Matplotlib

By Edward Smith

12:00-13:00
26th September 2018

10am

Plan

12:00 to 12:15 A Quick Overview of Matplotlib

12:15 to 12:30 Hands on

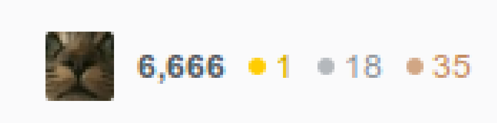
12:30 to 12:50 More Advanced Plotting

12:50 to 13:00 Hands on

13:00 to 14:00 Lunch – I will be here to answer questions over lunch and in the afternoon

Who am I?

- Currently a full time postdoc and software developer
  - Mechanical Engineering (Also Civ & Chem Eng at IC)
  - About 10 years of programming experience
  - Software Sustainability Fellow (www.software.ac.uk)
  - Python questions on Stackoverflow



6,666 ● 1 ● 18 ● 35

- I'm not using software carpentry sides as I taught this course in Python before
  - I learnt MATLAB as an undergrad in Mech Eng
  - My main incentive for the switch to Python is the long term potential and the ability to write more sustainable code, but it took me a year to kick the MATLAB habit
  - I wish I had learnt Python sooner, so I wrote a course to help people make the switch.

A Quick Overview

An Example vs MATLAB

```
%MATLAB
clear all
close all

x = linspace(0,2*pi,100);
y = sin(x);
z = cos(x);
plot(x,y,'-r');
hold all
plot(x,z,'-b')
```

```
#python
from numpy import *
from matplotlib.pyplot import *

x = linspace(0,2*pi,100)
y = sin(x)
z = cos(x)
plot(x,y,'-r')
plot(x,z,'-b')
show()
```

An Example vs MATLAB

```
%MATLAB
clear all
close all

x = linspace(0,2*pi,100);
y = sin(x);
z = cos(x);
plot(x,y,'-r');
hold all
plot(x,z,'-b')
```

```
#python
from numpy import *
from matplotlib.pyplot import *

x = linspace(0,2*pi,100)
y = sin(x)
z = cos(x)
plot(x,y,"-r")
plot(x,z,"-b")
show()
```

Import all

Better not to do this to
avoid nameclashes

plot function has been imported

Numerical and Plotting Libraries

- Numpy – The basis for all other numerical packages to allow arrays instead of lists (implemented in c so more efficient)
  - `x = np.array([1,2,3])`
  - `mean, std, linspace, sin, cos, pi, etc`
- Matplotlib – similar plotting functionality to MATLAB
  - `plot, scatter, hist, bar, contourf, imagesc (imshow), etc`
- Scipy
  - Replaces lots of the MATLAB toolboxes with optimisation, curve fitting, regression, etc. If it's not in numpy, probably in scipy
- Pandas ◄ N.B. We will not discuss plotting with Pandas dataframes
  - Dataframes to organise, perform statistics and plot data (using matplotlib behind the scenes with something like df.plot())

NOTE: Downloading/installing packages is easier with “pip” or conda

Importing Numerical and Plotting Libraries

- matplotlib – similar plotting functionality to MATLAB

```
import matplotlib.pyplot as plt
x = np.array([0,1,1,2,3,5,8,13])
plt.plot(x)
plt.show()
#Or plt.savefig("out.png")
```

We need the pyplot submodule of matplotlib for most things. Convention is to define to be plt

Here we use plot/show from matplotlib.pyplot

Use tab in notebooks to see what is available (or look online)

An Example vs MATLAB

```
%MATLAB

clear all

close all


x = linspace(0,2*pi,100);

y = sin(x);

z = cos(x);

plot(x,y,'-r');

hold all

plot(x,z,'-b')
```

```
#python

import numpy as np

import matplotlib.pyplot as plt


x = np.linspace(0,2*np.pi,100)

y = np.sin(x)

z = np.cos(x)

plt.plot(x,y,"-r")

plt.plot(x,z,"-b")

plt.show()
```

Import Plotting module matplotlib as plt

Use plot function from plt module

Plotting syntax based on MATLAB

MATLAB plotting syntax

- Matplotlib uses MATLAB shorthand syntax for styles
  - basically a string in any order which contains
    - Marker type
    - Colour

```
plt.plot(x,y,"-r")

plt.show()
```

