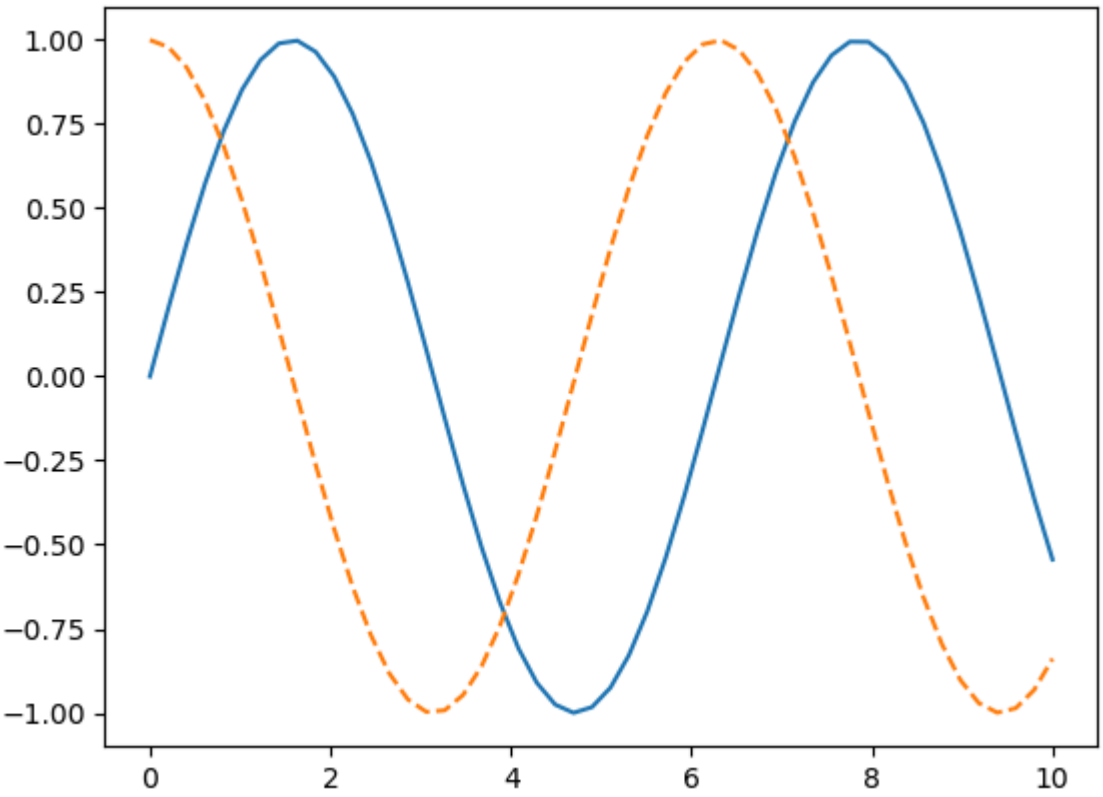
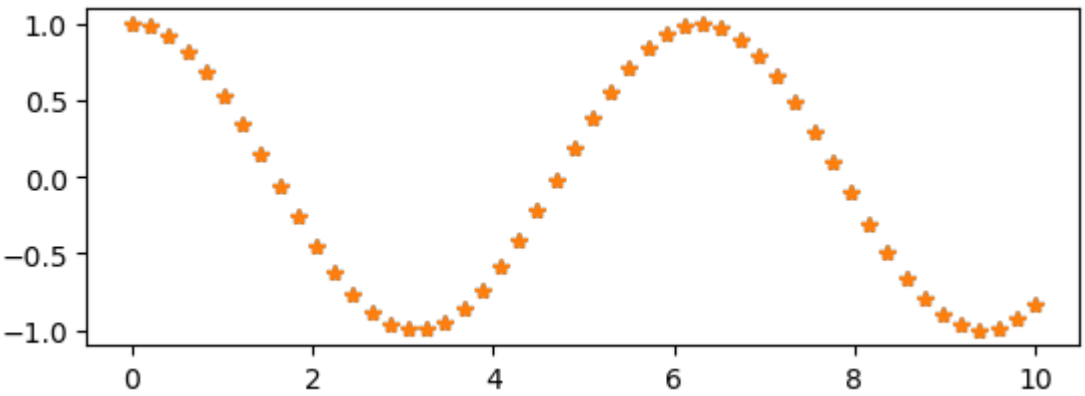
In [1]:
```
import numpy as np
```

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

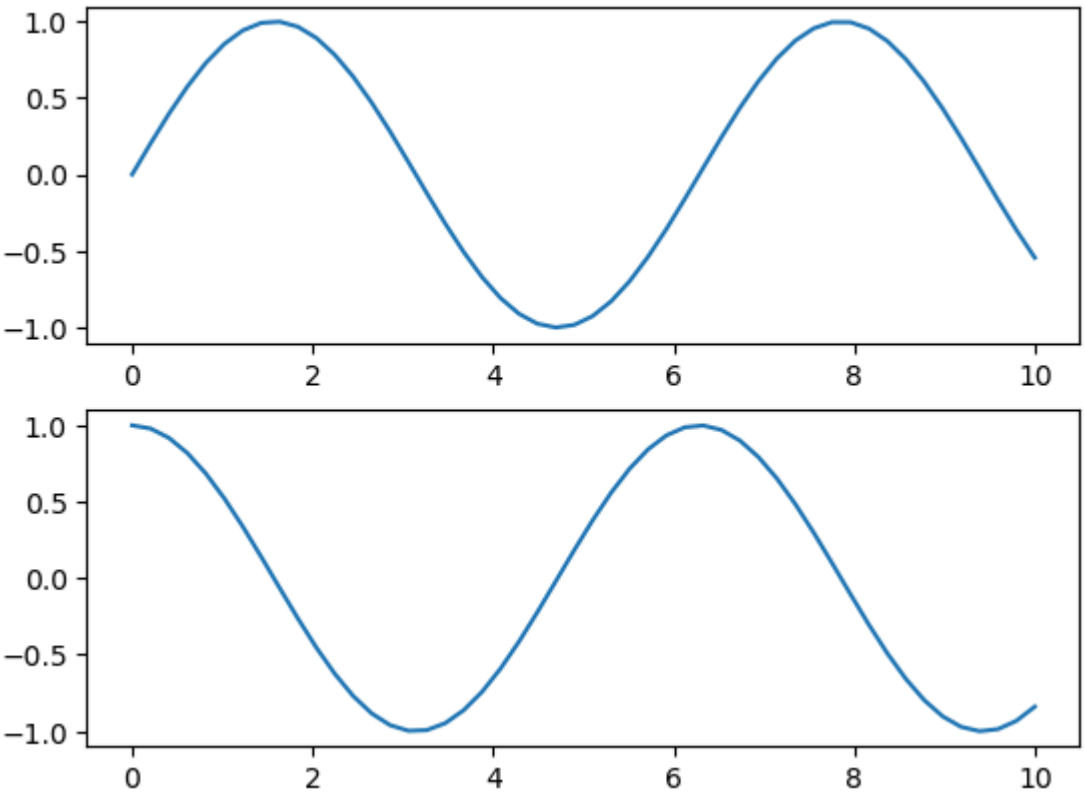
In [3]:
```
%matplotlib inline
x1 = np.linspace(0, 10, 50)

# 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()
```

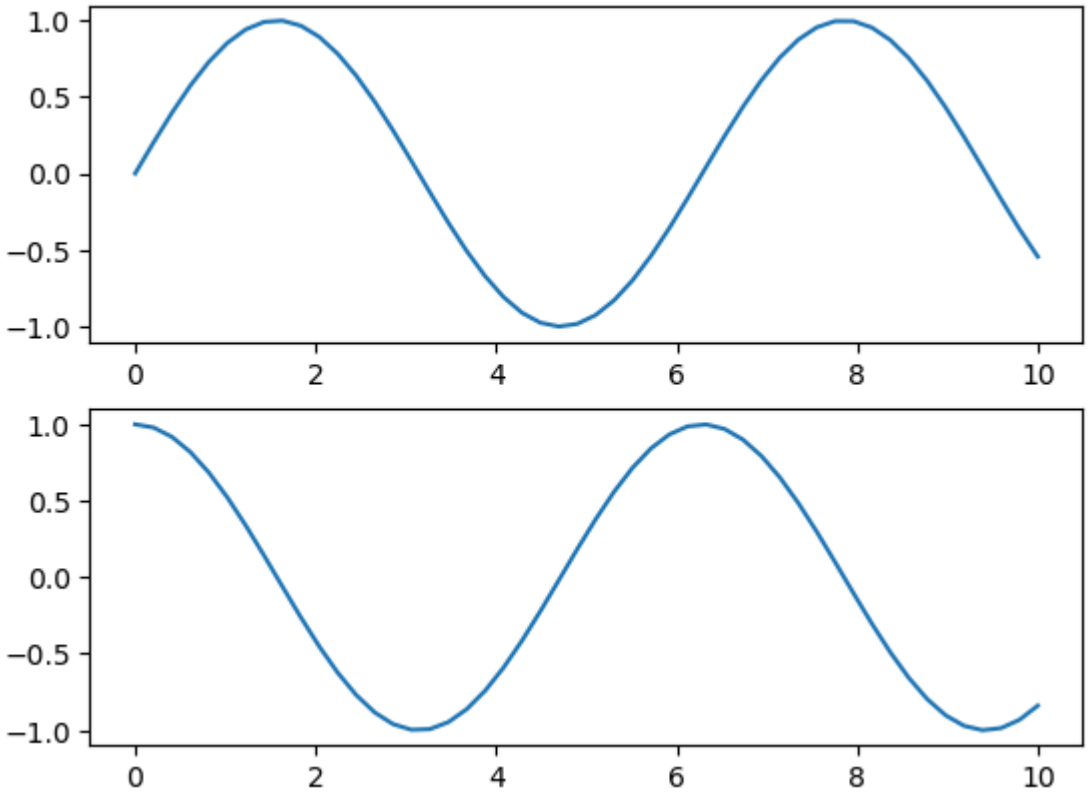


In [5]:
```
# create the first of two panels and set current axis
plt.subplot(2, 1, 1)   # (rows, columns, panel number)
plt.plot(x1, np.cos(x1), '*')
plt.show()
```



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



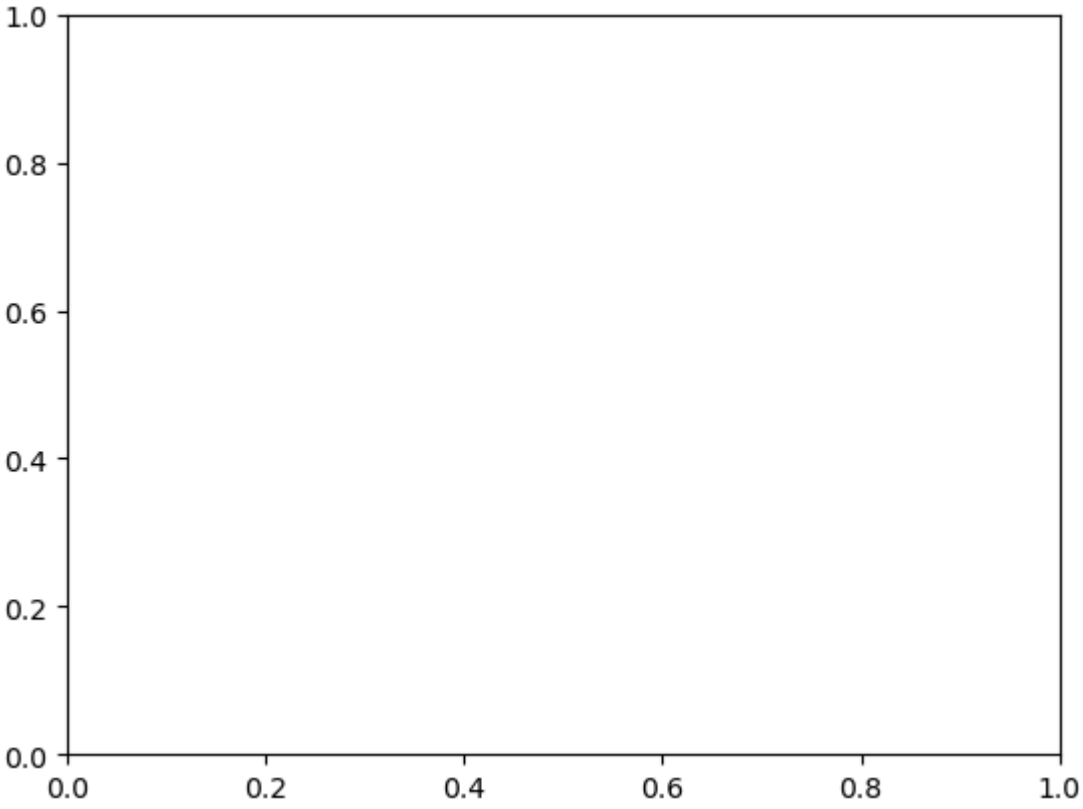


In [8]:
```
print(plt.gcf())
```

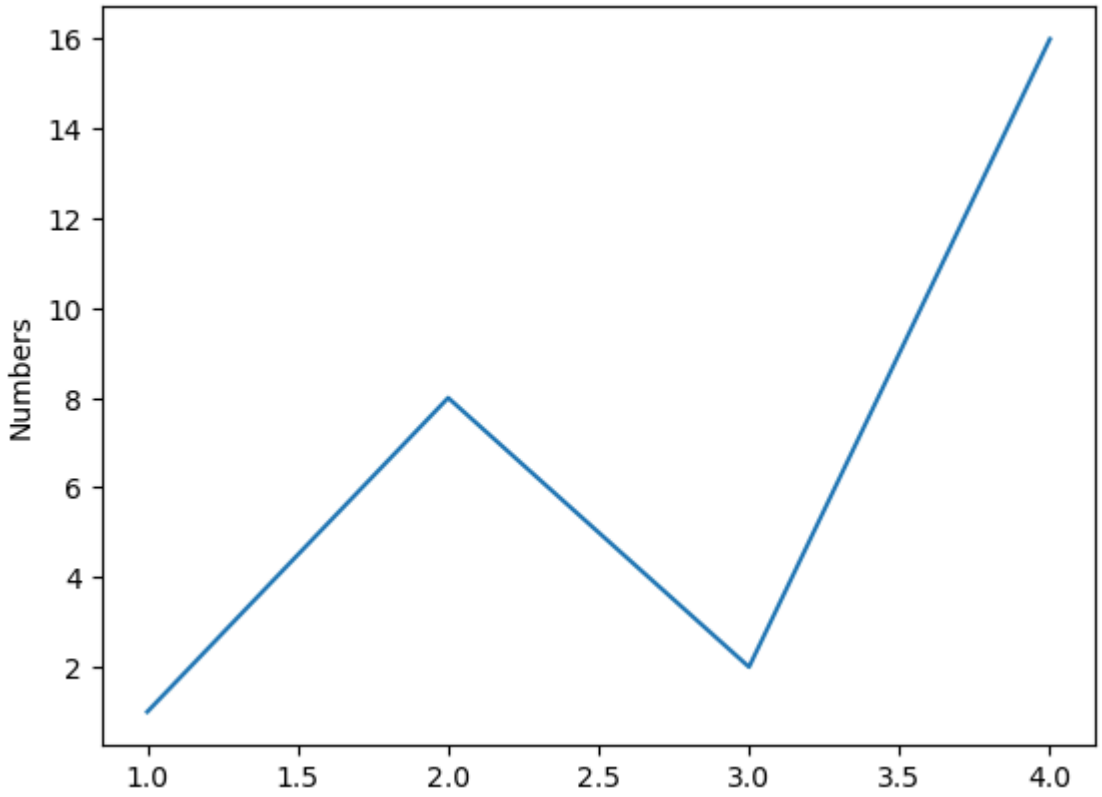
```
Figure(640x480)
```

In [10]:
```
# get current axis information

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

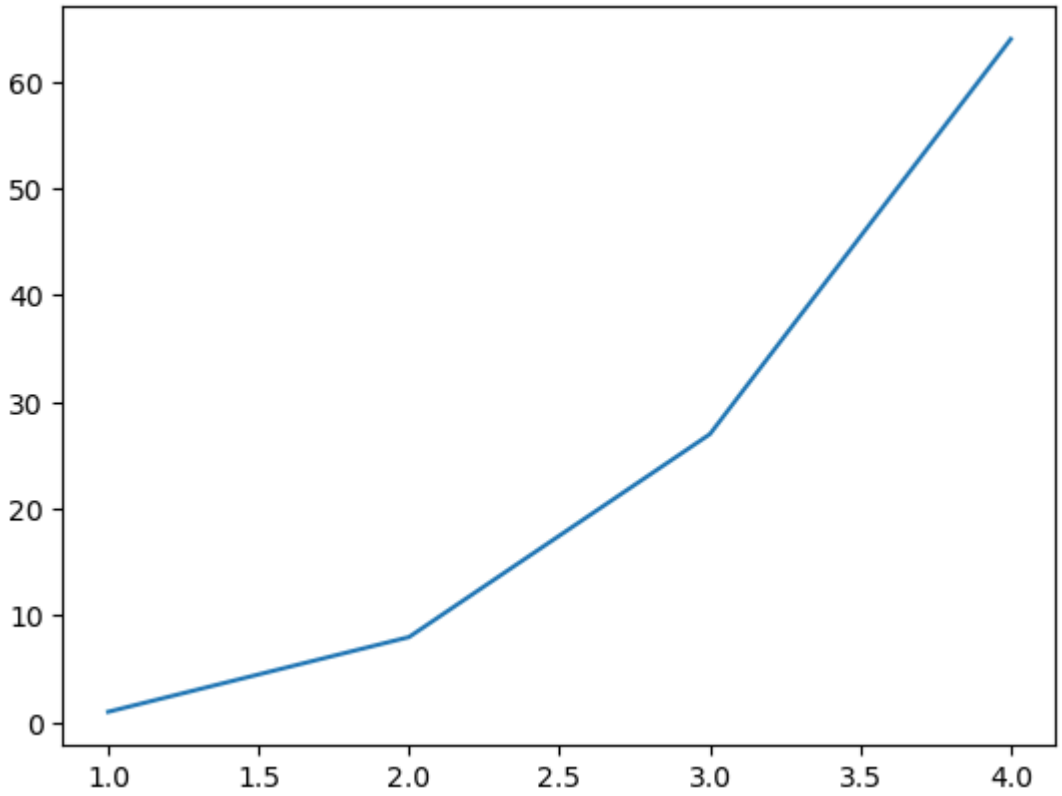
```
Axes(0.125,0.11;0.775x0.77)
```



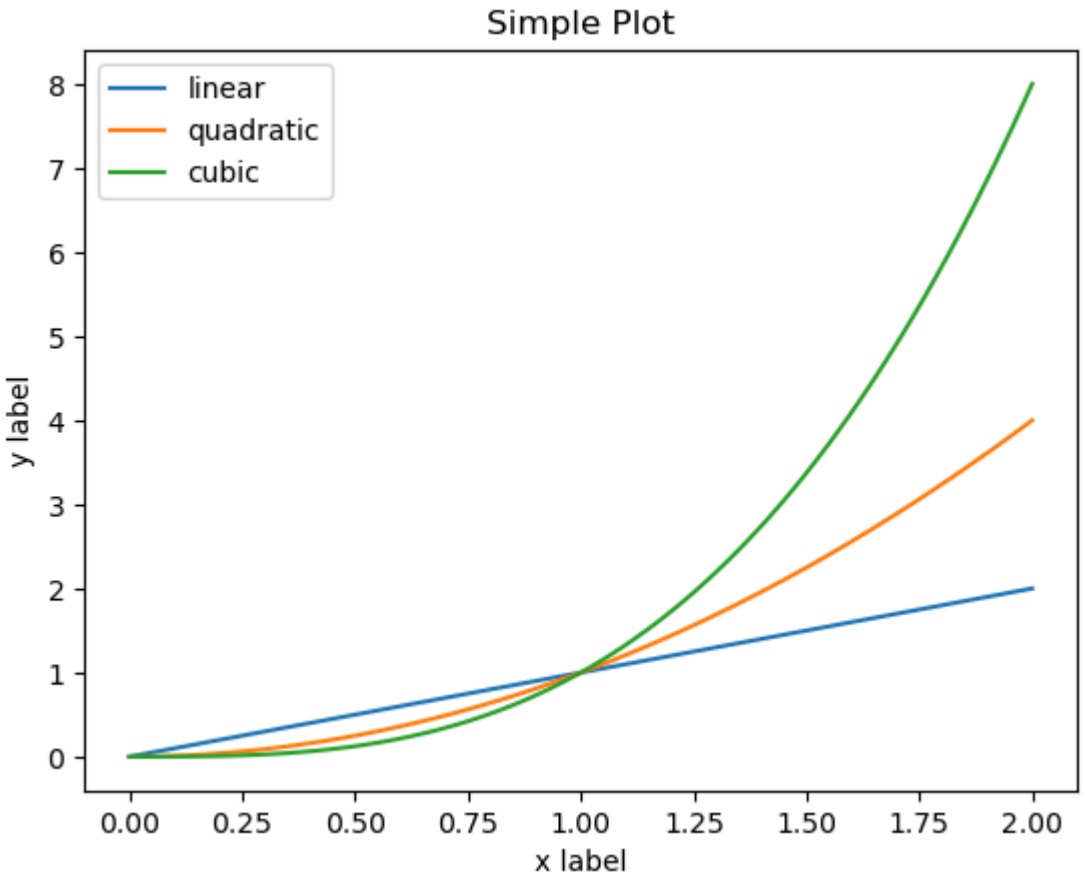
In [11]:
```
plt.plot([1,2,3,4], [1,8,2,16])
plt.ylabel('Numbers')
plt.show()
```



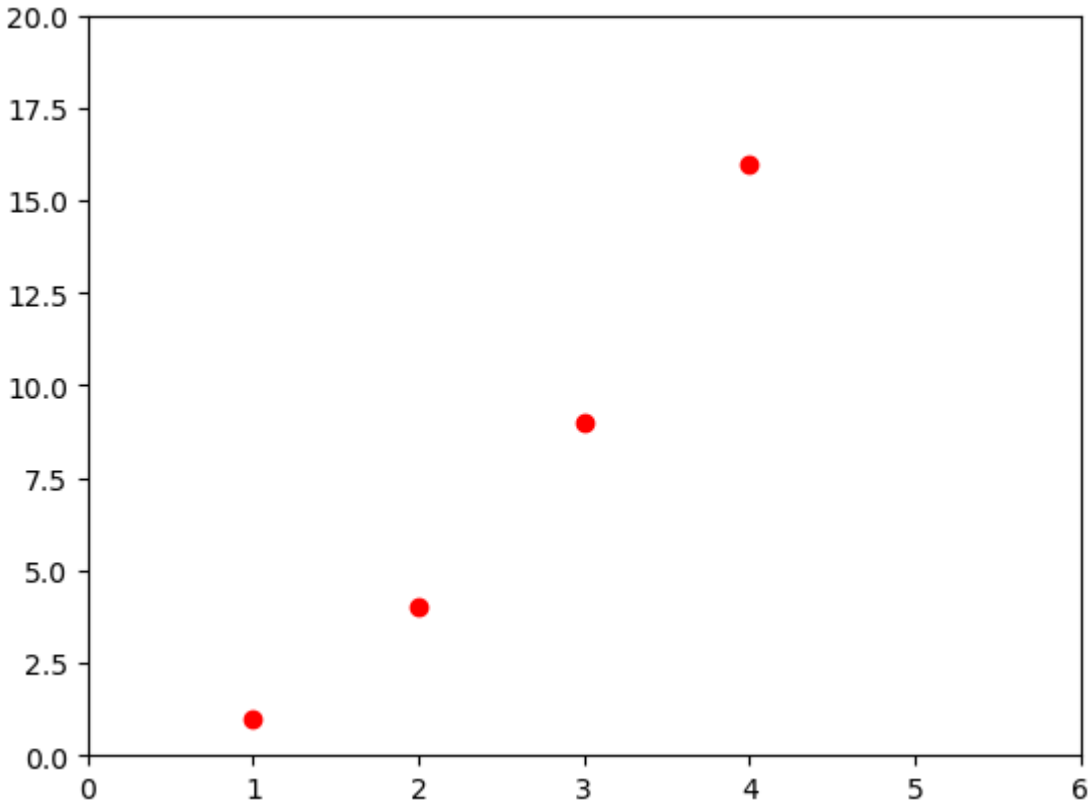
In [12]:
```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 8, 27, 64])
plt.show()
```



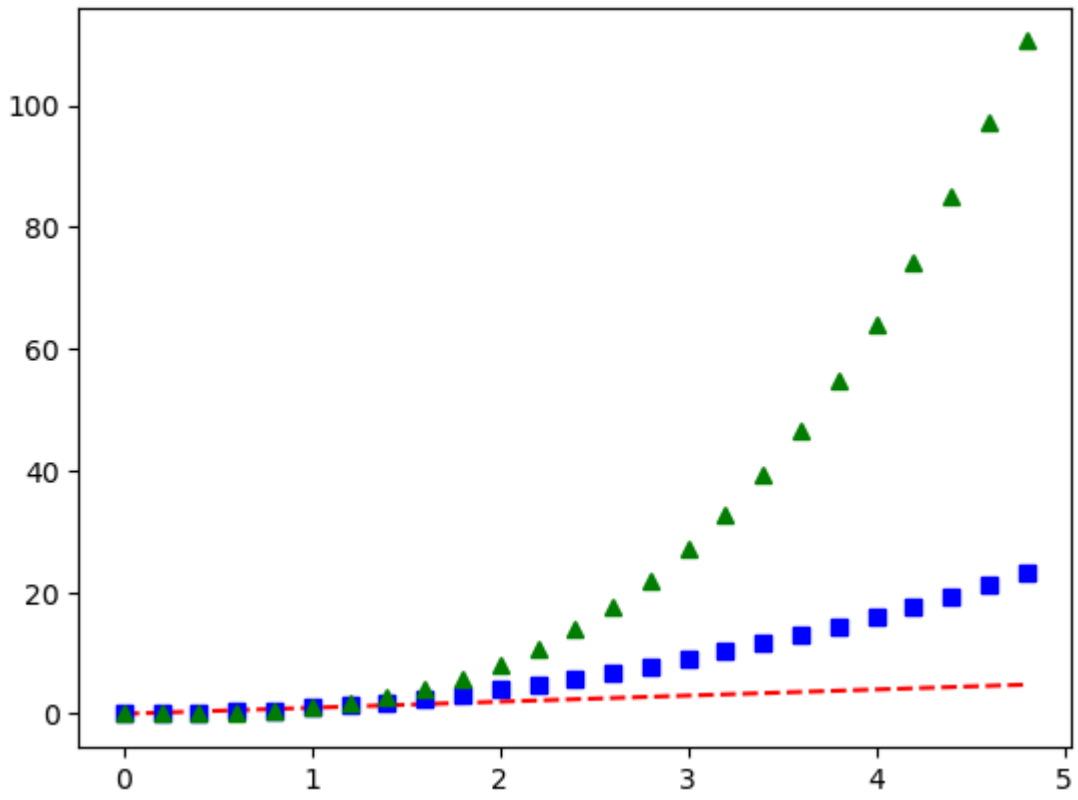
In [13]:
```
x = np.linspace(0, 2, 100)

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()
```



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

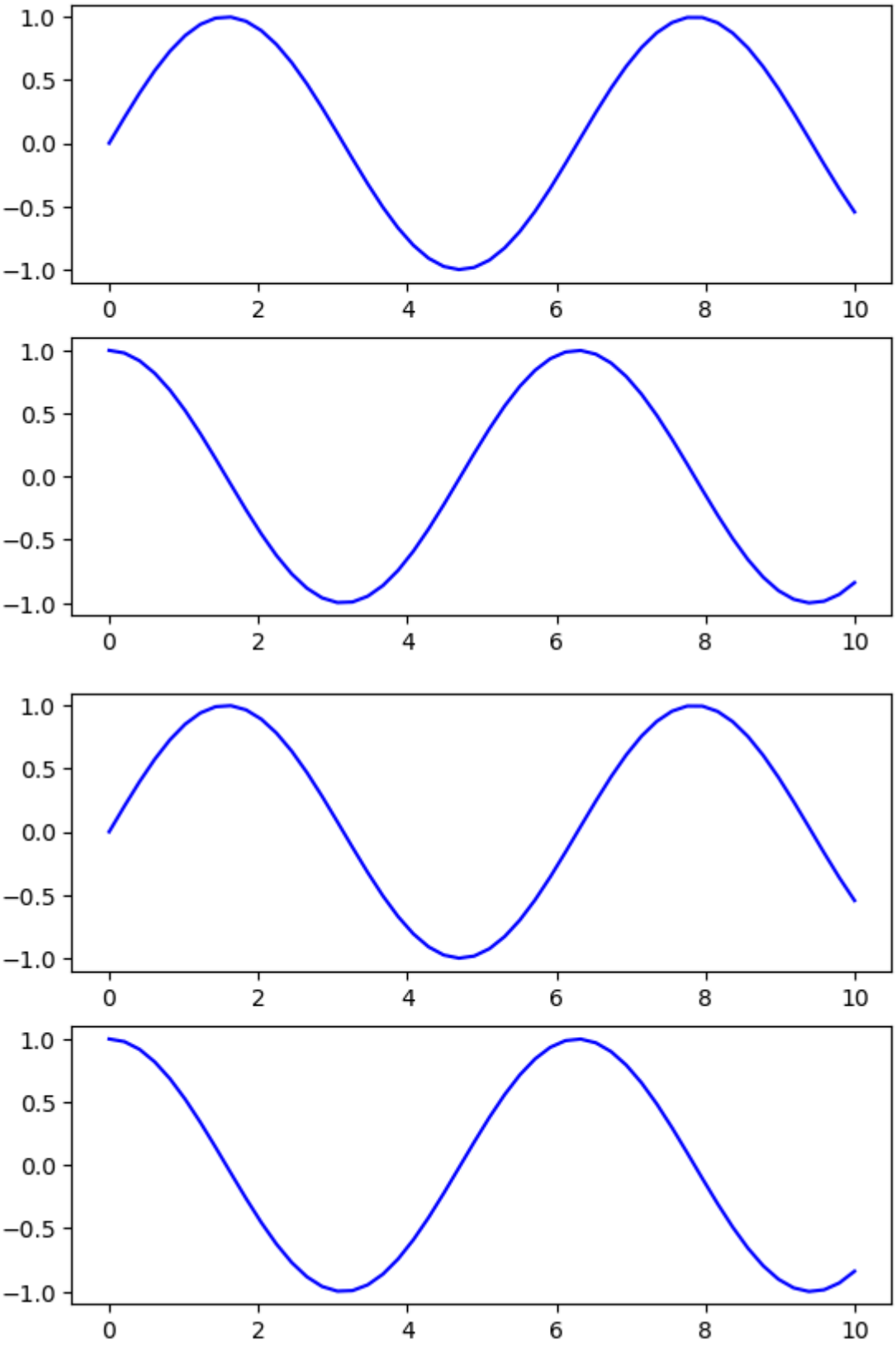


In [15]:
```
# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

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



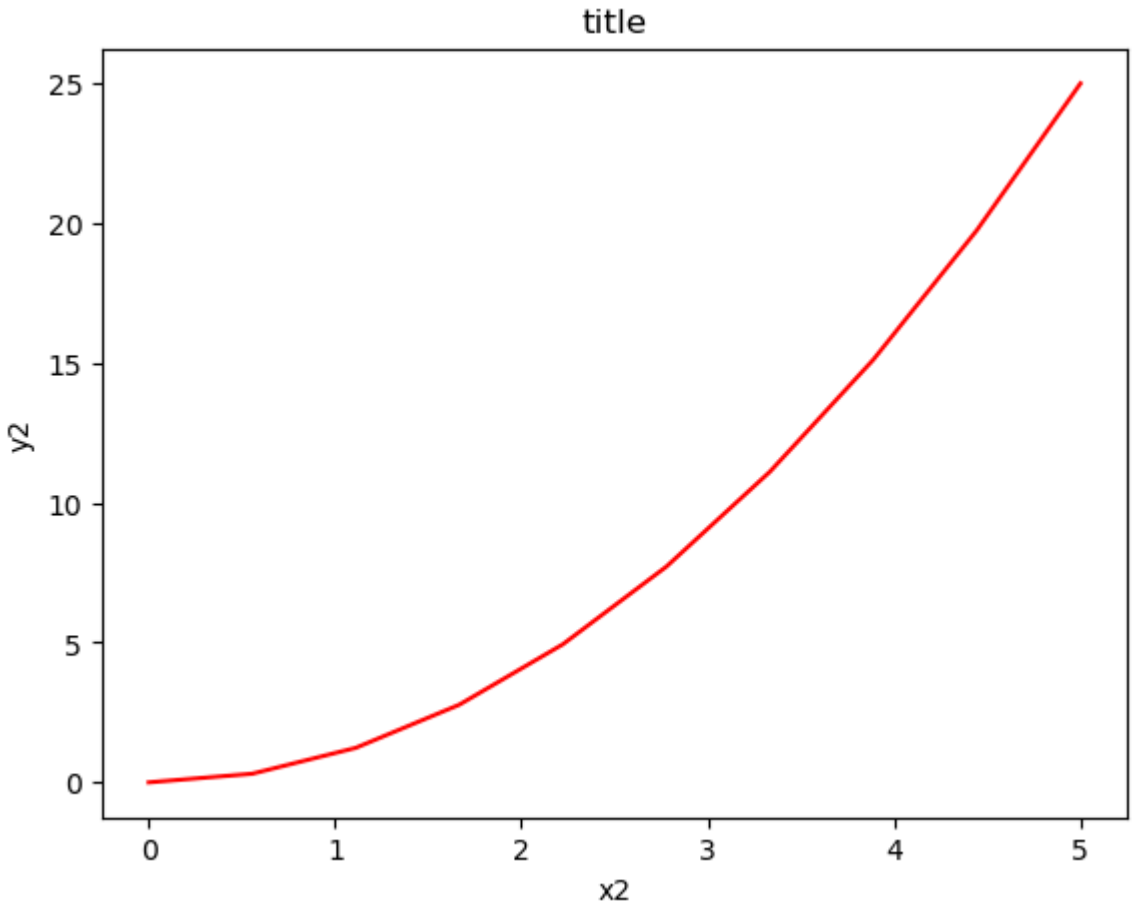
In [17]:
```
# First create a grid of plots
# ax will be an array of two Axes objects
```

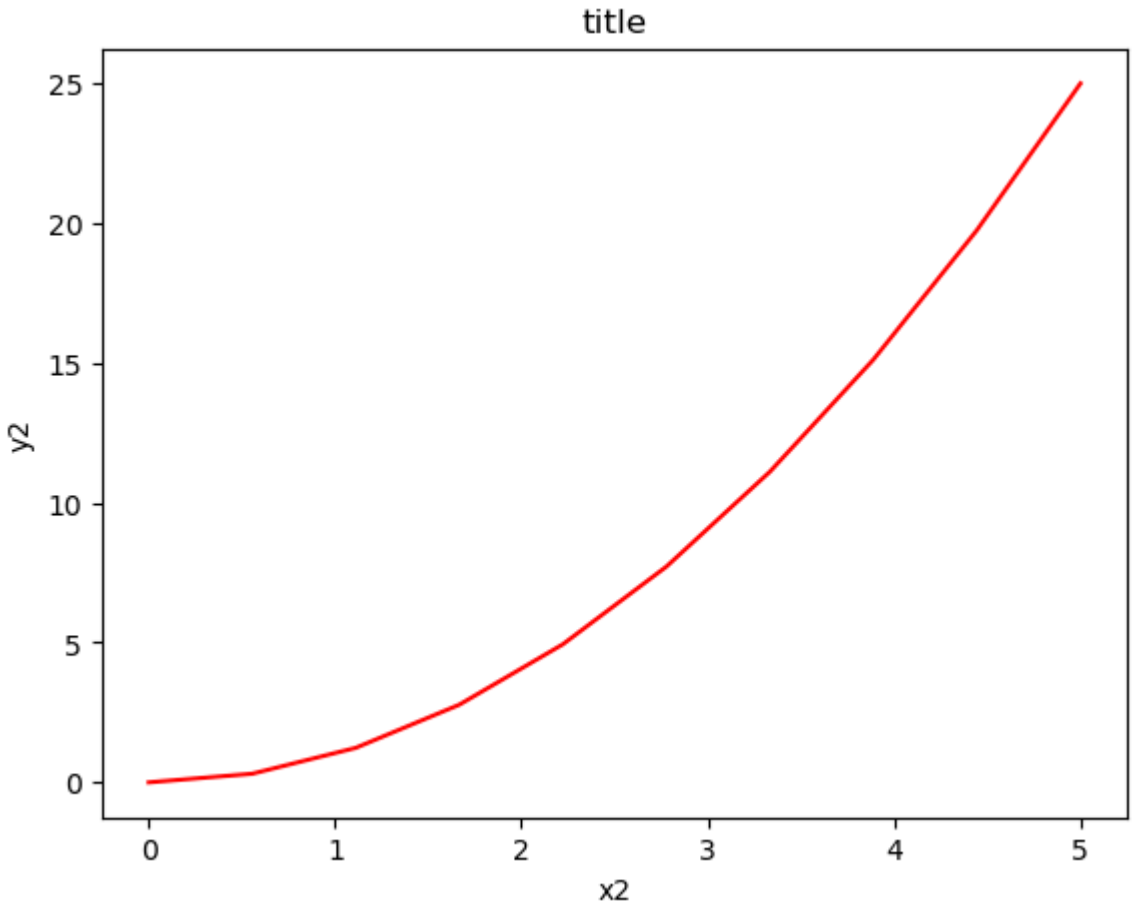
```
fig, ax = plt.subplots(2)


# 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()
```



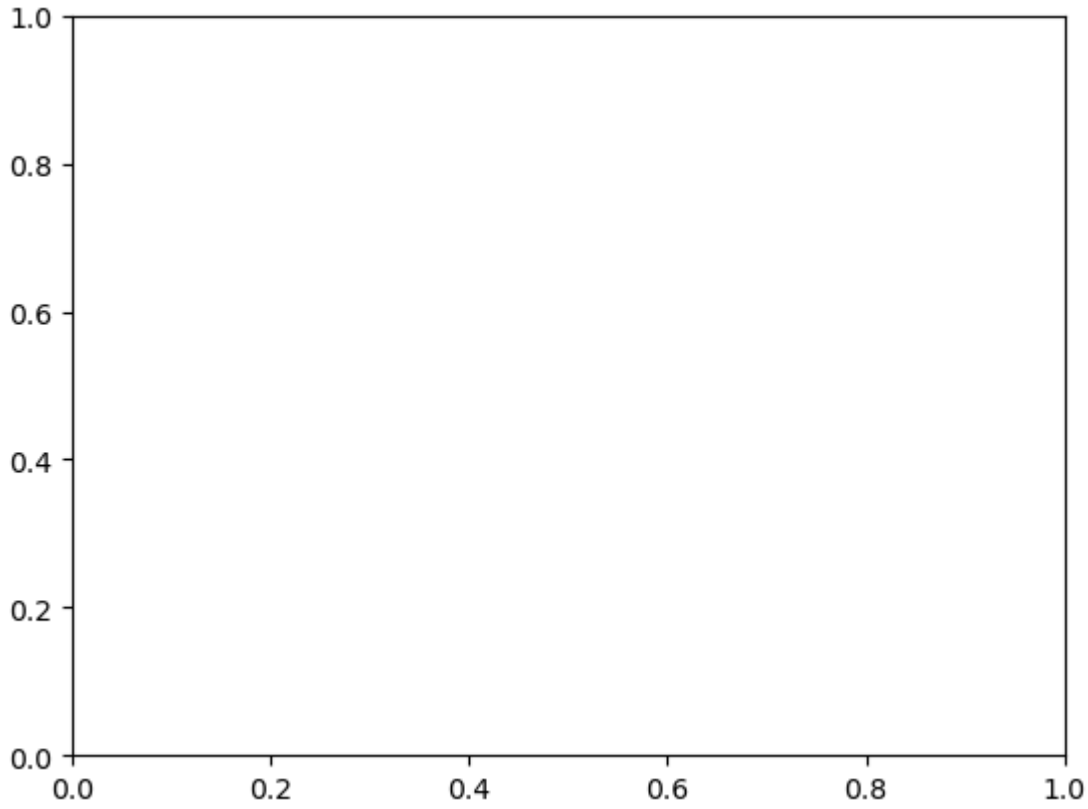
In [19]:
```
fig = plt.figure()
```

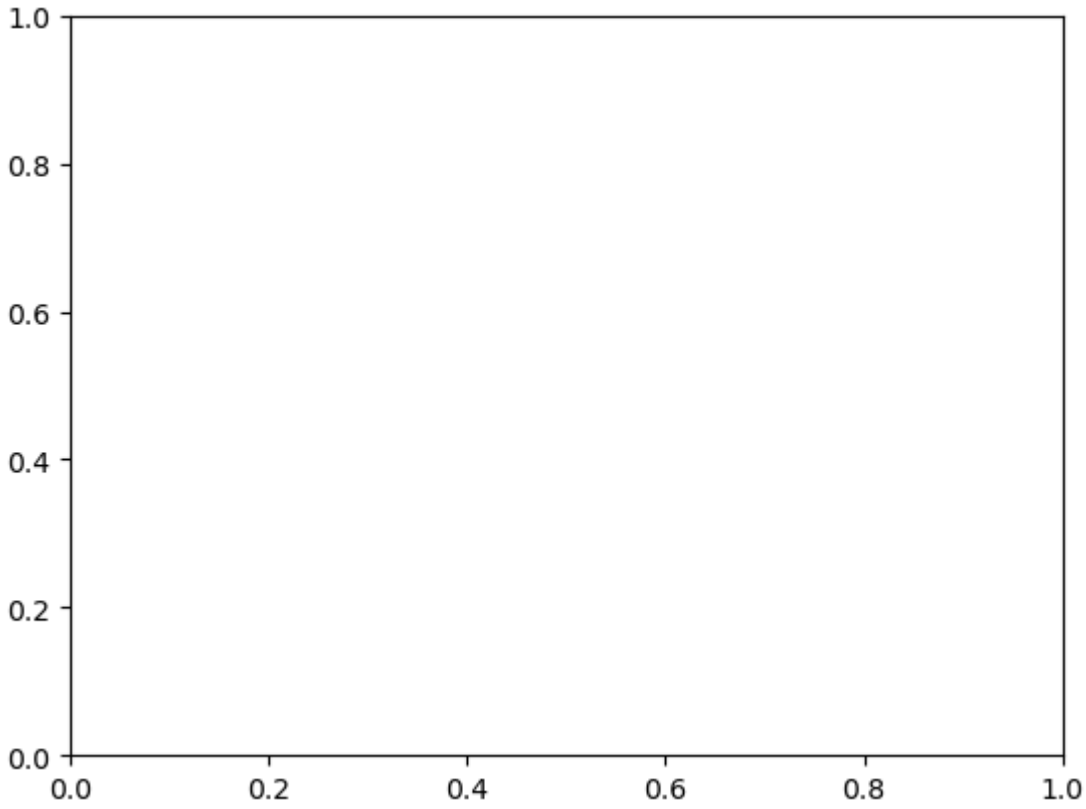
```
x2 = np.linspace(0, 5, 10)
y2 = x2 ** 2

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()
```



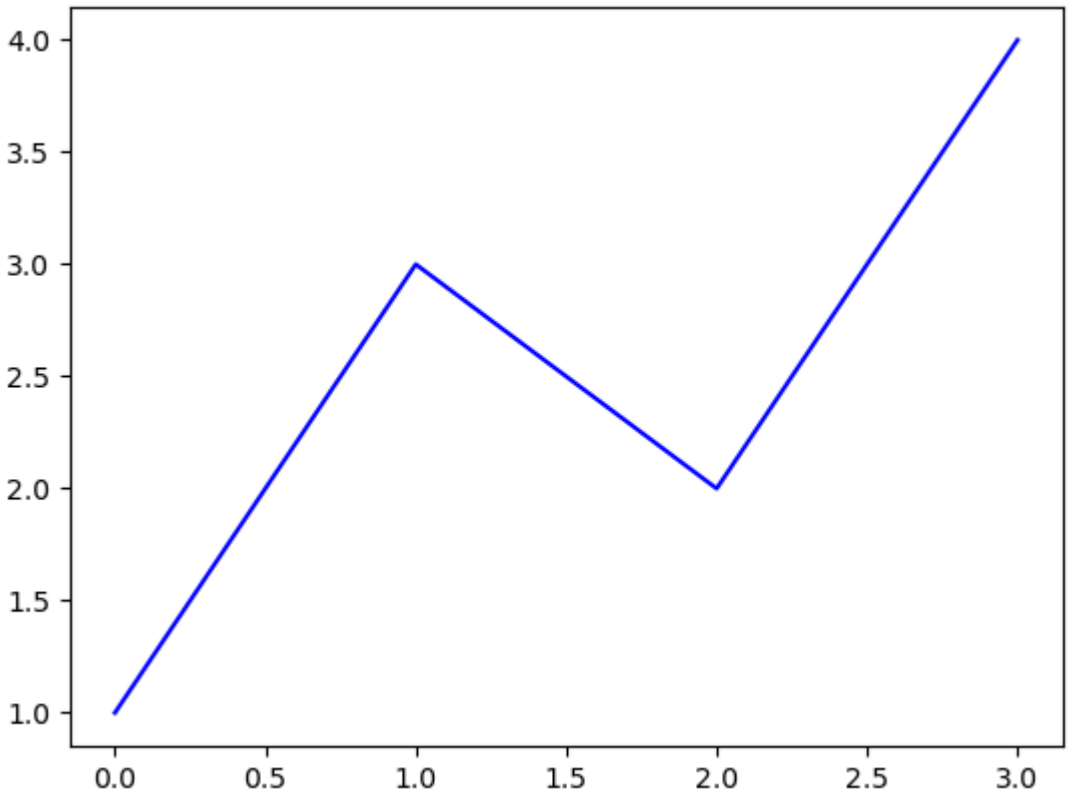


In [21]:
```
fig = plt.figure()

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

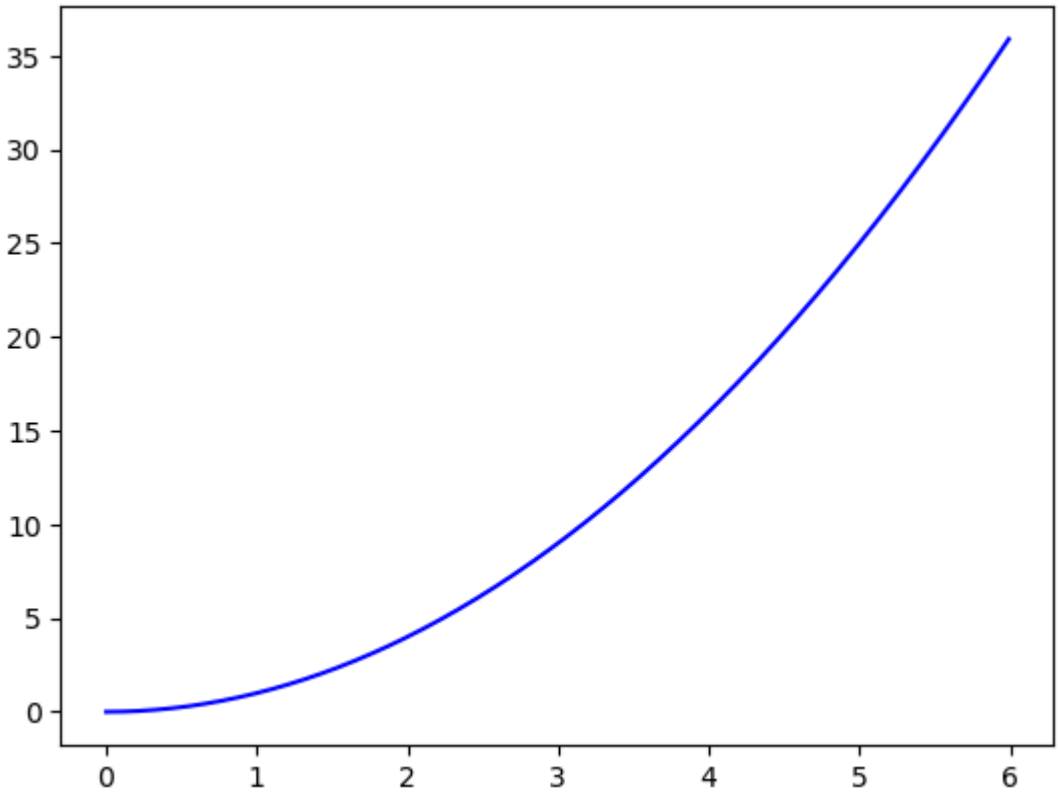




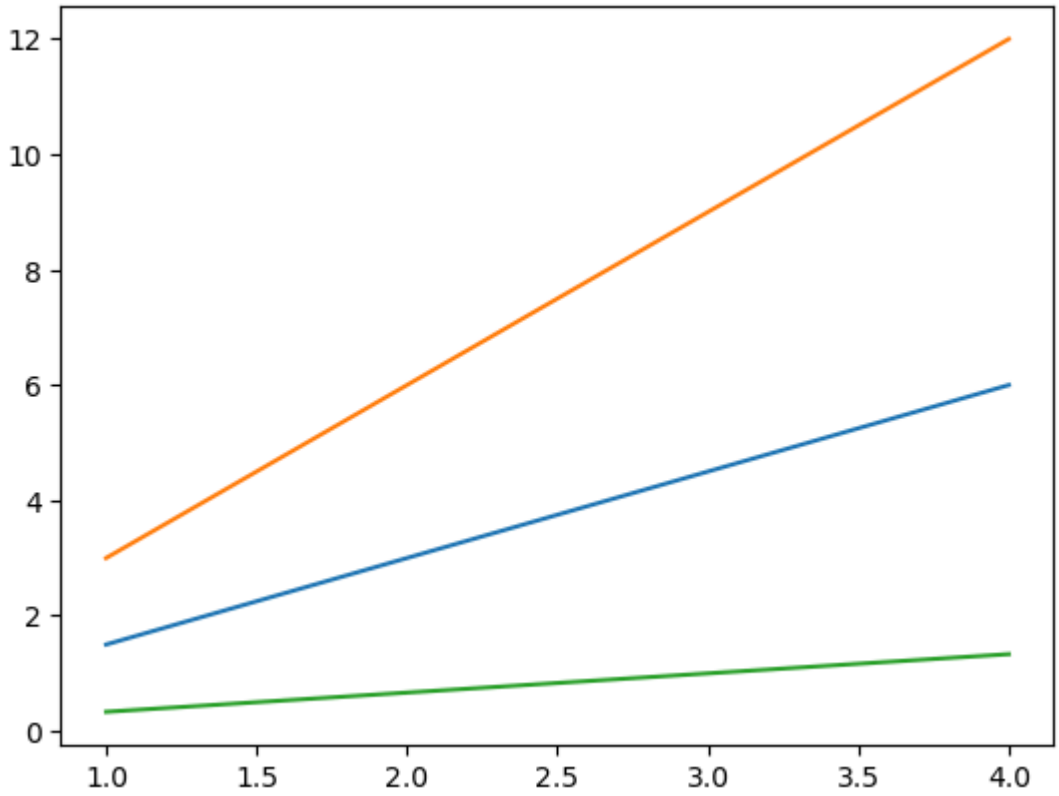
In [22]:
```
plt.plot([1, 3, 2, 4], 'b-')

plt.show( )
```



In [23]:
```
x3 = np.arange(0.0, 6.0, 0.01)

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

plt.show()
```



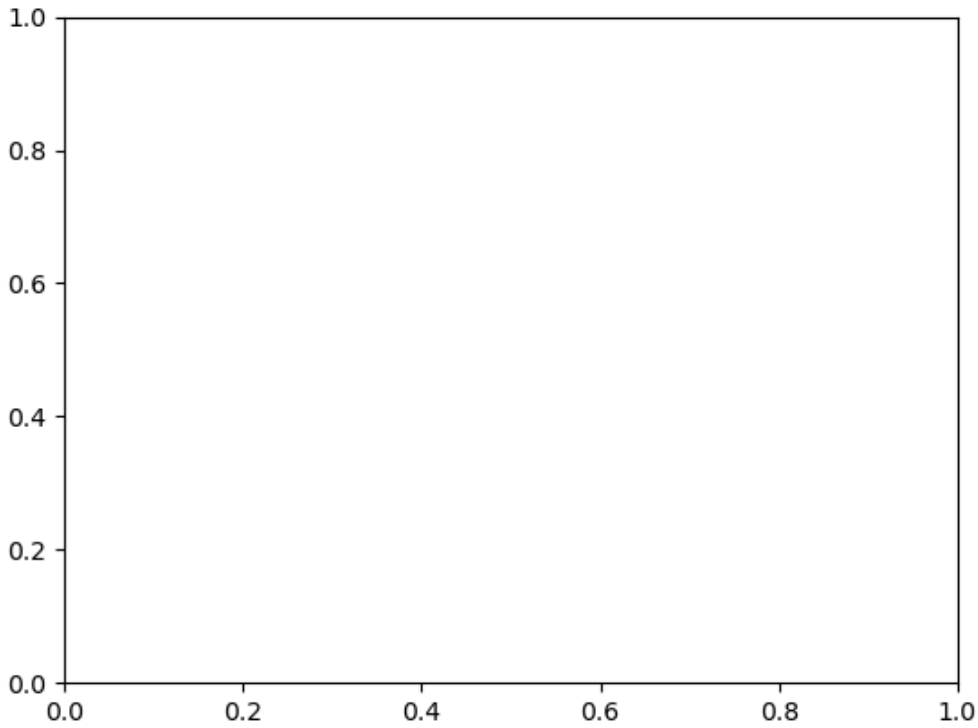
In [24]:
```
x4 = range(1, 5)

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()
```



In [25]:
```
# Saving the figure

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

In [26]:
```
# Explore the contents of figure

from IPython.display import Image

Image('plot1.png')
```

Out[26]:

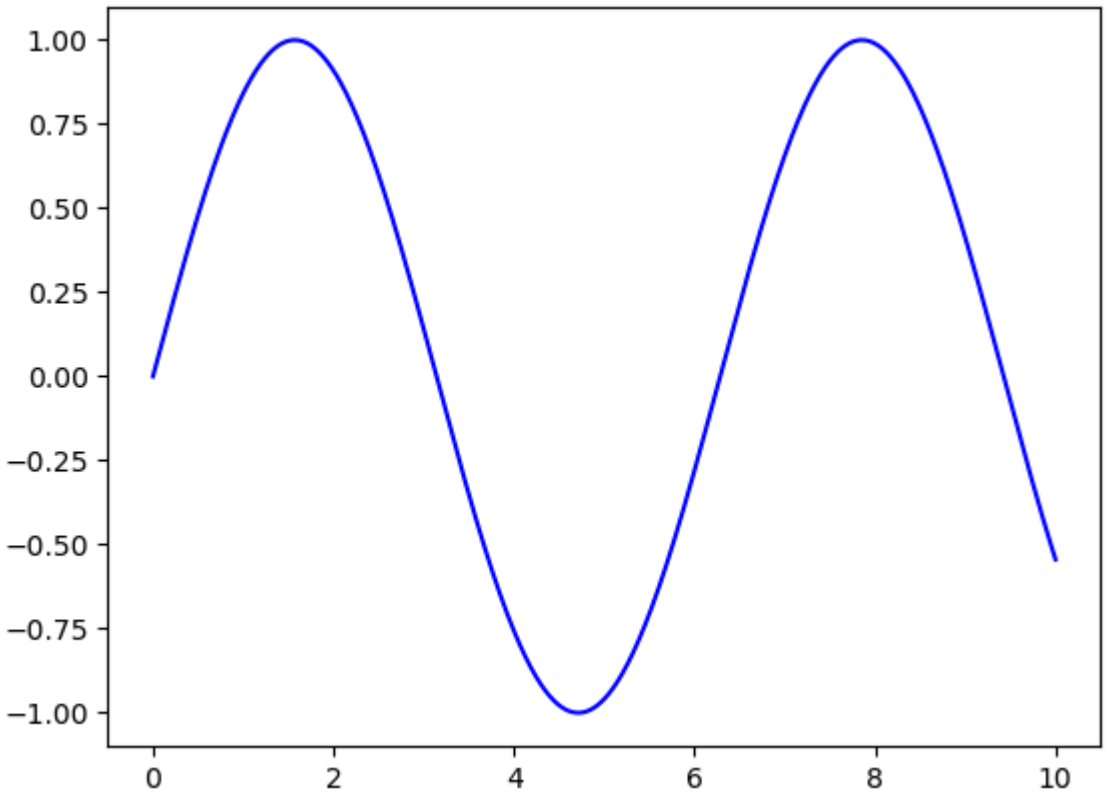


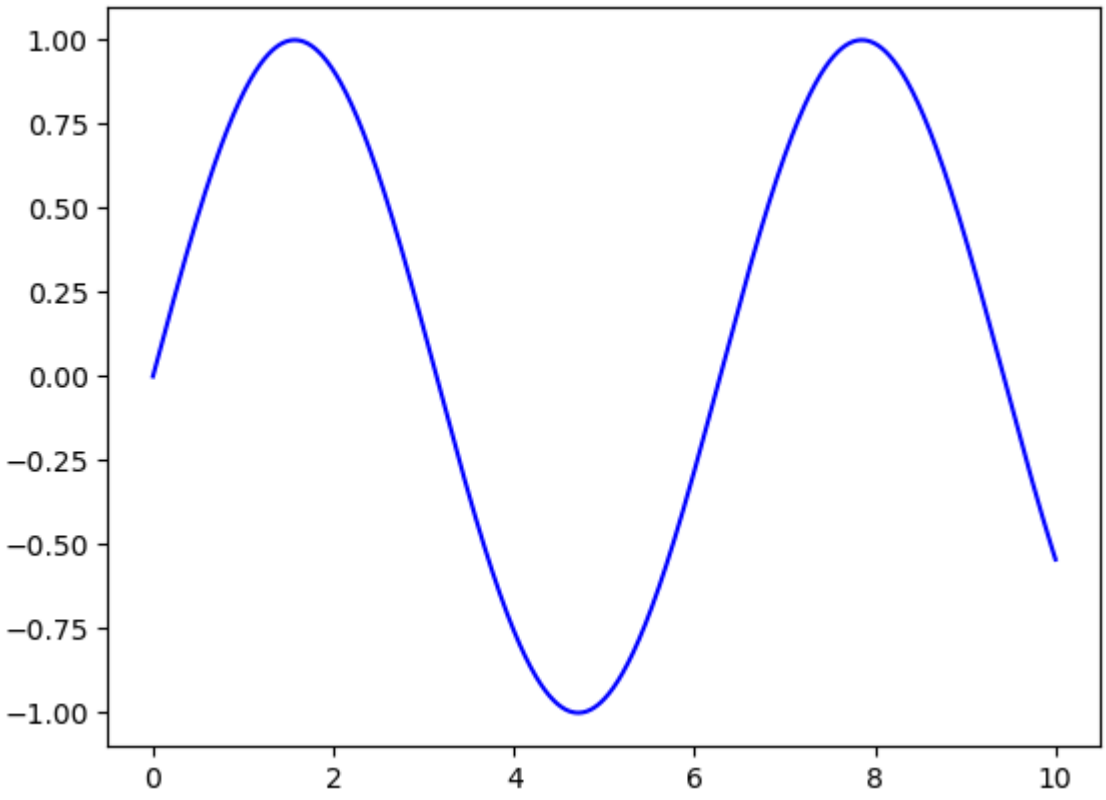
In [27]:
```
# Explore supported file formats


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'}
```

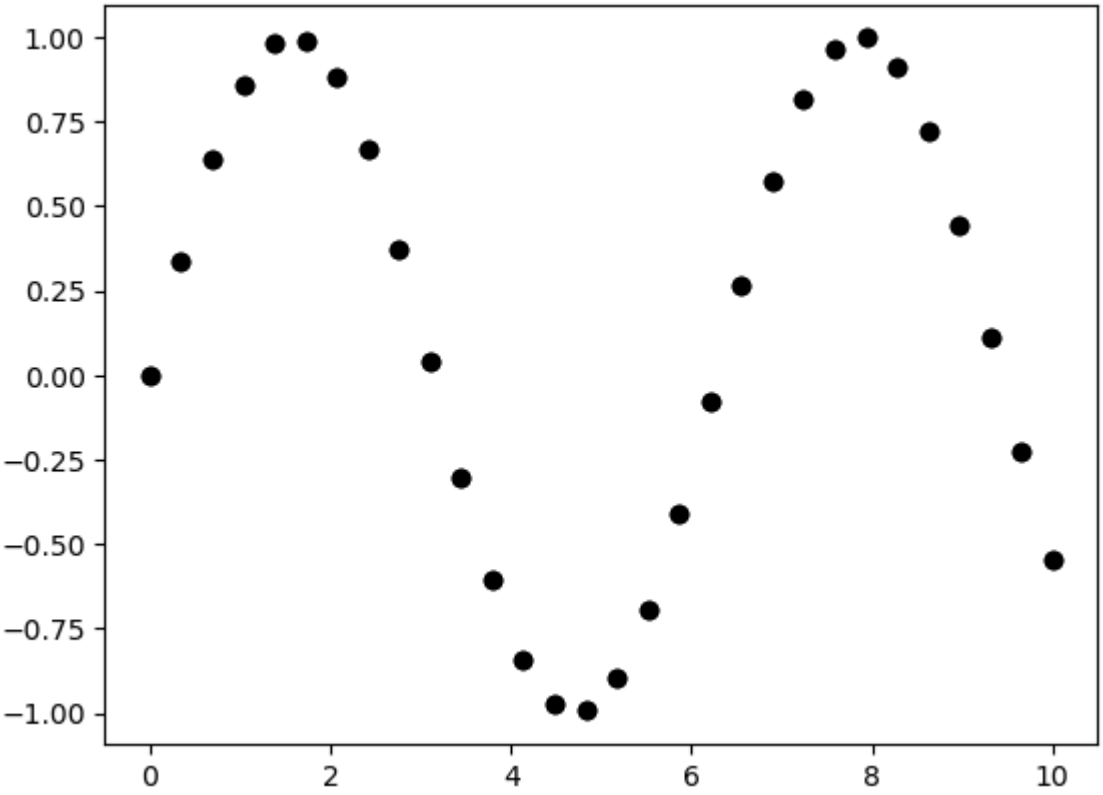
In [29]:
```
# Create figure and axes first
fig = plt.figure()
```

```
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()
```

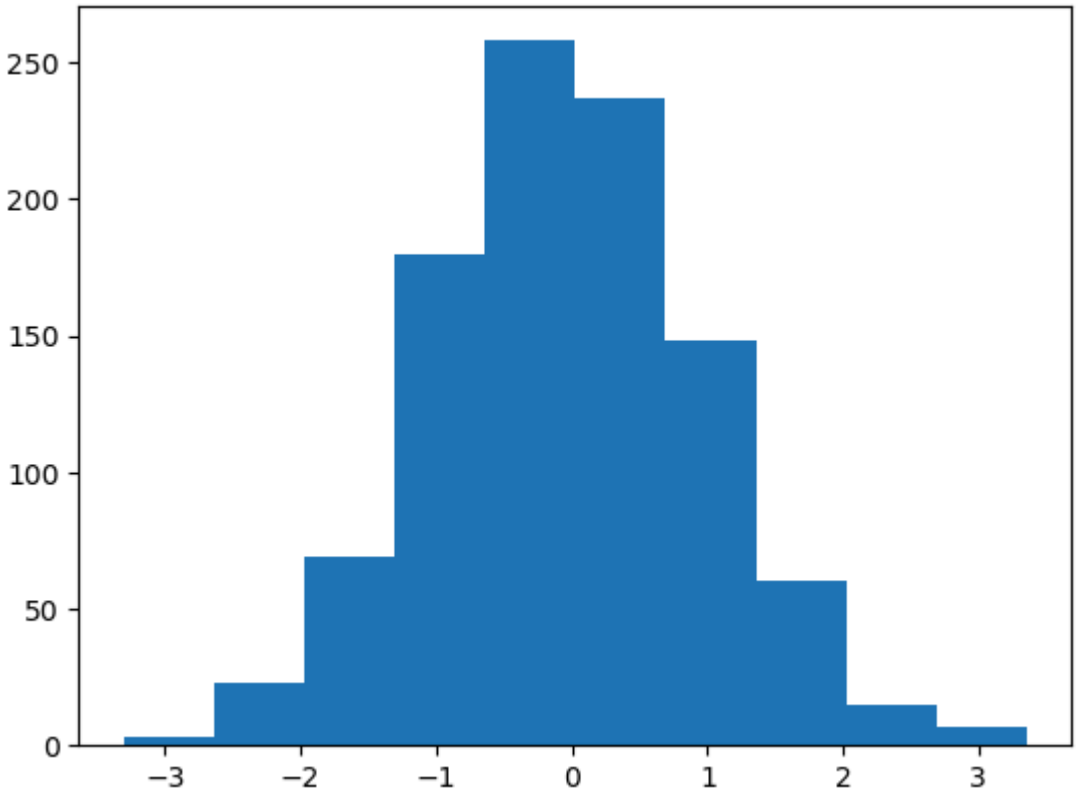




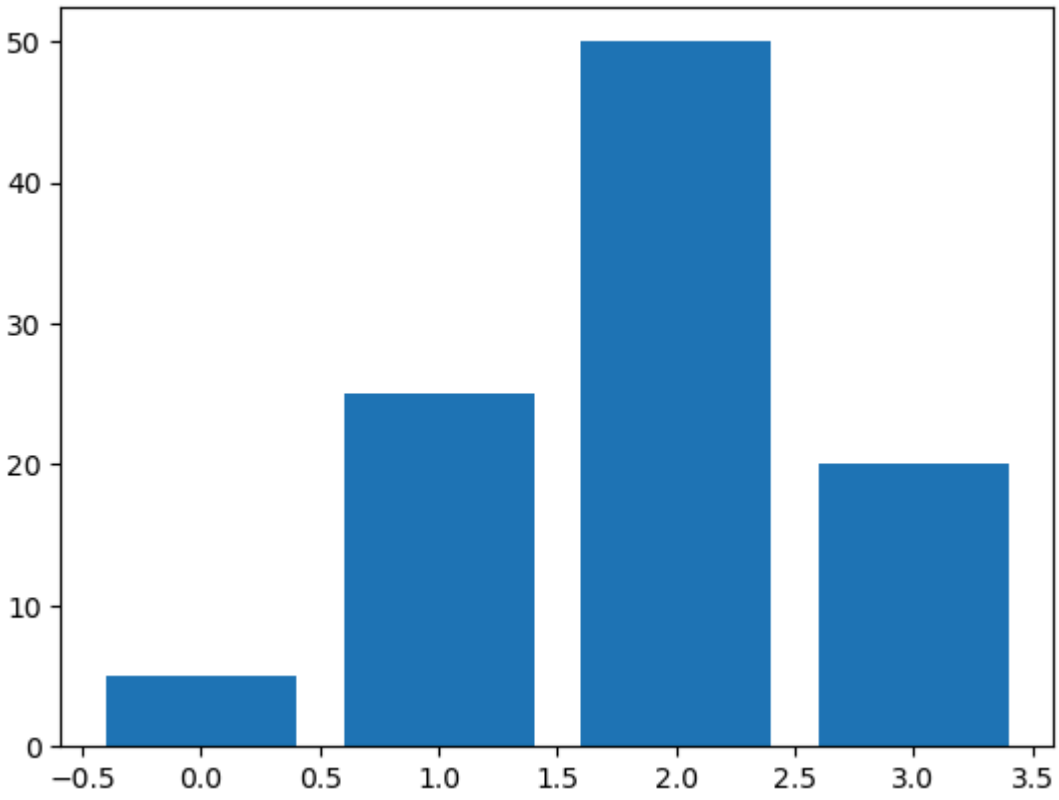
In [31]:
```
x7 = np.linspace(0, 10, 30)

y7 = np.sin(x7)

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



In [32]:
```
data1 = np.random.randn(1000)

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

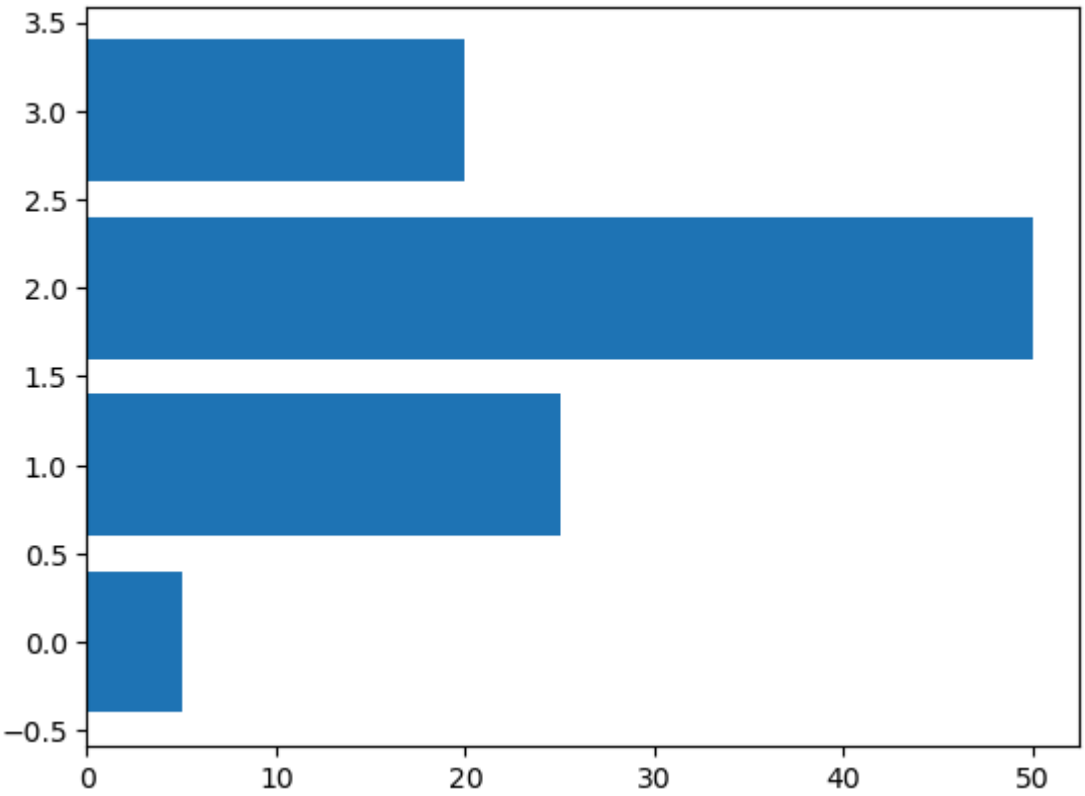


In [33]:
```
data2 = [5. , 25. , 50. , 20.]

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

plt.show()
```

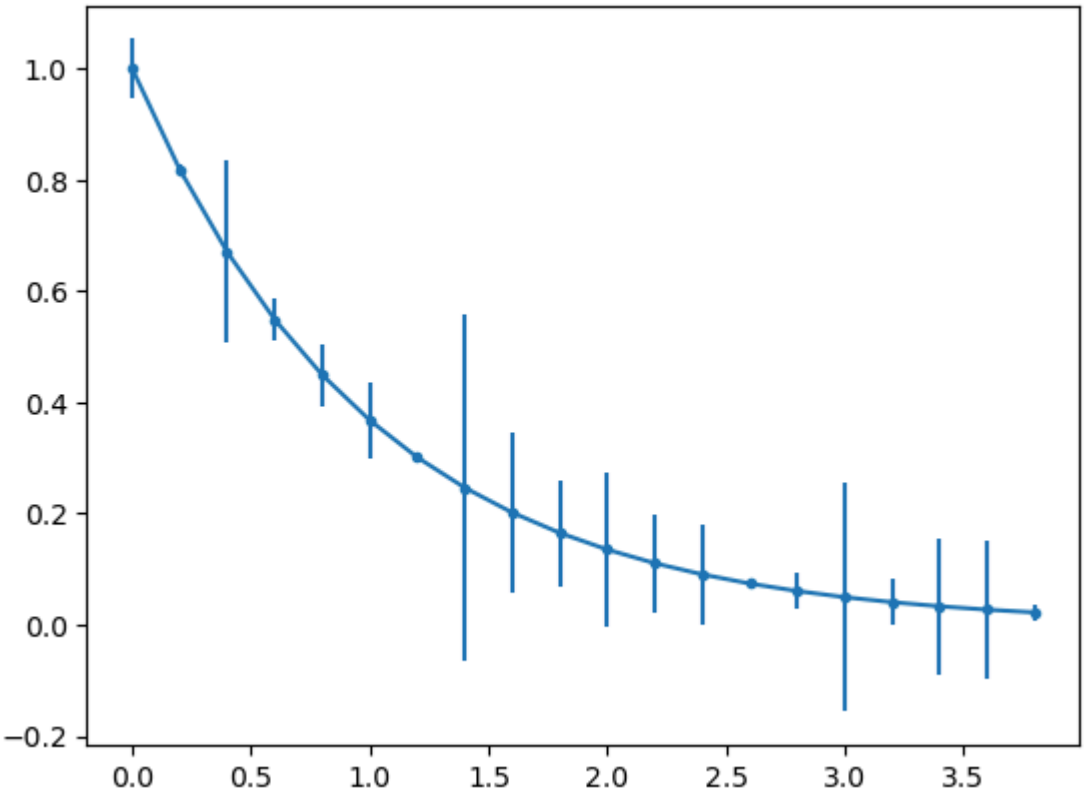


In [34]:
```
data2 = [5. , 25. , 50. , 20.]

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

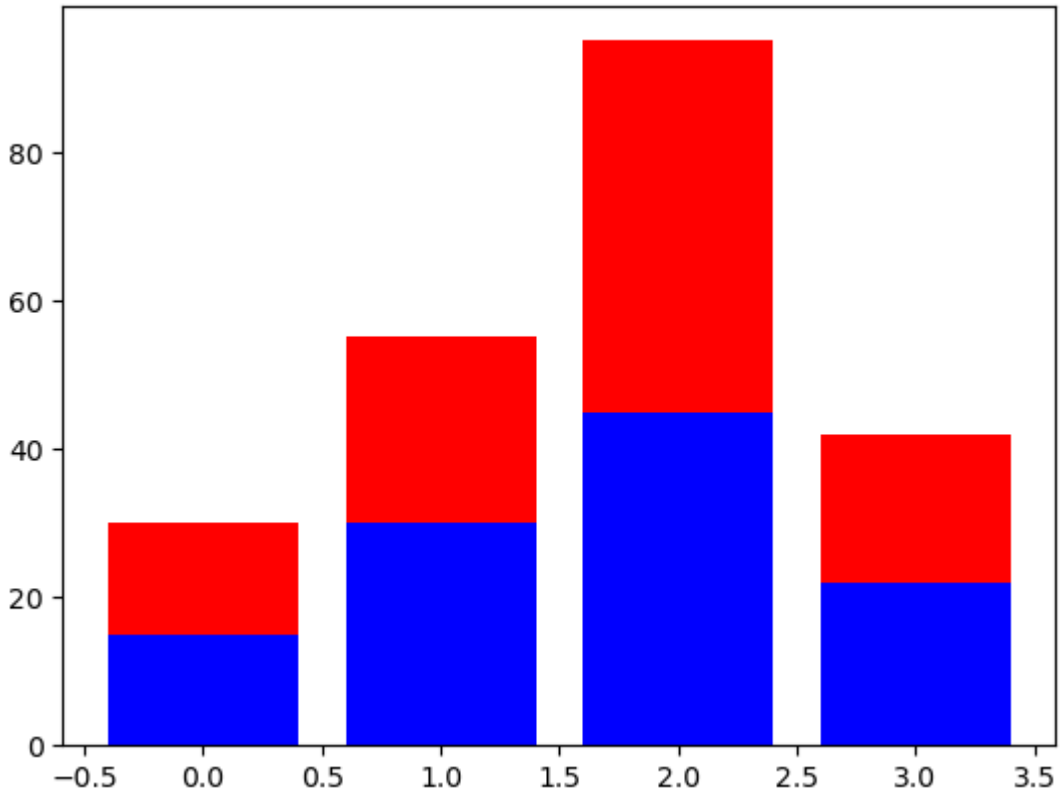
```
plt.show()
```



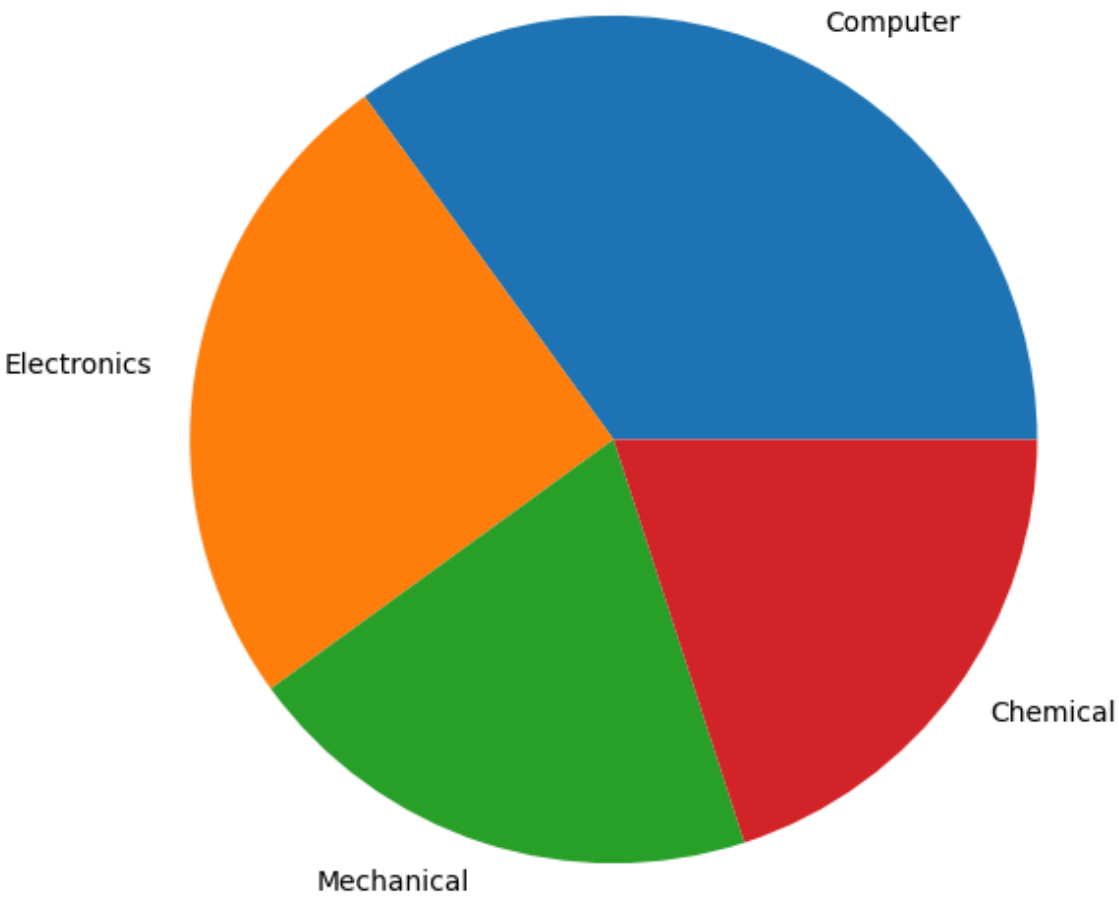
In [35]:
```
x9 = np.arange(0, 4, 0.2)

y9 = np.exp(-x9)

e1 = 0.1 * np.abs(np.random.randn(len(y9)))

plt.errorbar(x9, y9, yerr = e1, fmt = '.-')

plt.show();
```



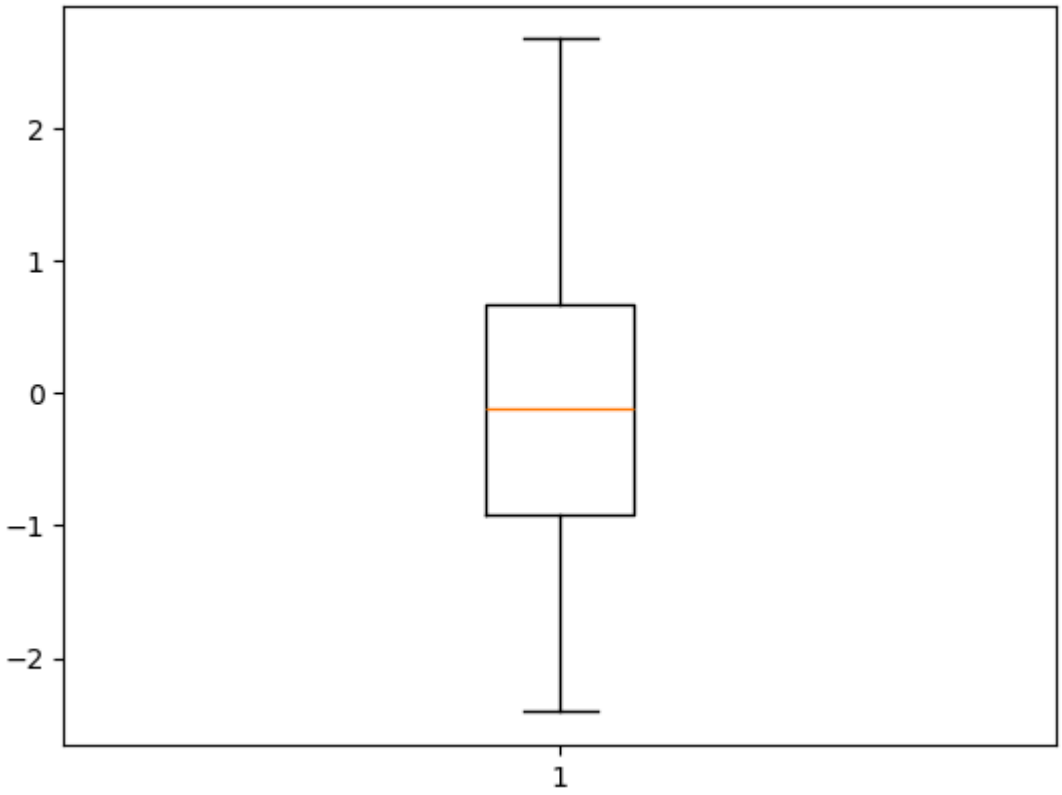
In [36]:
```
A = [15., 30., 45., 22.]

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

z2 = range(4)

plt.bar(z2, A, color = 'b')
plt.bar(z2, B, color = 'r', bottom = A)

plt.show()
```



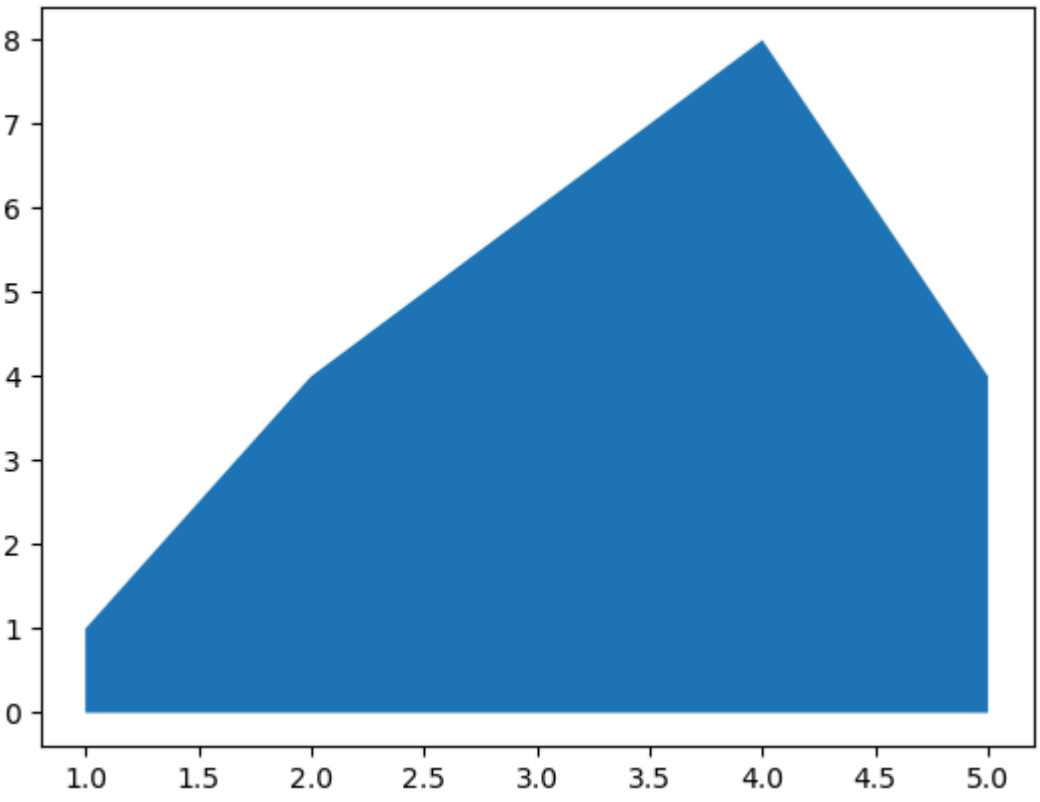
In [37]:
```
plt.figure(figsize=(7,7))

x10 = [35, 25, 20, 20]

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

plt.pie(x10, labels=labels);

plt.show()
```



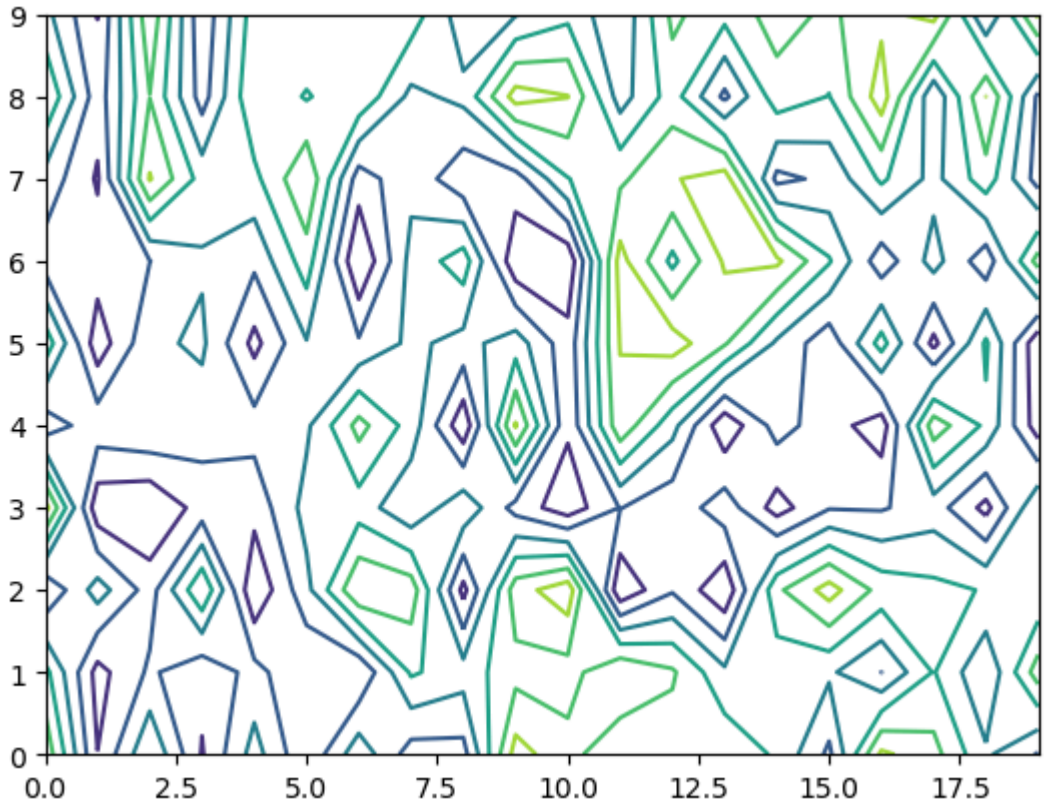
In [38]:
```
data3 = np.random.randn(100)

plt.boxplot(data3)

plt.show();
```



In [39]:
```
# Create some data
x12 = range(1, 6)
y12 = [1, 4, 6, 8, 4]

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



In [40]:
```
# Create a matrix
matrix1 = np.random.rand(10, 20)

cp = plt.contour(matrix1)

plt.show()
```



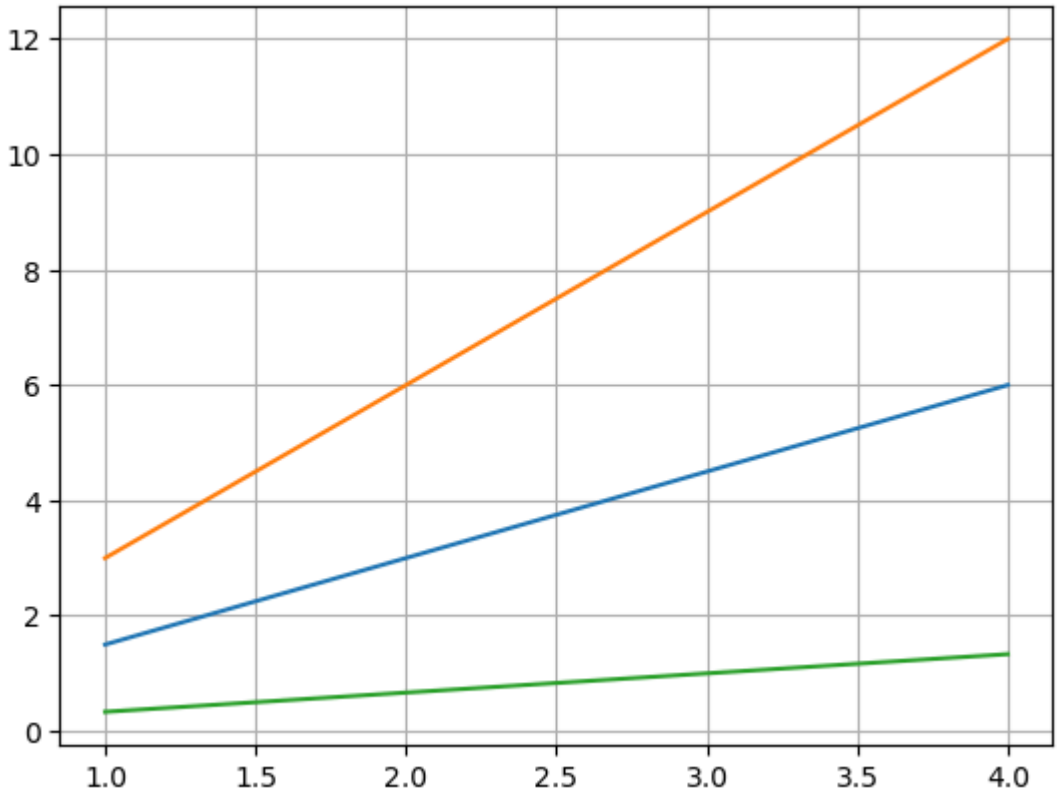
In [41]:
```
# View list of all available styles

print(plt.style.available)
```

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

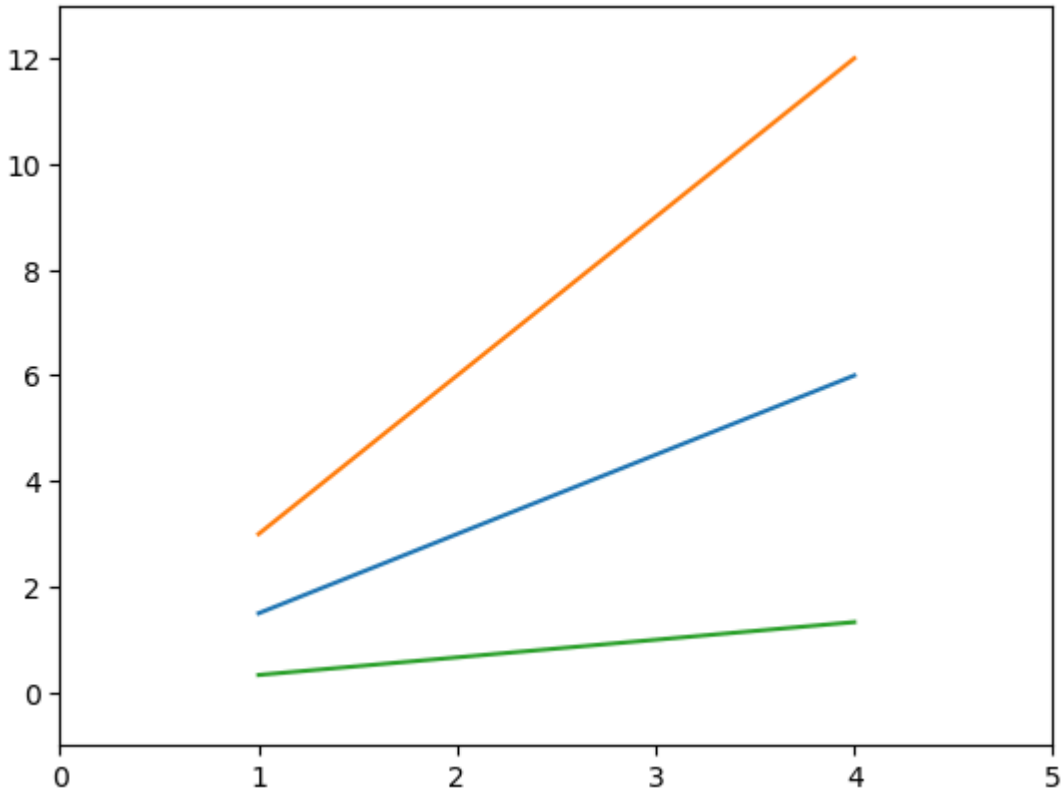
In [42]:
```
# Set styles for plots

plt.style.use('seaborn-bright')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\matplotlib\style\core.py:137, in use(style)
    136 try:
--> 137     style = _rc_params_in_file(style)
    138 except OSError as err:

File ~\anaconda3\Lib\site-packages\matplotlib\__init__.py:870, in _rc_params_in_f
ile(fname, transform, fail_on_error)
    869 rc_temp = {}
--> 870 with _open_file_or_url(fname) as fd:
    871     try:

File ~\anaconda3\Lib\contextlib.py:137, in _GeneratorContextManager.__enter__(sel
f)
    136 try:
--> 137     return next(self.gen)
    138 except StopIteration:

File ~\anaconda3\Lib\site-packages\matplotlib\__init__.py:847, in _open_file_or_u
rl(fname)
    846 fname = os.path.expanduser(fname)
--> 847 with open(fname, encoding='utf-8') as f:
    848     yield f

FileNotFoundError: [Errno 2] No such file or directory: 'seaborn-bright'

The above exception was the direct cause of the following exception:

OSError                                   Traceback (most recent call last)
Cell In[42], line 3
      1 # Set styles for plots
----> 3 plt.style.use('seaborn-bright')

File ~\anaconda3\Lib\site-packages\matplotlib\style\core.py:139, in use(style)
    137         style = _rc_params_in_file(style)
    138     except OSError as err:
--> 139         raise OSError(
    140             f"{style!r} is not a valid package style, path of style "
    141             f"file, URL of style file, or library style name (library "
    142             f"styles are listed in `style.available`)") from err
    143 filtered = {}
    144 for k in style:  # don't trigger RcParams.__getitem__('backend')

OSError: 'seaborn-bright' is not a valid package style, path of style file, URL o
f style file, or library style name (library styles are listed in `style.availabl
e`)
```

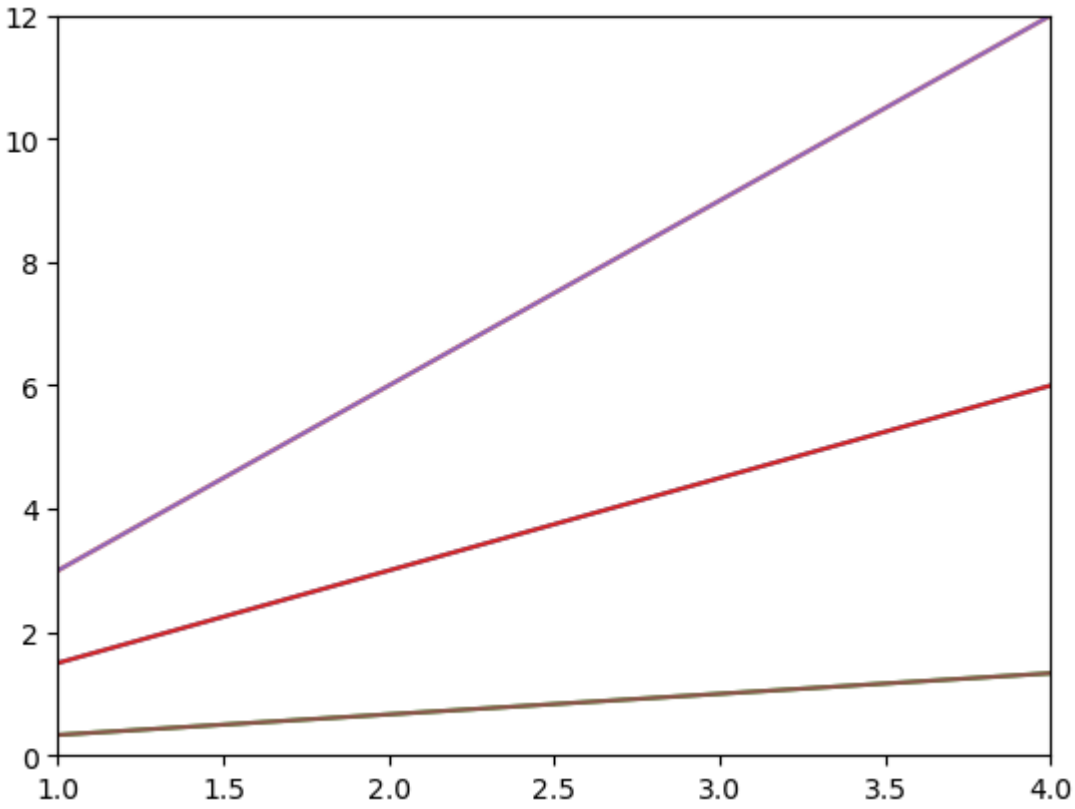
In [43]:
```
x15 = np.arange(1, 5)

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

plt.grid(True)

plt.show()
```



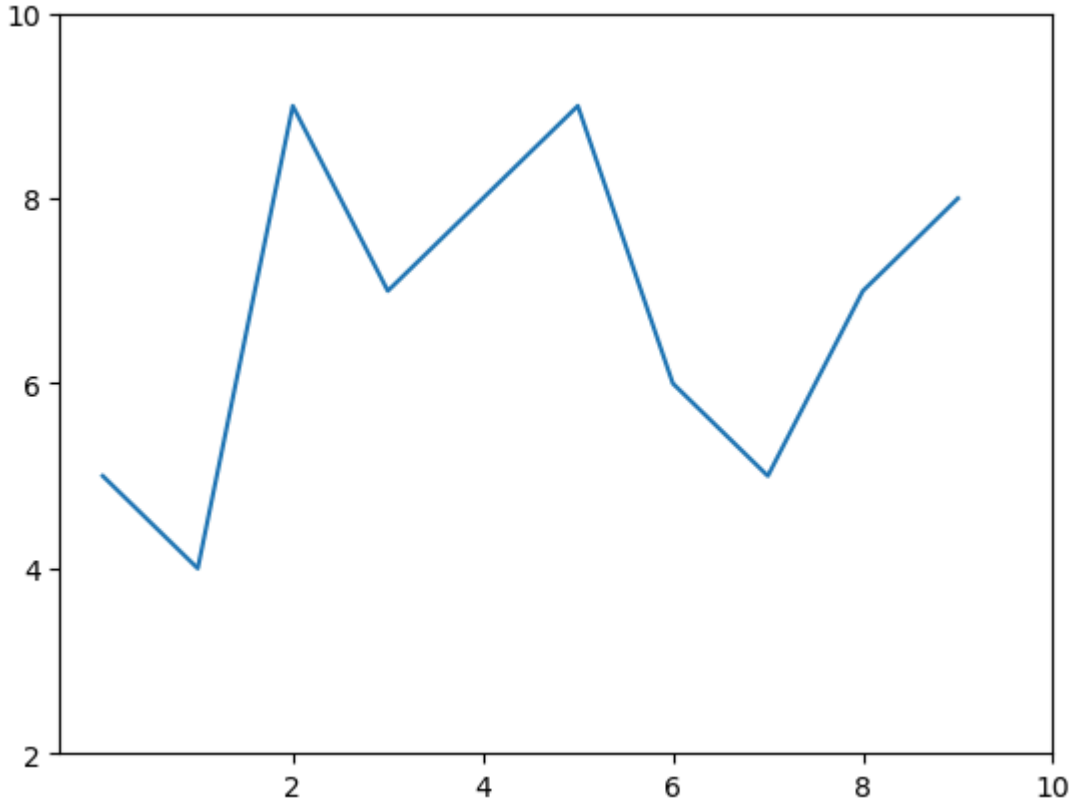
In [44]:
```
x15 = np.arange(1, 5)

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()
```



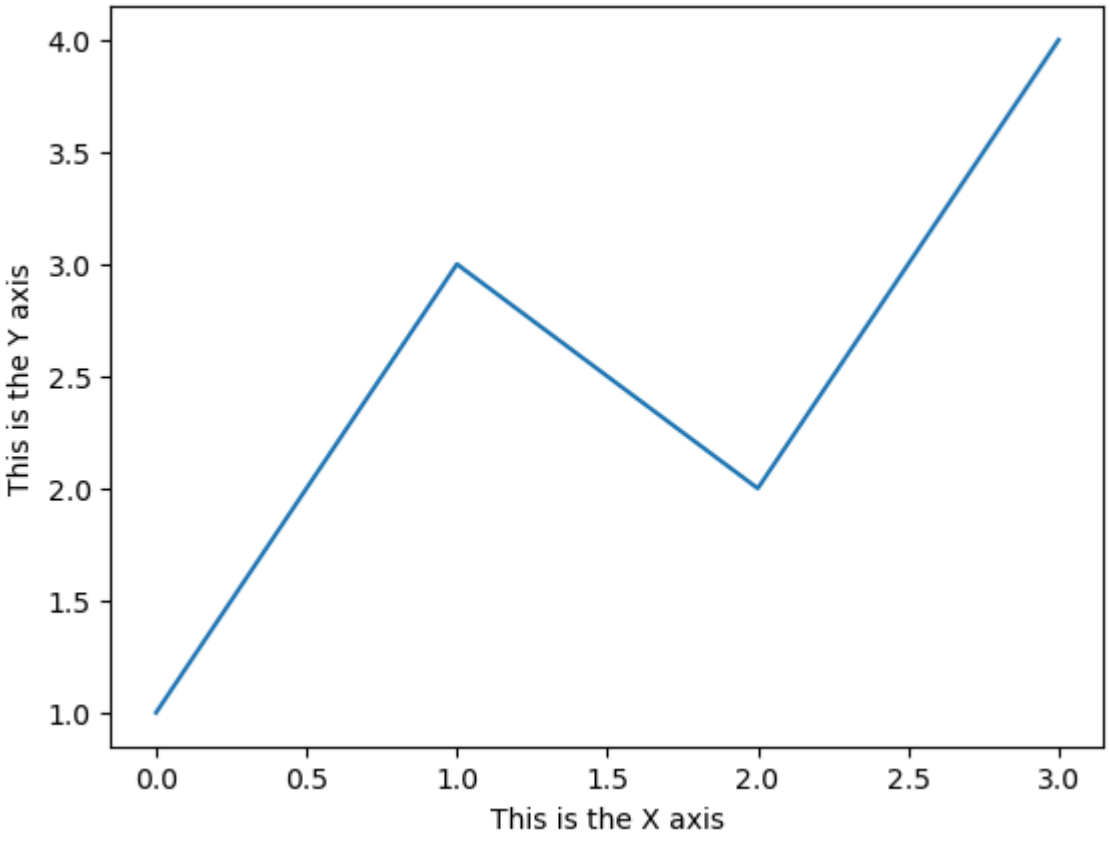
In [46]:
```
x15 = np.arange(1, 5)

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()
```



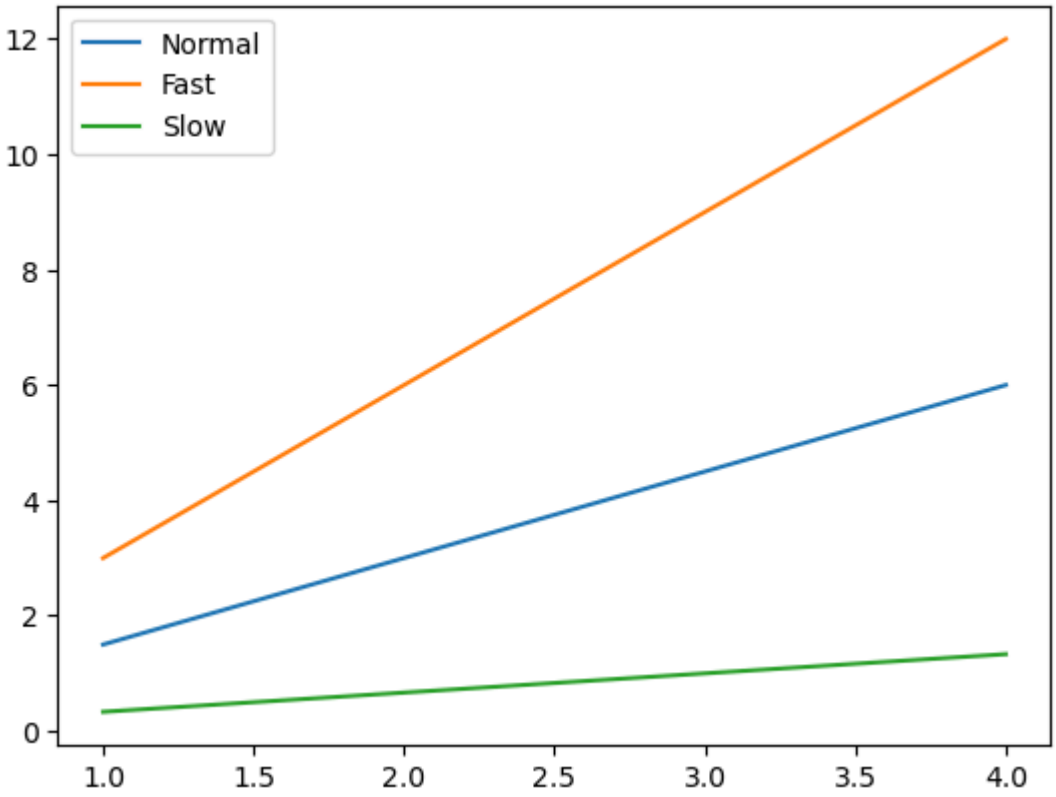
In [47]:
```
u = [5, 4, 9, 7, 8, 9, 6, 5, 7, 8]

plt.plot(u)

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

plt.show()
```

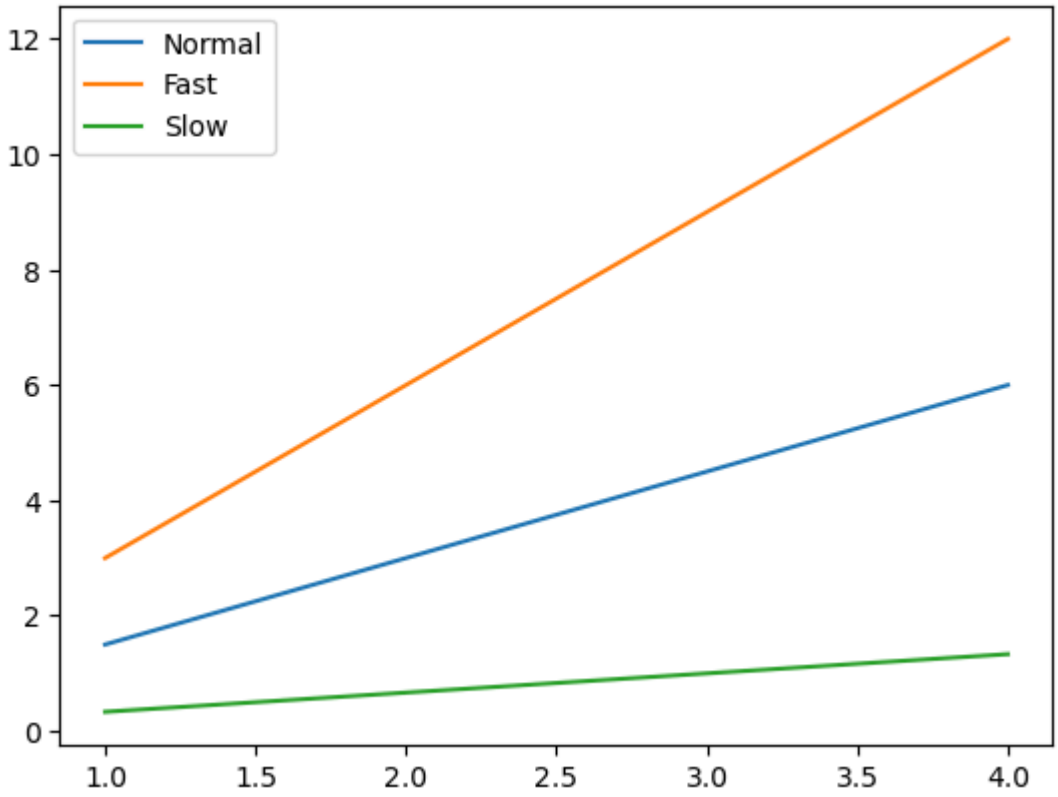


In [48]:
```
plt.plot([1, 3, 2, 4])

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

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

plt.show()
```

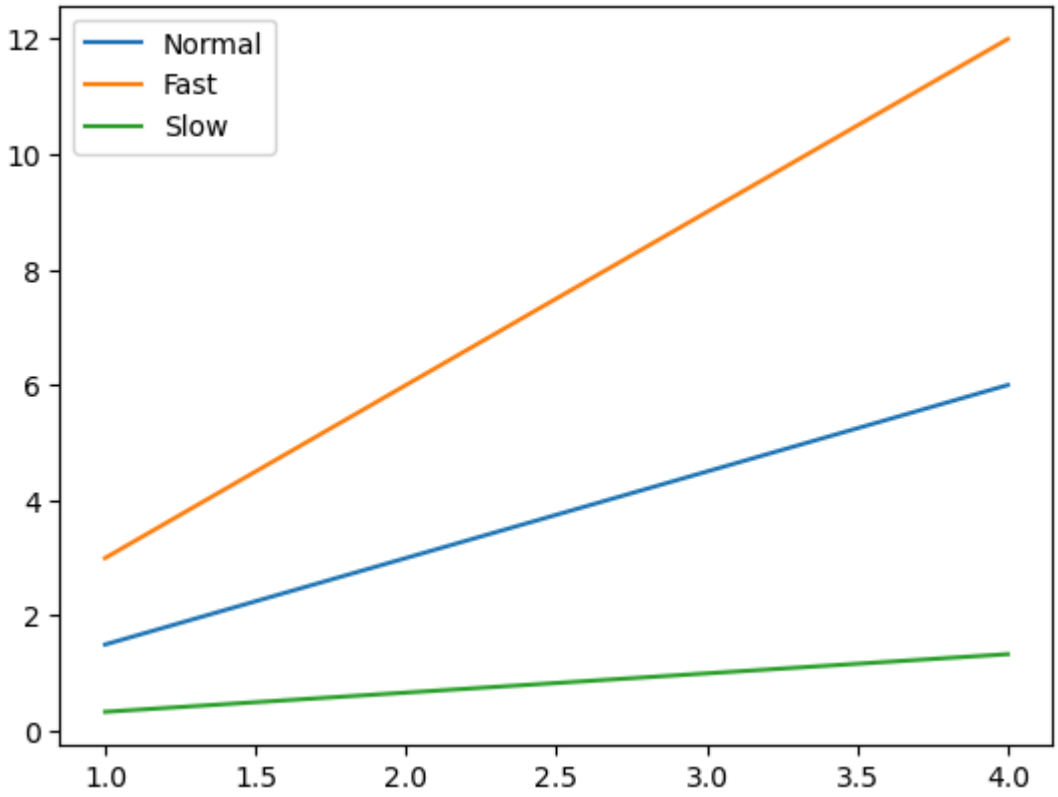


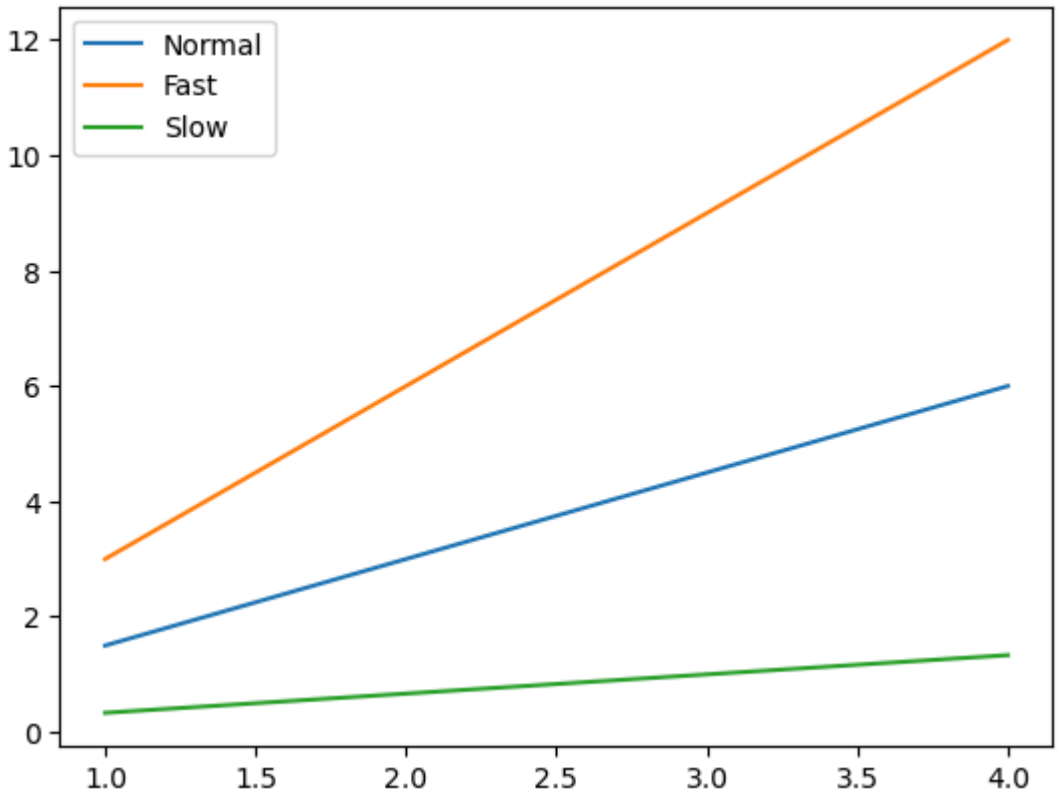
In [50]:
```
x15 = np.arange(1, 5)

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()
```



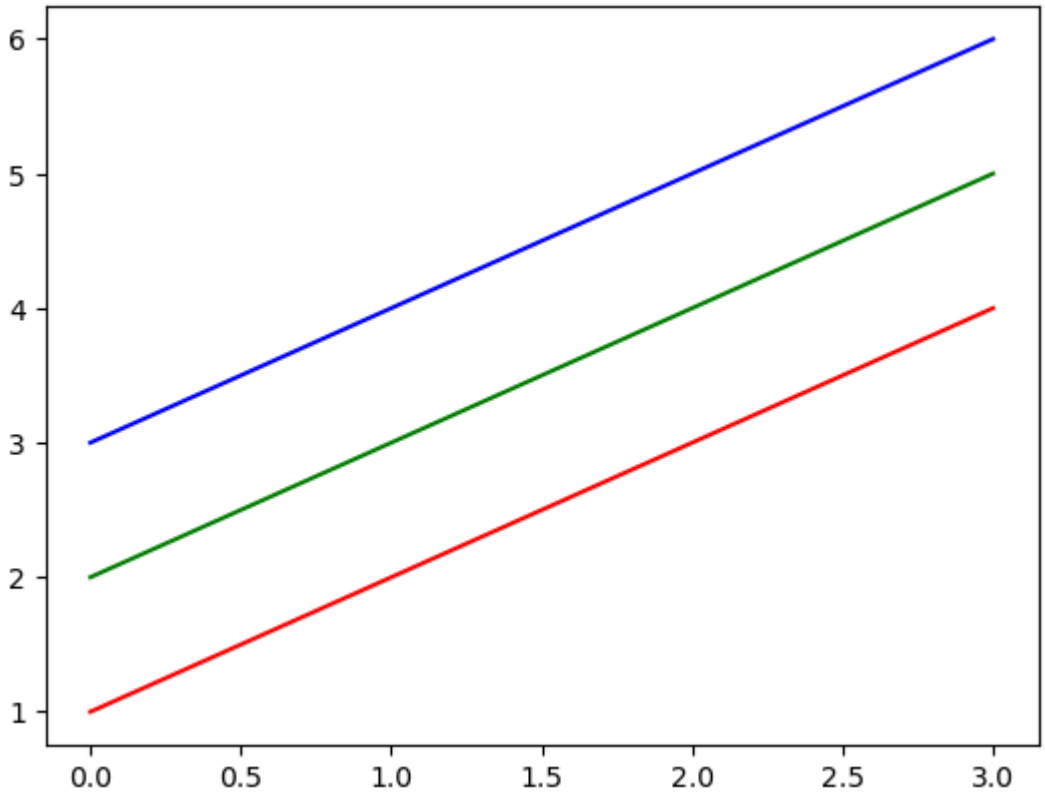


In [52]:
```
x15 = np.arange(1, 5)

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()
```

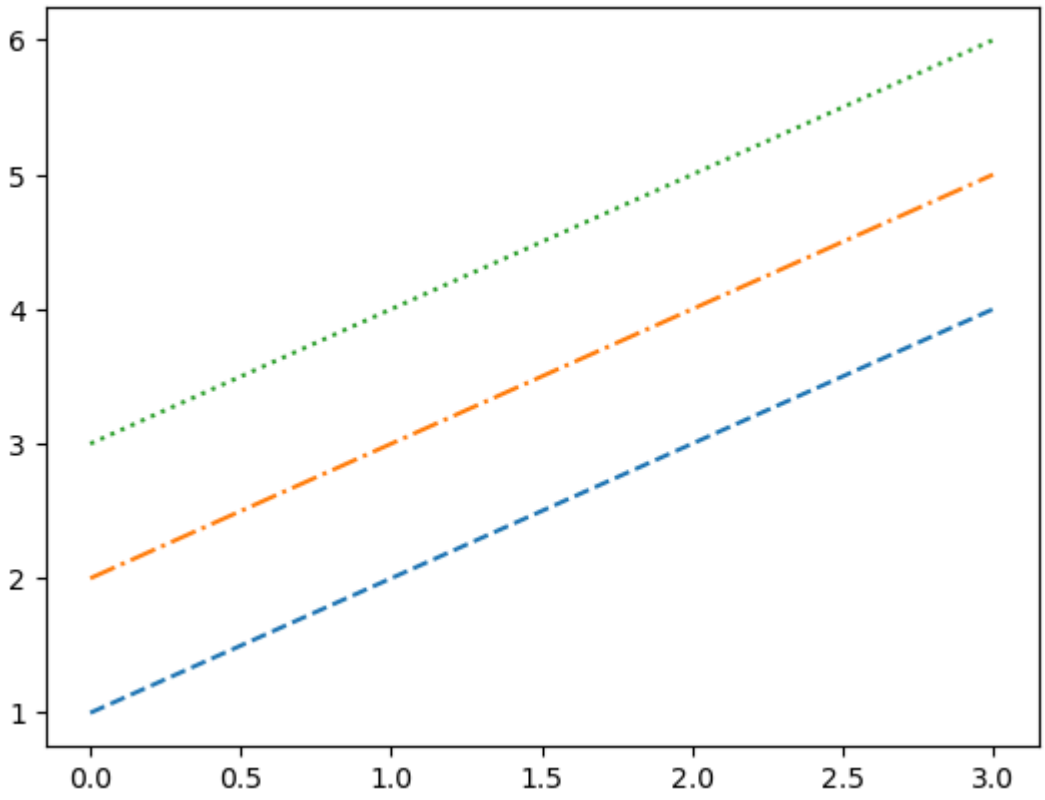




In [53]:
```
x16 = np.arange(1, 5)

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

plt.show()
```



In [54]:
```
x16 = np.arange(1, 5)

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

plt.show()
```



In [ ]: