Tutorial 5: Plotting (1)

We'll plot some things in IDL. This is a simple intro, but we'll do more plotting tutorials later on.

Take advantage of copy & paste and the up arrow in this tutorial. If you select text with the mouse, the text is "copied into the buffer" (as if you'd gone to the "Edit" menu and pressed "Copy" in a word processing program). You can past it by middle-clicking wherever you want it pasted.

BUT FIRST! Make a new folder in your `ASTR2600_assignments` folder and call it `Tutorials`. You'll store and upload your tutorial work to github from here. Navigate to the Tutorials folder, and open a terminal window if you haven't already. Confirm that it shows the correct path on the left. Alternately, the command `pwd` displays your current path. When your terminal is in the `Tutorials` folder, continue. Your journal and log files will save there, and you won't have to worry about moving them around after you're done.

Open IDL. Start a journal file called `tutorial6_plotting_[today's_date].pro`. Replace the part in square braces with today's date!

We'll make two `x` arrays: one "high resolution" and one "low resolution"

```
IDL> x1 = findgen(1000)/999*4*!pi
IDL> x2 = findgen(10)/9*4*!pi
```

Plot and overplot some functions (remember you can use the up arrow key to recall previous commands):

```
IDL> plot,x1,sin(x1)
IDL> oplot,x1,sin(x1)^2
IDL> oplot,x2,sin(x2)
IDL> oplot,x2,sin(x2)^2
```

That looks hideous. We'll add some color.

```
IDL> plot,x1,sin(x1)
IDL> oplot,x1,sin(x1)^2,color='0000FF'x
IDL> oplot,x2,sin(x2),color='00FF00'x
IDL> oplot,x2,sin(x2)^2,color='FF0000'x
```

Well... it's colorful, at least.

We can make the lines thicker:

```
IDL> oplot,x1,sin(x1)^2,color='0000FF'x,thick=3
```

Notice that there's some blank space on the right side. This can be cleaned up with the `xstyle` keyword:

```
IDL> plot,x1,sin(x1),xstyle=1,thick=3
IDL> oplot,x1,sin(x1)^2,color='0000FF'x,thick=3
IDL> oplot,x2,sin(x2),color='00FF00'x,thick=3
IDL> oplot,x2,sin(x2)^2,color='FF0000'x,thick=3
```

What does `xstyle` mean? Open the online help:

```
IDL> ?plot
```

Scroll down until you find the keyword `style`. Read the help. Also note all the myriad options on the way down!

Let's build up the longest sensible plot command we can (it might be a good idea to write this in a text file and copy and paste it so you can scroll around the line more easily):

```
IDL> plot, x1, sin(x1), xstyle=1, thick=3, title='My Plot', xtitle='The X
   Axis', ytitle='The Y Axis', ytickname=['minus 1','minus
   1/2','0','1/2','1']
IDL> oplot,x1,sin(x1)^2,color='0000FF'x,thick=3
IDL> oplot,x2,sin(x2),color='00FF00'x,thick=3
IDL> oplot,x2,sin(x2)^2,color='FF0000'x,thick=3
```

OK, clearly you can do a lot with a single IDL command, but it becomes overwhelming and excessive rather quickly.

There's a way around this: the "line continuation character" `$`

```
IDL> plot, x1, sin(x1), xstyle=1, thick=3, title='My Plot', $
        xtitle='The X Axis', ytitle='The Y Axis', $
        ytickname=['minus 1','minus 1/2','0','1/2','1']
```

Now you've entered three lines of text, but IDL treats them as one. This is a way to make your code neater: if you have a very long command, you can break it up the way you would break up normal text in a paragraph as long as you tell IDL by adding a `$` at the end.

ASIDE: You must use single quotes, not double quotes, if you have a number as the first character. For some quirky reason, IDL treats anything that looks like `"0231` or `"#...` (i.e., a double quote followed by any number) as an "octal" value.

More plotting - now python

Now do the same plot exercises, but now in python. If you decide to work with or for astronomers, odds are you will see both IDL and python being used. So, I want you to at least be able to read and do minimal tasks in both languages.

To start python, run the command

```
ipython -pylab
```

This is not 'vanilla python' - `ipython` is a special environment designed for interactive use. `pylab` means that the syntax (the names of functions and the way you call them) is meant to resemble matlab.

We'll do the same plots as in IDL:

```
>>> # IDL was: x1 = findgen(1000)/999*4*!pi
>>> x1 = linspace(0,4*pi,1000)
>>> # IDL was: x2 = findgen(10)/9*4*!pi
>>> x2 = linspace(0,4*pi,10)
>>> plot(x1,sin(x1))
>>> plot(x1,sin(x1)**2)
>>> plot(x2,sin(x2))
>>> plot(x2,sin(x2)**2)
```

Note the differences:

1. Python defaults to a white background with some buttons.
2. Python has the "convenience" command `linspace`
3. Python does *not* clear the plot by default: clearing the plot requires the command `clf()`
4. Python commands are followed by parentheses, while IDL procedures were followed by commas. We'll revisit this later: IDL uses two different syntaxes for "procedures" and "functions"...

5. The exponent in python is `**`, in IDL is `^`
6. Comments in python start with `#` instead of `;`
7. In "pylab mode", you can use `pi`, `sin`, `linspace`, and other functions just like in IDL. But in python scripting, you would have to call these as `np.pi`, `np.sin`, and `np.linspace` or something similar, because normally these functions and variables are hidden in the `numpy` package.

When you are finished, `exit` python. Now upload your IDL journal and python log to git. For step-by-step instructions look at the `uploading_to_github.pdf` file in the handouts folder (class website or `ASTR2600_materials` folder. Type `git pull` in the folder to get up-to-date).

Tutorial: How to break out of commands

What do you do if you've entered the wrong thing, and you haven't gotten your command line back? This applies to both IDL and the unix (bash) command line.

Say you enter the command

```
$ echo "I like to type
```

and press enter. You're now left with a prompt that looks like `>` instead of your friendly `cosmos ~$`. Bash (the unix command line) thinks you want to enter more text. So, you could type

```
multiple lines"
```

and press enter, and it should print out the text for you.

An alternative is to press and hold the "Control" key and press "C". This "breaks" the currently running command.

Let's try this in IDL. Try running this command, then quicky pressing Control-c: `print,findgen(1000000)` It should stop printing long before it gets to 999999.

You will probably want to use this key combination a lot.

Part 2: When stuff is really broken

NOTE: The following procedure is generally a bad idea.

The following command will wreck your IDL session:

```
IDL> while(1) do print,"Stop!"
```

It will print the word "Stop!" indefinitely. If you hit Control-c, it will not stop: it's an awful runaway process. Awful.

Press Control-z! It will "stop" IDL, sending it to the background. The process isn't really dead yet, we need to `kill` it.

Enter the command `jobs` to show that IDL is still there, hiding. Enter the command `ps` and look for the CMD idl. There should be a PID column - note the number associated with IDL.

Now we kill it [replace PID with the PID #]:

```
$ kill PID
```

Type `ps` again to see if it's still there. It might be, which is annoying.

Type `fg` to bring IDL back to the `foreground`. It should be dead for real now.

Part 3: Quitting things

Start up IDL.

Press Control-d.

This should quit your IDL session! It is a shortcut for that. Be careful, Control-d also quits your "terminal" session if you press it at the `$` prompt.