Vectorized String Operations:-

MAJJIGA JASWANTH
20BCD7171

Welcome to Jupyter!

In [1]:
```
import numpy as np
x = np.array([2, 3, 5, 7, 11, 13])
x * 2
```

Out[1]:
```
array([ 4,  6, 10, 14, 22, 26])
```

In [2]:
```
data = ['peter', 'Paul', 'MARY', 'gUIDO']
[s.capitalize() for s in data]
```

Out[2]:
```
['Peter', 'Paul', 'Mary', 'Guido']
```

In [3]:
```
data= ['peter', 'Paul', None, 'MARY', 'gUIDO']
[s.capitalize() for s in data]
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-3-e2d61c4b7509> in <module>
      1 data= ['peter', 'Paul', None, 'MARY', 'gUIDO']
----> 2 [s.capitalize() for s in data]

<ipython-input-3-e2d61c4b7509> in <listcomp>(.0)
      1 data= ['peter', 'Paul', None, 'MARY', 'gUIDO']
----> 2 [s.capitalize() for s in data]

AttributeError: 'NoneType' object has no attribute 'capitalize'
```

In [4]:
```
import pandas as pd
names = pd.Series(data)
names
```

Out[4]:
```
0    peter
1     Paul
2     None
3     MARY
4    gUIDO
dtype: object
```

In [5]:
```
names.str.capitalize()
```

Out[5]:
```
0    Peter
1     Paul
2     None
3     Mary
4    Guido
dtype: object
```

In [6]:
```
# convert all alphabets in to lower case
names.str.lower()
```

Out[6]:
```
0    peter
1     paul
2     None
3     mary
4    guido
dtype: object
```

In [7]:
```
# convert all alphabets in to upper case
names.str.upper()
```

Out[7]:
```
0    PETER
1     PAUL
2     None
3     MARY
4    GUIDO
dtype: object
```

In [9]:
```
# Swap the case of alphabets (convert upper case alphabets to lower cas
names.str.swapcase()
```

Out[9]:
```
0    PETER
1     pAUL
2     None
3     mary
4    Guido
dtype: object
```

In [10]:
```
# find the length of each string
names.str.len()
```

Out[10]:
```
0    5.0
1    4.0
2    NaN
3    4.0
4    5.0
dtype: float64
```

In [12]:
```
monte = pd.Series(['Graham Chapman', 'John Cleese', 'Terry Gilliam',
 'Eric Idle', 'Terry Jones', 'Michael Palin'])
monte
```

Out[12]:
```
0    Graham Chapman
1       John Cleese
2     Terry Gilliam
3         Eric Idle
4       Terry Jones
5     Michael Palin
dtype: object
```

In [13]:
```
monte.str.lower()
```

Out[13]:
```
0    graham chapman
1       john cleese
2     terry gilliam
3         eric idle
4       terry jones
5     michael palin
dtype: object
```

In [14]:
```
monte.str.len() # space is also counted as a character
```

Out[14]:
```
0    14
1    11
2    13
3     9
4    11
5    13
dtype: int64
```

In [15]:
```
monte.str.startswith('T')
```

Out[15]:
```
0    False
1    False
2     True
3    False
4     True
5    False
dtype: bool
```

In [16]:
```
monte.str.startswith('t') # Python is case-sensitive
```

Out[16]:
```
0    False
1    False
2    False
3    False
4    False
5    False
dtype: bool
```

In [17]:
```
monte.str.endswith('e')
```

Out[17]:
```
0    False
1     True
2    False
3     True
4    False
5    False
dtype: bool
```

In [18]:
```
monte.str.split() # default split condition is space
```

Out[18]:
```
0    [Graham, Chapman]
1       [John, Cleese]
2     [Terry, Gilliam]
3         [Eric, Idle]
4       [Terry, Jones]
5     [Michael, Palin]
dtype: object
```

In [19]:
```
# split the string when alphabet 'a' occurs in a string
monte.str.split('a') # 'a' will not be printed
```

Out[19]:
```
0    [Gr, h, m Ch, pm, n]
1            [John Cleese]
2         [Terry Gilli, m]
3              [Eric Idle]
```

```
5       [Mich, el P, lin]
dtype: object
```

In [20]:
```
# replace 'a' with '@'
monte.str.replace('a','@')
```

Out[20]:
```
0    Gr@h@m Ch@pm@n
1       John Cleese
2     Terry Gilli@m
3         Eric Idle
4       Terry Jones
5     Mich@el P@lin
dtype: object
```

In [21]:
```
monte.str[0:3]
```

Out[21]:
```
0    Gra
1    Joh
2    Ter
3    Eri
4    Ter
5    Mic
dtype: object
```

In [22]:
```
monte.str.split().str.get(-1)
```

Out[22]:
```
0    Chapman
1     Cleese
2    Gilliam
3       Idle
4      Jones
5      Palin
dtype: object
```

In [23]:
```
monte.str.split().str.get(0)
```

Out[23]:
```
0     Graham
1       John
2      Terry
3       Eric
4      Terry
5    Michael
dtype: object
```

This repo contains an introduction to Jupyter and IPython.

Outline of some basics:

- Notebook Basics
- IPython - beyond plain python
- Markdown Cells
- Rich Display System
- Custom Display logic
- Running a Secure Public Notebook Server
- How Jupyter works to run code in different languages.

You can also get this tutorial and run it on your laptop:

```
git clone https://github.com/ipython/ipython-in-depth
```

Install IPython and Jupyter:

with conda:

```
conda install ipython jupyter
```

with pip:

```
# first, always upgrade pip!
pip install --upgrade pip
pip install --upgrade ipython jupyter
```

Start the notebook in the tutorial directory:

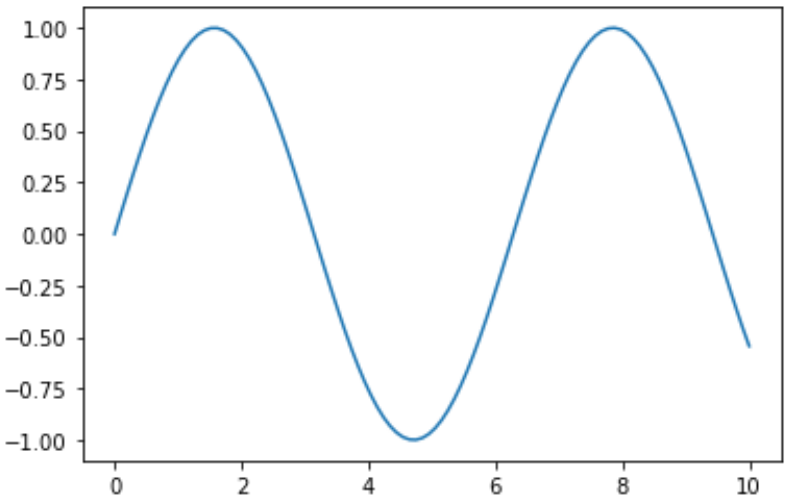
```
cd ipython-in-depth
jupyter notebook
```

Matplotlib:-

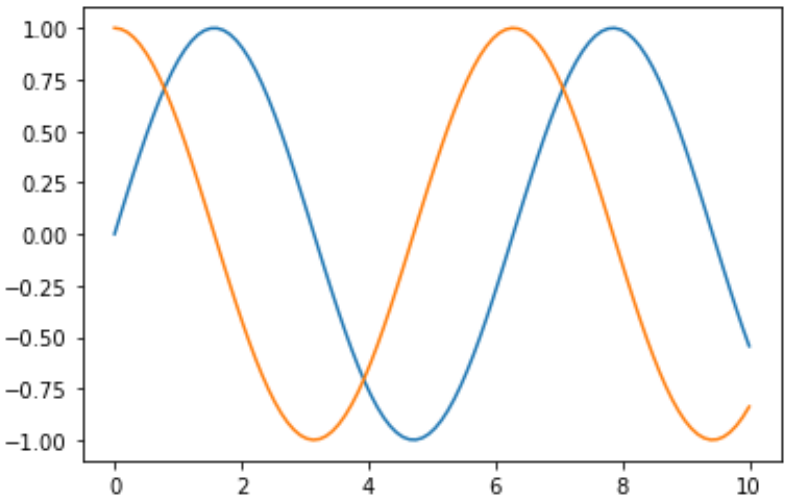
Welcome to Jupyter!

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

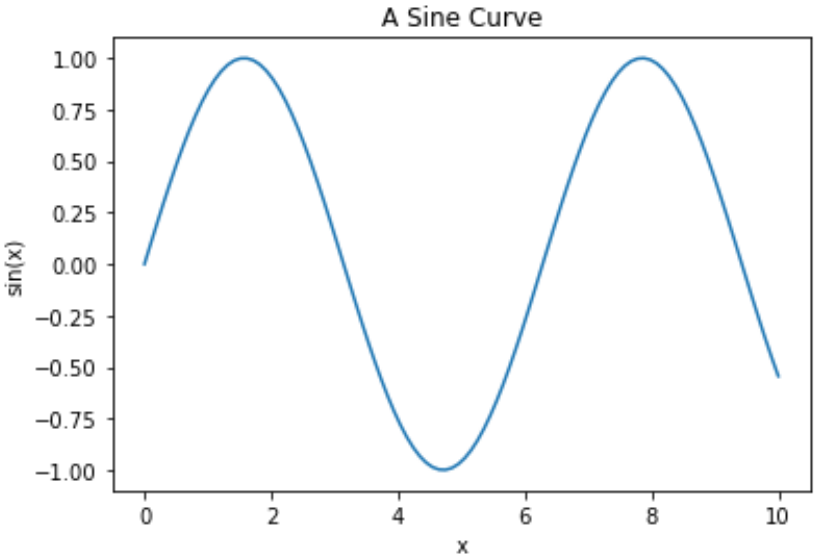
In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)
# Drawing sine curve
plt.plot(x, np.sin(x))
plt.show()
```



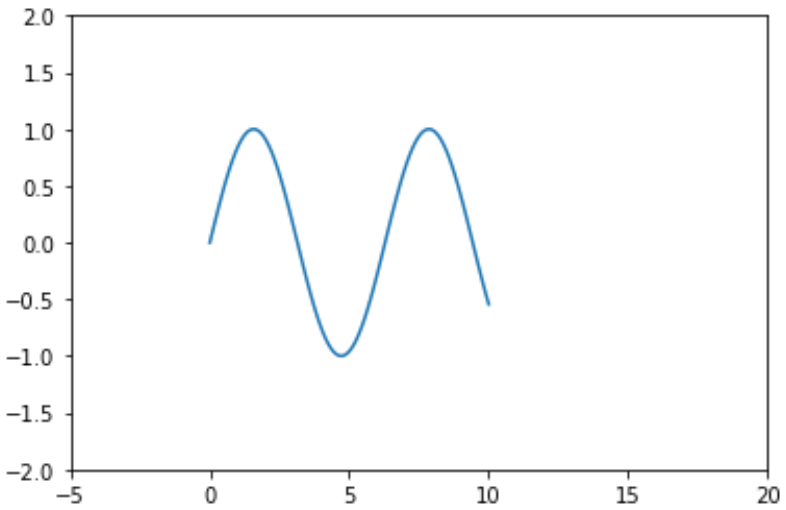
In [3]:
```
x = np.linspace(0, 10, 100)
# Drawing both sine and cosine curves on same plot
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
plt.show()
```



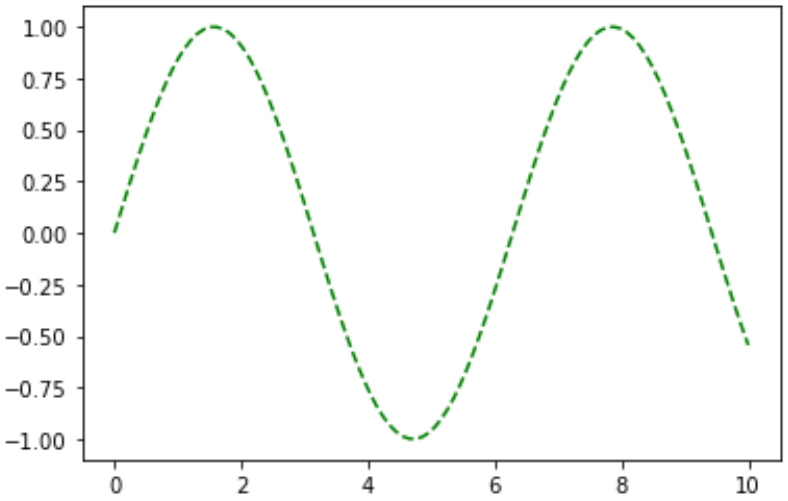
In [4]:
```
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x))
# Adding title of graph
plt.title("A Sine Curve")
# Adding label of x-axis and y-axis
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.show()
```



In [5]:
```
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()
```

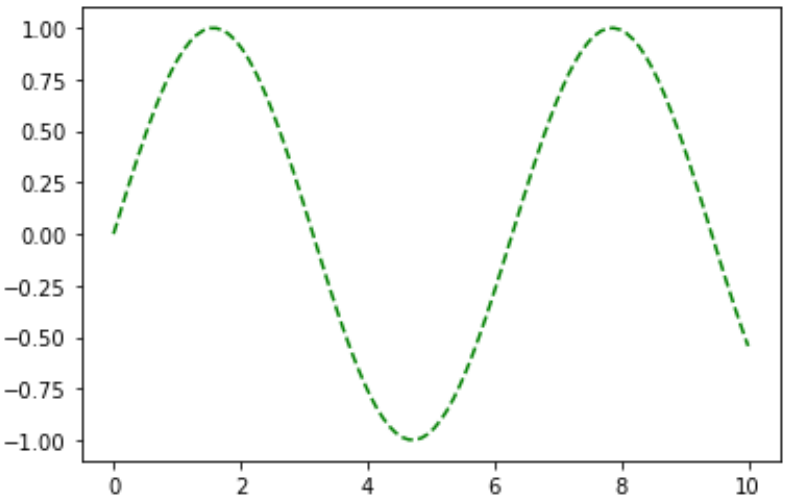


In [6]:
```
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x), color = 'green', linestyle='dashed')
plt.show()
```



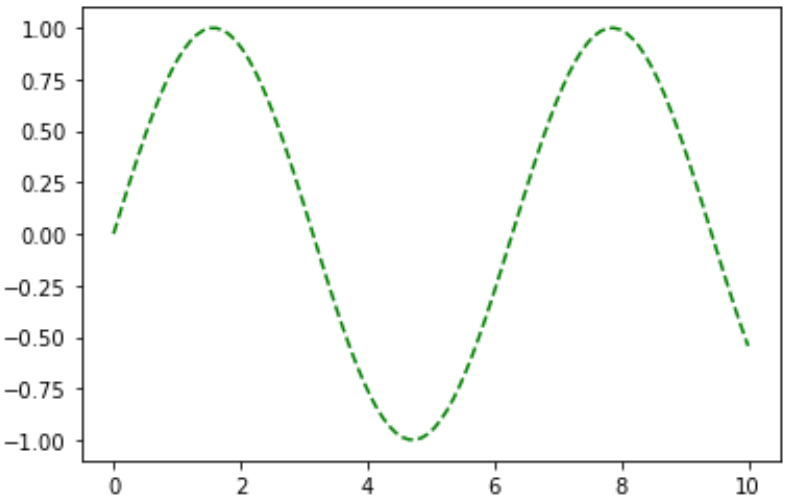
In [7]:

```
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x), color = 'g', linestyle='--')
plt.show()
```



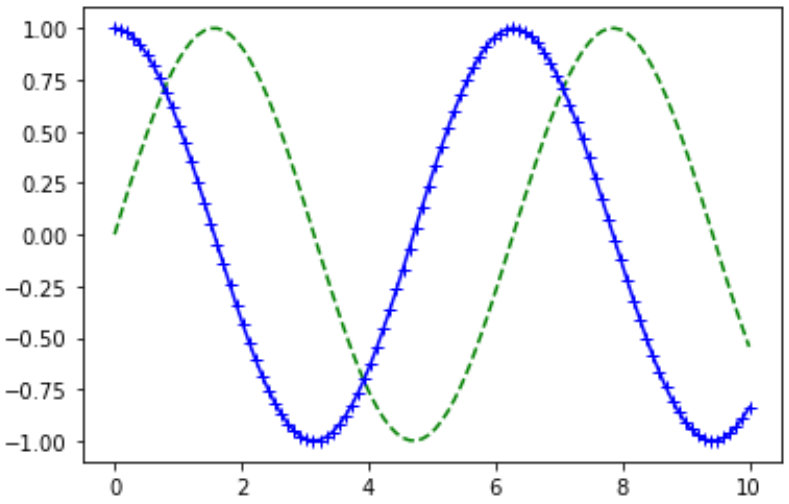
In [8]:

```
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x), 'g--')
plt.show()
```



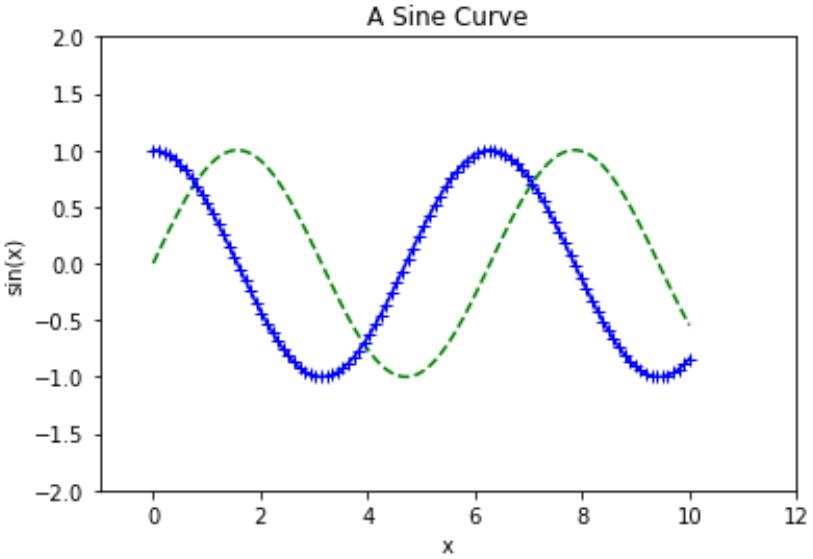
In [9]:

```
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x), 'g--')
plt.plot(x, np.cos(x), 'b-+')
plt.show()
```



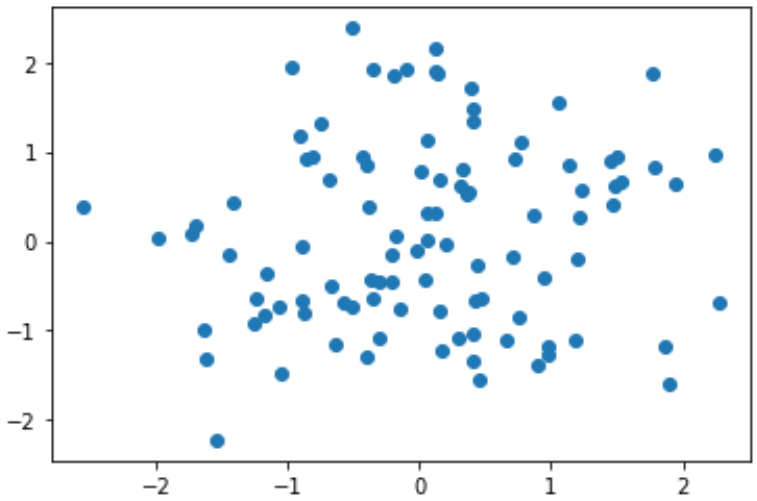
In [10]:

```
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()
```



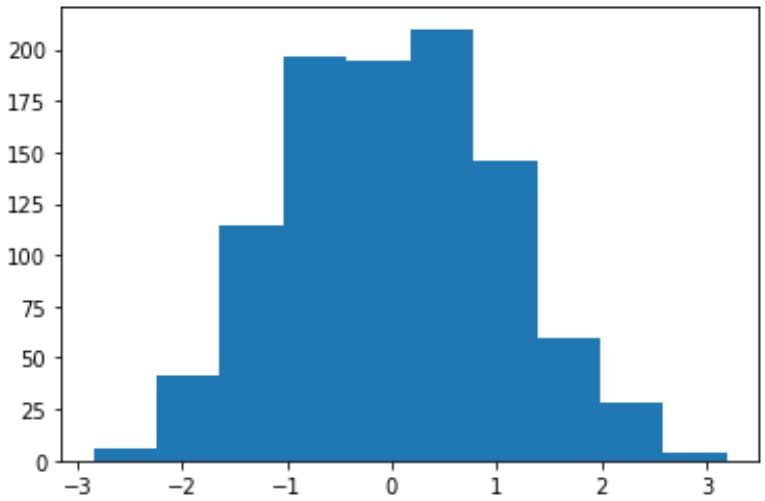
In [11]:

```
rng = np.random.RandomState(0)
x = rng.randn(100)
y = rng.randn(100)
plt.scatter(x, y)
plt.show()
```



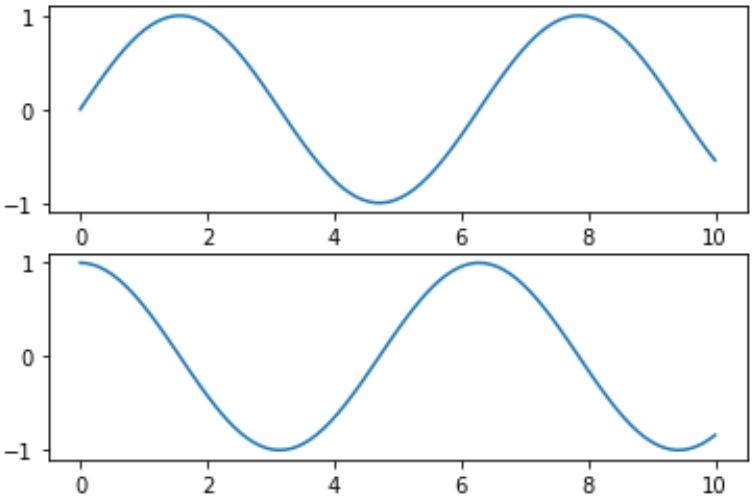
In [12]:

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



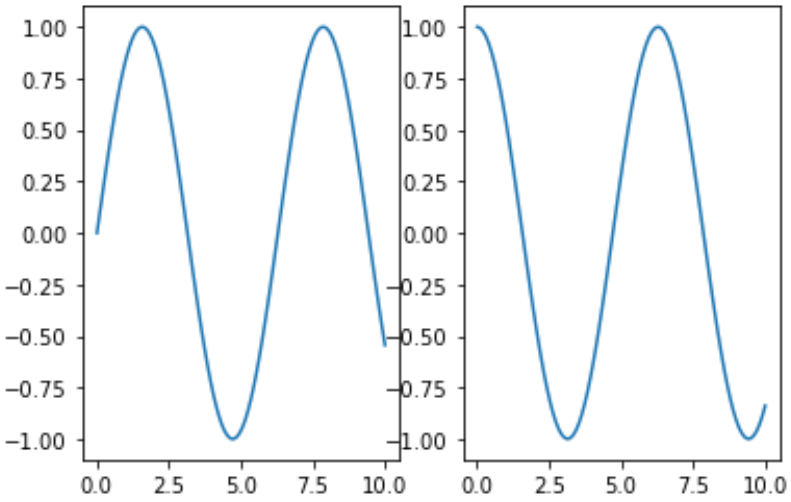
In [13]:

```
# 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));
```



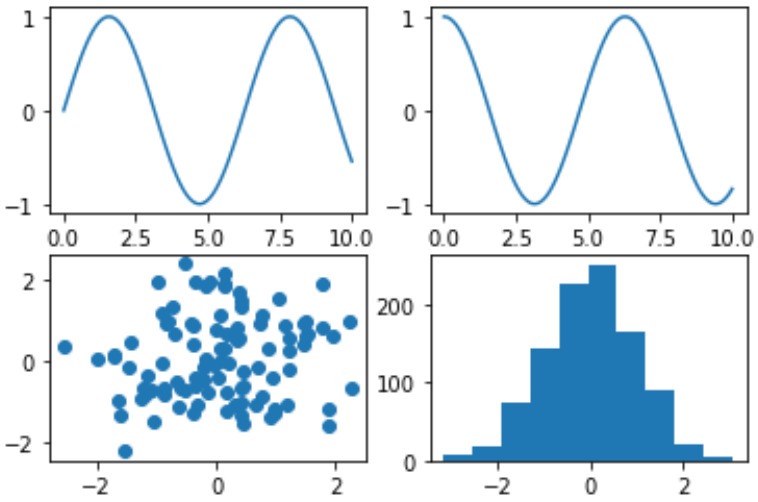
In [14]:

```
# 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 [15]:
```
# 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[15]:
```
(array([  7.,  17.,  73., 144., 225., 251., 166.,  89.,  22.,   6.]),
 array([-3.15832539, -2.53765069, -1.91697598, -1.29630128, -0.67562657,
        -0.05495186,  0.56572284,  1.18639755,  1.80707226,  2.42774696,
         3.04842167]),
 <BarContainer object of 10 artists>)
```



This repo contains an introduction to Jupyter and IPython.

Outline of some basics:

- Notebook Basics
- IPython - beyond plain python
- Markdown Cells
- Rich Display System
- Custom Display logic
- Running a Secure Public Notebook Server
- How Jupyter works to run code in different languages.

You can also get this tutorial and run it on your laptop:

```
git clone https://github.com/ipython/ipython-in-depth
```

Install IPython and Jupyter:

with conda:

```
conda install ipython jupyter
```

with pip:

```
# first, always upgrade pip!
pip install --upgrade pip
pip install --upgrade ipython jupyter
```

Start the notebook in the tutorial directory:

```
cd ipython-in-depth
```