Visualization with Matplotlib

Matplotlib is a multi-platform data visualization library built on NumPy arrays, and designed to work with the broader SciPy stack.

One of Matplotlib's most important features is its ability to play well with many operating systems and graphics backends.

Matplotlib supports dozens of backends and output types, which means you can count on it to work regardless of which operating system you are using or which output format you wish.

This cross-platform, everything-to-everyone approach has been one of the great strengths of Matplotlib. It has led to a large user base, which in turn has led to an active developer base and Matplotlib's powerful tools and ubiquity within the scientific Python world.

Importing Matplotlib

Just as we use the np shorthand for NumPy and the pd shorthand for Pandas, we will use some standard shorthands for Matplotlib imports:

In [1]:

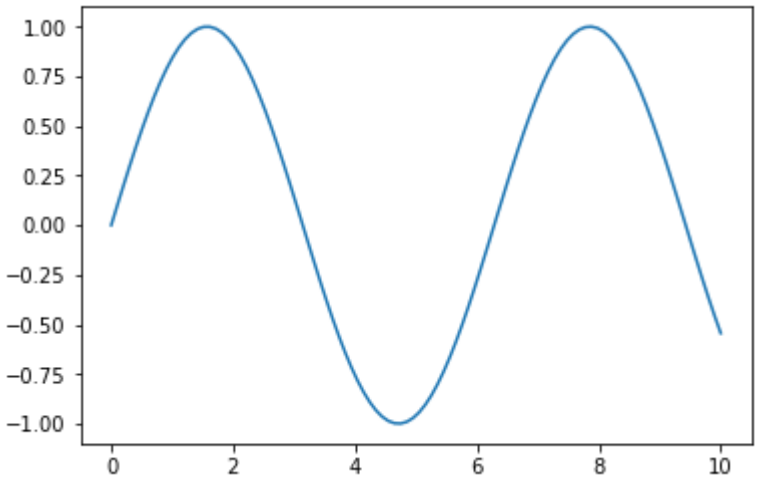
```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

The plt interface is what we will use most often

Drawing a simple Line plot

In [4]:

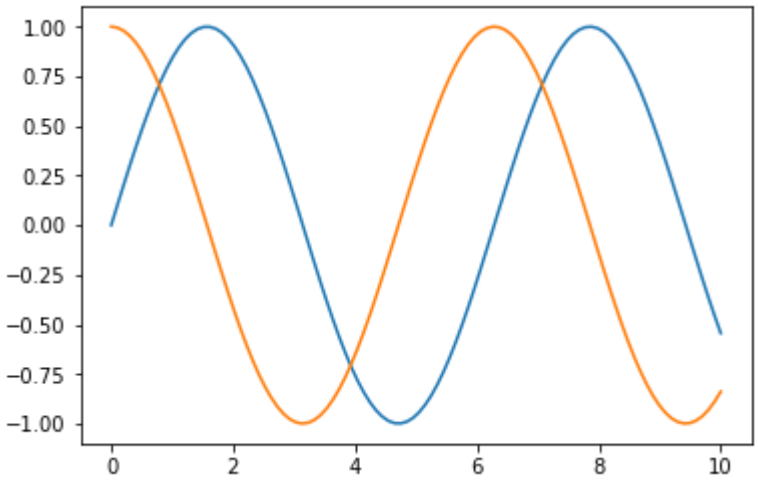
```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)

# Drawing sine curve
plt.plot(x, np.sin(x))

plt.show()
```



Drawing Multi-line Line plot

In [5]:

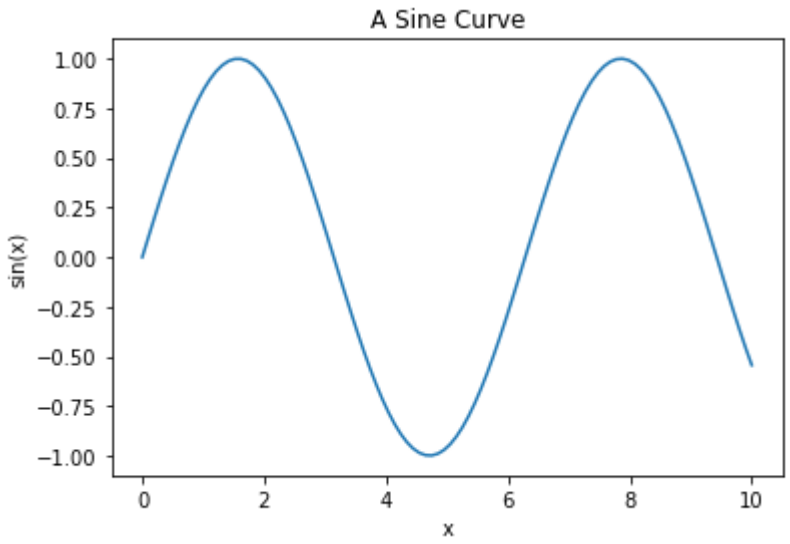
```
x = np.linspace(0, 10, 100)

# Drawing both sine and cosine curves on same plot
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))

plt.show()
```



Adding Labels

In [23]:

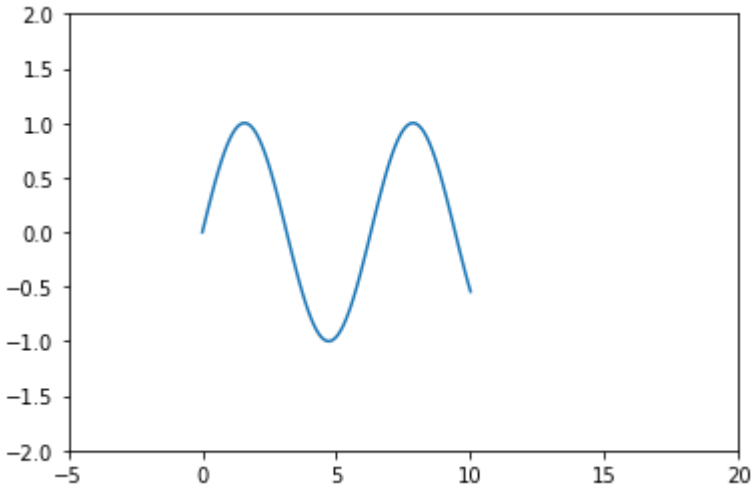
```
x = np.linspace(0, 10, 100)

plt.plot(x, np.sin(x))

# Adding title of graph
plt.title("A Sine Curve")

# Adding label of x-axis and y-axis
plt.xlabel("x")
plt.ylabel("sin(x)")

plt.show()
```



Adding Limits

In [22]:

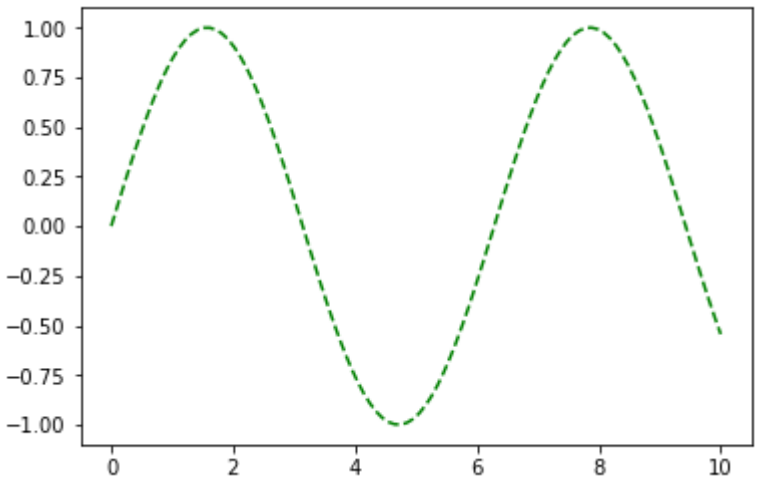
```
x = np.linspace(0, 10, 100)

plt.plot(x, np.sin(x))

# Adding limits of x-axis and y-axis
plt.xlim(-5, 20)
plt.ylim(-2, 2);

plt.show()
```



Specifying Line Color and Line Type

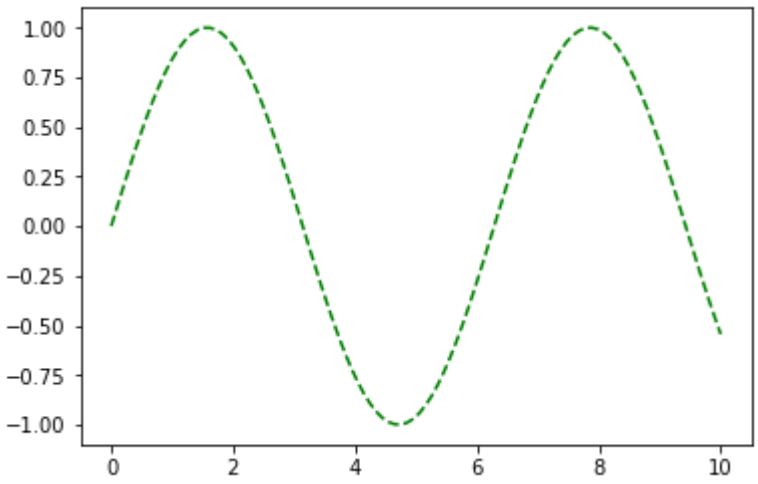
In [9]:

```
x = np.linspace(0, 10, 100)

plt.plot(x, np.sin(x), color = 'green', linestyle='dashed')

plt.show()
```



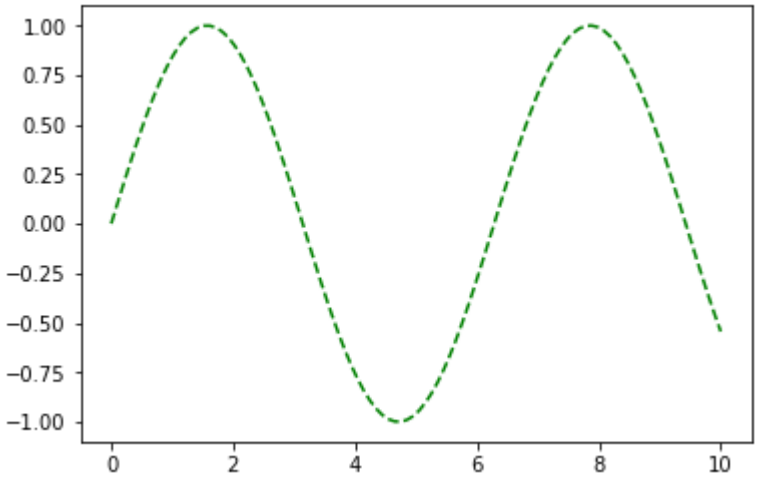
In [17]:

```
x = np.linspace(0, 10, 100)

plt.plot(x, np.sin(x), color = 'g', linestyle='--')

plt.show()
```



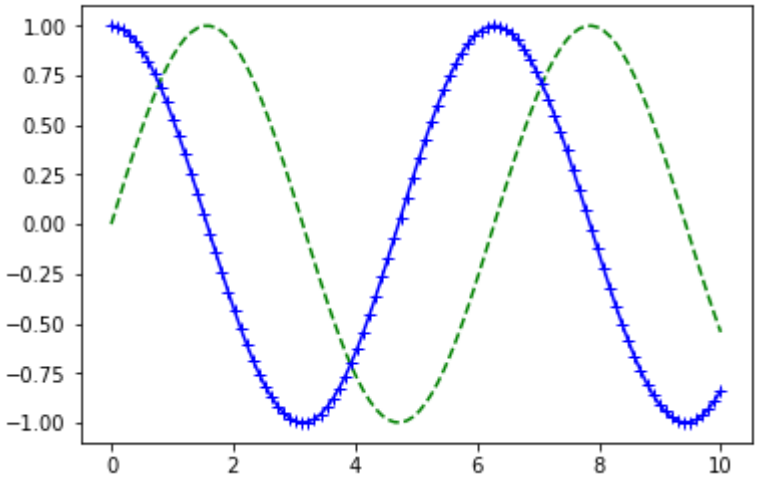
In [19]:

```
x = np.linspace(0, 10, 100)

plt.plot(x, np.sin(x), 'g--')

plt.show()
```



In [21]:

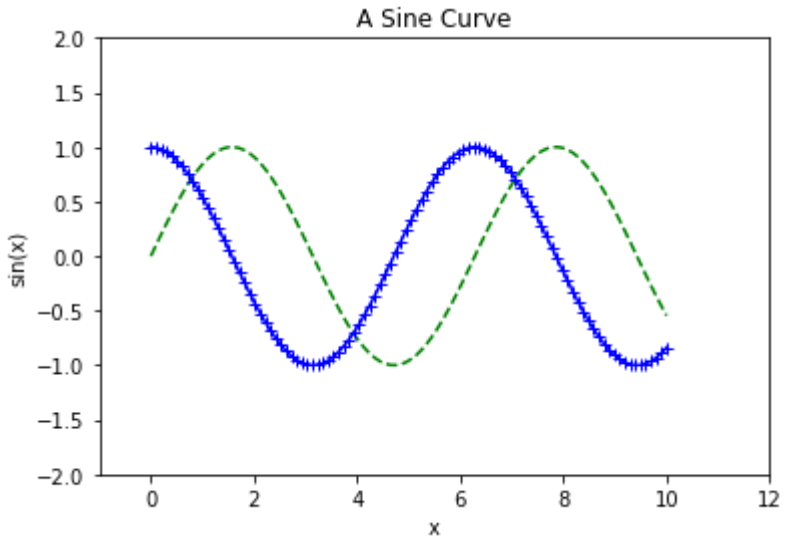
```
x = np.linspace(0, 10, 100)

plt.plot(x, np.sin(x), 'g--')
plt.plot(x, np.cos(x), 'b-+')

plt.show()
```



Drawing a graph with all attributes mentioned above

In [30]:

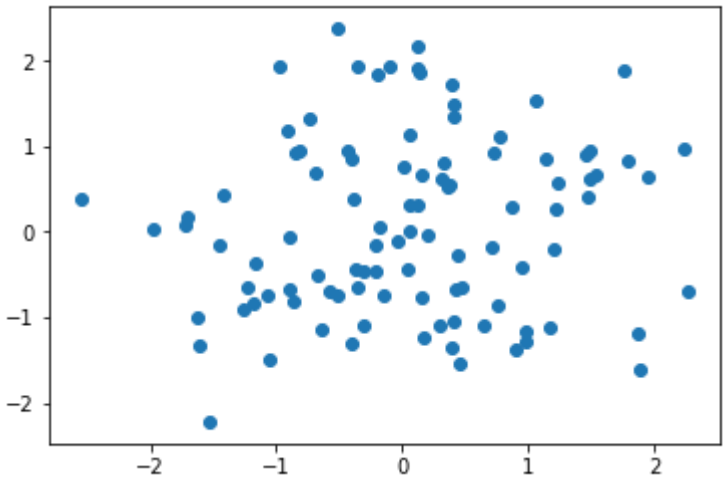
```
x = np.linspace(0, 10, 100)

plt.plot(x, np.sin(x), 'g--')
plt.plot(x, np.cos(x), 'b-+')

# Adding limits of x-axis and y-axis
plt.xlim(-1, 12)
plt.ylim(-2, 2);

# Adding title of graph
plt.title("A Sine Curve")

# Adding label of x-axis and y-axis
plt.xlabel("x")
plt.ylabel("sin(x)")


plt.show()
```



Drawing a Scatter Plot

In [26]:

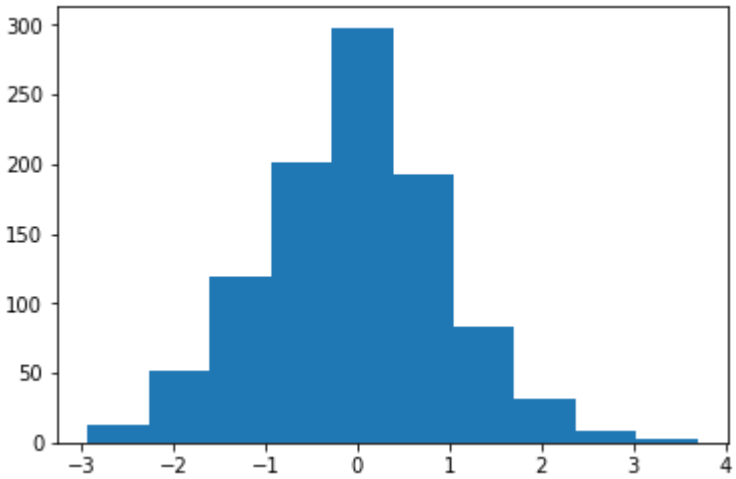
```
rng = np.random.RandomState(0)
x = rng.randn(100)
y = rng.randn(100)

plt.scatter(x, y)

plt.show()
```



Drawing a Histogram

In [27]:

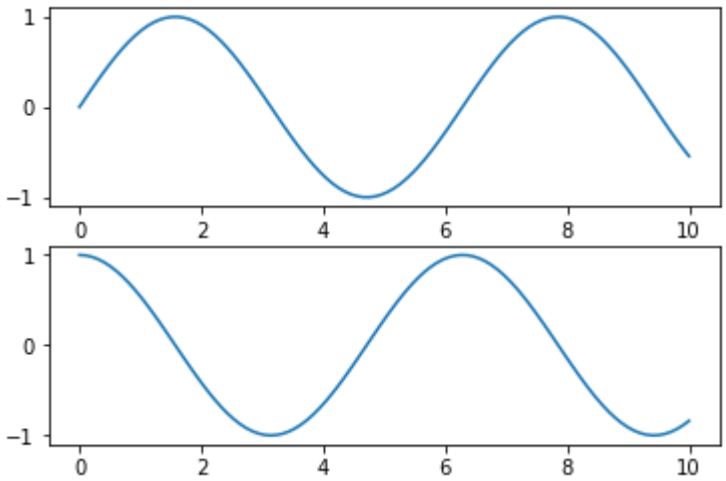
```
data = np.random.randn(1000)

plt.hist(data)

plt.show()
```



sub-graphs

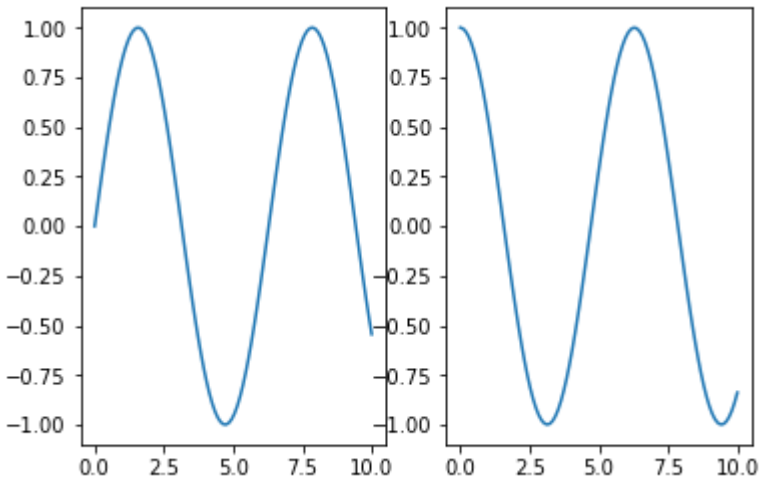
In [32]:

```
# Draw a 2 by 1 graph
x = np.linspace(0, 10, 100)

# create the first of two panels and set current axis
plt.subplot(2, 1, 1) # (rows, columns, panel number)
plt.plot(x, np.sin(x))

# create the second panel and set current axis
plt.subplot(2, 1, 2)
plt.plot(x, np.cos(x));
```



In [33]:

```
# Draw a 1 by 2 graph
x = np.linspace(0, 10, 100)

# create the first of two panels and set current axis
plt.subplot(1, 2, 1) # (rows, columns, panel number)
plt.plot(x, np.sin(x))

# create the second panel and set current axis
plt.subplot(1, 2, 2)
plt.plot(x, np.cos(x));
```

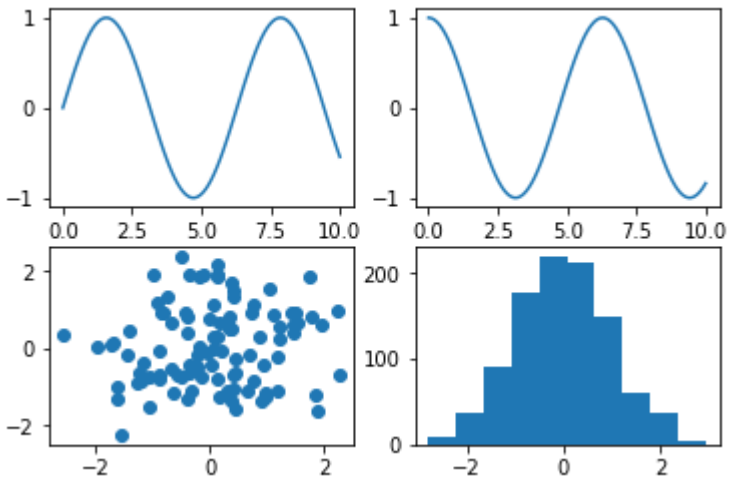


In [34]:

```
# Draw a 2 by 2 graph
x = np.linspace(0, 10, 100)

# create the first panel and set current axis
plt.subplot(2, 2, 1) # (rows, columns, panel number)
plt.plot(x, np.sin(x))

# create the second panel and set current axis
plt.subplot(2, 2, 2)
plt.plot(x, np.cos(x));

# create the third panel and set current axis
plt.subplot(2, 2, 3)
rng = np.random.RandomState(0)
x = rng.randn(100)
y = rng.randn(100)

plt.scatter(x, y)

# create the fourth panel and set current axis
plt.subplot(2, 2, 4)
data = np.random.randn(1000)

plt.hist(data)
```

Out[34]:

```
(array([ 10.,  36.,  91., 178., 220., 213., 148.,  61.,  38.,   5.]),
 array([-2.79172089, -2.21836312, -1.64500535, -1.07164759, -0.49828982,
         0.07506795,  0.64842571,  1.22178348,  1.79514125,  2.36849901,
         2.94185678]),
 <a list of 10 Patch objects>)
```



In [ ]: