Matplotlib

- Matplotlib is a multi-platform data visualization library built on NumPy arrays.
- Matplotlib is a graph plotting library in python that serves as a visualization utility.

Importing Matplotlib

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

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the `plt` alias. The `plt` interface is what we will use most often for plotting.

Matplotlib Version

In [3]:
```
mpl.__version__
```

Out[3]:
```
'3.4.2'
```

Import NumPy

In [7]:
```
import numpy as np
```

Setting Potting Styles

The `plt.style` directive is used to choose appropriate aesthetic styles for the figures.

In [5]:
```
plt.style.use('classic') # Here, the classic Matplotlib style is used
```

In [6]:
```
# Available style in Matplotlib
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot
', 'grayscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark', 'seaborn-dark-palette', 'seabor
n-darkgrid', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-post
er', 'seaborn-talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']
```

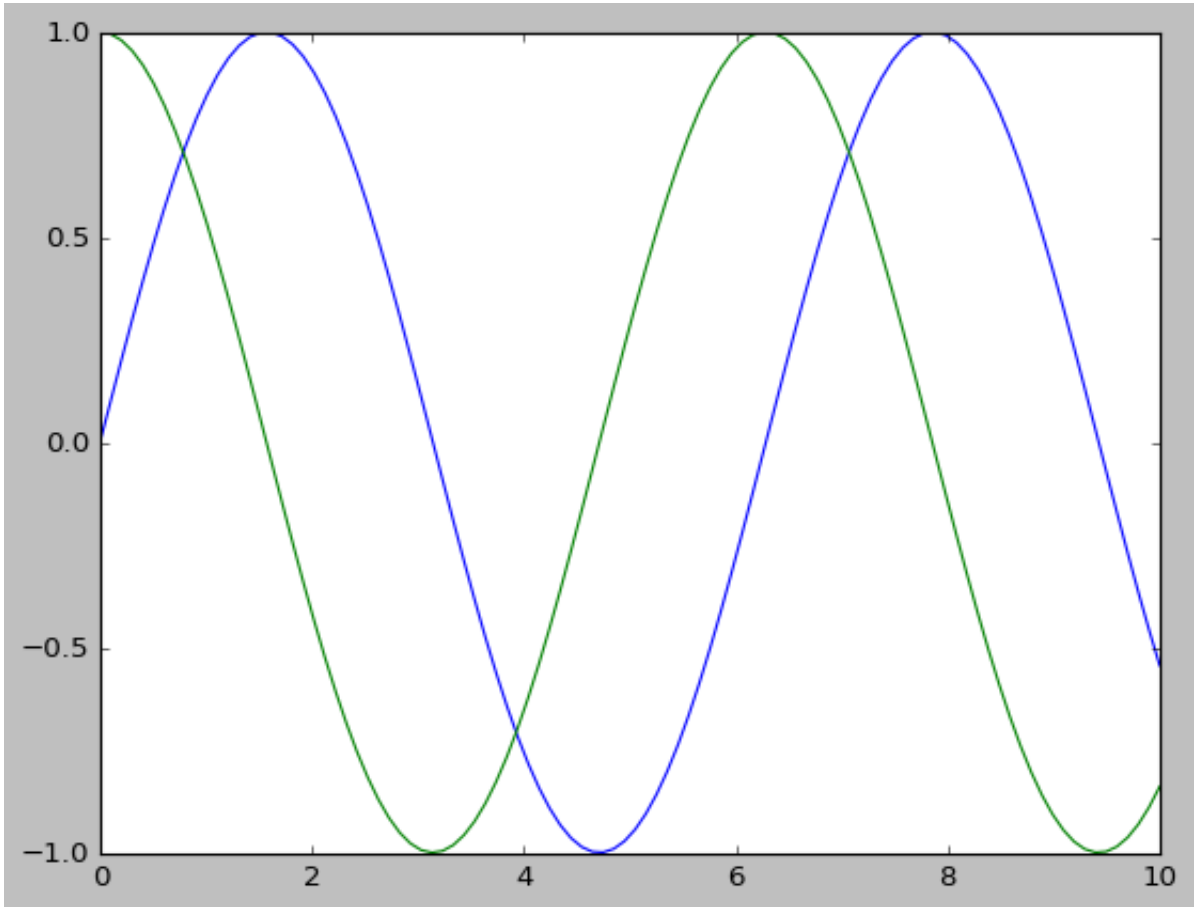
How to display plots?

- `plt.show()` can be used to display plots.
- Opens interactive window that display the figure.
- One thing to be aware of: the `plt.show()` command should be used only once per Python session, and is most often seen at the very end of the script. Multiple show() commands can lead to unpredictable backend-dependent behavior, and should mostly be avoided.

In [10]:
```
x = np.linspace(0, 10, 100)

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

plt.show()
```

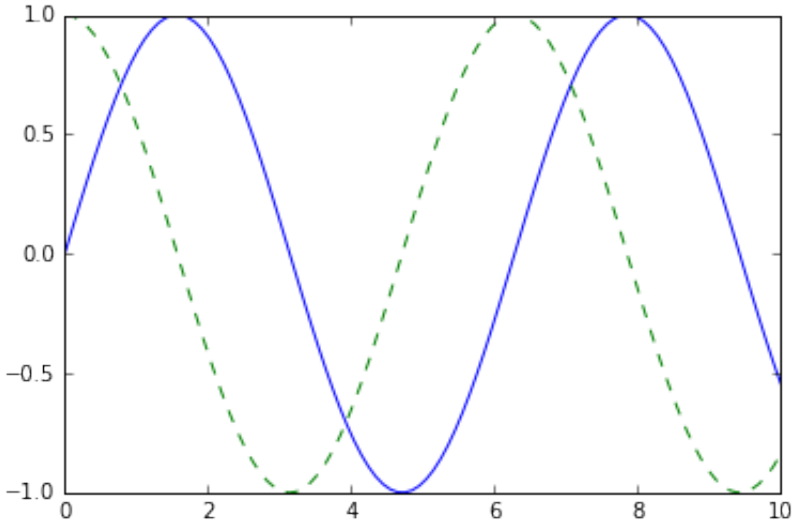


- Besides, plotting interactively within an notebook can be done with the `%matplotlib` magic command.
- Embed graphics directly in the notebook.
- In this case there are two options for embedding graphics directly in the notebook:
  - `%matplotlib notebook` will lead to interactive plots embedded within the notebook.
  - `%matplotlib inline` will lead to static images of your plot embedded in the notebook.

In [11]:
```
"""
    This command needs to be done only once per kernel/session.
    Any cell within the notebook that creates a plot will embed
    a PNG image of the resulting graphic.
"""
%matplotlib inline
```

In [12]:
```
x = np.linspace(0, 10, 100)

fig = plt.figure() # create a plot figure
plt.plot(x, np.sin(x), '-')
plt.plot(x, np.cos(x), '--');
```



Saving Figures to File

- Saving a figure can be done using the savefig() command.

In [13]:
```
# Saving the above figure as a PNG file in the current working directory.
fig.savefig('my_figure.png')
```

In [16]:
```
# List of supported file types can be found for your system
# by using this command.
fig.canvas.get_supported_filetypes()
```

Out[16]:
```
{'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'}
```

MATLAB-style Interface

- Matplotlib was originally written as a Python alternative for MATLAB users, and much of its syntax reflects that fact.
- The MATLAB-style tools are contained in the `pyplot` (`plt`) interface.
- This interface is fast and convenient for simple plots, it is easy to run into problems.
- For example, once the second panel is created, it is a bit clunky to go back and add something to the first.
- Fortunately, there is a better way, that is, Object-oriented interface

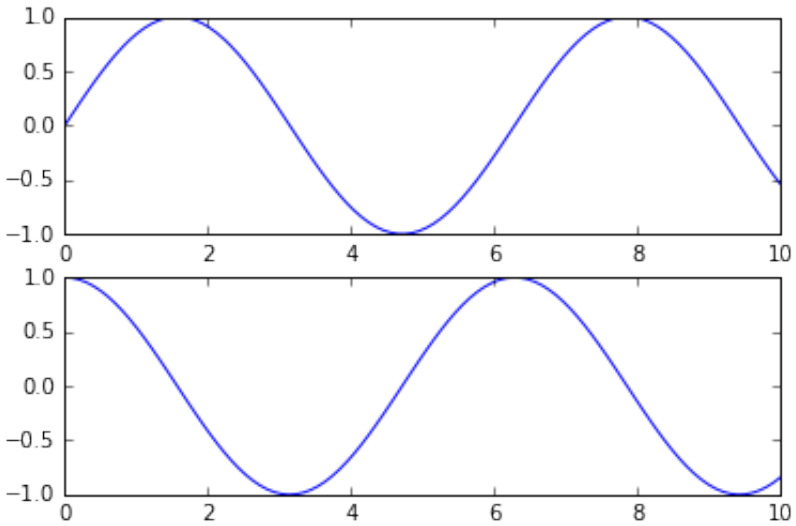
In [17]:

```
# Example of MATLAB-style Interface

plt.figure()  # create a plot figure

# 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));
```



Object-oriented interface

- Provides more control over your figure.
- In object-oriented interface the plotting functions are methods of explicit `Figure` and `Axes` objects.

In [18]:

```
# re-create the previous plot using this style of plotting

# 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(x, np.sin(x))
ax[1].plot(x, np.cos(x));
```