

Scientific Programming in Python
Notes
30 September 2009

Pylab

We will use the Pylab library to plot graphs. Since this library contains some functions and variables that conflict with the SciPy library, we will start our programs with

```
from __future__ import division
from scipy import *
import pylab
```

We will then call functions from the Pylab library using the dot notation.

Plotting data

The `plot` command can draw any number of lines and markers onto a set of axes to generate very attractive graphs. The basic command has the form

```
pylab.plot(x,y)
```

where `x` and `y` are lists or arrays. The lists must have the same length. To generate the graph, Pylab will draw straight line segments between every pair of coordinates, (x_i, y_i) . An alternative usage is or

```
pylab.plot(y)
```

In this, it is implicit that `x = [0, 1, 2, ...]`.

Formatting lines

It is also possible to specify a format for each plot. For example

```
pylab.plot(x, y, 'o')
```

will draw circles at the data points rather than drawing a line through them. Some of the available format commands are listed in Table 1. Note that some of the commands change the colour of the plot rather than the line style. You can combine different format characters to draw composite plots. For example,

```
pylab.plot(x, y, 'gs-')
```

will draw a green plot (`g`) with a solid line (`-`) and square symbols (`s`) on the data points.

Controlling figure windows

By default Pylab plots multiple graphs on the same set of axes in one window. You can create a new window with the `figure()` command. Every time that you call `figure()`, a new empty plot window is created. Any subsequent `plot` commands draw on this new figure. You will notice that each figure window has a number, for example a figure might be called **Figure 1**. If you want to select an existing figure, use `figure(number)`.

The `show()` command makes all of the figure windows visible. You *have* to call this command to be able to see your plots. If you forget to call `show()`, there will be no output.

-	solid line	--	dashed line
-.	dash-dot line	:	dotted line
.	points	,	pixels
o	circle symbols	^	triangle up symbols
v	triangle down symbols	<	triangle left symbols
>	triangle right symbols	s	square symbols
+	plus symbols	x	cross symbols
D	diamond symbols		
b	blue	g	green
r	red	c	cyan
m	magenta	y	yellow
k	black	w	white

Table 1: Some of the available format characters for changing the line style, data point symbols and colours.

Example The following commands create two figures, plot x^2 and x^4 on the first figure and plot x^3 on the second figure.

```

from __future__ import division
from scipy import *
import pylab

x = linspace(0, 1, 100)

pylab.figure(1)           # Create the first figure
pylab.plot(x, x**2)      # Plot on Figure 1

pylab.figure(2)           # Create the second figure
pylab.plot(x, x**3)      # Plot on Figure 2

pylab.figure(1)           # Return to Figure 1
pylab.plot(x, x**4)      # Plot on Figure 1

pylab.show()

```

Changing the axes

The `axis` command is used to change the ranges of the x and y axes. Calling

```
pylab.axis([xmin, xmax, ymin, ymax])
```

sets the ranges. Calling `axis()` without any parameters returns the current x and y ranges. You can also use the functions `xlim` and `ylim` to set the range of only one of the axes.

By default Pylab scales the axes so that all of the data fits on the plot window. This means that the x and y might have different length scales. This behaviour can be changed with the following commands.

- `axis('equal')` sets the x and y axes to have the same scale. This will make a circle look circular.
- `axis('scaled')` also makes a circle look circular, but does it by changing the dimensions of the plot window (and not the axes).

- `axis('tight')` changes the axes so that the data fits exactly into the plot. This means that the ranges of the x and y axes will be the same as the ranges of the x and y coordinates of the data.
- `axis('auto')` selects the default behaviour.

Finally, you can use `axis('off')` to remove the axes and axis labels completely.

Adding text

You can add text labels to your plot, to the axes and to the graphs on the plot.

- `title('...')` adds a title to the plot.
- `xlabel('...')` adds text to the x axis.
- `ylabel('...')` adds text to the y axis.

The `legend()` command creates a legend with a text description for each of the lines on your plot. If you have 3 lines, you can use

```
pylab.legend('Text for line 1', 'Line 2', 'And line 3')
```

to create the legend. Pylab chooses a good location for the legend on the plot. If you don't like the placement you can use, for example,

```
pylab.legend('Text for line 1', 'Line 2', 'And line 3', loc='upper left')
```

to position the legend yourself. The available locations are

```
'best'
'upper right'
'upper left'
'lower left'
'lower right'
'center left'
'center right'
'lower center'
'upper center'
'center'
```

The 'best' option is the default that lets Pylab choose the location.

Other types of plots

Rather than using the linear axes of `plot()`, you can also use log scaled axes.

- `loglog` makes both the x and y axes log scaled.
- `semilogx` makes the x axis log scaled and the y axis linear.
- `semilogy` makes the y axis log scaled and the x axis linear.

The parameters for these three commands are the same as for the `plot` command.