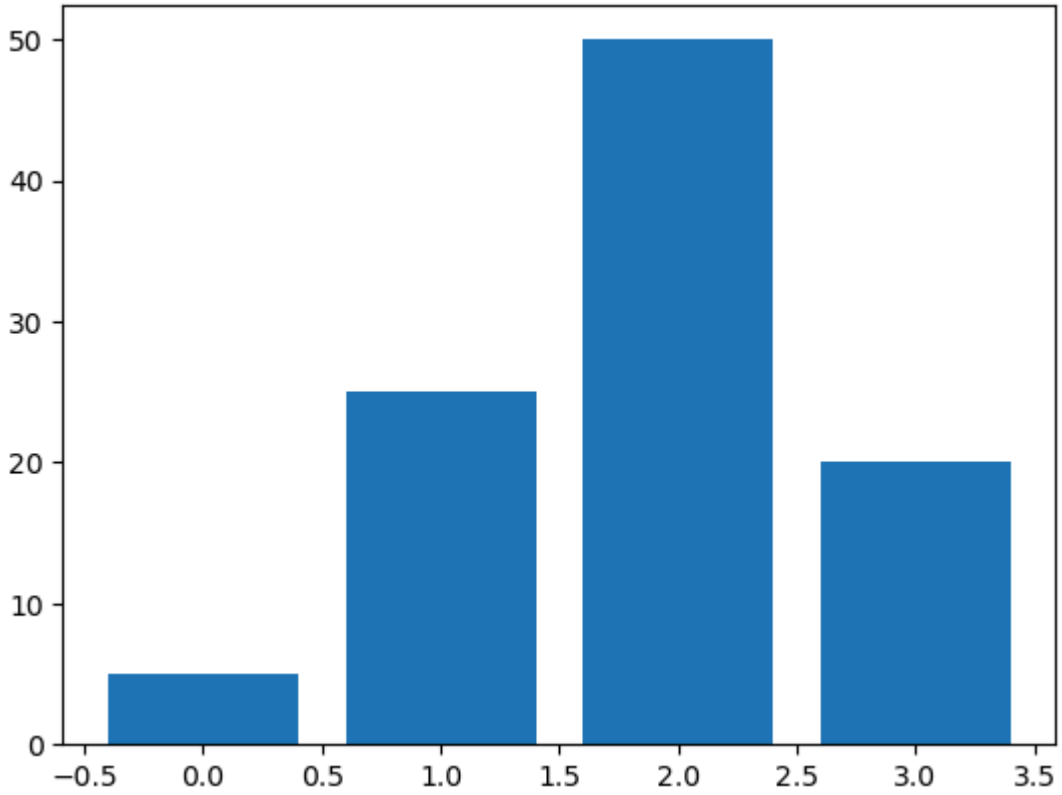
```
data1 = np.random.randn(1000)
plt.hist(data1);
plt.show()
```



bar chart

In [110…

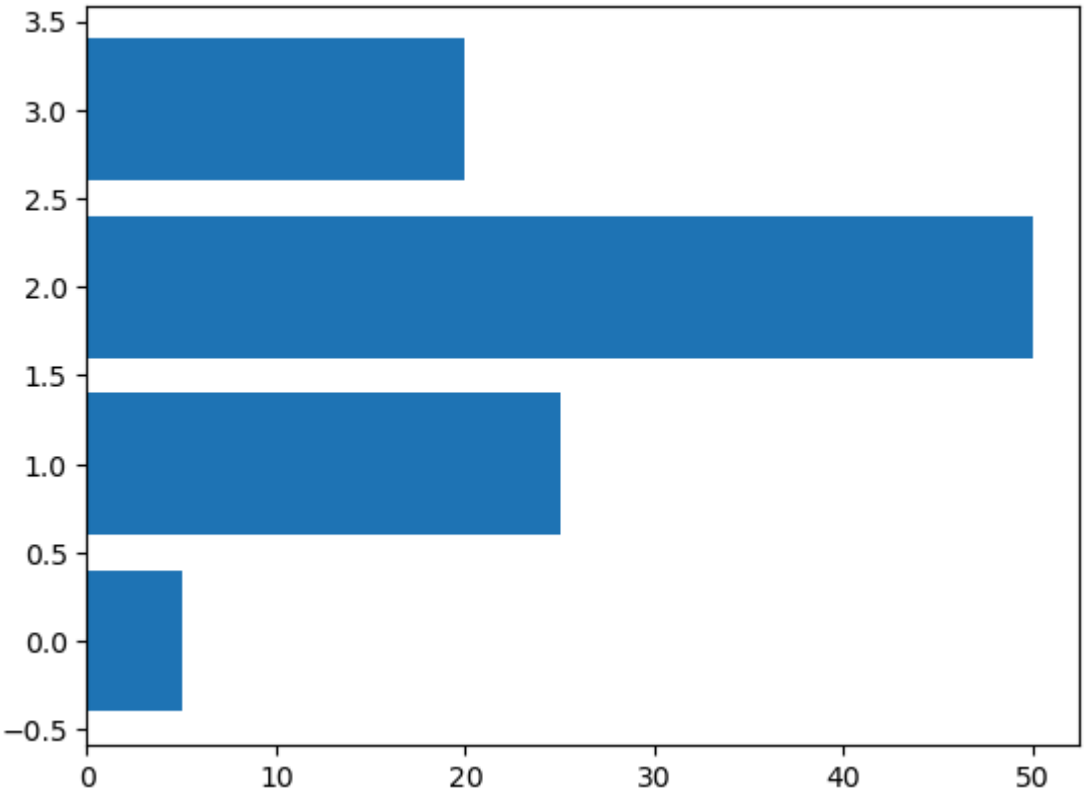
```
data2 = [5.,25.,50.,20.]
plt.bar(range(len(data2)),data2)
plt.show()
```



horizontal barchart

In [111…

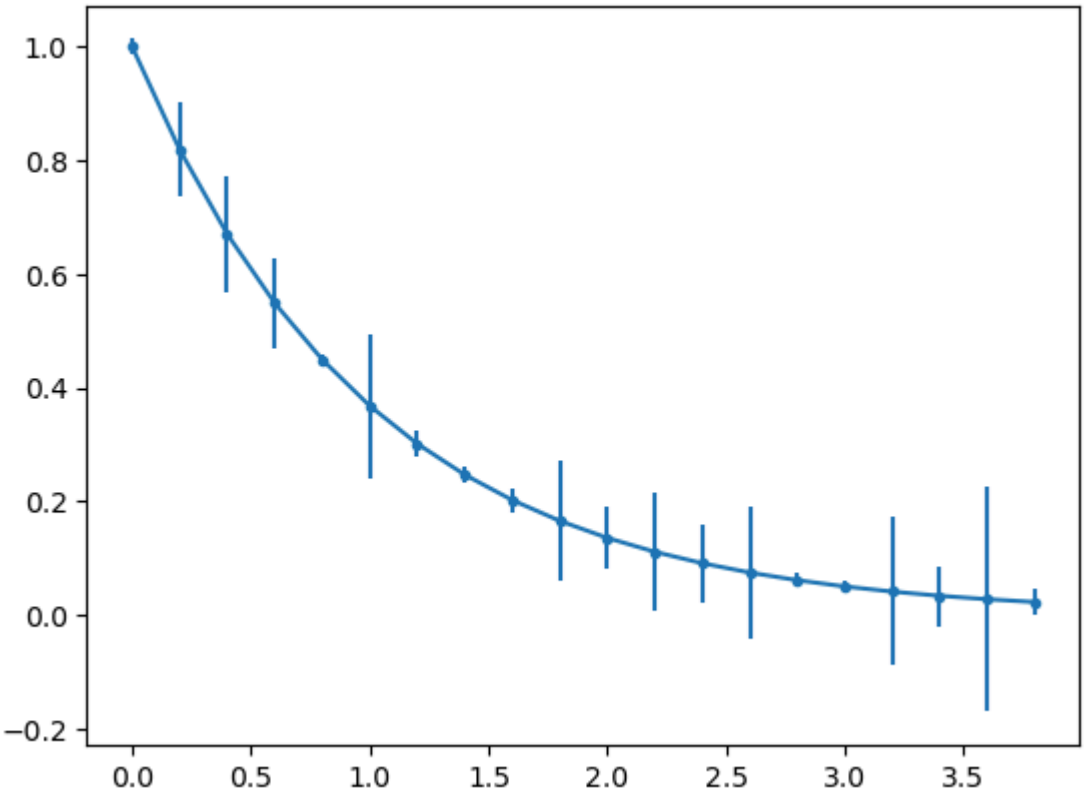
```
data2 = [5.,25.,50.,20.]
plt.barh(range(len(data2)),data2)
plt.show()
```



error bar chart

In [117…

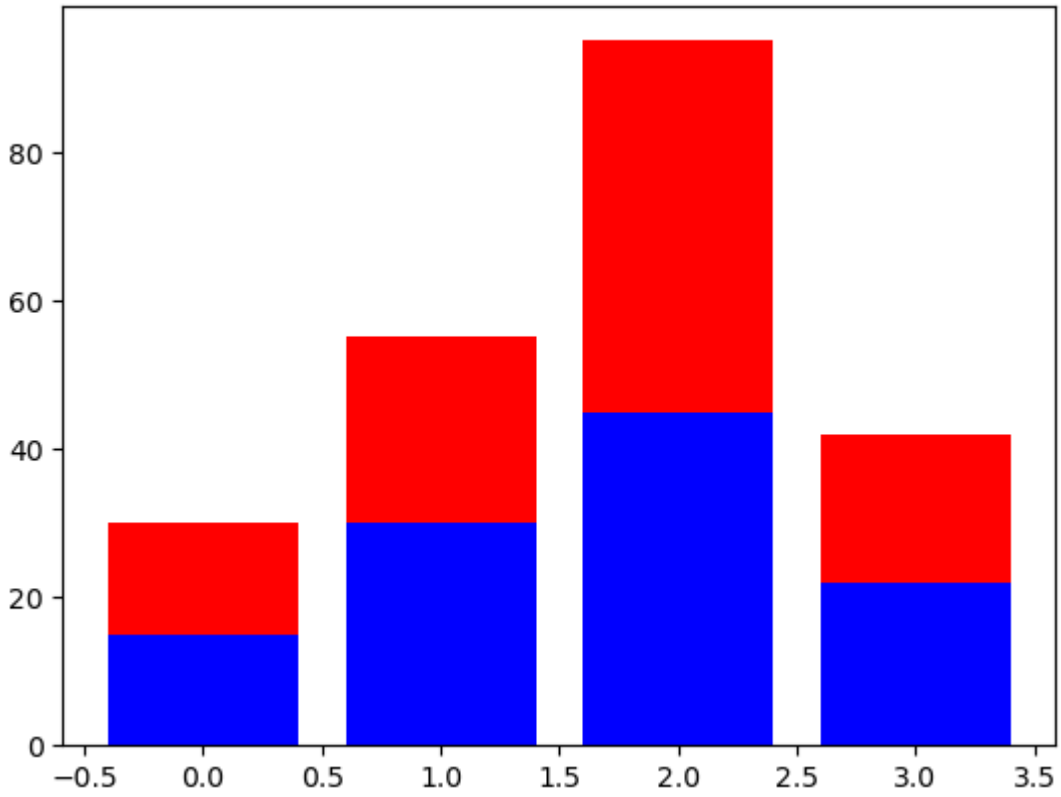
```
x7 = np.arange(0, 4, 0.2)
y7 = np.exp(-x7)

e1 = 0.1 * np.abs(np.random.randn(len(y7)))

plt.errorbar(x7, y7, yerr = e1,fmt='.-')

plt.show();
```



stacked bar chart

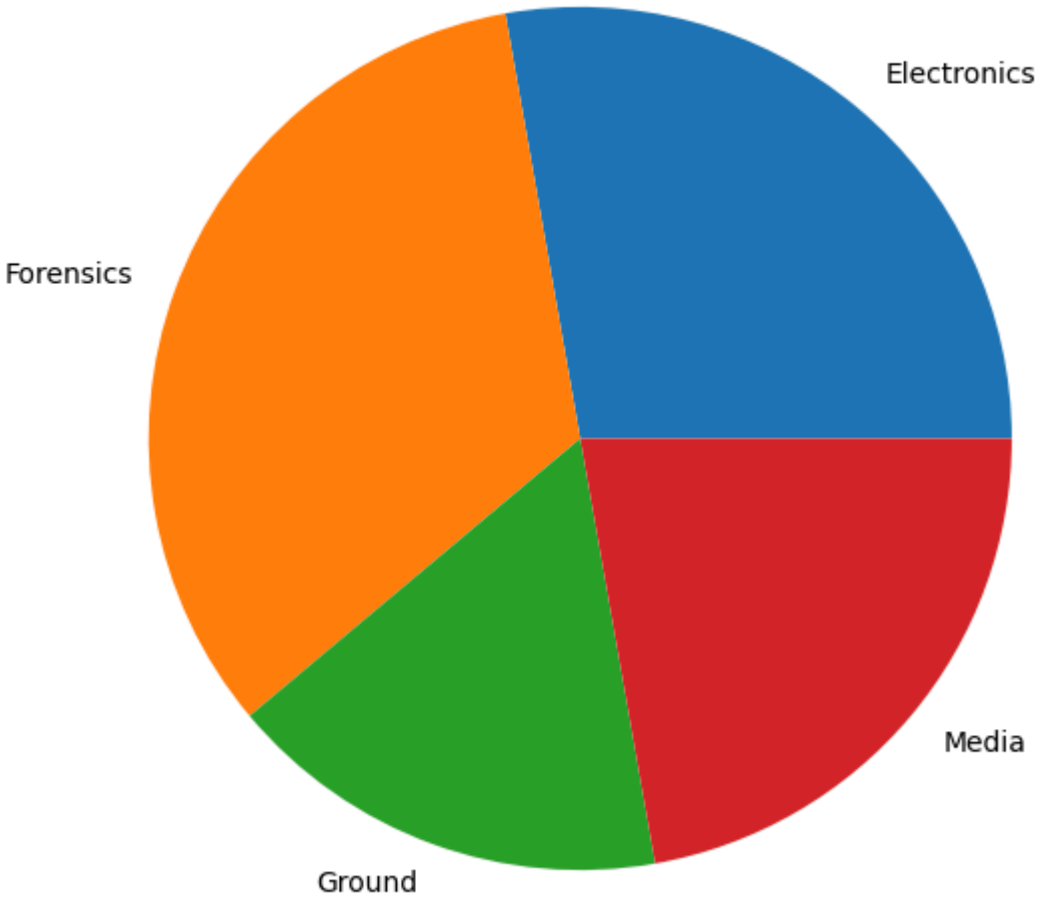
In [119…

```
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) #we use 'bottom' parameter for stack

plt.show()
```



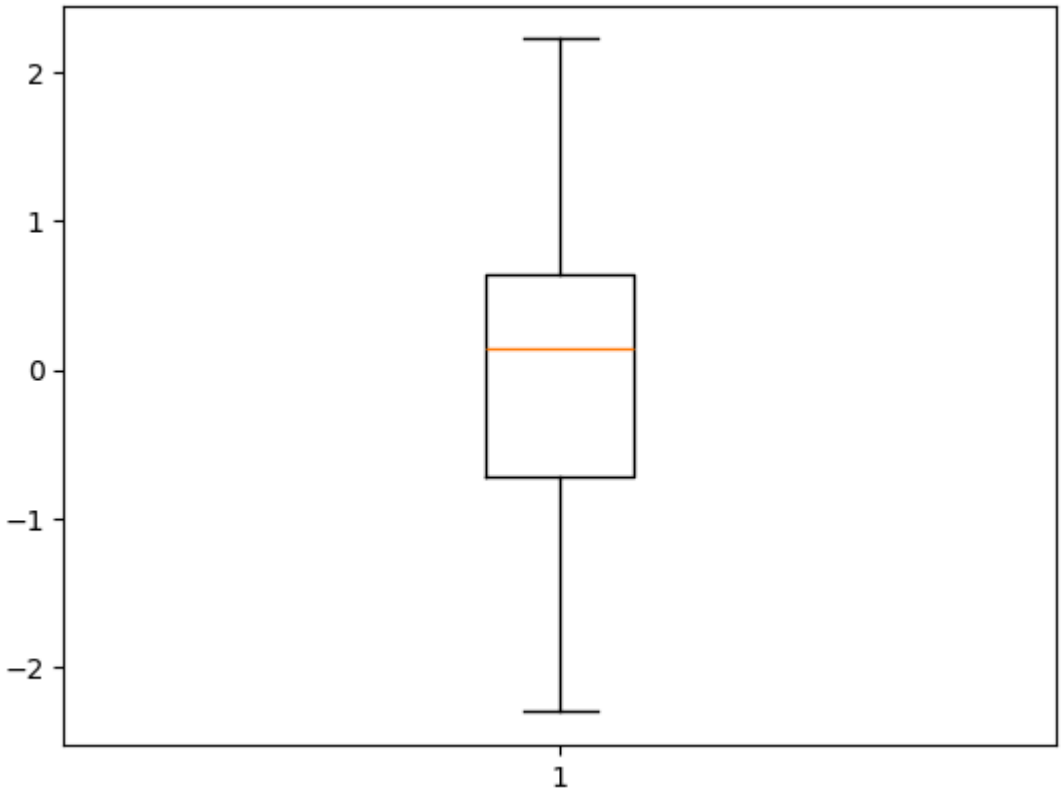
pie chart

In [4]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(7,7))
x10 = [25,30,15,20]
labels = ['Electronics','Forensics','Ground','Media']
plt.pie(x10,labels=labels);

plt.show()
```



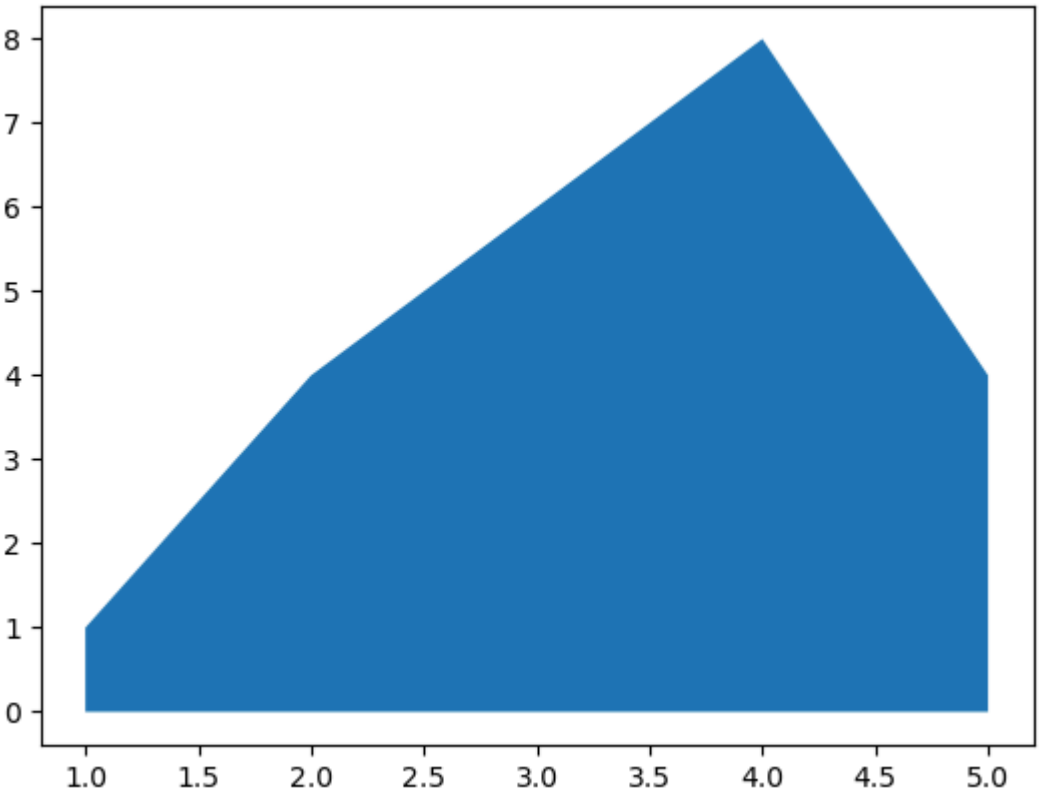
box plot

In [8]:
```
import numpy as np
data3 = np.random.randn(100)
plt.boxplot(data3)
plt.show()
```



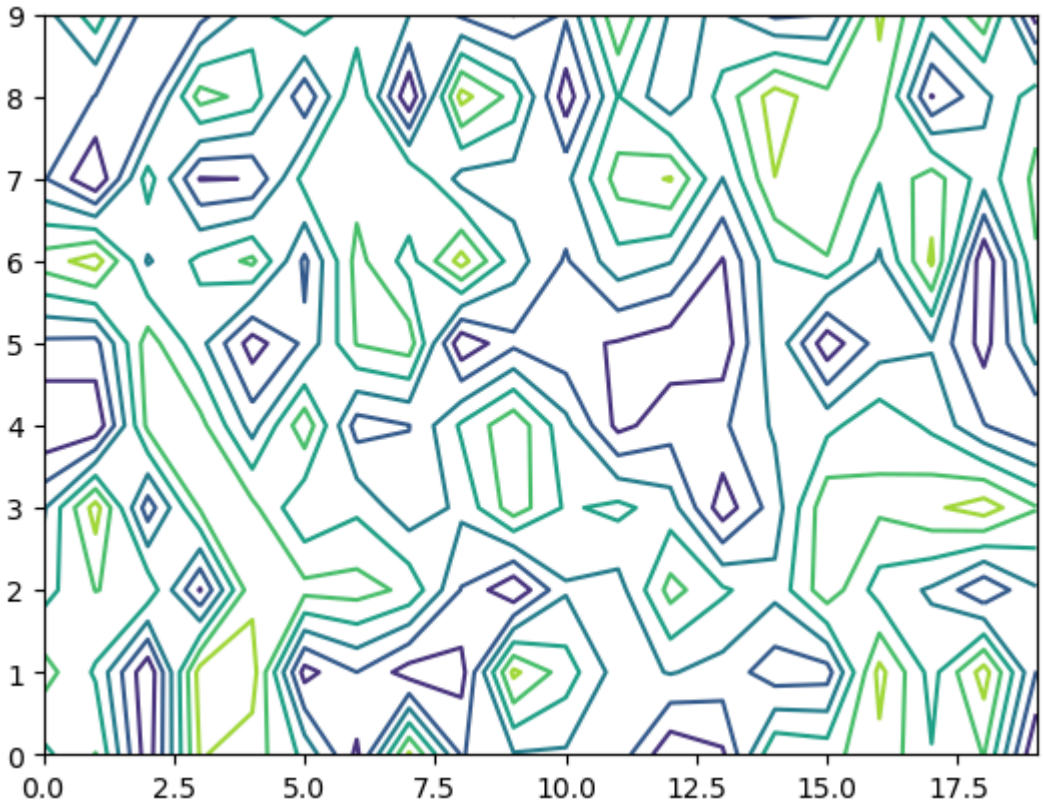
area chart

In [10]:
```
x12 = range(1,6)
y12 = [1,4,6,8,4]

plt.fill_between(x12,y12)
plt.show()
```



contour plot

In [12]:
```
matrix1 = np.random.rand(10,20)
cp = plt.contour(matrix1)
plt.show()
```

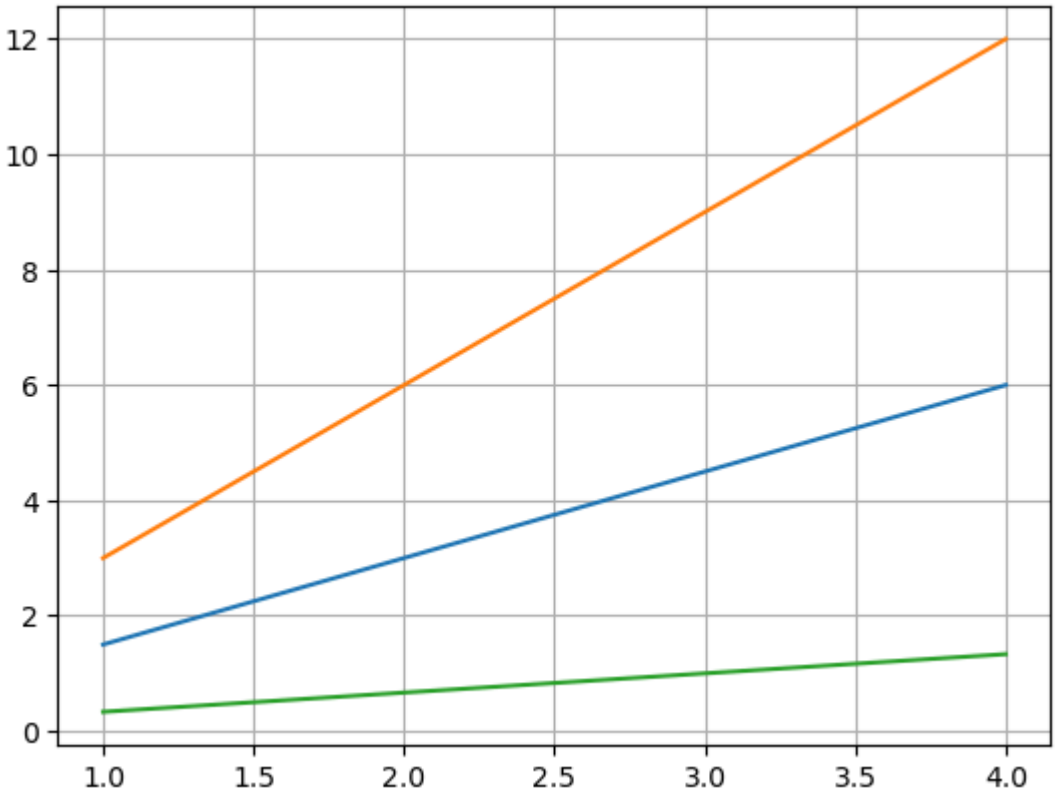


In [14]:
```
print(plt.style.available) #views list of all available styles
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid',
'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'graysc
ale', '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-d
eep', 'seaborn-v0_8-muted', 'seaborn-v0_8-notebook', 'seaborn-v0_8-paper', 'seabo
rn-v0_8-pastel', 'seaborn-v0_8-poster', 'seaborn-v0_8-talk', 'seaborn-v0_8-tick
s', 'seaborn-v0_8-white', 'seaborn-v0_8-whitegrid', 'tableau-colorblind10']
```

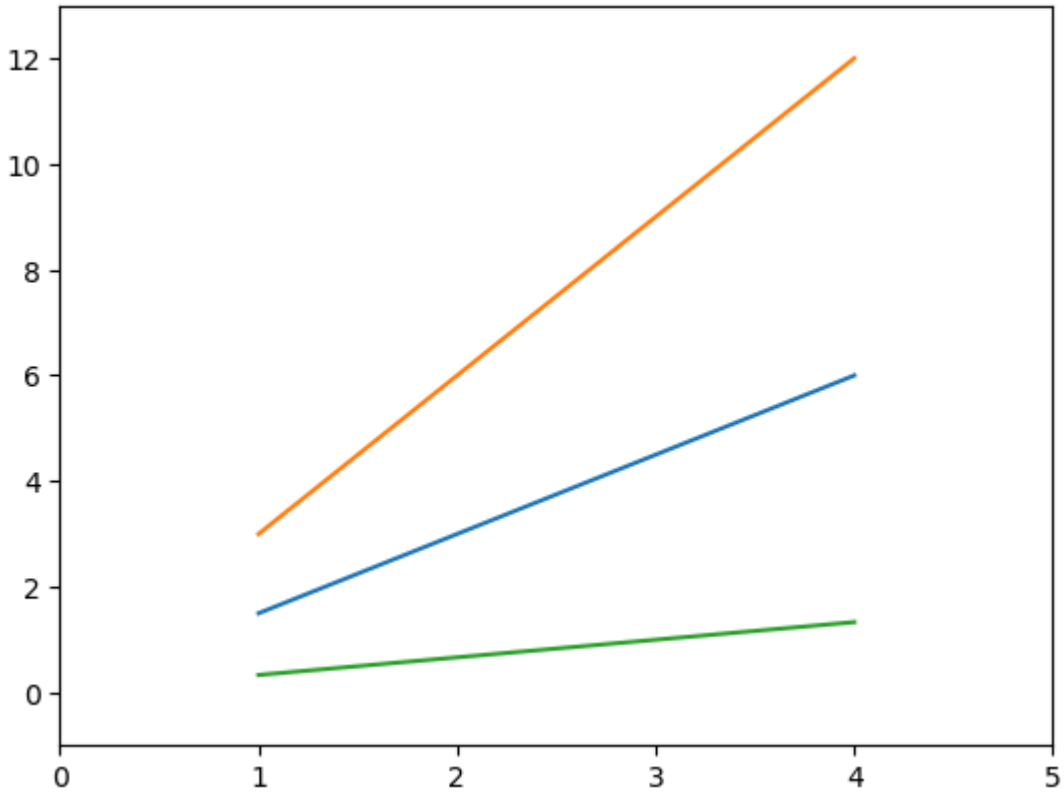
adding a grid

In [16]:
```
x15 = np.arange(1, 5)

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

plt.grid(True)

plt.show()
```



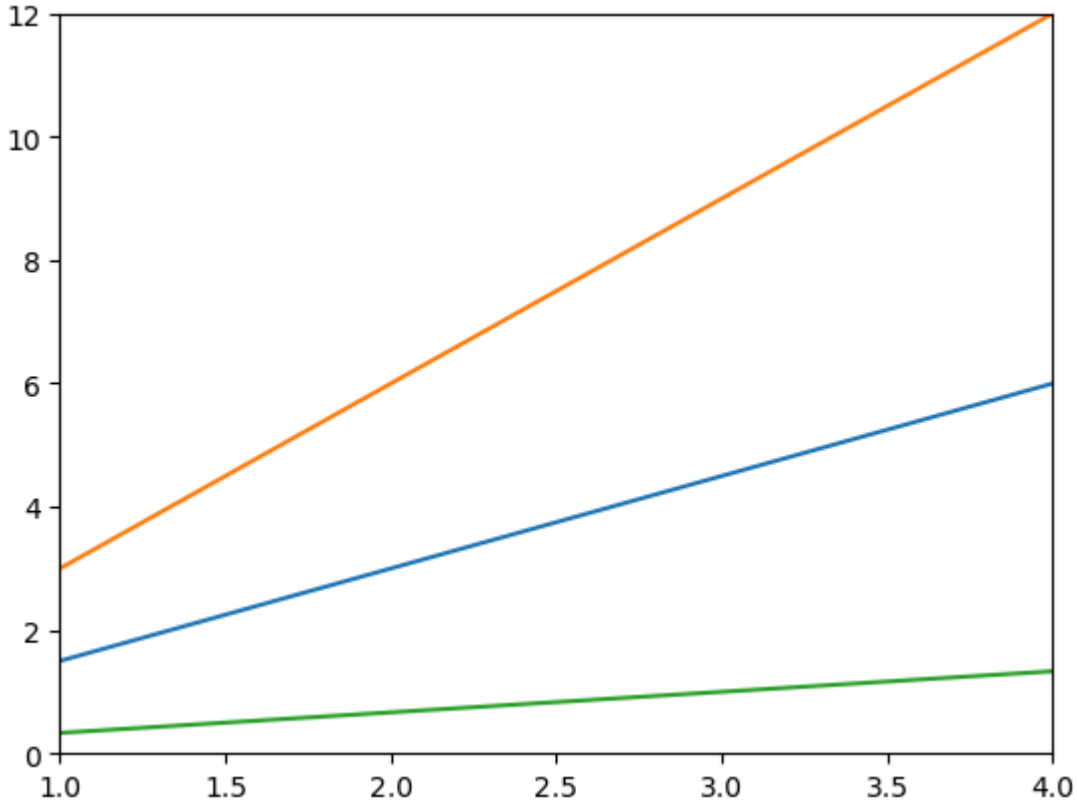
handling axes

In [17]:
```
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()
```

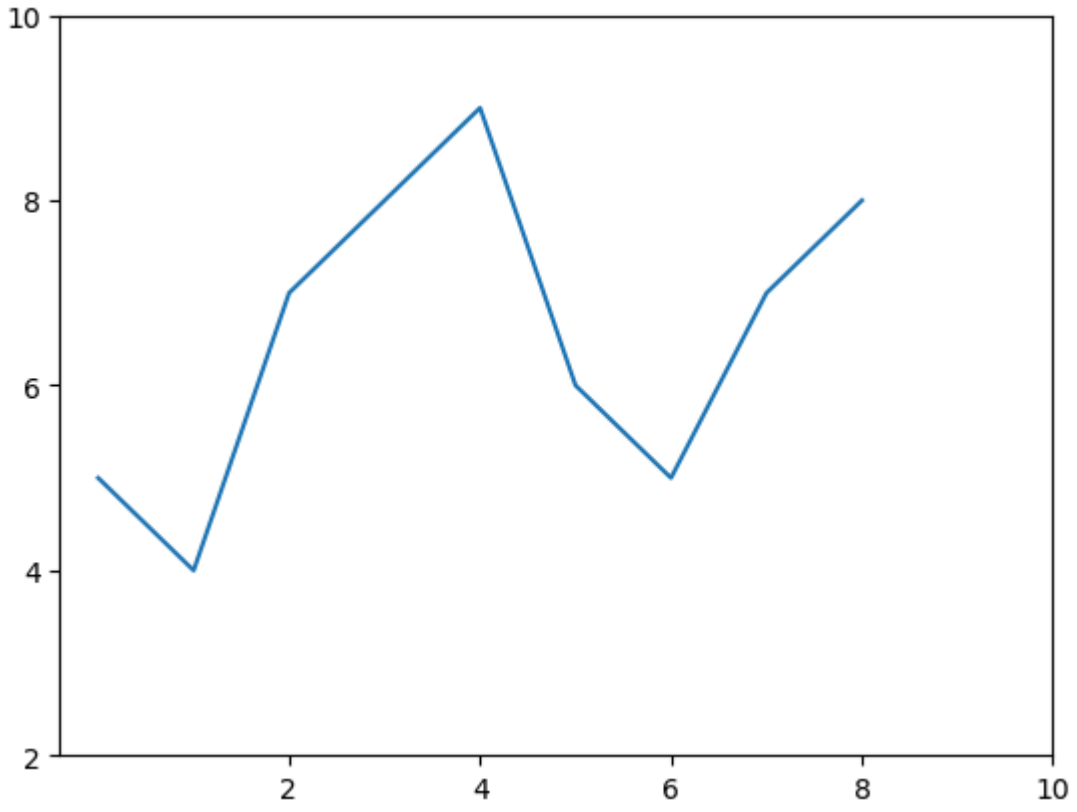


In [18]:
```
#We can control the limits for each axis separately using the xlim() and ylim()

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[18]: (0.0, 12.0)

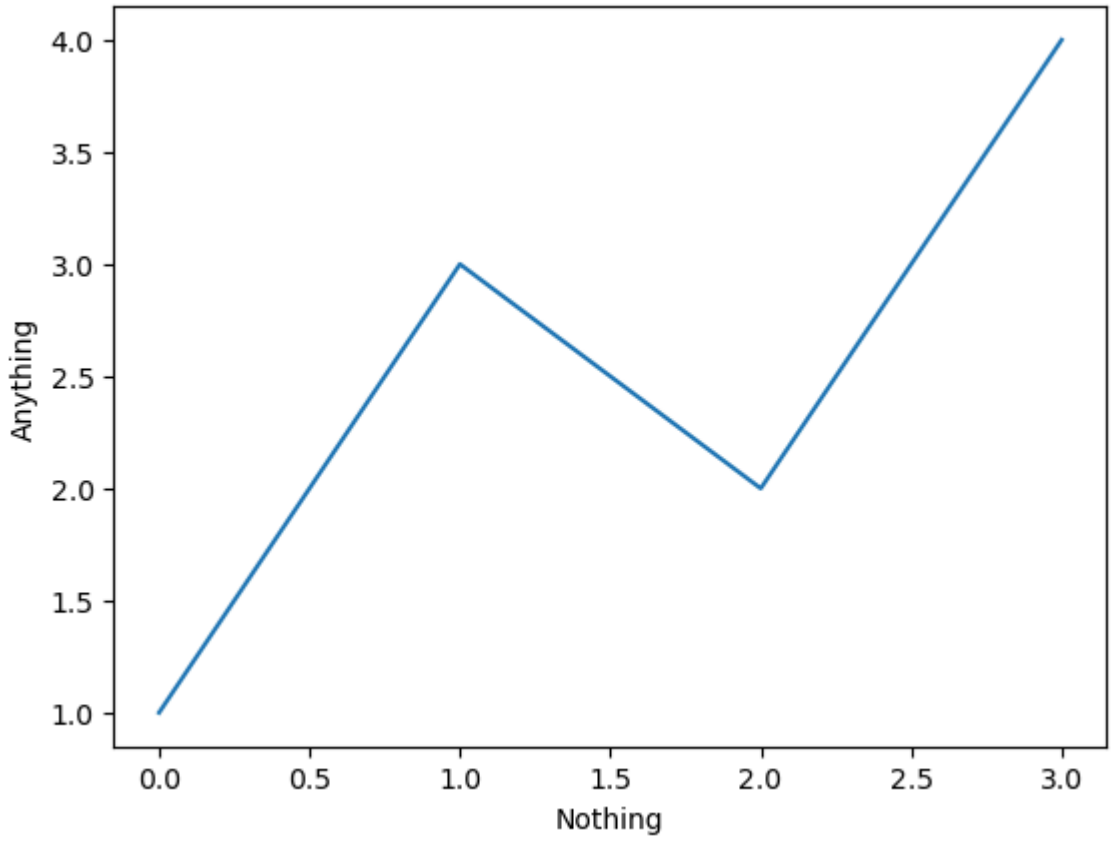


In [19]:
```
u = [5,4,7,8,9,6,5,7,8]
plt.plot(u)

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

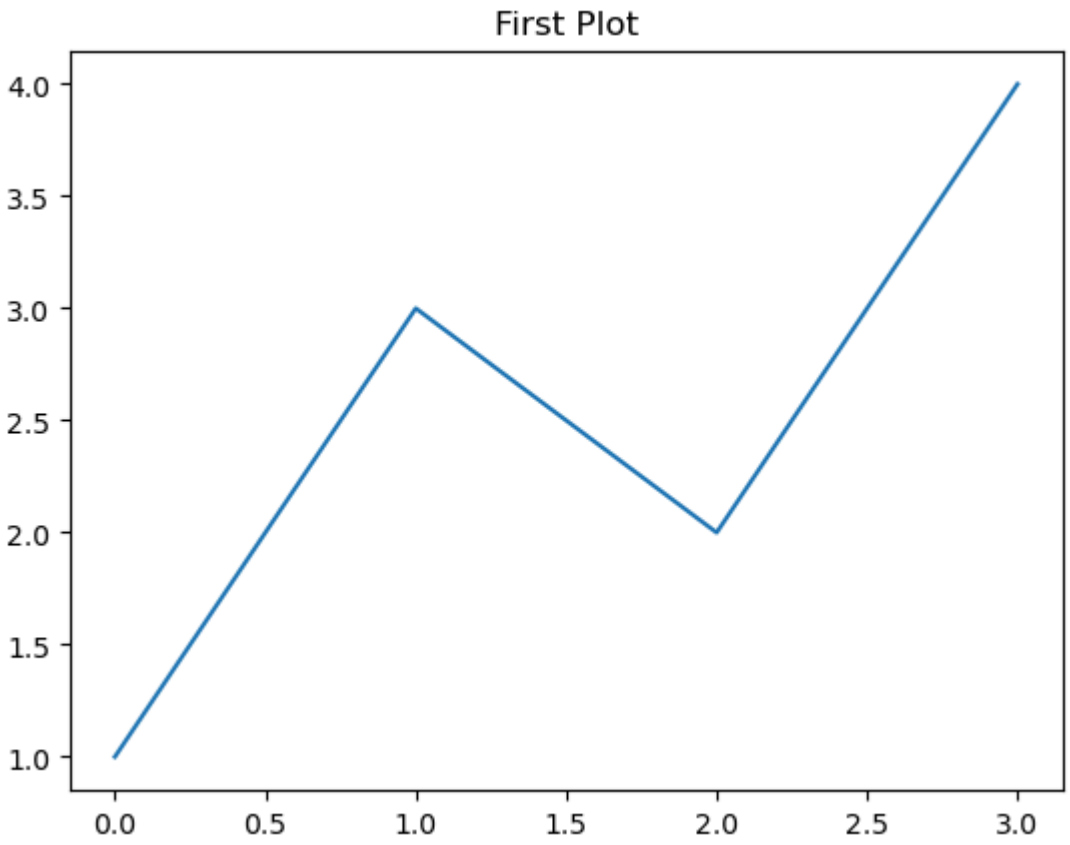


In [23]:
```
#adding labels
import matplotlib.pyplot as plt
```

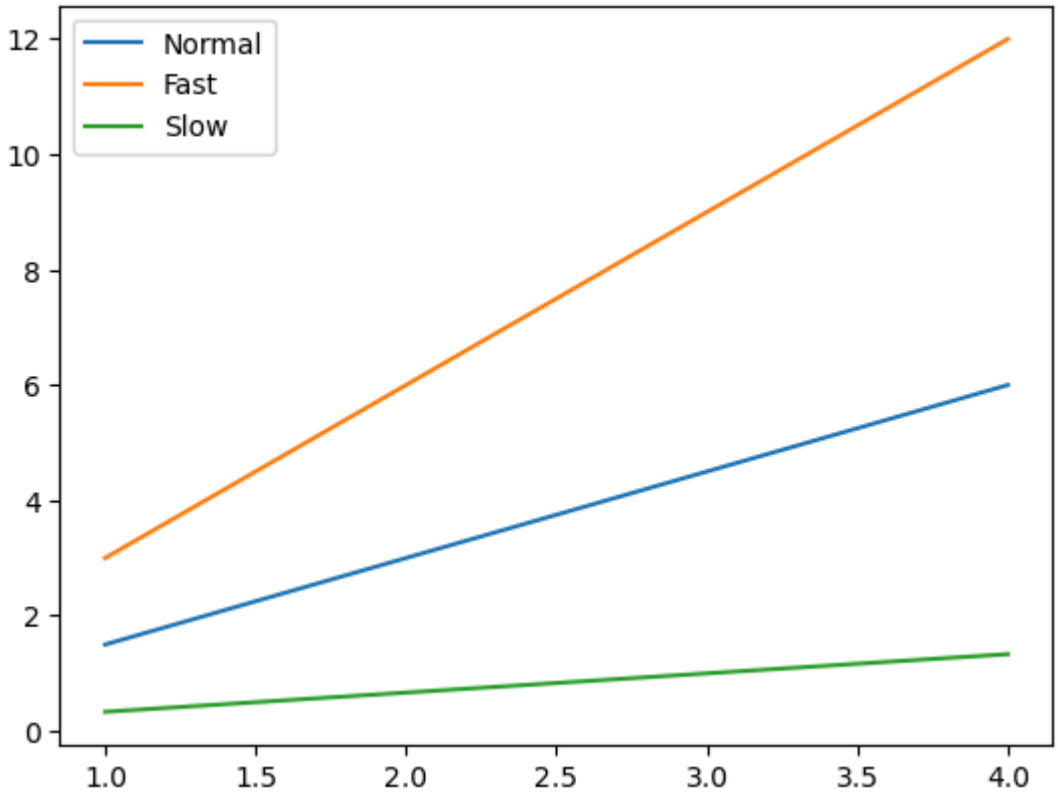
```
plt.plot([1,3,2,4])
plt.xlabel('Nothing')
plt.ylabel('Anything')

plt.show()
```



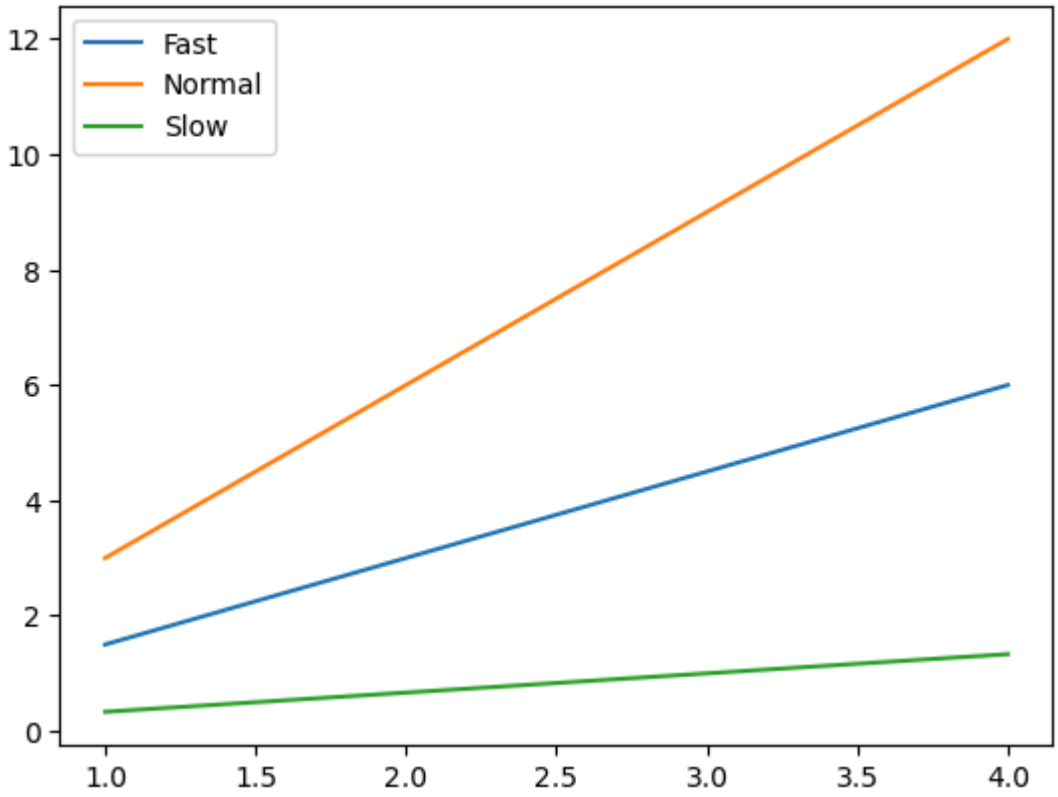
In [24]:
```
plt.plot([1, 3, 2, 4])

