Intro to IDL for ASTR301

Starting, Stopping, Getting Help
On 'ladyhuggins' you can enter the IDL command-line mode by typing
`idl`
at a unix prompt. To exit this environment type:
`IDL> exit`
To interrupt (without exiting) and resume IDL: use `^z` and `fg` UNIX commands.
To get help with IDL, normally you could type ? in IDL. But this won't work for us, so instead consult the website:
`http://idlastro.gsfc.nasa.gov/idllibsrch.html`
also, there are several IDL resources sitting on the bookshelf outside Wendy Bauer's office, and she has offered to help with IDL problems when I'm not around.

Basic commands `print & help`
`IDL> PRINT,'Hello World'`
`IDL> PRINT, 10.0^(0.4*(4.7+20.5))` – IDL makes a good calculator
`IDL> PRINT,` *`variable`*

`IDL> HELP,` *`optional variable`* – to display information on a variable or all variables and compiled procedures.

Variables
`IDL> a='Hello World'` – makes 'a' a *string* scalar
`IDL> b=2.0e5` – makes 'b' a *floating point* scalar
`IDL> c=3` – makes 'c' a *integer* variable.
`IDL> d=1.63d50` – makes 'd' a *double-precision floating point* scalar
`IDL> e=2062651` – (that is the letter 'l' at the end) makes 'e' a *long integer* scalar
`IDL> f=FLOAT(c)` – makes 'f' a floating point version of c
Other similar 'recasting' commands are `FIX(), LONG(), DOUBLE(), STRING()`
To Do: Experiment with trying to operate on these symbols with math variables: + - * ^ /

`IDL> g=[1,1,2,3,5,8,13]` – makes 'g' an *array* of integers
`IDL> PRINT, g[0], g[2:4]` – prints the first and 3rd-5th elements of 'g'; the 0 and 2:4 are called subscripts
`IDL> h=FLTARR(10)` – makes 'h' a 10-element floating point array, all entries are zero
`IDL> i=FINDGEN(10)` – makes 'i' a 10-element floating point array, values from 0.0 to 9.0
`IDL> j=FLTARR(10,10)` – makes 'j' a two-dimensional array, individual elements accessed e.g. `j[4,3]`
`IDL> k=i*c` – multiply all elements of 'i' with scalar 'c'

Functions and Procedures
Functions return a value:
`IDL> variable_name=FUNCTION_NAME(input_parameter,KEYWORD=keyval)`
Procedures are similar to functions, but don't return a value.
`IDL> PROCEDURE_NAME, input_parameter,KEYWORD=keyval`
You have already met procedures `HELP` and `PRINT`. IDL doesn't care about upper- and lower-case, but capitalizing functions and procedures is a good habit. One of the procedures you will use all the time

in this class is the PLOT procedure.
`IDL> PLOT,i,k`
The arrays 'i' and 'k' are the *input parameters*. We can get fancier:
`IDL> plot,i,k,xtitle='X Axis', ytitle='Y Axis'`
Here `xtitle` and `ytitle` are some of the many *keywords* of the `plot` command.
To do: Look up the PLOT command on the idlastro website listed above and add a title to your plot, change the range of the axes and make the line thick.

Create your own procedures and functions!
Here is a simple example IDL procedure. Open a file in emacs in your astro301 directory called hello_world.pro, then copy in the following text.:

```
PRO HELLO_WORLD
;comment, this is first my program to try out IDL
a='Hello World
print, a
END
```

Note that every procedure should start with a `PRO` line and end with `END`.
To run this program first save it in emacs, then in IDL type:
`IDL> .r hello_world` – this will *compile* the procedure
`IDL> hello_world` – this will *run* the procedure

More generically, you can create procedures using the following syntax for the first line:
`PRO PROCEDURE_NAME, input_parameter, KEYWORD=keyval`
and functions using:
`FUNCTION FUNCTION_NAME, input_parameter, KEYWORD=keyval`
Functions also require that you return a value before the end of the program using:
`RETURN, value`
Both types of programs can have multiple input parameters and keywords. It is good to have the program's file name (in lower case with a .pro attached) and the procedure/function name match.

Debugging
Code is never perfect the first time around. The trick is learning how to diagnose the problems. Two suggestions on how to do this.
`STOP` – putting one of these in your code will stop the program at that location and dump you out to an `IDL>` prompt where you can explore what is going wrong with your program.
`PRINT` – use to print out the value of a variable, or just to check that part of the program is executing.
`HELP, variable` – will tell you what type of variable it is and if it is defined.

Creating Output Files
Text Files: To create an output file and print to it use the commands:

```
OPENW,1,'filename.txt' – the 1 here is the IDL file unit
printf,1,'Hello World'
CLOSE,1
```

Figures: You have two options for creating figures. (1) You take a snapshot of the figure on your screen using command-shift-4 on your Mac. This will save the file on your local computer, not on ladyhuggins!
Another way to create a file is to create a postscript output file. Postscript, like PDF, is a type of file designed for printing. To create a postscript file in IDL requires several extra lines of code (and quite a few complications!):

`SET_PLOT, 'PS'` – changes the plotting window from screen to file
`DEVICE,file='`*`filename.ps`*`'` – give IDL a filename, can also specify size etc.
do all plotting commands here
`DEVICE,/CLOSE` – closes up postscript file
`SET_PLOT,'X'` – returns plotting window to the screen

To view the file, you can use the unix 'gv' command.

Plotting a Blackbody Curve
Create your own program to plot a blackbody and compare fluxes in different bands.

1. Open a file called plot_blackbody.pro in emacs.
2. In the first line, create a procedure which has temperature as an input parameter.
3. Create an array of wavelength values between 3000 and 8000 Ångstroms using the FINDGEN() function.
4. Look up the PLANCK() function on the idlastro website. Use this to create a blackbody curve with your wavelength array and for the temperature you specified.
5. Now create a plot of your blackbody curve vs. wavelength. Label all axes with appropriate units (see PLANCK() program for details).
6. Use the WHERE() function to determine the indices of the array that are at blue wavelengths (4000-5000 Å) and red wavelengths (6000-7000 Å). WHERE() is one of the most useful functions in IDL!
7. Now compute the total amount of flux in the red and blue parts of the spectrum using the TOTAL() function.
8. Print out the temperature and the ratio of blue to red flux.
9. End the program.