python™

Programming with Python for Data Science

Unit Topics

- Matplotlib.pyplot
- Pandas dataframe plotting capabilities
- Seaborn
- Plotly and Cufflinks

Learning objectives

- Visualize your data!

matplotlib.pyplot

Matplotlib is a Python plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that “axes” here and in most places in the documentation refers to the *axes part of a figure* and not the strict mathematical term for more than one axis).

Basic Example

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 5, 11)
y = x ** 2
```

We have imported the library and created two arrays of data

```
plt.plot(x, y, 'r') # 'r' is the color red
plt.xlabel('X Axis Title Here')
plt.ylabel('Y Axis Title Here')
plt.title('String Title Here')
```

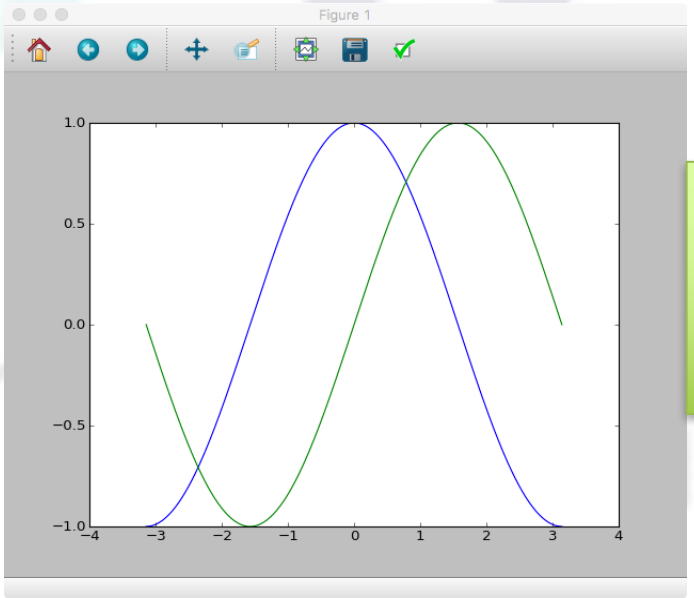
Note that if you write the commands on the Spyder IPython console **you must input them toghether (*Ctrl+Enter*)**, else you will end up with four different graphs, or in the future examples with no graph at all; in fact the default is that the graphs are shown "inline".

If you write them on the Python console you will have a separate canvas, on which you will see the effect of each command. Sometimes a `plt.show()` or `fig.show()` command is needed.

You can obtain the interactive canvas again in IPython changing the preferences.

With Jupyter use `%matplotlib inline`

```
# Create a new figure of size 8x6 points, using 100 dots per inch
plt.figure(figsize=(8,6), dpi=100)
# Create a new subplot from a grid of 1x1
plt.subplot(1,1,1)
X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
C,S = np.cos(X), np.sin(X)
# Plot cosine using blue color with a continuous line of width 1 (pixels)
plt.plot(X, C, color="blue", linewidth=1.0, linestyle="-")
# Plot sine using green color with a dotted line of width 1 (pixels)
plt.plot(X, S, color="green", linewidth=1.0, linestyle="--")
plt.xlim(-4.0,4.0)                                  # Set x limits
plt.xticks(np.linspace(-4,4,9,endpoint=True))       # Set x ticks
plt.ylim(-1.0,1.0)                                  # Set y limits
plt.yticks(np.linspace(-1,1,5,endpoint=True))       # Set y ticks
# Save figure using 72 dots per inch
# savefig("../figures/exercice_2.png",dpi=72)
# Show result on screen
plt.show()
```




```
#auto
C,S = np.cos(X),np.sin(X)
plt.plot(X,C)
plt.plot(X,S)
plt.show()
```

Object Oriented Method

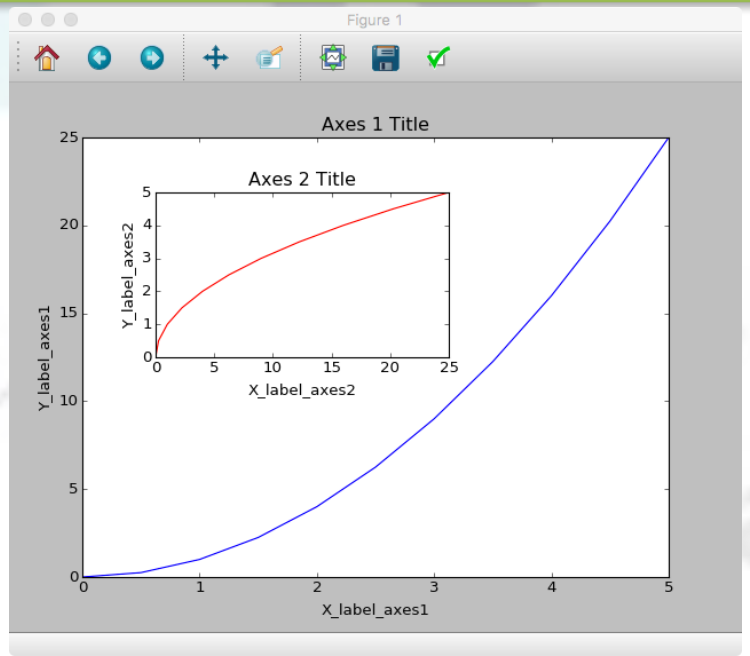
```
# Create Figure (empty canvas)
fig = plt.figure()
```

We instantiate figure objects and then call methods or attributes from that object.

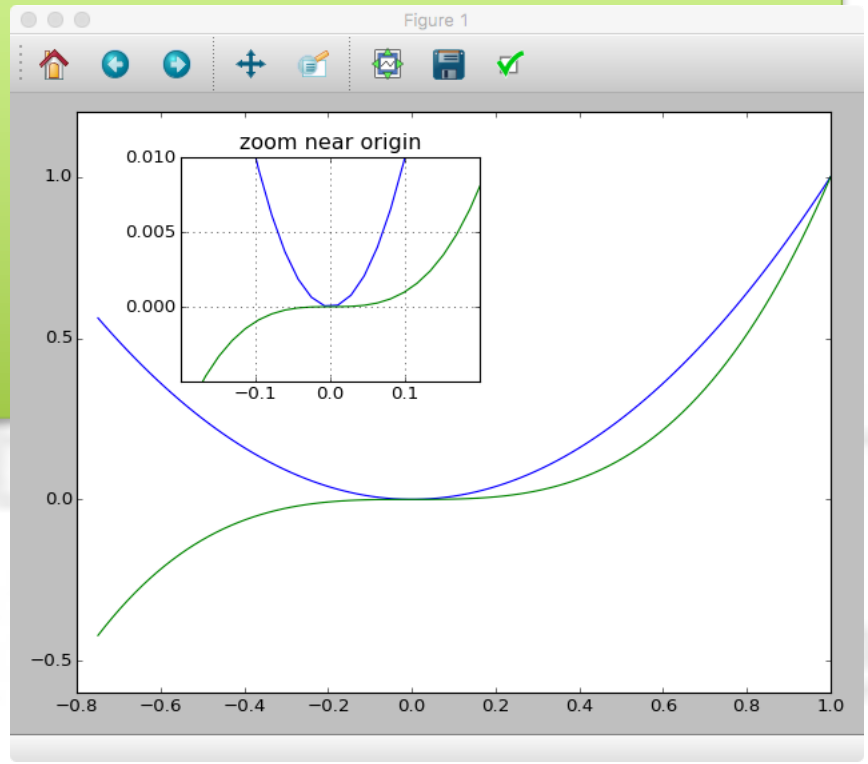
```
# Add set of axes to figure
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])  # left, bottom, width, height
                                           # (range 0 to 1)
# Plot on that set of axes
axes.plot(x, y, 'b')
axes.set_xlabel('Set X Label') # Notice the use of set_ to begin methods
axes.set_ylabel('Set y Label')
axes.set_title('Set Title')
```

Code is a little more complicated, but the advantage is that we now have full control of where the plot axes are placed, and we can easily add more than one axis to the figure

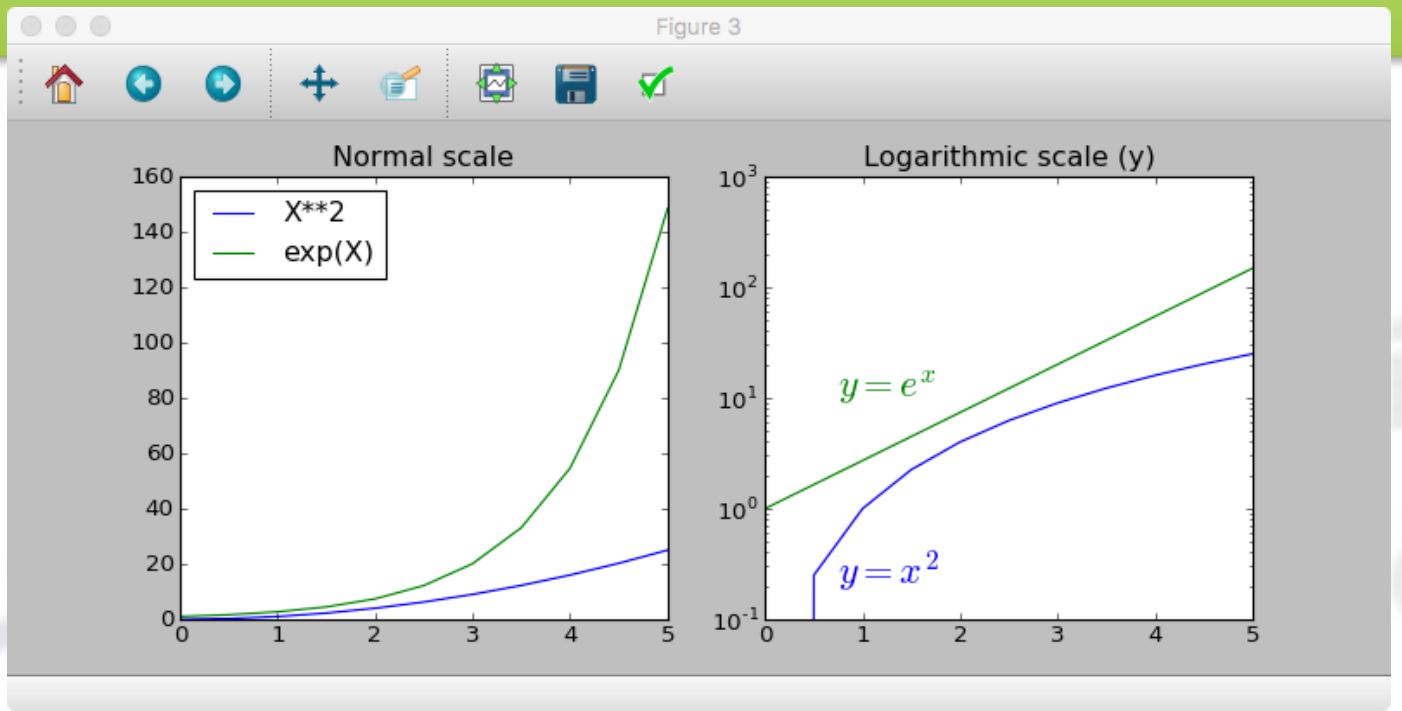
```
# Creates blank canvas
fig = plt.figure()
axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # main axes
axes2 = fig.add_axes([0.2, 0.5, 0.4, 0.3]) # inset axes

# Larger Figure Axes 1
axes1.plot(x, y,'b')
axes1.set_xlabel('X_label_axes1')
axes1.set_ylabel('Y_label_axes1')
axes1.set_title('Axes 1 Title')

# Insert Figure Axes 2
axes2.plot(y, x, 'r')
axes2.set_xlabel('X_label_axes2')
axes2.set_ylabel('Y_label_axes2')
axes2.set_title('Axes 2 Title');
```



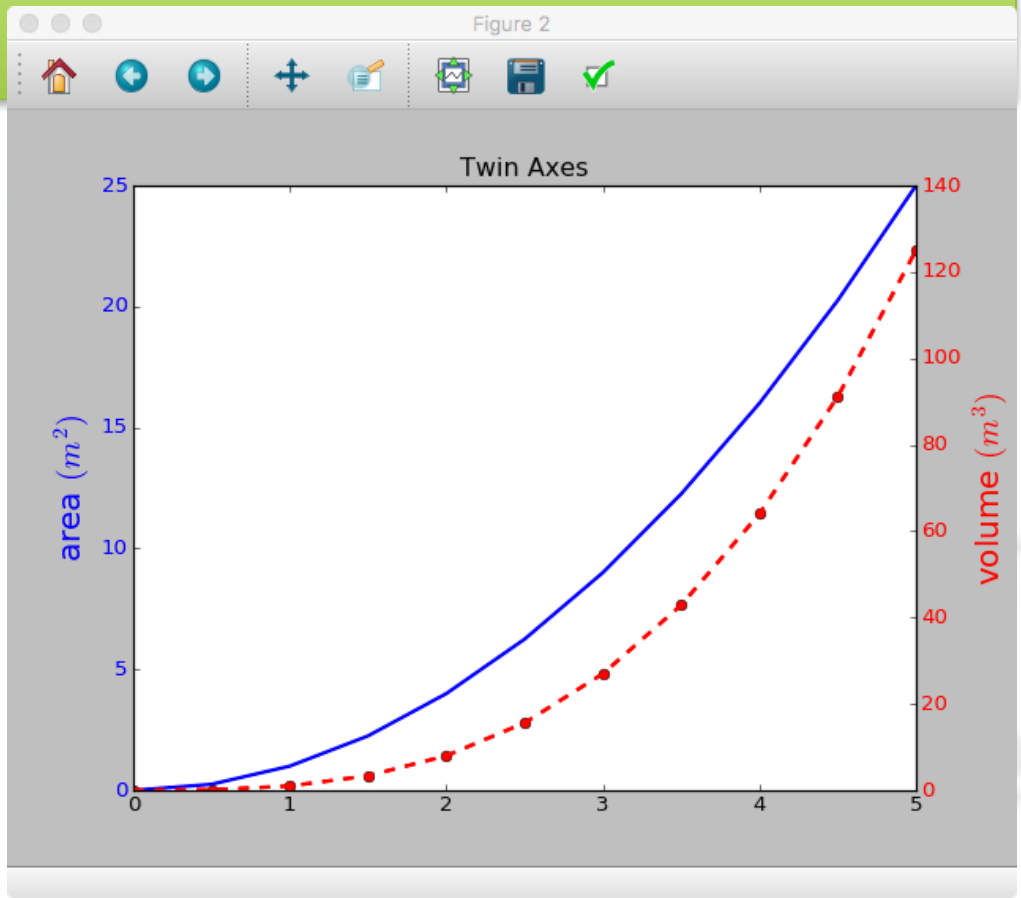
```
xx = np.linspace(-0.75, 1., 100)

fig, ax = plt.subplots()
ax.plot(xx, xx**2, xx, xx**3)
fig.tight_layout()
# manually add inset
inset_ax = fig.add_axes([0.3, 0.55, 0.35, 0.35]) # X, Y, width, height
inset_ax.plot(xx, xx**2, xx, xx**3)
inset_ax.set_title('zoom near origin')

# set axis range
inset_ax.set_xlim(-.2, .2)
inset_ax.set_ylim(-.005, .01)

# set axis tick locations
inset_ax.set_yticks([0, 0.005, 0.01])
inset_ax.set_xticks([-0.1,0,.1]);
inset_ax.grid(True)
```



```
fig, axes = plt.subplots(1, 2, figsize=(10,4))

axes[0].plot(x, x**2, label="X**2")
axes[0].plot(x, np.exp(x), label="exp(X)")
axes[0].set_title("Normal scale")
axes[0].legend(loc=0)          # let matplotlib decide the optimal location

axes[1].plot(x, x**2, x, np.exp(x))
axes[1].set_yscale("log")
axes[1].set_title("Logarithmic scale (y)");
axes[1].text(0.75, 0.2, r"$y=x^2$", fontsize=20, color="blue")
axes[1].text(0.75, 10, r"$y=e^x$", fontsize=20, color="green")

fig.savefig("scales.png", dpi=200)
```



```
fig, ax1 = plt.subplots()
ax1.plot(x, x**2, lw=2, color="blue")
ax1.set_ylabel(r"area $(m^2)$", fontsize=18, color="blue")
for label in ax1.get_yticklabels():
    label.set_color("blue")

ax2 = ax1.twinx()
ax2.plot(x, x**3, lw=2, color="red", ls='--', marker='o')
ax2.set_ylabel(r"volume $(m^3)$", fontsize=18, color="red")
for label in ax2.get_yticklabels():
    label.set_color("red")

ax1.set_title("Twin Axes")
```



Matplot exercise

Create a figure object, and put two axes ax1 and ax2 on it. Locate them respectively at [0,0,1,1] and [0.2,0.5,.4,.4]

```
fig = plt.figure()
ax1 = fig.add_axes([0,0,1,1])
ax2 = fig.add_axes([0.2,0.5,.4,.4])
```

Plot y=x**3 on the first axis

```
x = np.arange(0,100)
y = x**3
ax1.plot(x,y)
```

The second axis will be the zoom in the range 0-10

```
ax2.plot(x,y)
ax2.set_xlim(0,10)
ax2.set_ylim(0,800)
fig
```

Matplotlib home exercise!

Plot this figure using `plt.figure()`