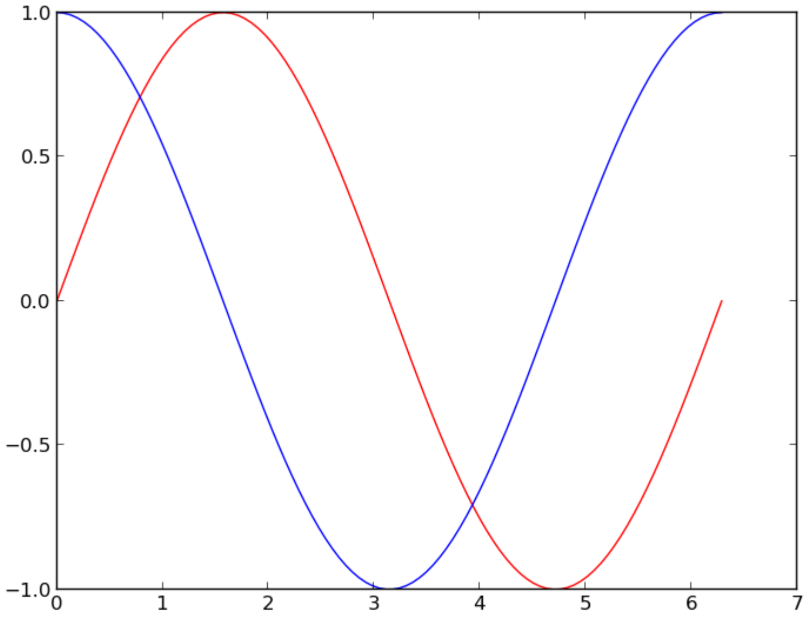
Some Styles
- = line
-- = dotted line
: = finer dotted
o = points
x = crosses
^ = triangles
s = squares

Some Colours
r = red
b = blue
g = green
y = yellow
c = cyan
k = black

- This is ”syntactic sugar”, all plot features can be specified exactly using keywords with far more control over any aspects of the plot

An Example plot

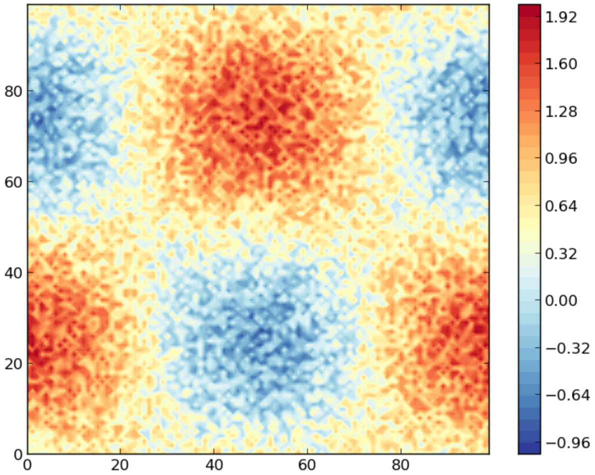
```
#python
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
z = np.cos(x)
plt.plot(x, y)
plt.plot(x, z)
plt.show()  #Or plt.savefig("out.png")
```



An Example plotting a histogram

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

#10,000 Uniform random numbers
x = np.random.random(10000)
#10,000 Normally distributed random numbers
y = np.random.randn(10000)
#Plot both on a histogram with 50 bins
plt.hist(y, 50)
plt.hist(x, 50)
plt.show()   #Or plt.savefig("out.png")
```

An Example plotting a 2D Array (matrix)



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

N = 100
x = np.linspace(0,2*np.pi,N)
y = np.sin(x); z = np.cos(x)
#Create 2D field from outer product of previous 1D functions
noise = np.random.random(N**2)
u = np.outer(y,z) + noise.reshape(N,N)
plt.contourf(u, 40, cmap=plt.cm.RdYlBu_r)
plt.colorbar()
plt.show()  #Or plt.savefig("out.png")
```

Reshape an N**2 1D array into N by N 2D array

Creates a 2D array from two 1D arrays

Don't use Jet colormap!

Hands on session

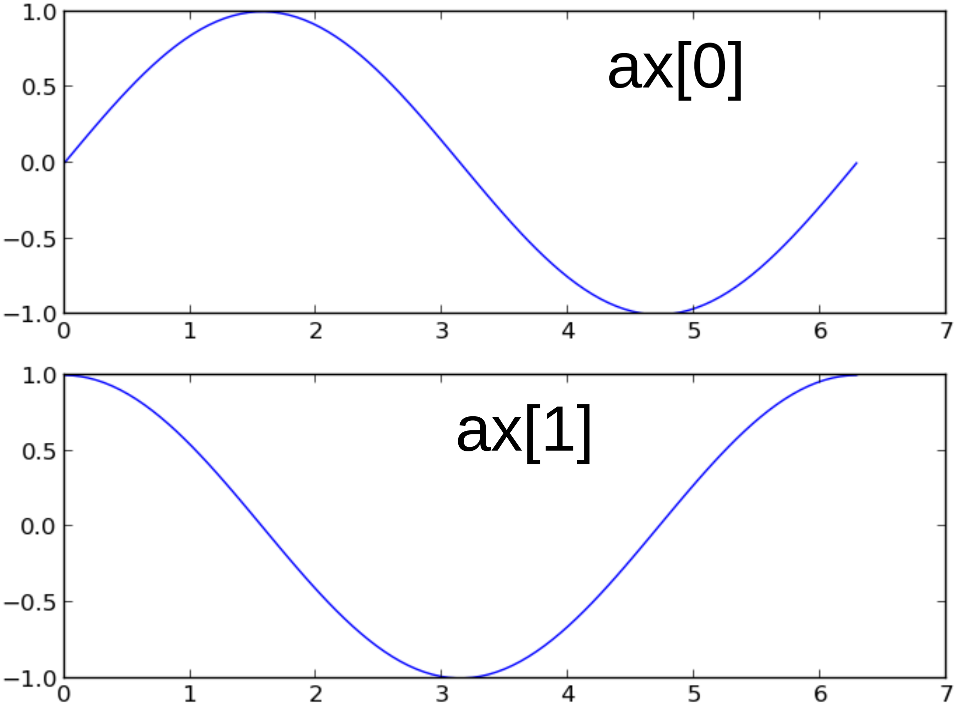
1) Plot a tanh function in the range -2 pi to 2 pi using
   - x = np.linspace(-2*np.pi,2*np.pi,100) and matplotlib to plot the function np.tanh(x)

2) Create a 1D array of 10,000 normally distributed random numbers with numpy t = np.random.randn(10000)
   - Plot the array t as a line
   - Plot a histrogram of the array t from with 50 bins
   - Convert array t to a 2D array using field=t.reshape(100,100) and plot using contourf

More Advanced Plotting

A Plot of Two Axes

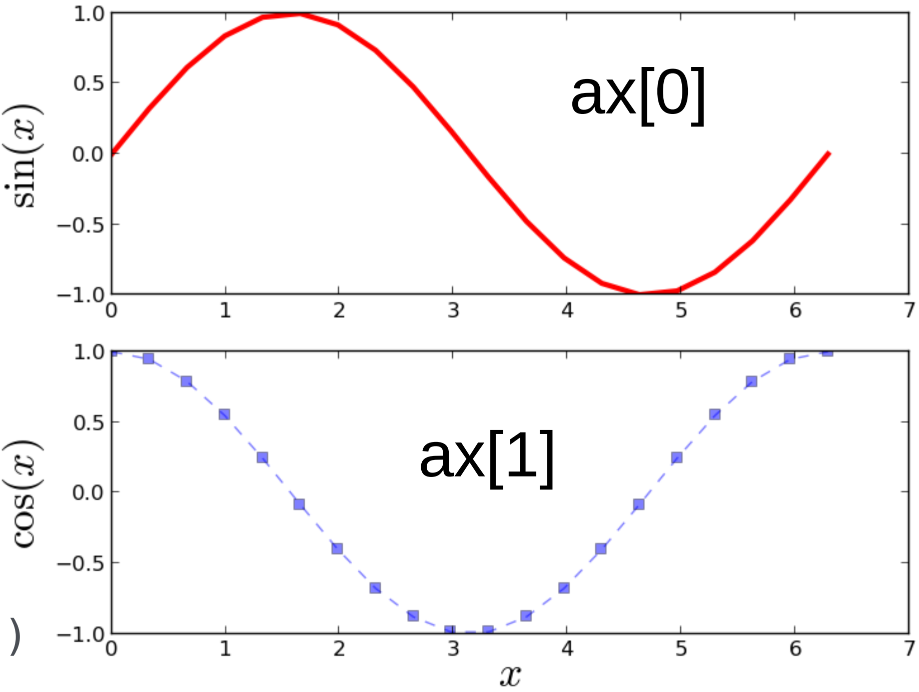
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
z = np.cos(x)
fig, ax = plt.subplots(2,1)
ax[0].plot(x, y)
ax[1].plot(x, z)
ax[1].set_xlabel("x axis", fontsize=24)
plt.show()
```



Two subplots, ax is a list of so called axis handles and we use the plot method of these handles.

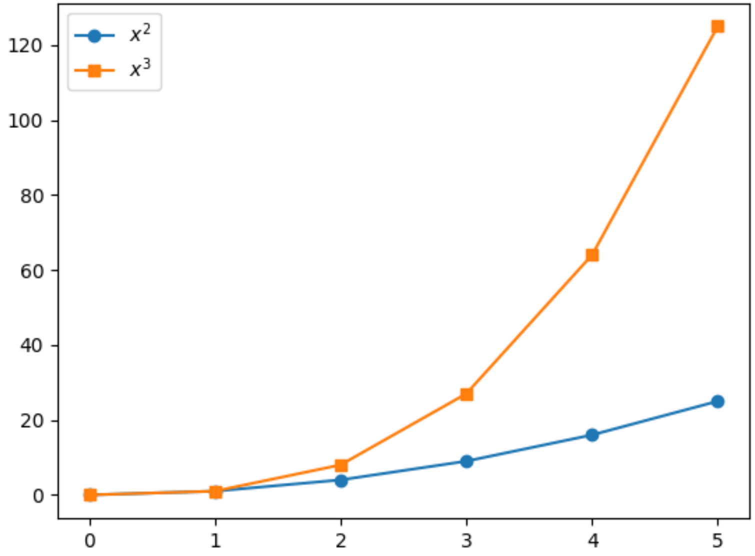
A Plot of Two Axes with Labels and Styles

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 2*np.pi, 20)
y = np.sin(x)
z = np.cos(x)
fig, ax = plt.subplots(2,1)
ax[0].plot(x, y, lw=3., c='r')
ax[1].plot(x, z, '--bs', alpha=0.5)
ax[1].set_xlabel("$x$", fontsize=24)
ax[0].set_ylabel("$\sin(x)$", fontsize=24)
ax[1].set_ylabel("$\cos(x)$", fontsize=24)
plt.show()
```



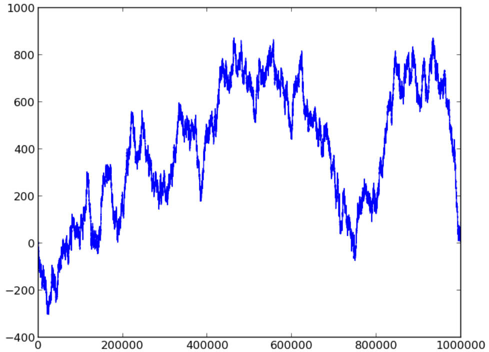
An Example with a legend

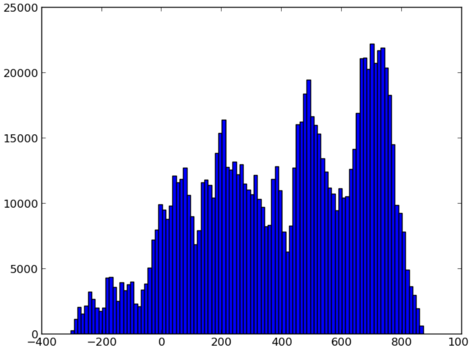
```
import numpy as np
import matplotlib.pyplot as plt

#Get six values as a numpy array
x = np.arange(6)
#Plot with latex syntax
plt.plot(x, x**2, "-o", label="$x^2$")
plt.plot(x, x**3, "-s", label="$x^3$")
plt.legend()
plt.show()
```

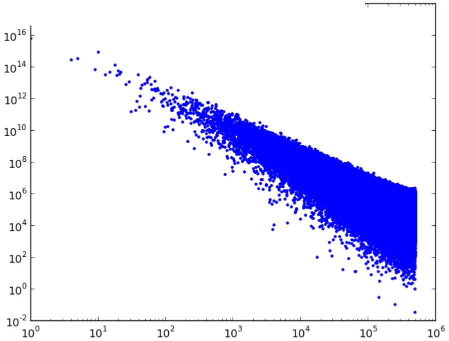


An Example using time series

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


N = 1000000
signal = np.cumsum(np.random.randn(N))
plt.plot(signal); plt.show()
plt.hist(signal, 100); plt.show()
Fs = np.fft.fft(signal)**2
plt.plot(Fs.real[:N/2], ".")
plt.xscale("log"); plt.yscale("log")
plt.show()
```

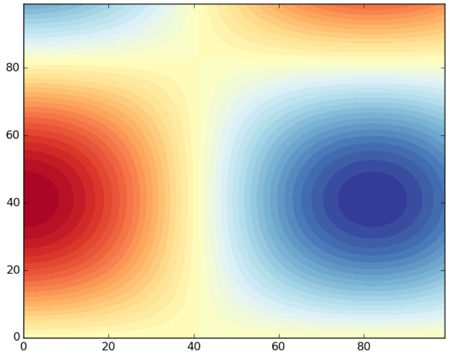


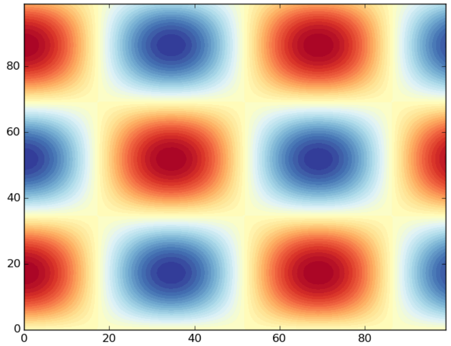




An Example of Animation

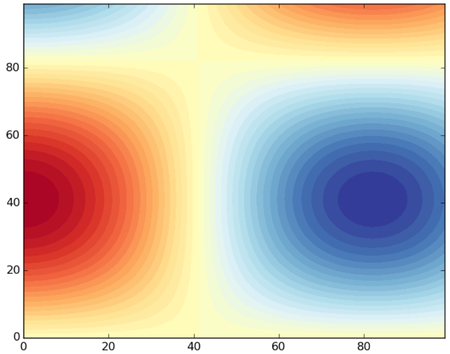
```
import numpy as np
import matplotlib.pyplot as plt
def get_field(a, N = 100):
    x = a*np.linspace(0,2*np.pi,N)
    y = np.sin(x); z = np.cos(x)
    return np.outer(y,z)
plt.ion(); plt.show()   #Interactive plot
for i in np.linspace(0., 5., 200):
    u = get_field(i)    #Call function with new
    plt.contourf(u, 40, cmap=plt.cm.RdYlBu_r)
    plt.pause(0.01)   #Pause to allow redraw
    plt.cla()   #Clear axis for next plot
```

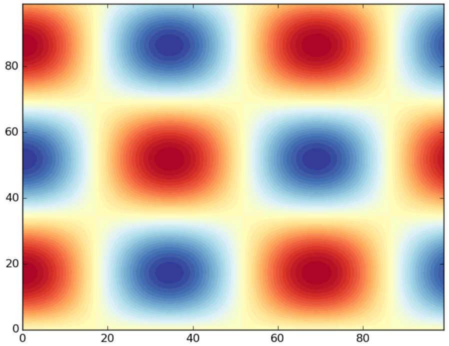




An Example of making a video

```
import numpy as np
import matplotlib.pyplot as plt
def get_field(a, N = 100):
    x = a*np.linspace(0,2*np.pi,N)
    y = np.sin(x); z = np.cos(x)
    return np.outer(y,z)
plt.ion(); plt.show()   #Interactive plot
for n, i in enumerate(np.linspace(0., 5., 200)):
    u = get_field(i)    #Call function with new i
    plt.contourf(u, 40, cmap=plt.cm.RdYlBu_r)
    plt.pause(0.01)   #Pause to allow redraw
    plt.savefig("filename{:05}".format(n),
                bbox_inches="tight")
    plt.cla()   #Clear axis for next plot
```

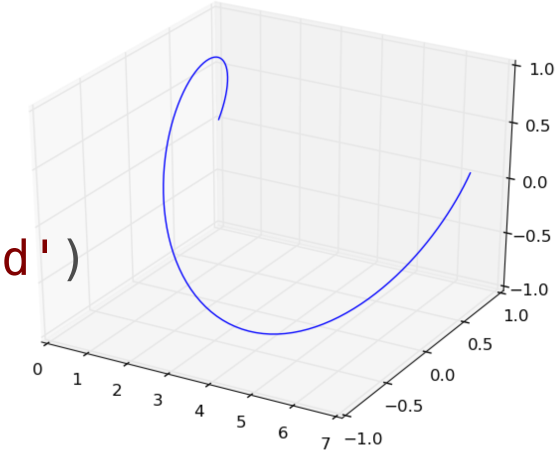




Three dimensional Plots

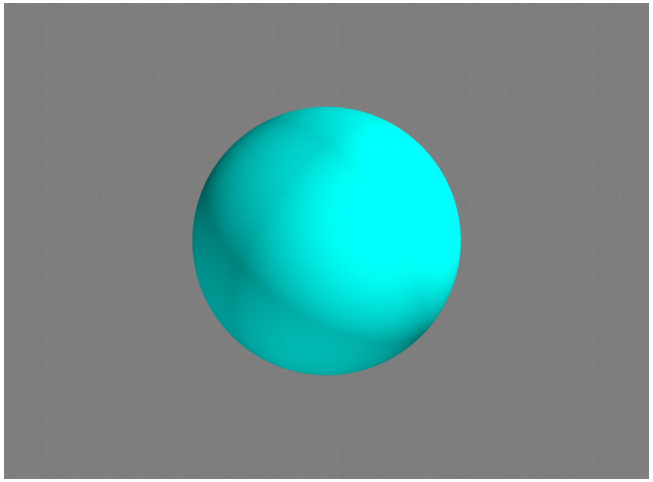
- Some 3D plotting in matplotlib (but limited)

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
x = np.linspace(0.,2*np.pi,100)
ax.plot(x, np.cos(x), np.sin(x))
plt.show()
```



- Generate isosurface data using mayavi (better 3D than matplotlib)

```
import numpy as np
import mayavi.mlab as mlab
x = np.linspace(-1., 1., 100)
y = x; z = y
[X,Y,Z] = np.meshgrid(x,y,z)
out1 = mlab.contour3d(X**2+Y**2+Z**2,
                      contours=[0.8])
mlab.show()
```



A GUI with a Slider

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

#Setup initial plot of sine function
x = np.linspace(0, 2*np.pi, 200)
l, = plt.plot(x, np.sin(x))

#Adjust figure to make room for slider
plt.subplots_adjust(bottom=0.15)
axslide = plt.axes([0.15, 0.05, 0.75, 0.03])
s = mw.Slider(axslide, 'A value', 0., 5.)

#Define function
def update(A):
    l.set_ydata(np.sin(A*x))
    plt.draw()

#Bind update function to change in slider
s.on_changed(update)
plt.show()
```

Adjust figure to make room for the slider and add a new axis axslide for the slider to go on

Define a function to change figure based on slider value. Here this updates the plot data and redraws the plot

Bind function update to slider change

Curve Fitting with Scipy

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

x = np.linspace(0, 4., 30)
y = x + (2.*(np.random.random(30)-.5))
plt.plot(x, y, 'ob')

def linear(x, m, c):
    "Define line function"
    return m*x + c

params, cov = curve_fit(linear, x, y)
yf = linear(x, params[0], params[1])
plt.plot(x, yf, 'r-')

plt.show()
```

Function from scipy. Takes function handle for the fit you want with x and y data. It returns fit parameters (here m and c) as a list with 2 elements and the covariance (for goodness of fits, etc)

We use params (m and c) with the linear function to plot the fit

Hands-On Session 2

1) Create x=np.linspace(0., 10.,1000) and plot x^2 and x^3 on axes ax[0] and ax[1] from plt.subplots(2,1). Change line colour, markers and size
2) Change the y axes on 1) to logarithmic and label the x and y axes
3) Create 2D data from 1D arrays y and z using x=np.outer(y,z) and plot using imshow, contour, contourf and pcolormesh (try different 1D arrays)
4) Fit an appropriate line to

   x = np.linspace(0, 2*np.pi, 100)

   y =np.sin(x) + (2.*(np.random.random(100)-.5))

   Advanced
5) Create fig, ax = plt.subplots(1,1), switch interactive mode on and plot ax.plot(np.sin(A*x)) for A in np.linspace(-5,5,100) using plt.pause(0.1) to redraw and plt.cla() to clear the axis (NOTE WON'T WORK IN NOTEBOOK)
6) Run the slider example and adapt to plot sin(Ax^2) with the value of A specified by the slider value.
7) Develop a slider example with both sine and cosine on the plot updated by slider. Adapt this to add a new slider for a second coefficient B for cos(Bx).