plt.title('First Plot') #adding title

plt.show()
```



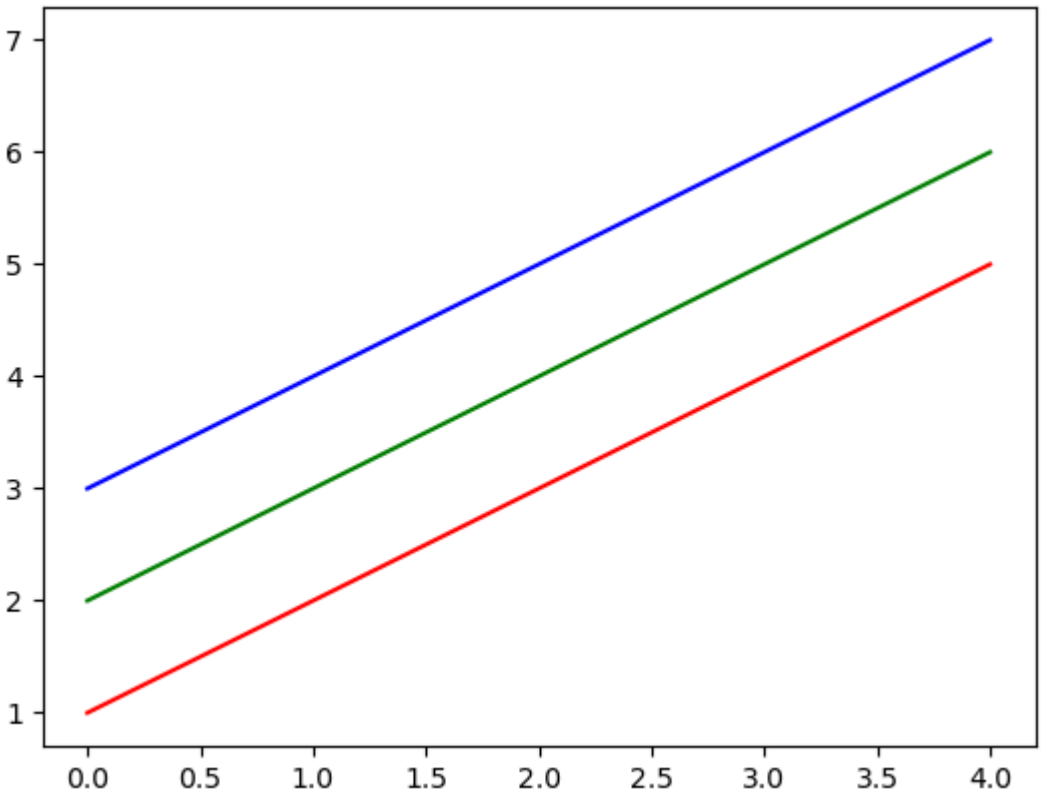
In [25]:
```
#adding legends
x15 = np.arange(1, 5)

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']);
```



In [27]:
```
x15 = np.arange(1, 5)

fig, ax = plt.subplots()

ax.plot(x15, x15*1.5, label='Fast')
ax.plot(x15, x15*3.0,label='Normal')
ax.plot(x15, x15/3.0, label='Slow')

ax.legend();
plt.show()
```

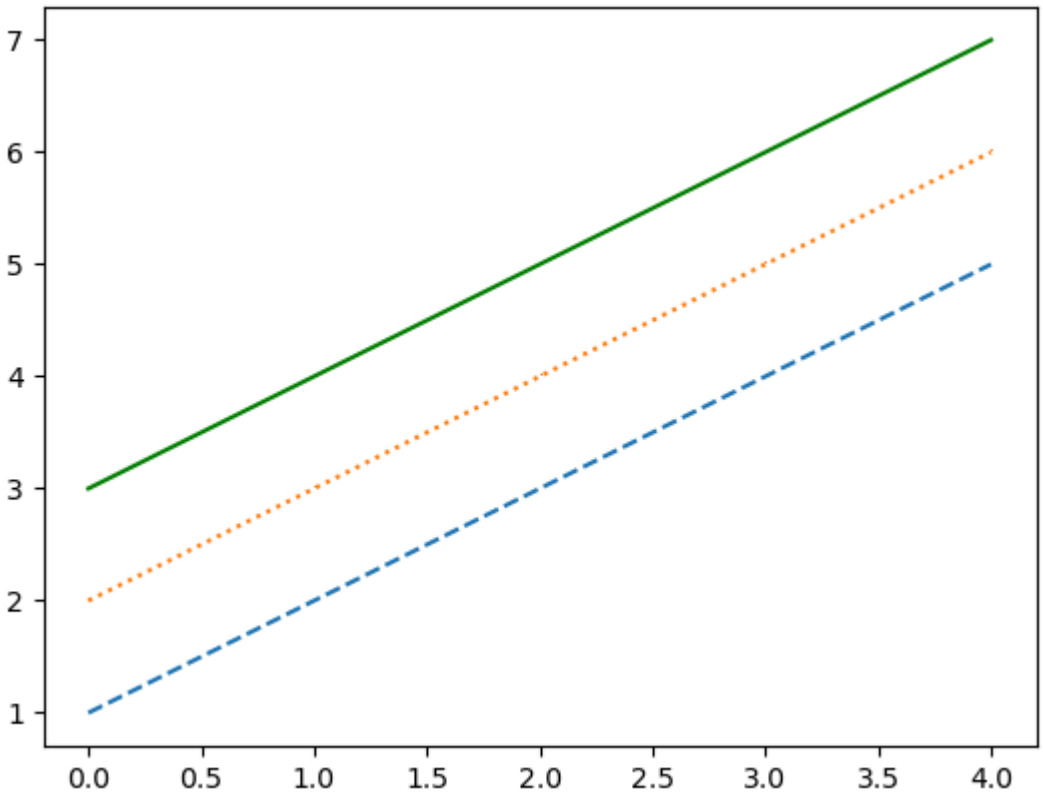


In [29]:
```
x16 = np.arange(1,6)
plt.plot(x16,'r')
plt.plot(x16+1,'g')
plt.plot(x16+2,'b')
plt.show()
```



In [31]:
```
x17 = np.arange(1,5)
plt.plot(x16,'--',x16+1,':,',x16+2,'g')
```

```
plt.show()
```



In [ ]: