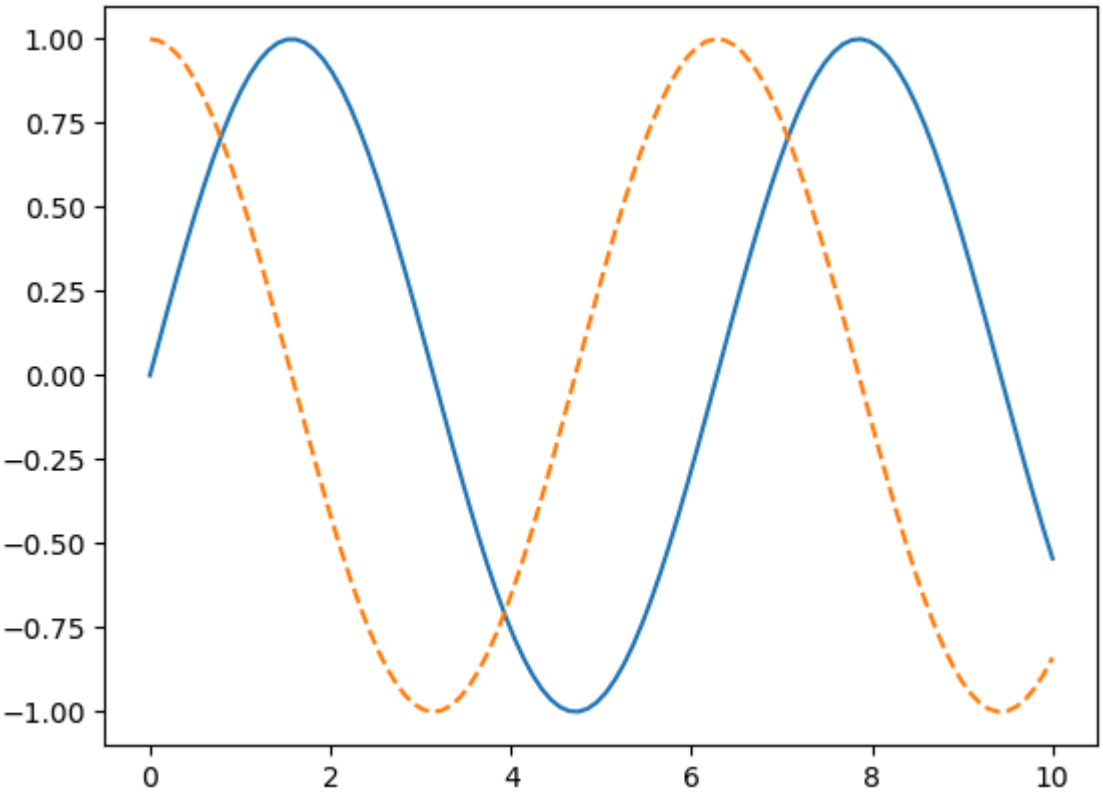
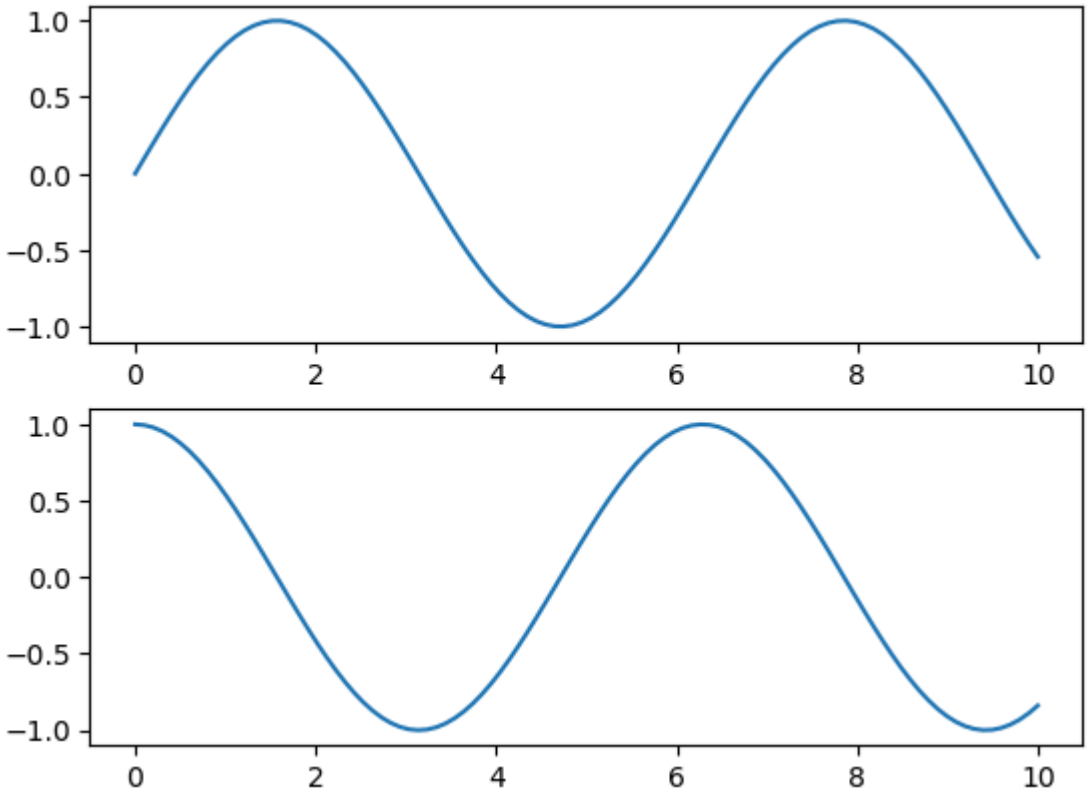
In [10]:
```
import numpy as np
import pandas as pd
```

In [12]:
```
import matplotlib.pyplot as plt
```

In [36]:
```
%matplotlib inline

x1 = np.linspace(0, 10, 100)
# It creates an array x1 of 100 evenly spaced values between 0 and 10.

# Create a Plot Figure
#fig = plt.figure()

plt.plot(x1, np.sin(x1), '-') # Performs sine function over range [0,10]
plt.plot(x1, np.cos(x1), '--') # Performs cosine function over range [0,10]

plt.show()
```



In [44]:
```
# This activates the first panel ( 2 rows, 1 column ) in the grid.
plt.subplot(2, 1, 1)   # (rows, columns, panel number)
plt.plot(x1, np.sin(x1)) # creates a sine wave plot in this first panel

# This activates the second panel ( 2 rows, 1 column ) in the grid.
plt.subplot(2, 1, 2) # (rows, columns, panel number)
plt.plot(x1, np.cos(x1)) # creates a cosine wave plot in this second panel

plt.show()
```



In [52]:
```
# WE ARE PRINTING THE CURRENT FIGURE INFORMATION
```

In [50]:
```
print(plt.gcf())
plt.show()
```

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

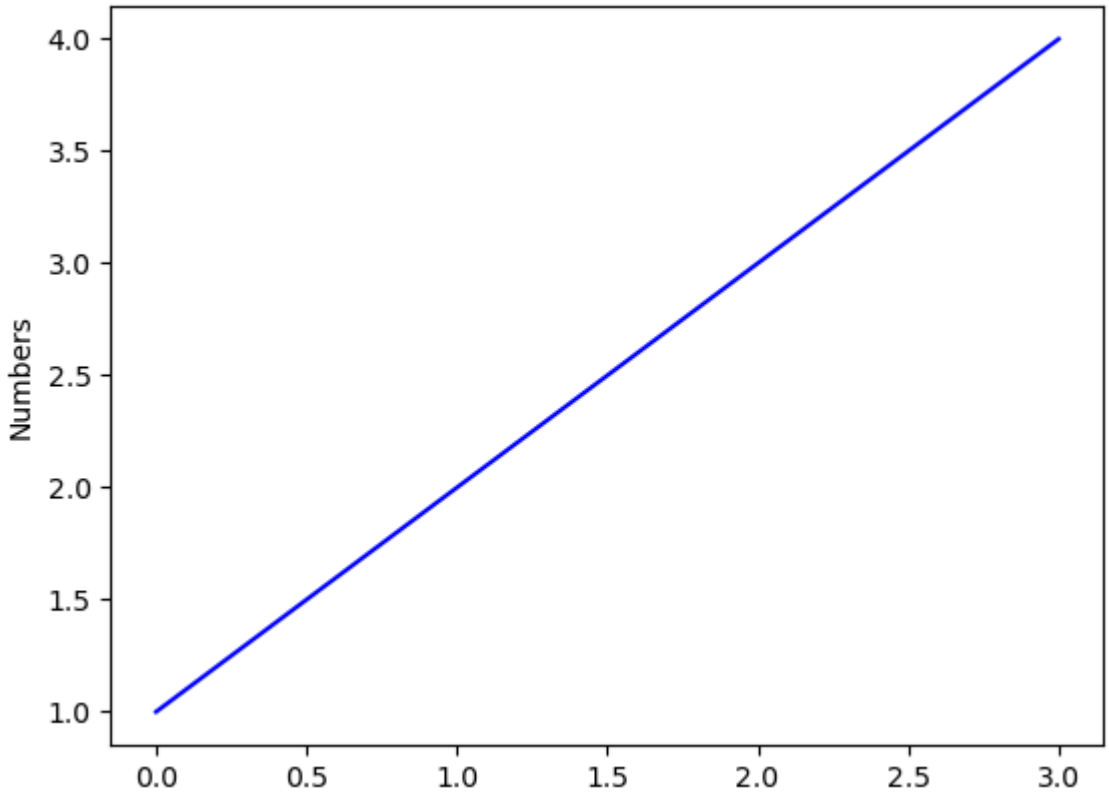
In [54]:
```
# WE ARE PRINTING THE CIRRENT AXIS INFORMATION
```

In [56]:
```
print(plt.gca())
```

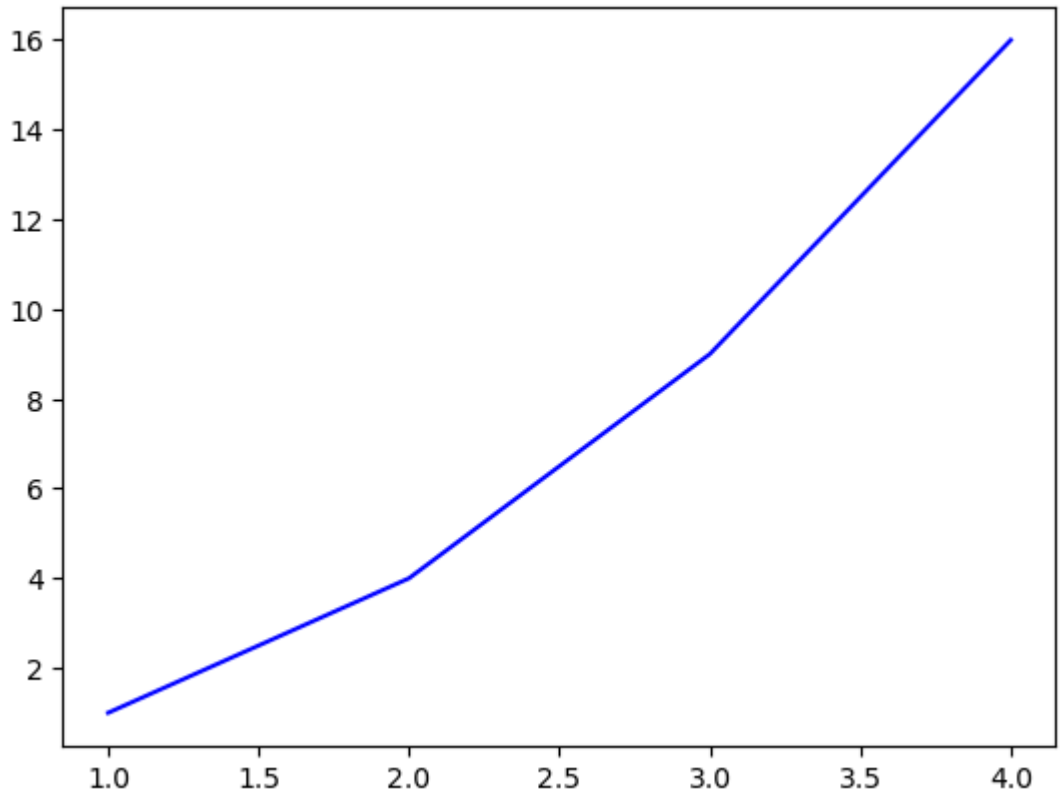
```
Axes(0.125,0.11;0.775x0.77)
```

Visualization with Pyplot

In [69]:
```
plt.plot([1,2,3,4], c = 'blue') # Line Plot
# These are the y-coordinates of the plots

# The x-coordinates are automatically generated

plt.ylabel('Numbers') # Adds a name to the y-axis
plt.show()
```

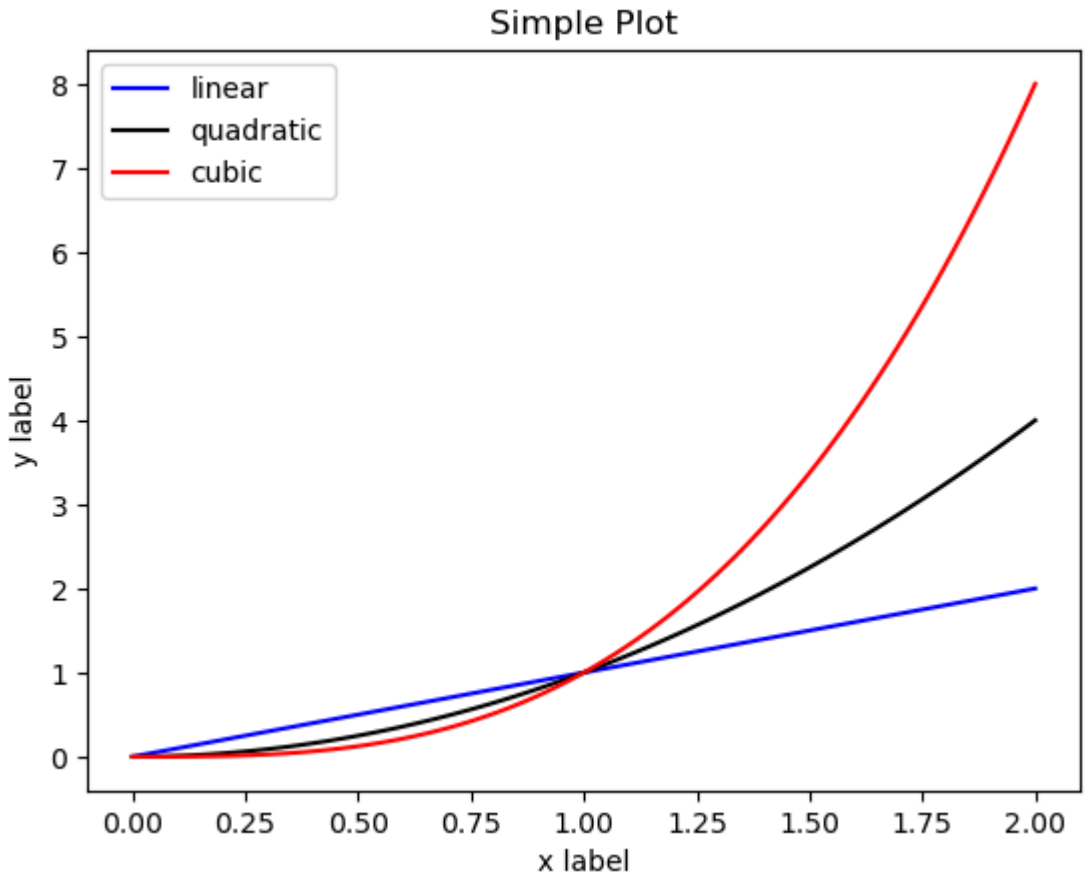


In [75]:
```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], c = 'blue')
# This has x-coordinates and y-coordinates

# It plots like (1,1), (2, 4), (3, 9), (4, 16)

plt.show()
```

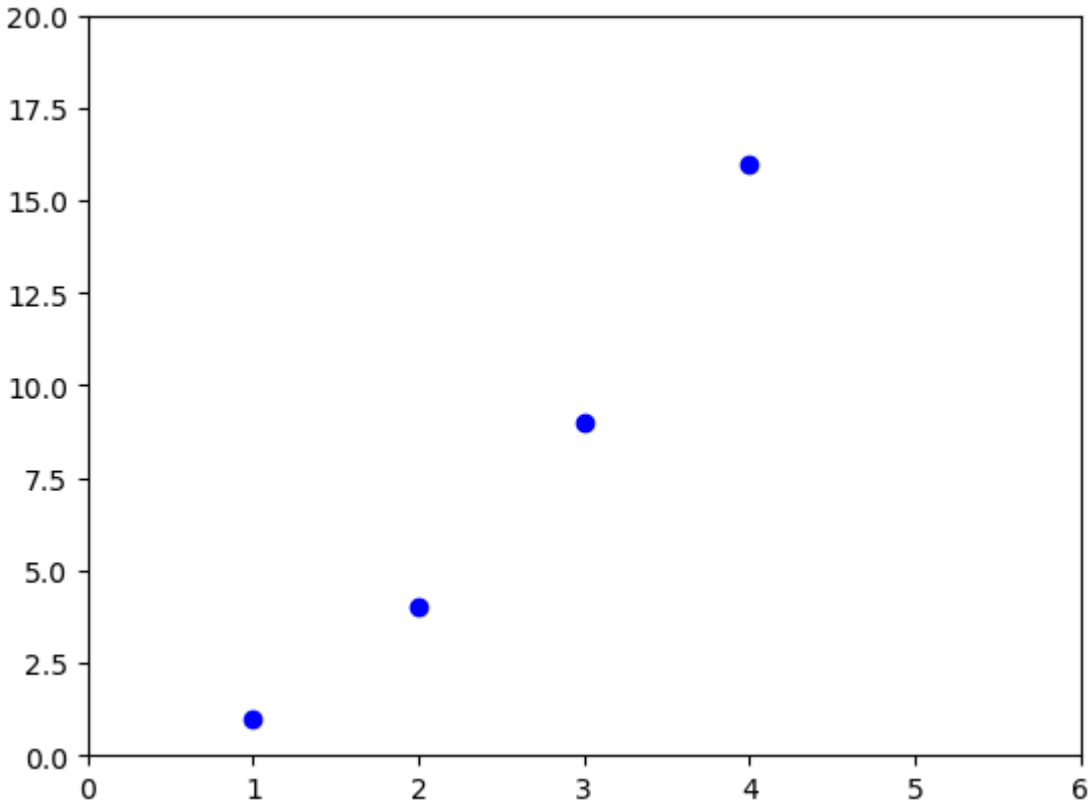


In [79]:
```
x = np.linspace(0, 2, 100)
# This creates an array of 100 values from 0 to 2
```

```
plt.plot(x, x, label='linear', c = 'blue') # Plots a linear function y=x, with l
plt.plot(x, x**2, label='quadratic', c = 'black') # Plots a quadratic function y
plt.plot(x, x**3, label='cubic', c = 'red') # Plots a cubic function y=x**3, wit

plt.xlabel('x label') # Name for x-axis
plt.ylabel('y label') # Name for y-axis

plt.title("Simple Plot") # Heading/Title for the plot

plt.legend() # Gives context

plt.show()
```



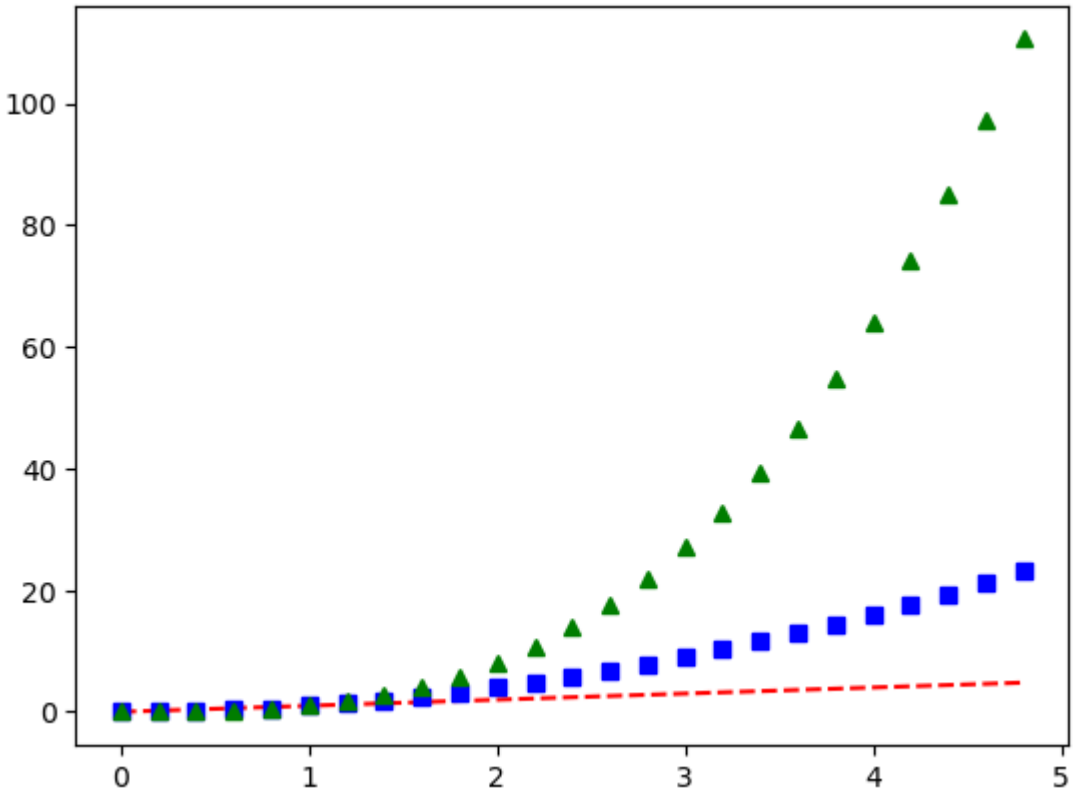
In [95]:
```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'bo')
# This has x-coordinates and y-coordinates
# Colour is Blue, Marker is Circle

# It plots like (1, 1), (2, 4), (3, 9), (4, 16)

plt.axis([0, 6, 0, 20])
# This sets the ranges for x-axis and y-axis

# It sets like x-axis ranges from 0 to 6
# It sets like y-axis ranges from 0 to 20

plt.show()
```



Working with Numpy Arrays

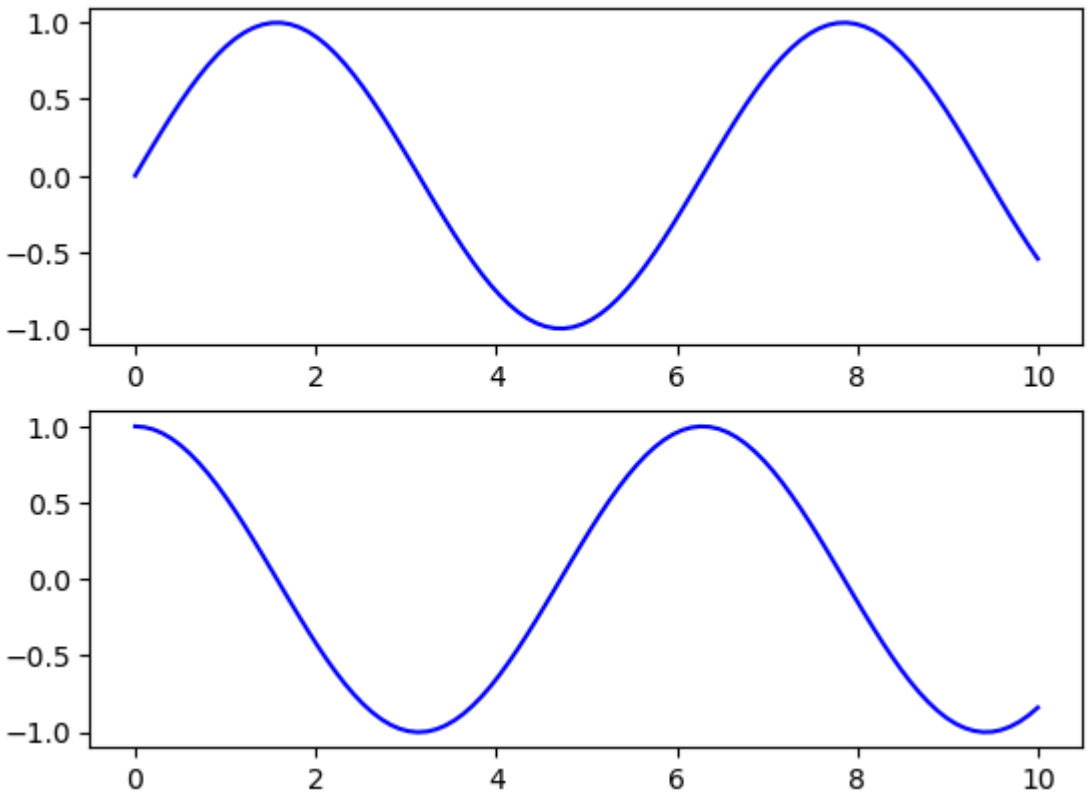
In [102…

```
t = np.arange(0., 5., 0.2)
# Generates an array starting from 0.0, ending till 5.0, with a step of 0.2

plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
# y = t, to plot linear function, colur red, dashed line
# y = t**2, to plot quadratic function, colour blue, marker sqaure
# y = t**3, to plot cubic function, colour gree, marker triangle

plt.show()
```



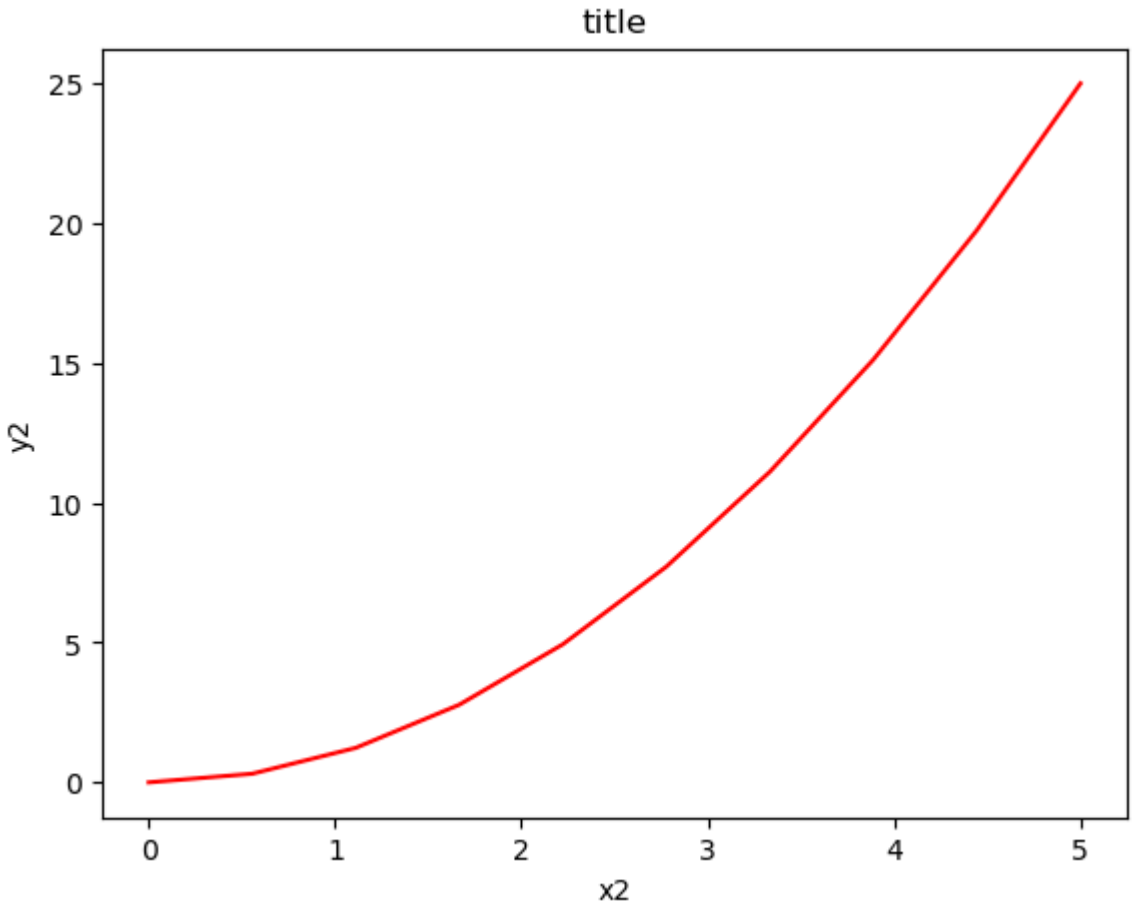
In [108…

```
fig, ax = plt.subplots(2)
# Creates a grid of subplots with nrows rows and ncols columns
# nrows = 2, ncols = 1 (default when not specified)
# ax is an array of two Axes objects

ax[0].plot(x1, np.sin(x1), 'b-') # y = sin(x), blue line
ax[1].plot(x1, np.cos(x1), 'b-') # y = cos(x), blue line

plt.show()
```



In [122…

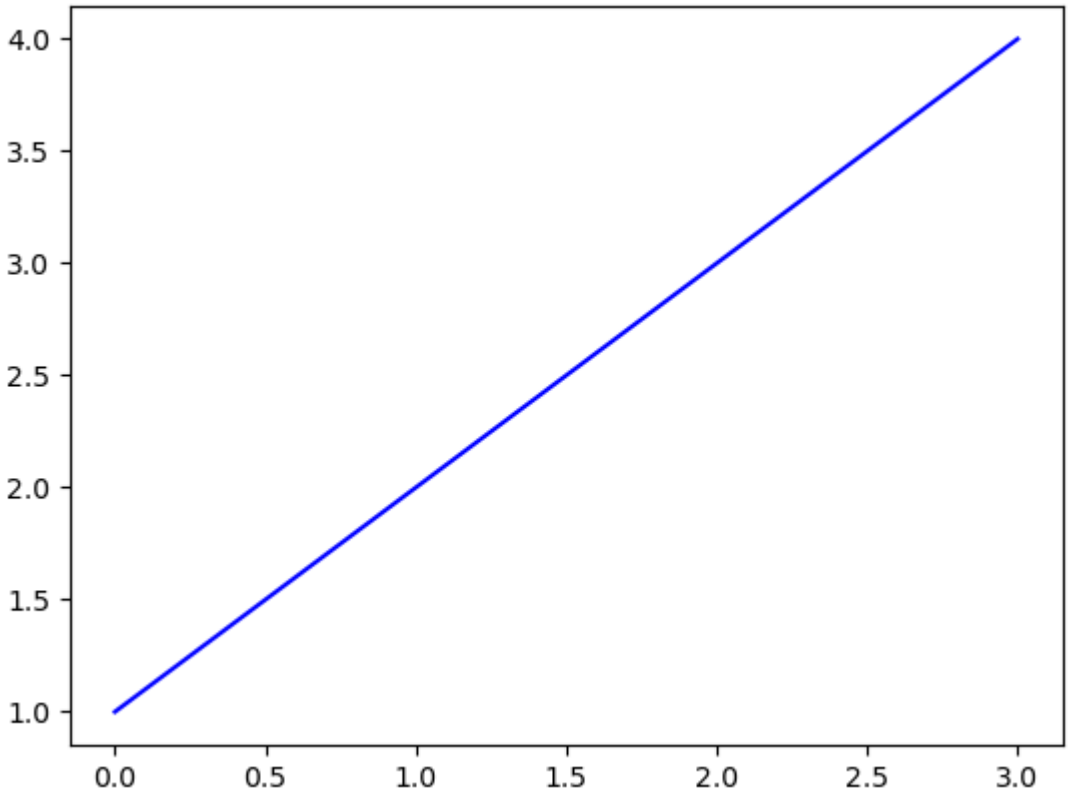
```
fig = plt.figure()

x2 = np.linspace(0, 5, 10)
# This creates an array of 10 values from 0 to 5

y2 = x2 ** 2 # Applying the square operation to each value in x2

axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
# [left, bottom, width, height]
# adds some modifications at above positions

axes.plot(x2, y2, 'r') # Plots the data on x-axis, y-axis, with red line

axes.set_xlabel('x2') # Name for x-axis
axes.set_ylabel('y2') # Name for y-axis

axes.set_title('title') # Adds a Name/Title for the plot

plt.show()
```



First Plot with Matplotlib

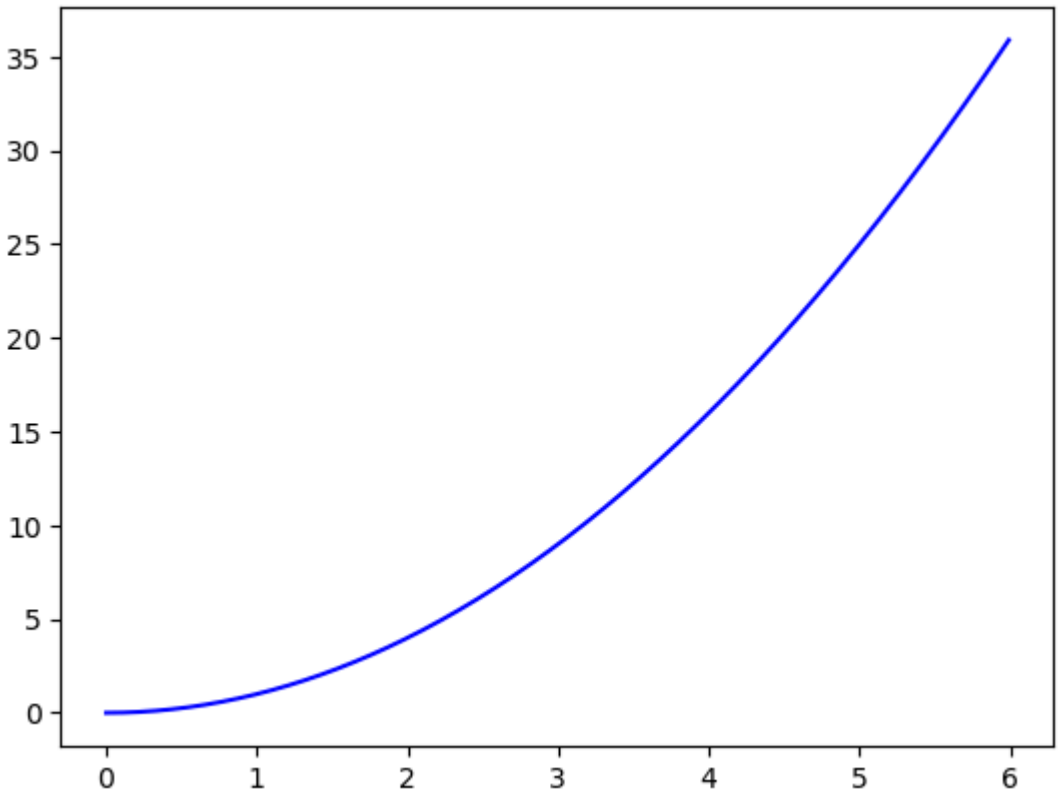
In [125…

```
plt.plot([1,2,3,4], 'b-') # Line Plot
# These are the y-coordinates of the plots

# The x-coordinates are automatically generated

plt.show()
```



In [127…

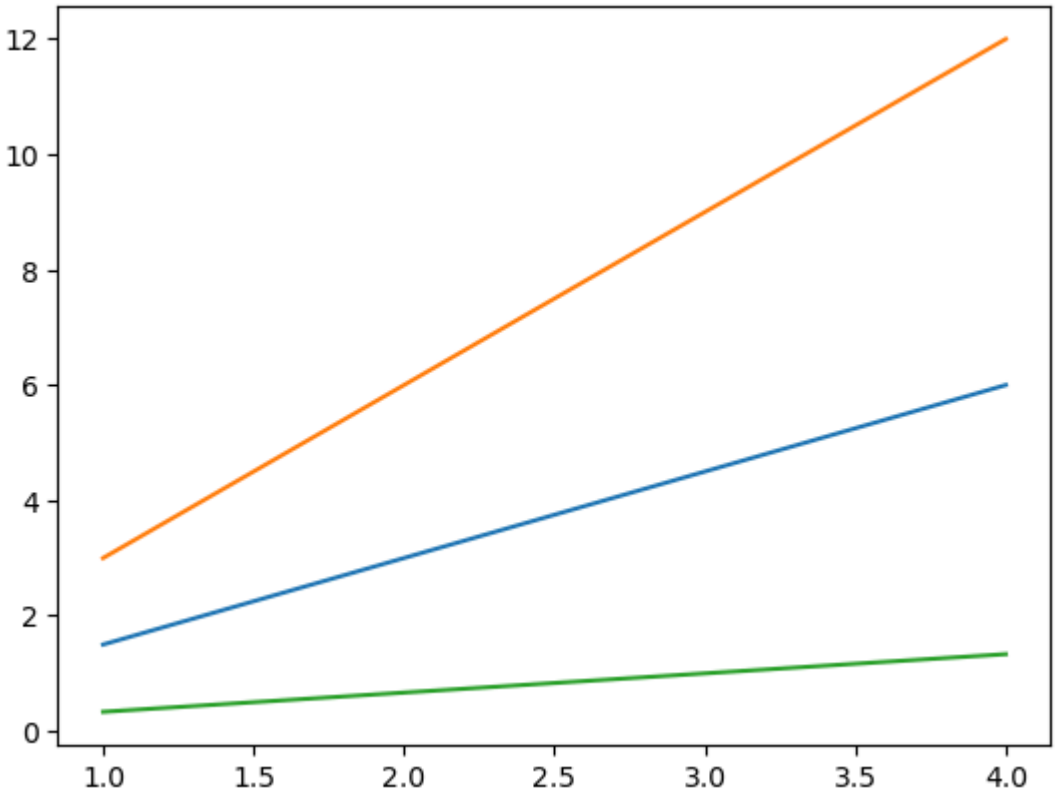
```
x3 = np.arange(0.0, 6.0, 0.01)
# Generates an array values starting from 0.0, ending till 6.0, with a step of 0

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

plt.show()
```



In [131…

```
x4 = range(1, 5)
# Creates a range of values from 1 to 4
```

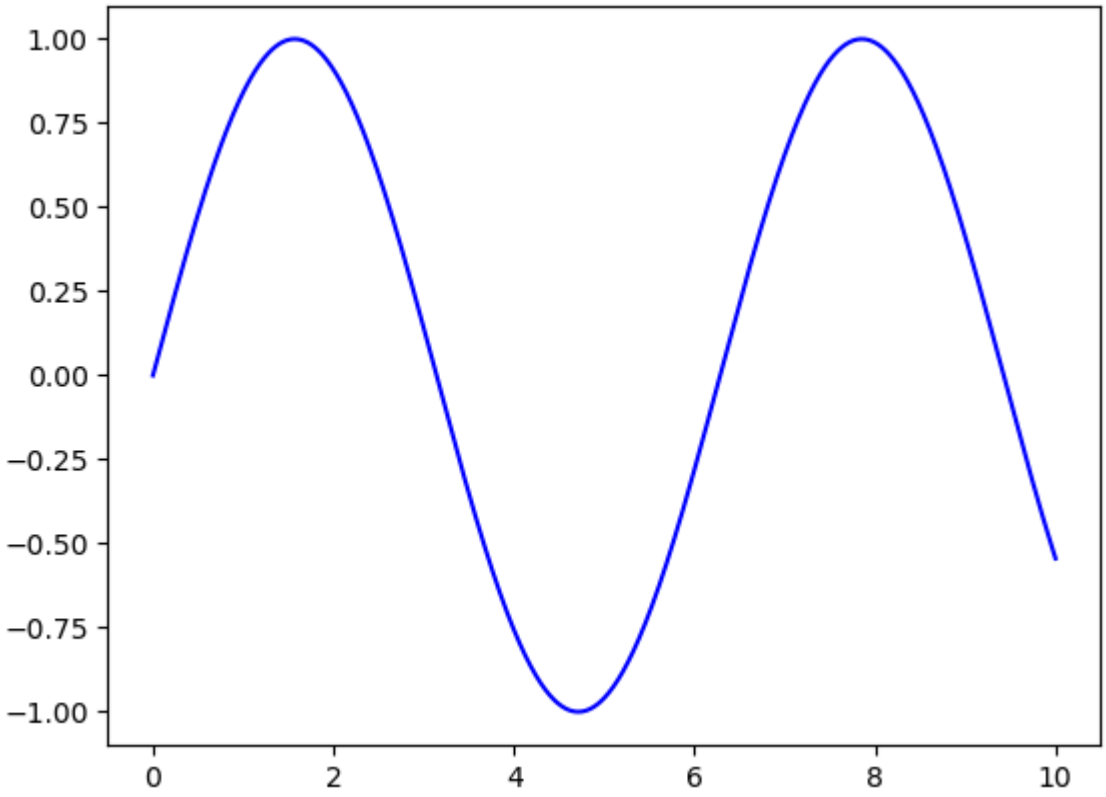
```
plt.plot(x4, [xi*1.5 for xi in x4]) # Multiplication with 1.5 to the values in x

plt.plot(x4, [xi*3 for xi in x4]) # Multiplication with 3 to the values in x4

plt.plot(x4, [xi/3.0 for xi in x4]) # Division with 3.0 to the values in x4

plt.show()
```



Line Plot

In [141…

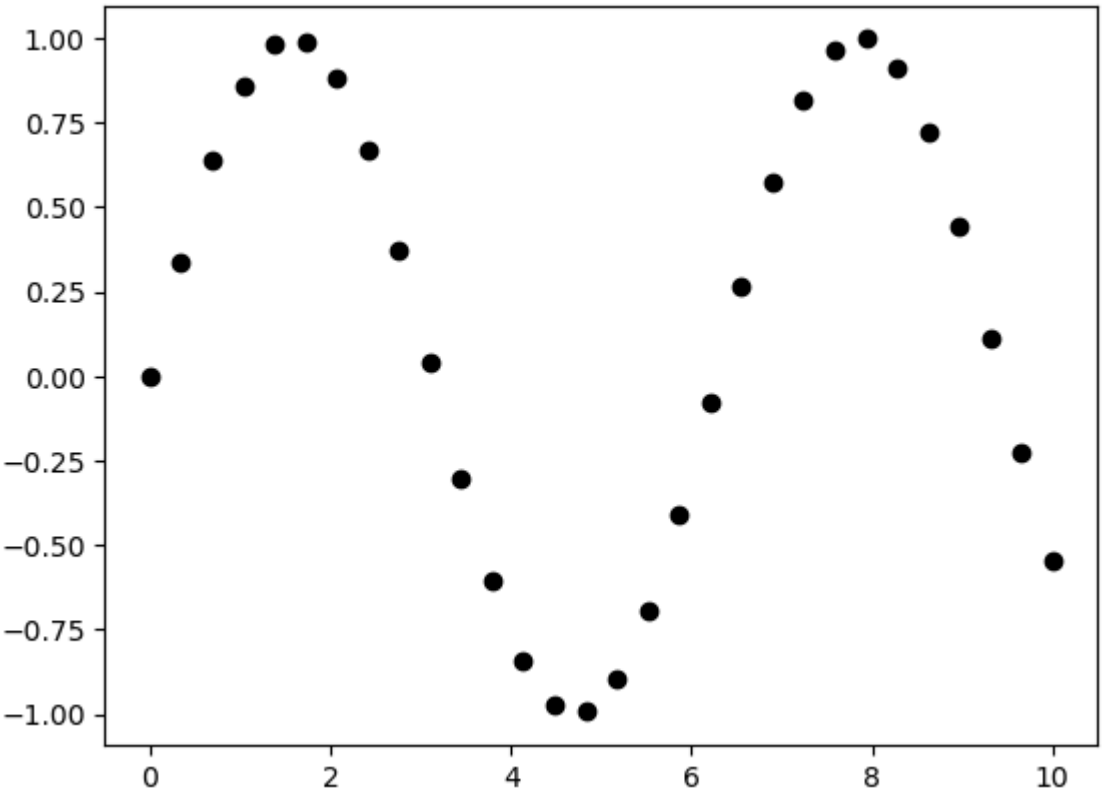
```
# Create figure and axes first
fig = plt.figure()

ax = plt.axes()

# Declare a variable x5
x5 = np.linspace(0, 10, 1000)
# This generates 1000 values from 0 to 10

# Plot the sinusoid function
ax.plot(x5, np.sin(x5), 'b-') # plots the sin function

plt.show()
```



Scatter Plot

In [143…

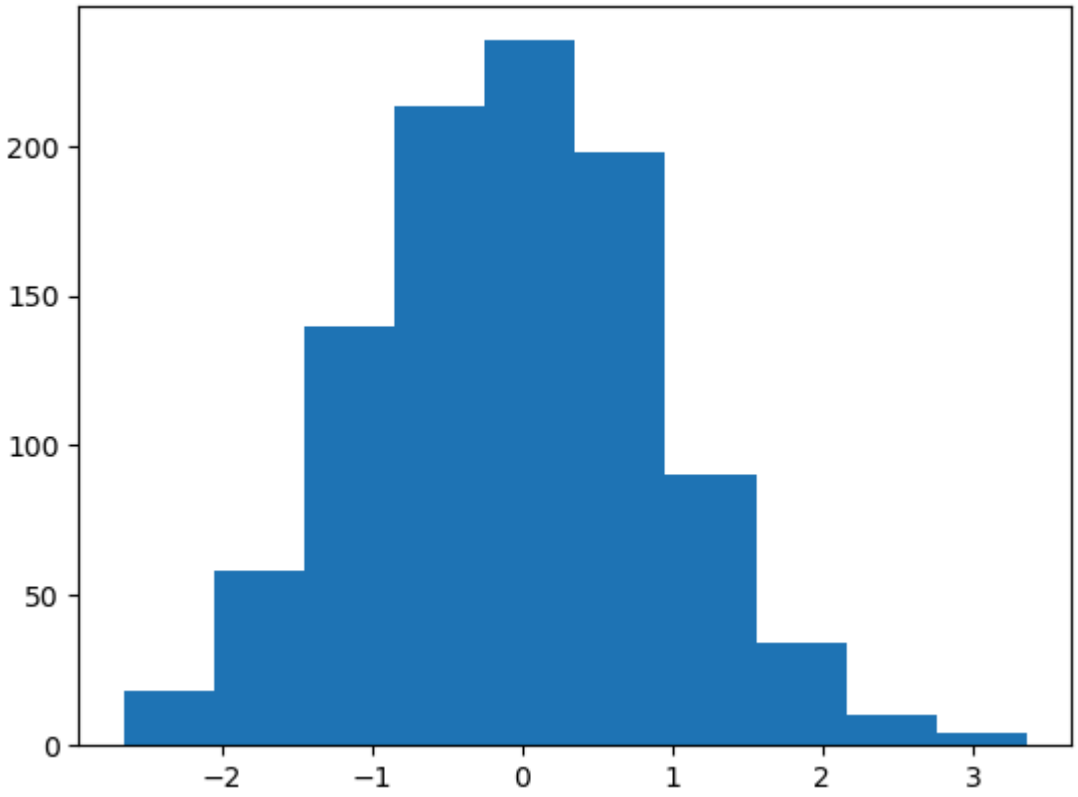
```
x7 = np.linspace(0, 10, 30)
# This generated 30 values between 0 to 10

y7 = np.sin(x7)
# This peforms the sine function on the x7 values

plt.plot(x7, y7, 'o', color = 'black');
# Plots on x-axis (x7), y-axis (y7), marker 'o', colour black

plt.show()
```



Histogram

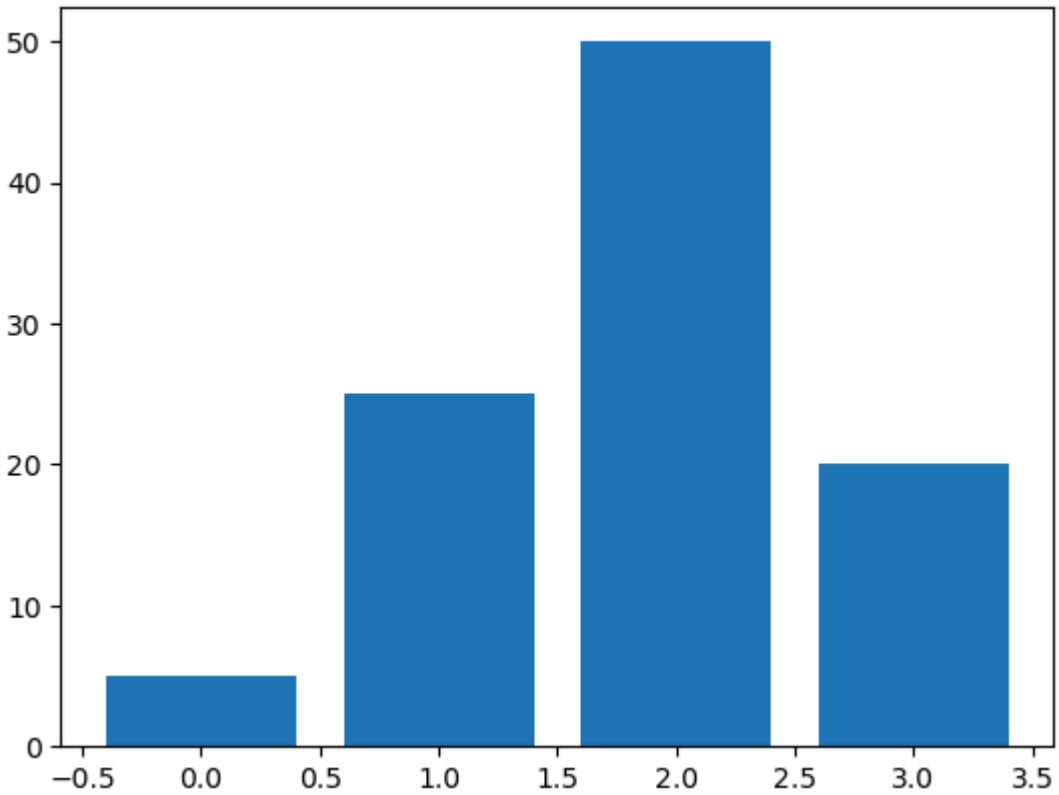
In [150…

```
data1 = np.random.randn(1000)
# Creates 1000 random numbers

plt.hist(data1); # plots the hist data of those numbers

plt.show()
```



In [152…

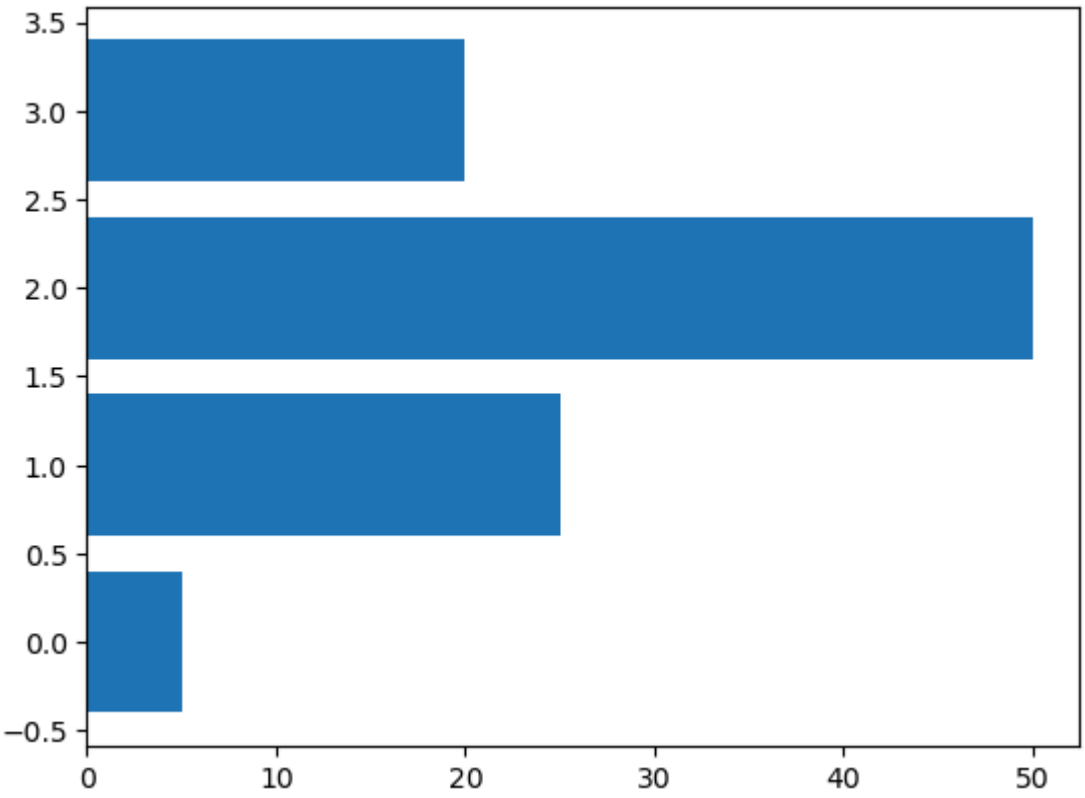
```
data2 = [5. , 25. , 50. , 20.]
# This will be used as heights in the y-axis

plt.bar(range(len(data2)), data2)
# The length will be used in x-axis

plt.show()
```



Horizontal Bar Chart

In [155…

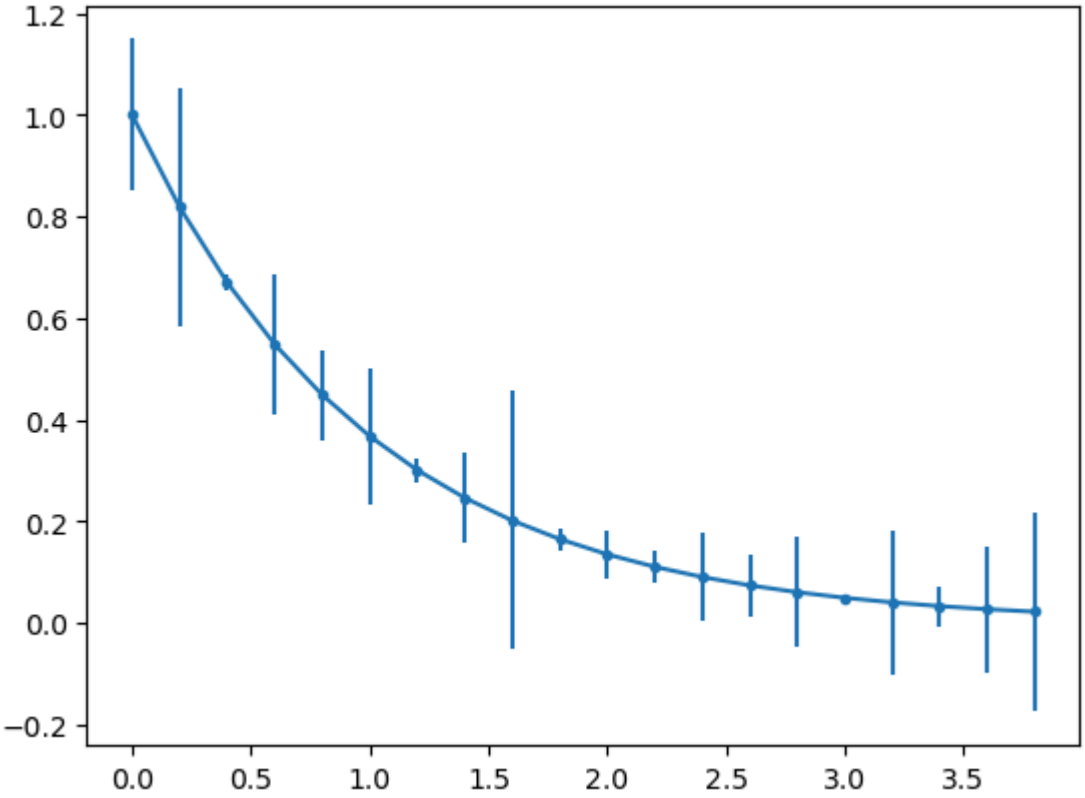
```
data2 = [5. , 25. , 50. , 20.]
# This will be used as heights in the y-axis

plt.barh(range(len(data2)), data2)
# The length will be used in x-axis

plt.show()
```



Error Bar Chart

In [158…

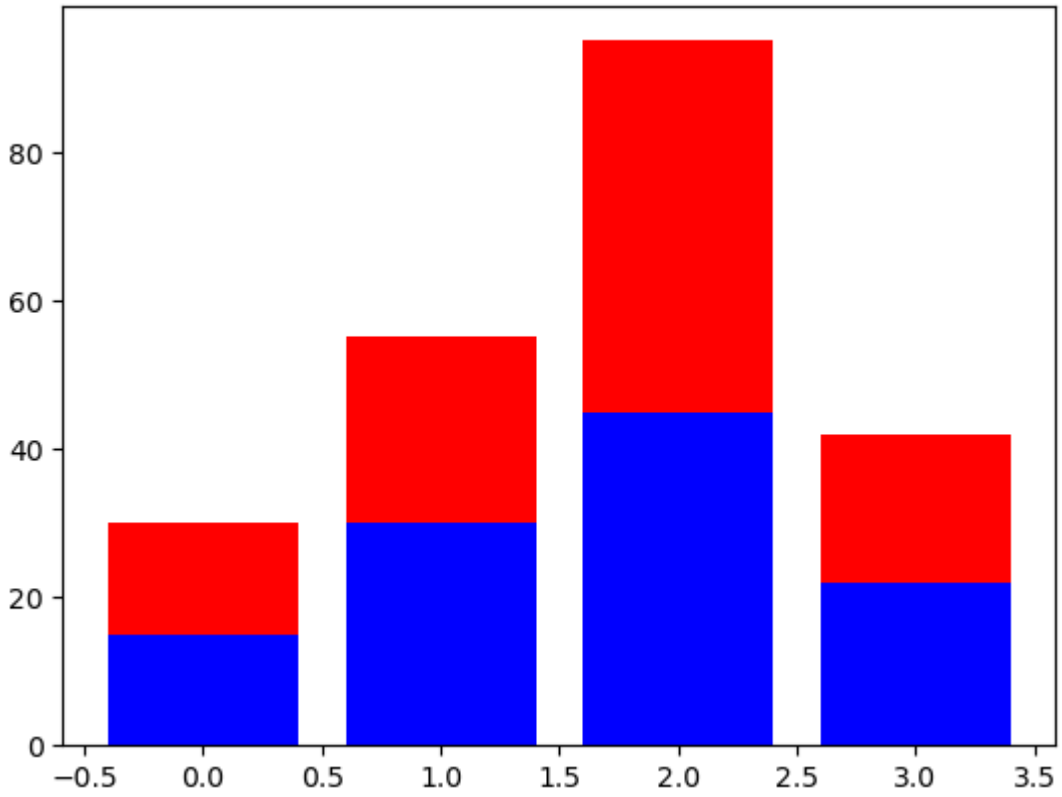
```
x9 = np.arange(0, 4, 0.2)
# Generates value from 0 to 4, with step count of 0.2

y9 = np.exp(-x9) # Applies exponential function to the values in x9

e1 = 0.1 * np.abs(np.random.randn(len(y9))) # Reperesents the errors in the bar

plt.errorbar(x9, y9, yerr = e1, fmt = '.-')
# x9 is x-coordinates, y9 is y-coordinates, e1 is error, format is .-

plt.show();
```



Stacked Bar Chart

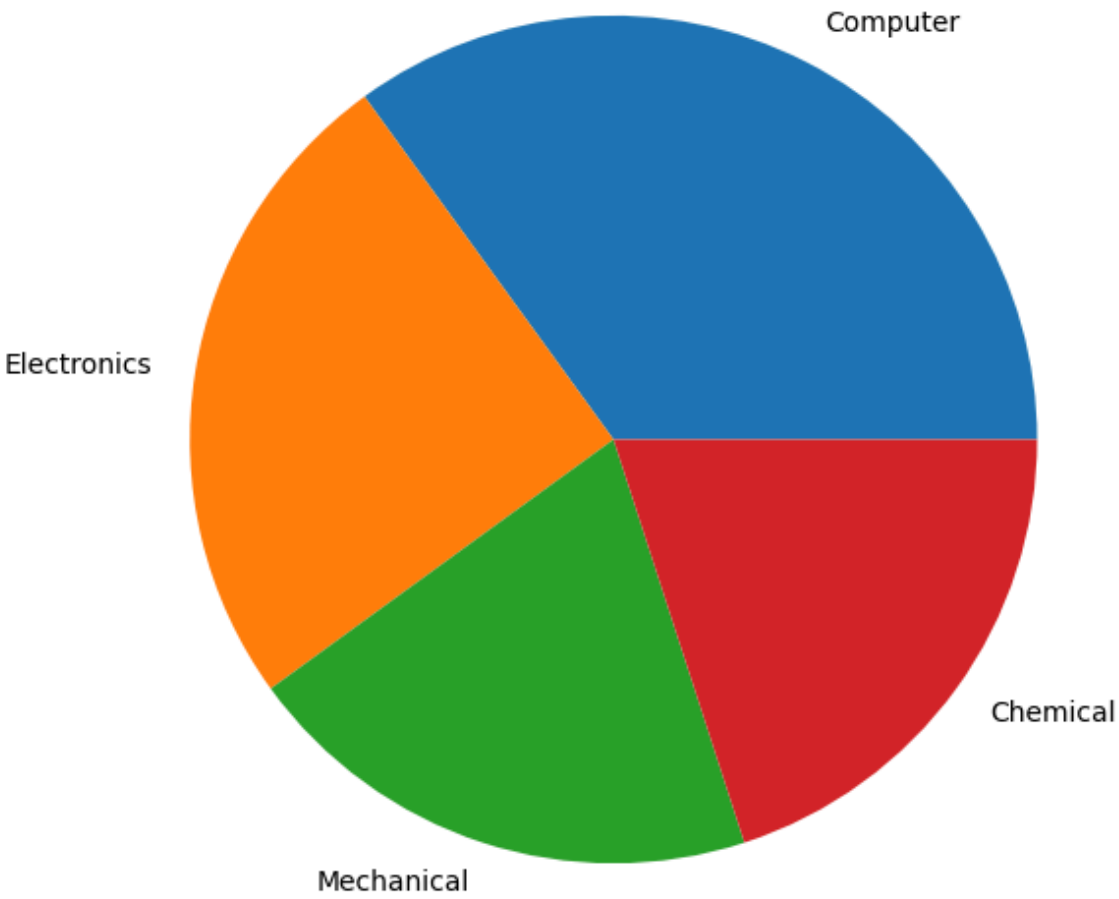
In [161…

```
A = [15., 30., 45., 22.]

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

z2 = range(4) # this represent x-xoordinates

plt.bar(z2, A, color = 'b')
# z2 is x-coordinates, A is y-coordinates, colour blue

plt.bar(z2, B, color = 'r', bottom = A)
# z2 is x-coordinates, B is y-coordinates, colour red, and we are saying A is bo

plt.show()
```



In [163…

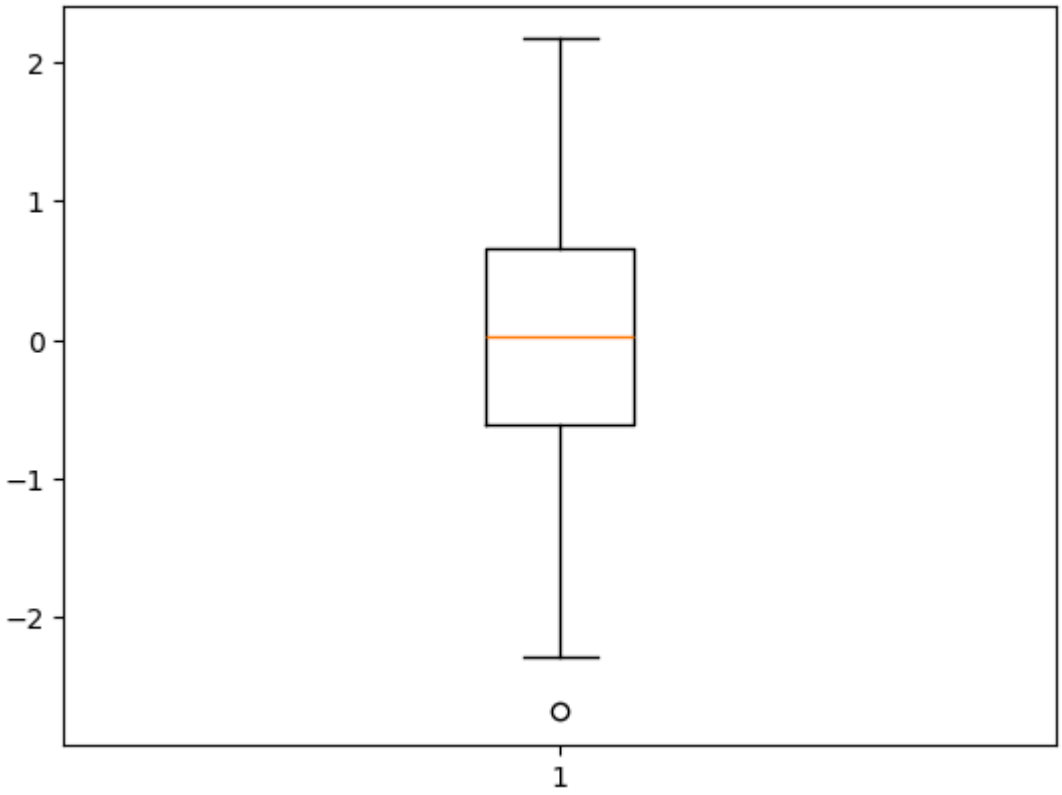
```
plt.figure(figsize=(7,7))
# The size of the figure

x10 = [35, 25, 20, 20]

labels = ['Computer', 'Electronics', 'Mechanical', 'Chemical']
# Labels of each sector respectively to the x10 values

plt.pie(x10, labels=labels) # Plotting the pie chart

plt.show()
```



Box Plot

In [166…

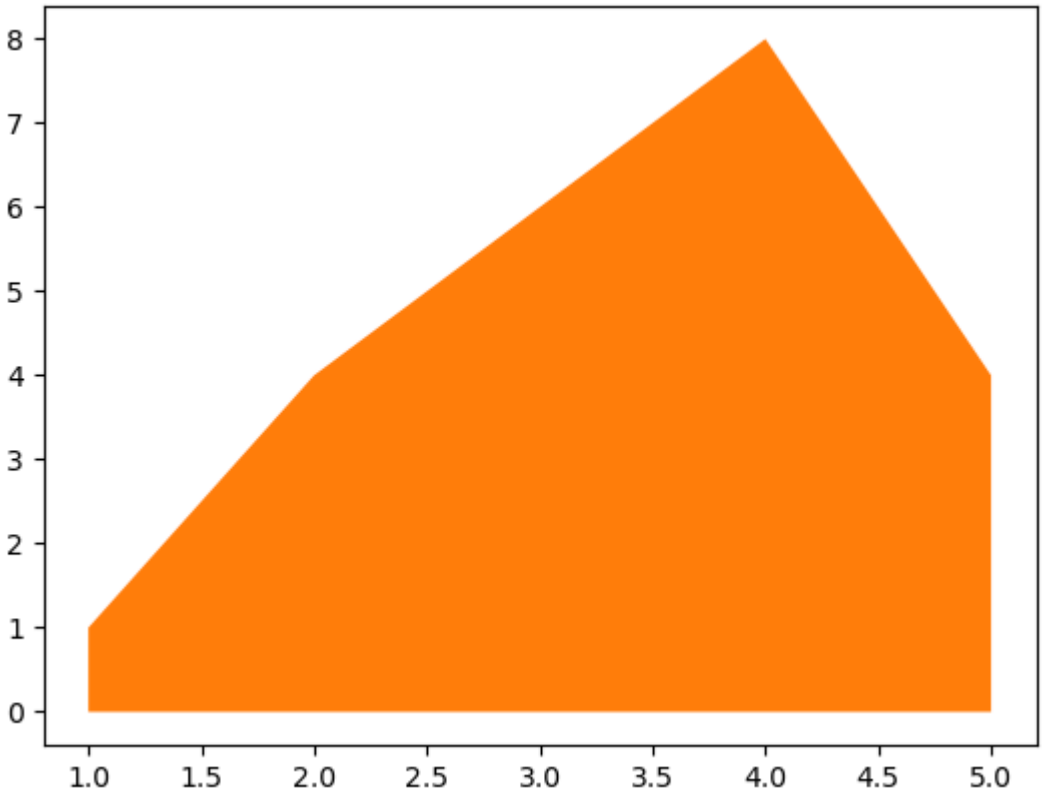
```
data3 = np.random.randn(100)
# Generated 100 random numbers

plt.boxplot(data3)

plt.show();
```



Area Chart

In [171…

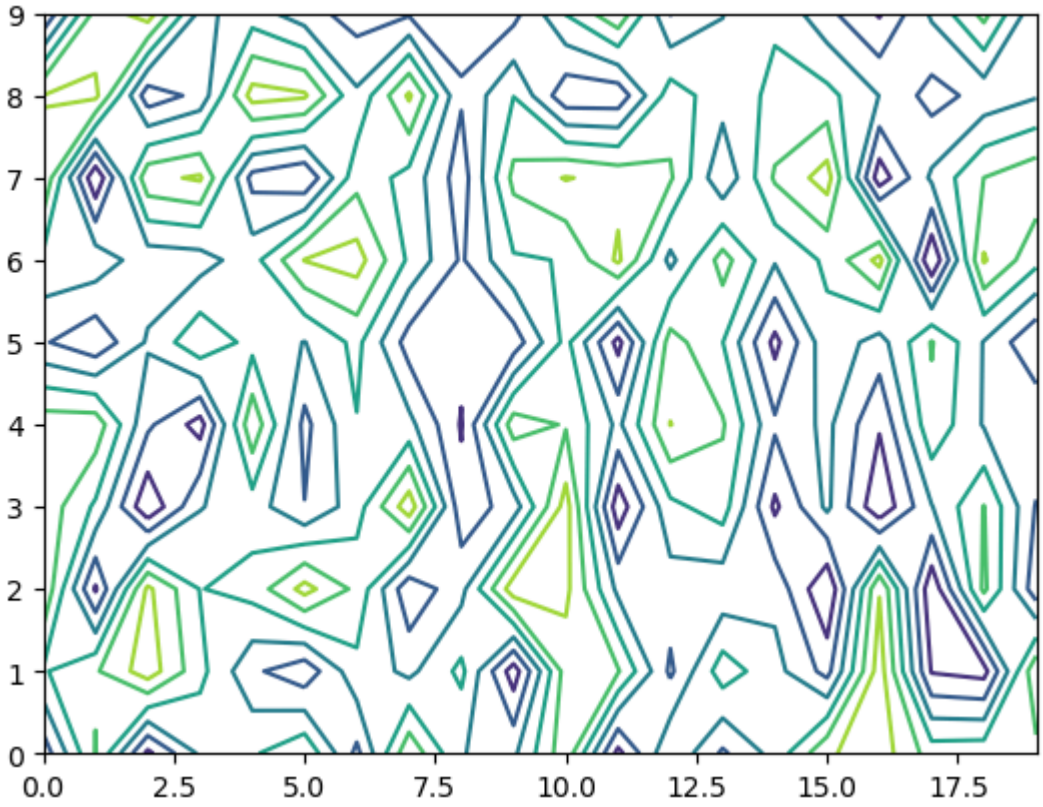
```
x12 = range(1, 6)
y12 = [1, 4, 6, 8, 4]

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



Contour Plot

In [174…

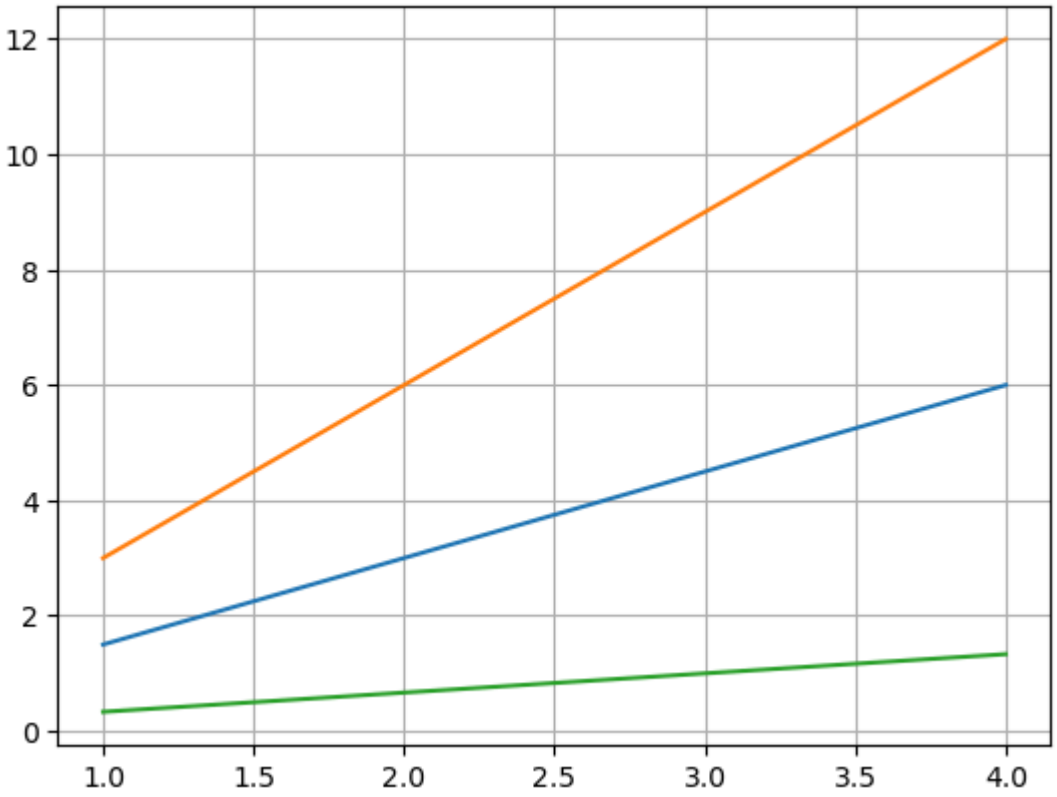
```
matrix1 = np.random.rand(10, 20)
# Generates random values from 10 to 19

cp = plt.contour(matrix1)

plt.show()
```



Adding a Grid

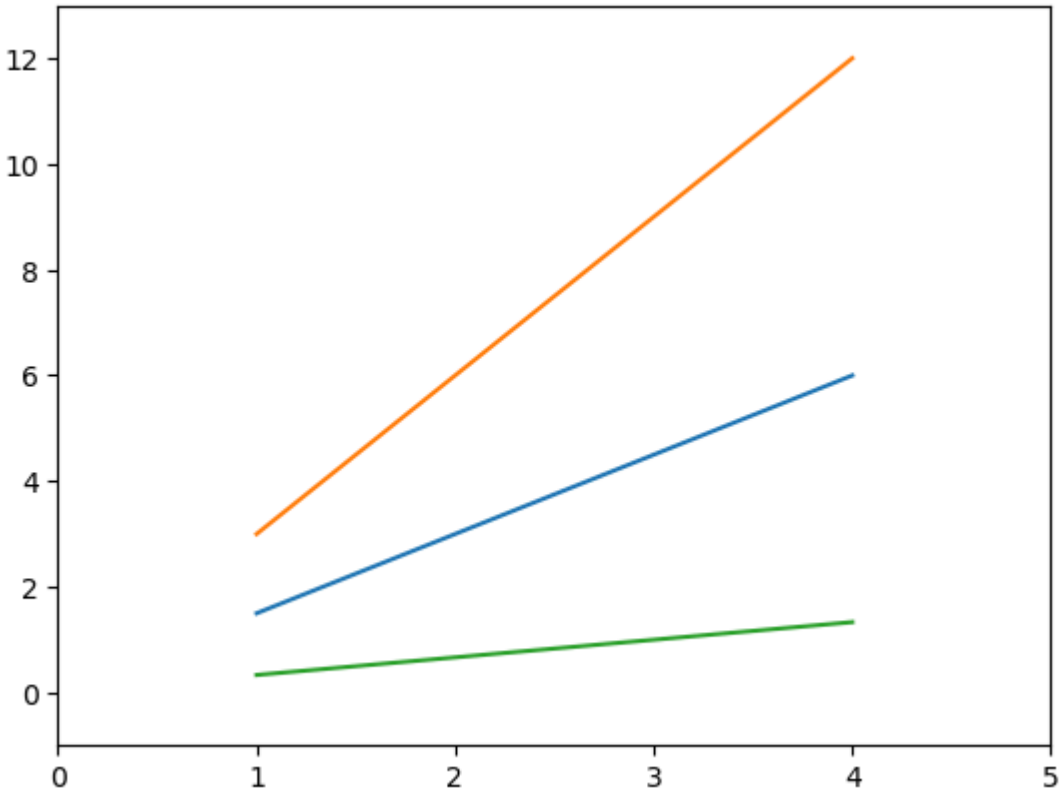
In [181…

```
x15 = np.arange(1, 5)
# Values between 1 to 4

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

plt.grid(True)

plt.show()
```



In [185…

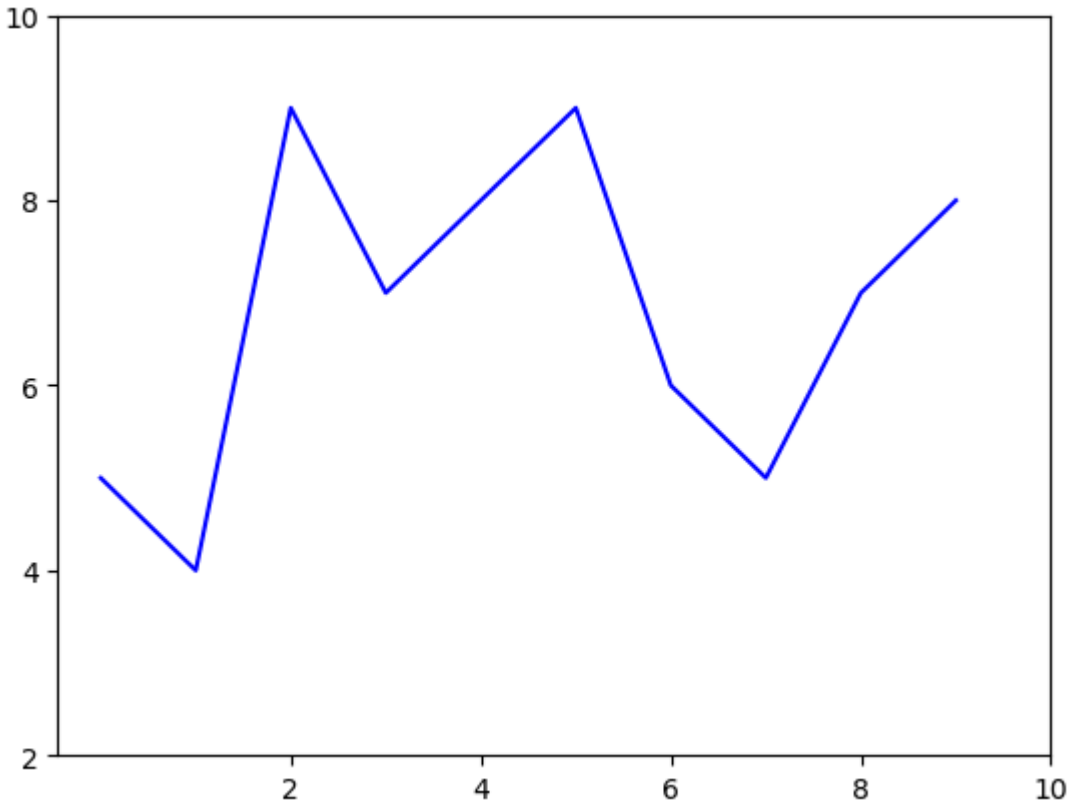
```
x15 = np.arange(1, 5)
# Values from 1 to 4

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])
# Sets min, max values for x-axis and y-axis

plt.show()
```



Handling X and Y Ticks

In [190…

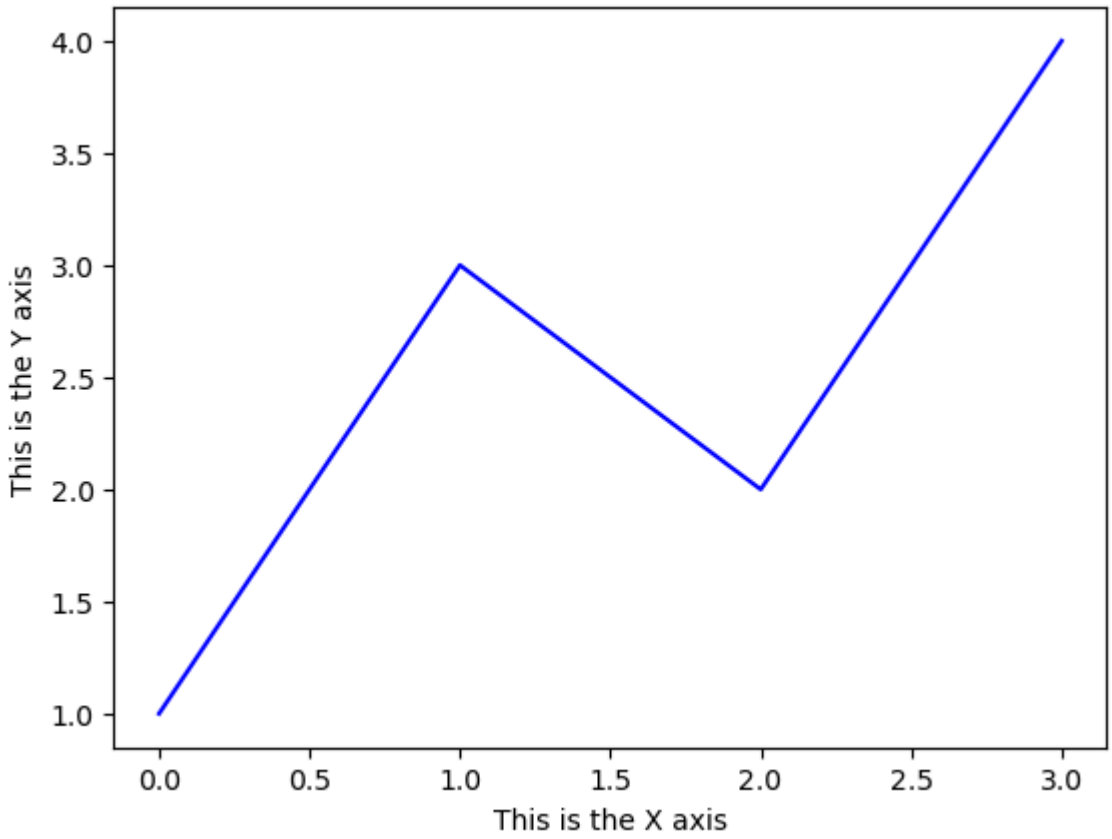
```
u = [5, 4, 9, 7, 8, 9, 6, 5, 7, 8]

plt.plot(u, c='blue')

plt.xticks([2, 4, 6, 8, 10]) # Specifies where the tick marks should appear on t
plt.yticks([2, 4, 6, 8, 10]) # Specifies where the tick marks should appear on t

plt.show()
```



Adding Labels

In [197…

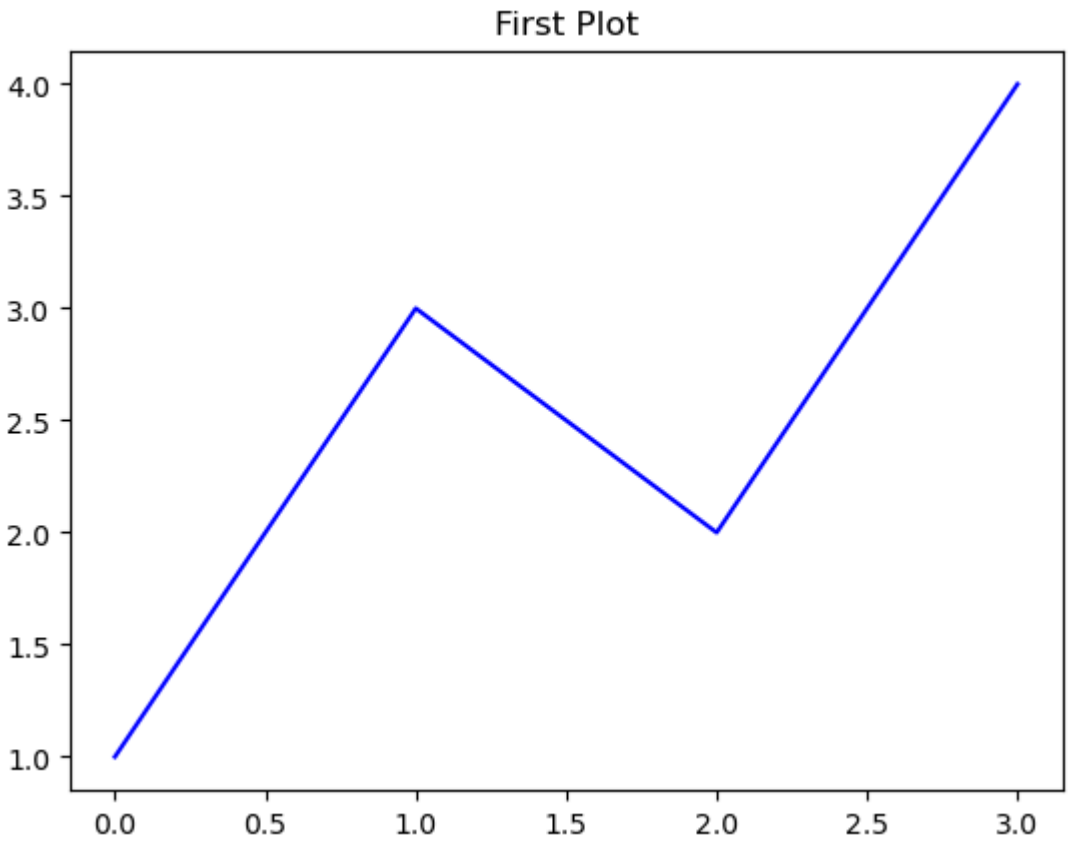
```
plt.plot([1, 3, 2, 4], c = 'Blue')

plt.xlabel('This is the X axis') # Name for the x-axis

plt.ylabel('This is the Y axis') # Name for the y-axis

plt.show()
```



Adding a Plot

In [200…

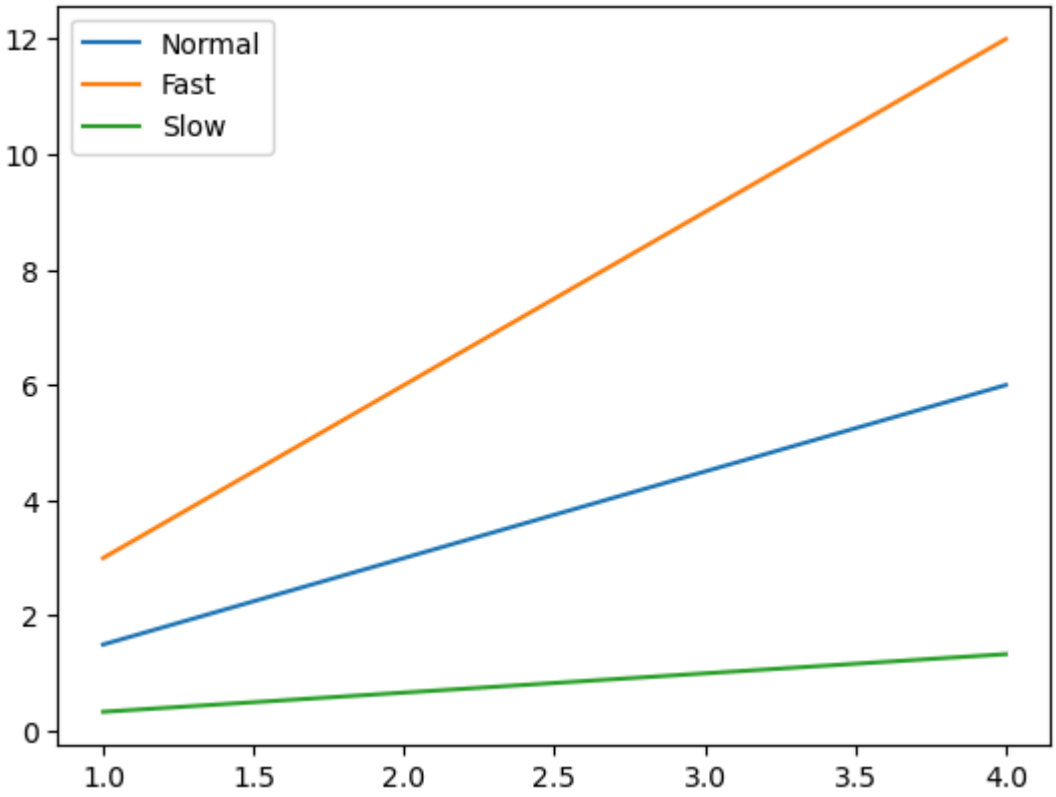
```
plt.plot([1, 3, 2, 4], c = 'Blue')

plt.title('First Plot') # Adds a title

plt.show()
```

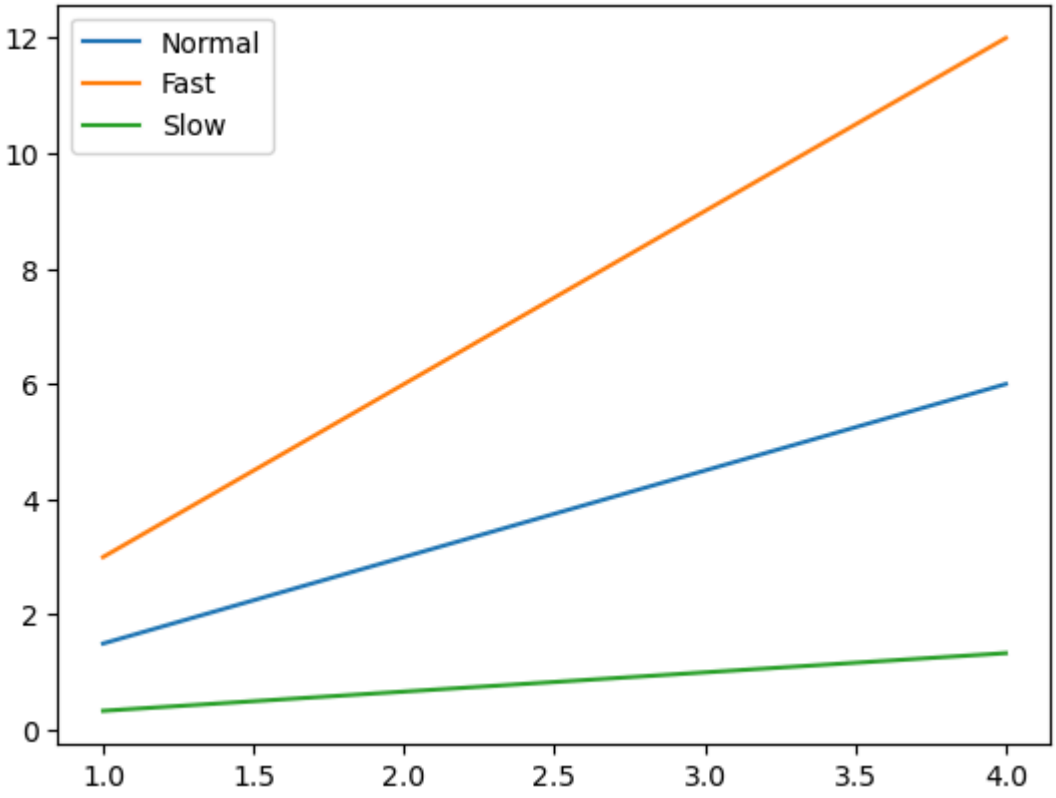


Adding a Legend

In [205…

```
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']) # Adds what each line represents

plt.show()
```



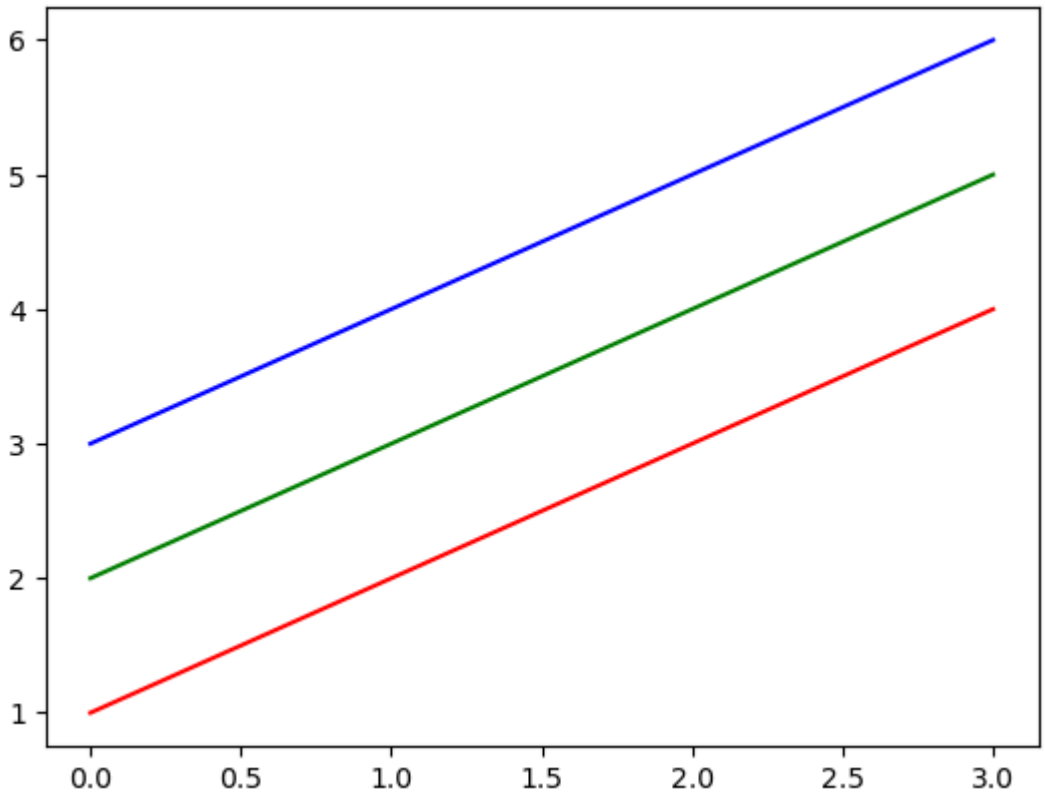


Control Colours

In [208…

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

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

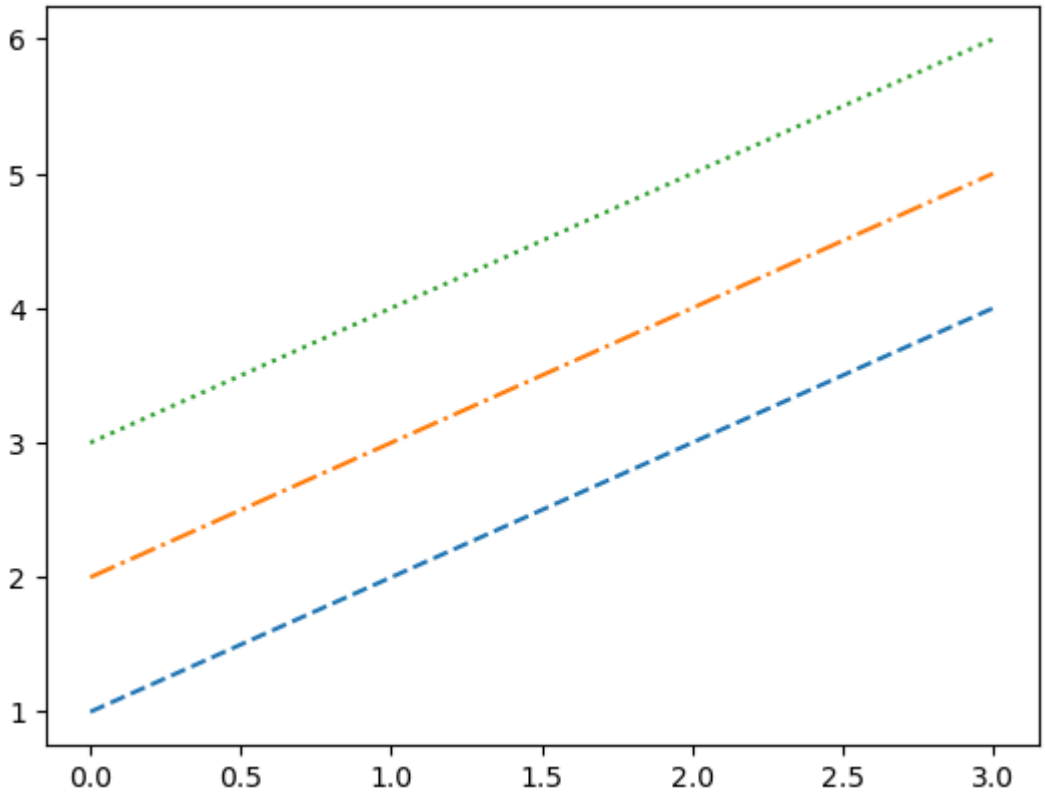
```
plt.show()
```



Control Line Styles

In [211…

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

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

plt.show()
```



In [ ]: