Guide to Plotting a Function in Python (written by J. Bradford)

Suppose we want to plot some function in Python. You have two choices: either type commands line-by-line into the python command window, or type all commands into a file and execute the script on the python command line (> run myscript.py).

To begin, you must include the functionality that allows you to plot, make a set of consecutive numbers. Note that I have added comments to the code by using "#".

```
# Import and define some libraries, this section of code is for GLOBAL code

# pylab is a tool that replicates matlab plotting windows in python
import pylab

# numpy is a package for scientific computing, need this for a nice array object
# np is an alias, you can refer to the modules as np.module instead of numpy.module
import numpy as np

# matplotlib is essentialy what we plot with
import matplotlib.pyplot as plt

# general system module, handles i/o, etc.
import sys
```

You generally want to define constants first and then variables.

```
#  Make variables and arrays
  G = 6.67e-11 #grav const, m^3 kg^-1 s^-2
  m_2 = 1.90e30 #kg, mass of the sun
  m_1 = 5.97e24 #kg, mass of the earth
  r = np.arange(10,10e12,10e10) #array of sequential floats, in m
  F_g = G*m_1*m_2/(r**2) #Newtons
```

You can see that I have used exponential notation (e.g., 6.67e-11) for numbers. I am also using comments to keep track of units, which is good practice. I use a function in the numpy library called "arange". This creates a range (hence the name) of numbers that starts with 10 and increases in increments of 10e10 until I reach 10e12. Finally, I use the variables I defined to generate some function. Note the operations of multiplication (*), division (/) and exponentiation (**). You can read more about operations here:

http://www.tutorialspoint.com/python/python_basic_operators.htm

Now, we can initialize the plot object.

```
# Set up the plot
pylab.figure(1) #initializes or focuses on figure 1
pylab.clf() #clears the figure window
```

This code will open up a figure with index 1, or focus on that figure if you have already plotted in figure 1. Then it will clear the figure window in case you have already plotted to that figure.

Finally, we write the code to plot in our figure. Since we have set focus on figure 1, we don't have to specify what figure we want to do our plotting.

```
# Do the plotting
plt.plot(r,F_g) #send data to the plot
plt.ylabel('log(F (Newtons))') #label the y axis
plt.xlabel('log(r (meters))') #label the x axis
str_title = 'Gravity! of m_1 = ' + str(m_1) + ' kg and m_2 = ' + str(m_2) + ' kg.'
plt.title(str_title) #title
plt.yscale('log') #log scale for the y axis
plt.xscale('log')
plt.axis([0,10e13,0,10e50]) #[xmin,xmax,ymin,ymax]
print 'Figure is plotted, your command line is blocked until you close the plot.'
plt.show() #show the plot and give control to the plotting window
```

Most of this code is explained in the comments, but note that we are manipulating this plot object's properties. We manipulate the plot data (plot), y and x labels, the plot title, the scale of the axes and the range of the axes. We could manipulate dozens of properties of this object, for more properties and and explanation of each see:

http://matplotlib.sourceforge.net/