This is a short exercise that should help acquaint you with the basics of IRAF.

Starting up IRAF can be a bit complicated (you have to be in the right directory, using the right kind of terminal window, etc.) but on our local machines here there is a script to take care of all of this for you - just type "xiraf" and it should start up iraf in a new window, as well as opening an image display program called ds9. (If this is the first time you've run iraf, it will ask you about creating a login.cl [which is the main iraf user preferences file] - just hit return to accept the default action.)

In this, as in the other exercises, the commands after the prompt (either % or cl>) are meant to be typed by the user; the #-sign indicates a comment.

The exercise uses two images, with file names im010.fits and im011.fits, which you may already find in your ~/iraf directory (which is the directory you're already in by default if you started IRAF as indicated above). If they aren't there, you can copy them to your current directory with

```
cl> cp /data/astro121/iraf_intro/im*.fits .
```

Notice that there is a period at the end of that command, separated from the filename by a space - that looks like punctuation, but in fact it is part of the command, since '.' is linux-speak for 'the current directory'. Also, if you know a little unix/linux, you may recognize that this is just the standard linux cp command - we have IRAF configured locally to pass a common subset of linux commands directly out to the shell to be executed. For those that aren't pre-defined, you can always execute a linux command from within IRAF by preceding it with a !, as in

```
cl> ! emacs file.log &   # Starts an emacs window under linux.
```

Our local installation of IRAF is configured to work directly with FITS files by default, where "FITS" stands for Flexible Image Transport System, a standard astronomical image format. Older versions of IRAF (V2.10.4 and earlier, or anything earlier than the end of 1997) worked with a different format where each file was split into a separate header and pixel files. If you encounter files with extensions .imh and .pix, these are IRAF formatted image header and pixel files, respectively.

```
cl> dir          # list the files in the current directory
                 # and make sure you have the .fits files.
```
The next step is to familiarize yourself a bit with some of the basic operations of IRAF. Tasks with similar/related functionality are grouped together in packages. Tasks, and sometimes packages, have parameter files that control their operation.

```
cl> diskspace       # show the available disk space
cl> path           # show the current working directory
cl> dir            # list the files in your current directory
cl> dir *.fits     # list just the fits files in your directory
cl> dir uparm      # list the files in the uparm directory - a place where parameter files are stored
cl> dir l+         # do a long directory listing
cl> package        # show what packages are currently loaded
cl> help images    # help for the package "images"
cl> phelp imheader # help for task "imheader" using a more sophisticated pager - it lets you:
   <space bar>    # go to next page of help
   b              # go back one page
   ?              # view options
   q              # quit the help for this task
cl> lpar imh       # this doesn't work (try it!) - task names and parameters need enough characters specified to be identifiable
cl> lpar imheader  # note query and hidden parameters - query parameters are listed first with hidden parameters following in parentheses
cl> unlearn imhead # unlearn parameters - go back to defaults
cl> imhead im010   # short header listing - what does it tell you?
cl> imhead         # by default imheader will list all images with the extensions listed in the parameter imlist
cl> imhead im010 l+ # Long version of the output - shows all header fields
cl> epar imhead    # modify the task so it always types the long header listing - choose yes for the parameter entry for "longheader" followed by a return
   :q              # exit eparam mode and "save" the new parameters [exiting with ctrl-C does NOT update the parameter file] You can also exit and save with ctrl-D.
cl> lpar imhead    # was the edited parameter saved?
cl> imhead im011 | page # what is the uparm directory used for? Can you find the parameter file for "imheader"? The name is a combination of the package (imutil) and task name, i.e. imlimhear.par.
cl> dir uparm      # what is the uparm directory used for? Can you find the parameter file for "imheader"? The name is a combination of the package (imutil) and task name, i.e. imlimhear.par.
cl> unlearn imhead # go back to the default parameter setting
cl> dir uparm      # what just happened?
cl> imhead im010 l+ | page # note the keyword "exptime"
cl> unlearn hselect
cl> lpar hselect
```
cl> hselect im*.fits $I,exptime yes
      # see the help for hselect - it allows us
      # to look at selected keywords from image
      # headers
      # What does $I do?
      # What does the "yes" argument do?

cl> unlearn hedit
cl> lpar hedit
cl> hedit im* notice "test data" add+
    # add a keyword plus value to the headers of
    # the two images - hit return after each query
cl> imhead im010 l+   # do you see the new keyword?
cl> hselect im* $I,notice yes

What have you learned about parameters? What is the difference between
query and hidden parameters? What is the uparm directory? Do you
understand the different syntaxes that we have been using in task executions?
If you have questions see the Beginner's Guide mentioned below in the
References section.

A Short Exercise
---------------

There are 2 images in your directory, im010 and im011, of the same
field, but one frame has been slightly shifted from the other. We want
to shift the second frame so it aligns with the first frame, and then
average these two frames together.

cl> dir im*.fits       # note the two frames
cl> imhead im*.fits    # check to see if they are the same field
cl> unlearn display
cl> lpar display      # look at the parameters for the display task
cl> display im010 1   # load the first image into the image display

Let us diverge for a moment. This is probably a good time to acquaint
yourself with the image display program ds9, if you have not used it
before. You should know how to zoom, pan, window, and blink. Most
options can be done with the mouse. Notice the pull down menus at the
top of the window. Pull each menu down and review the selections -
use the left mouse button to pull down the menus. At the center of
the window are some point and click options - clicking in the top row
of buttons changes what buttons are available in the second row for
doing various tasks (e.g. clicking the "Zoom" button displays "+", 
"-", "to fit", etc.). At the top of the window are rough coordinates
of the current cursor position as well as an estimate of the pixel
value at that cursor position. The two boxes in the upper right
corner are "panner" boxes - one shows you what part of the image you
are currently looking at (right now we are displaying the full image),
and one shows you a zoomed-in view under the current mouse position.

The lookup table can be modified by moving the cursor up and down in
the window (contrast) or left and right (brightness) while holding
down the right mouse button. Try it. (You can always get back to
default settings by clicking "Frame", and then "reset".)
Panning (changing which part of the image you're displaying) is done with the middle mouse button. Move the cursor to an interesting part of the image - click once with the middle mouse button and that part of the image will be centered. You can also drag the blue box in the top panner window to move the region displayed. To zoom in, choose the Zoom button, then click "+" or "-" to zoom in or out; you can also zoom in and out with the scroll wheel on the mouse. Play with this until you understand how it works. Go back to the unzoomed image when you are finished.

We want to blink this field and that of im011. So we will need to load the second image into the second frame buffer - you have 4 frame buffers available for images.

    cl> display im011 2

Now try to match the lookup tables of both images. Adjust the contrast and brightness for the second image that you should be looking at now. (Remember, just click and drag with the right mouse button.) Twiddle things until you have black stars on a white sky background. Now look at Frame 1 again (push "Frame", then "next"). Go back to frame 2, and select "Frame -> Match -> Colorbar" from under the "Frame" menu. Now go to frame 1 - they should have the same contrast and brightness now. Then blink between the two frames by pressing the "Blink" button. (You can stop the blinking by clicking "single", or you can blink manually by just repeatedly clicking "Next".) Notice the small shift between the frames? Stop the blink mode by pressing the "single" button.

Play a bit with ds9 until you feel comfortable with it. It has many more features that we will not cover here - see the documentation online, or under the "Help" menu in ds9.

Now back to our original problem of computing and correcting for these shifts.

    cl> display im010 1       # redisplay im010
    cl> unlearn imexamine
    cl> lpar imexamine         # we want to compute the shifts using this task interactively
    cl> imexamine             # move the cursor into the ds9 window - notice that it has changed to a blinking circle - imexamine has put us into "interactive image cursor mode"

    a. put the cursor on some stars and press the "a" key - information will be printed in the xgterm window: the x-coord and y-coord values are on the first line with additional information on the second line - see the help page for imexamine for details

    b. with the cursor in the image window type "?" to see the cursor help - exit cursor help with "q" in the xgterm window
c. what we would like to do is select 3 relatively bright stars and measure their positions in both frames and then compute the differences - we will use the average of the differences to shift the second frame so it aligns with the first

mark 3 stars with the "a" key - space them over the image

d. with the cursor in the image window, type "d" - you will be queried in the xgterm window for the next image - enter "im011" and frame "1"

measure the same three stars in this image

e. quit "imexamine" with "q" in the ds9 window

Compute the average shift, i.e. what shift do we want to apply to the second image so that it aligns with the first? The shifts that I computed are -0.53 and -1.68 - do you agree?

cl> lpar help
cl> help imshift sec=description  # look at just one part of the help page for the task that we want to use

cl> help imshift sec=example
cl> unlearn imshift
cl> lpar imshift
cl> imshift im011 shift011 ??? ???  # shift im011 with the appropriate values

cl> dir  # Note that you have created a new image!

Now, blink im010 and shift011 and see if we did a good job. Do you remember how to do this? Congratulate yourself if things look ok!!

Let's try a little image arithmetic now. We have two frames, im010 and shift011, that we want to average. Let's do it two ways.

cl> unlearn imsum imarith

a. cl> lpar imsum
cl> imsum im010,shift011 aver1 pixt=r calct=r option=average v+  
<or try "epar imsum", modify all of the parameters, and then type ":go" >

[Note the concern here about the pixel type, both for the calculation and for the output image.]

b. cl> lpar imarith
cl> imarith im010 + shift011 aver2 pixt=r calct=r v+  
<try "epar imarith", edit all parameters, type ":go">
cl> imarith aver2 / 2 aver2  # notice we are overwriting the input image
c. [Hopefully the results are the same for both operations.]

```bash
cl> unlearn imstatistics
cl> lpar imstat
cl> imstat aver*.fits
```

Notice that when you change hidden parameters on the command line that they are NOT "learn"ed! How do you "learn" parameters?

**History**

---------

IRAF can redirect terminal output to a file, as well as pipe output from one task into the input of another task. There is also a history and recall mechanism.

```bash
cl> history              # prints history tree
cl> ^                    # recall and execute last command, can also include number to execute any task in tree
cl> e lpar               # recall last lpar command - allows you to edit command line before executing with "return" - use the arrow keys to move cursor, delete or insert to the left of the cursor
cl> e                    # recall last command - use up/down arrows to go up/down history tree
cl> history 100          # look at last 100 commands
cl> history 100 > hfile  # redirect output to a file
cl> page hfile           # page the file
cl> history 100 | page   # alternate method avoiding intermediate file
```

What is the difference between ">" and "|"?

**More on Plotting**

-------------

Let's explore plotting in IRAF a bit. Some plot tasks are interactive and others are not. You can always replot whatever was in the last plot buffer and play with it.

```bash
cl> phelp plot         # list help for "plot" package
cl> contour shift011   # make a contour plot of your shifted image - if it is not already the frame that you are looking at in the ds9 window, re-display it. Compare the plot with the displayed image
cl> surface shift011   # draw a surface plot of the same image
cl> implot shift011    # this is an interactive plotting task that is useful for inspecting 2-d images - type "?" - note the cursor
```
commands (small letters and : commands) - these will differ from
task to task (if they are interactive) - they are NOT global

c - column plot, column read from cursor position

:1 100 - plot line 100

:c 150 200 - plot average of columns 150-200

:.write meta # save plot in file

( Spend some time becoming familiar with "implot" - you
will use it often - try the global keys as well. )

How can you expand the plot other than with "Z"? look at
the "e" key - put both cursors at the lower left corner of
a box defining the region you wish to zoom, press "e",
then move both cursors to the upper right corner of the
box and press "e" again

:1 100 - to get the full plot size back

q # exit

cl> implot dev$wpix # plot the image that is distributed
with the IRAF system

:w world # this image has RA and DEC information
stored in its header; use it for the x axis. (Why doesn't the y axis
display this way as well?)

:f %H # convert to hh:mm:ss

:.write meta

:c # place the cursor on the middle of the
galaxy and type "c" to get a column plot

:f %h # convert to dd:mm:ss

:.write meta

q # exit

We have saved several plots in a file called meta. Let's look at those plots now.

cl> unlearn gkimosaic gkidir
cl> gkidir meta # list plot file
cl> lpar gkimosaic
cl> gkimos meta # plot the meta file
q # to quit plot mode

Now let's print out one of the plots. Choose one of them (use the
gkidir or gkimosaic commands again to look them over). Note its frame
number (as given by gkidir) and then plot it using
cl> gkiextract meta 1 | stdplot

Here I chose frame 1 - use a different one if you wish. In general, you can print from within most plotting tasks by hitting the "=" key (or by typing :.snap), which takes a snapshot of the graphics window and sends it to the printer. Type "help cursor" for more info on this and other graphics cursor commands.

For the next few minutes it may be profitable to spend a bit more time familiarizing yourself with the task IMEXAMINE, since it is such a powerful tool.

cl> display dev$pix 1 # this is the IRAF distributed image, M51. Wpix and pix are the same image but wpix has the world coordinate information in its header

[window the image until it looks good]

cl> imexamine

[put the cursor on the star at 224, 131]

? # list cursor options - type "q" in the text window to exit list
z # print out pixel values around cursor
m # print statistics within box

[move the cursor to a glob on a spiral arm]

s # surface plot
:epar # edit the parameters for the last plot - you must type this from the ds9 window although the command will appear in the graphics window - exit epar mode in the xgterm window with :q

l # plot a line
g # go to "interactive graphics cursor mode" - the global cursor keys can now be used to fuss with the graphics plot
:naverage 10 # average 10 lines - since the graphics window is now the active window type this command in it

i # go back to "interactive image cursor mode"

[try other options]
q # quit the task - you can quit in either interactive mode
One last thing: a pretty picture. Let's overlay a grid in the ds9 window on the image dev$wpix.

```
c> display dev$wpix 1 xmag=0.8 ymag=0.8
cl> wcslab dev$wpix 1          # invert the image to black background and window - if nothing happens when the prompt returns type "gflush"
```

What did the xmag and ymag parameters do?

Print a copy of your image by choosing "Print" from the "File" menu in ds9.

If you are finished with this exercise you may want to clean your directory up a bit with the deletion commands.

```
c> dir
cl> delete hfile,meta          # deletes ordinary files
cl> imdelete *.fits ver+       # deletes image files - you may want to keep im*.fits around but we will not be using them again
```

Why there is a special IMDELETE command for the IRAF images? This is actually a holdover from using IRAF format images which were stored in two separate files, header and pixels. The IMDELETE command would delete both files at once. With FITS files, everything is in one file, so you could also just say

```
c> delete *.fits
or
cl> rm *.fits    # Actually passes it out to the shell
```

It's best to use the ver+ option (notice that this is how you "turn on" yes/no parameters without editing the parameter file) when you delete anything so that you don't accidentally wipe out a bunch of files! To set this as the default, you can type "epar imdelete", change "verify" to "yes", and save the parameters; the same for the "delete" command. The parameter files are saved in your "uparm" directory for your next IRAF session.

```
c> logout
```

You should always log out of IRAF first before shutting down the window environment with the mouse. Then log off the linux system, if you wish.

```
% logout
```

References

DS9 version 1.9.7.1 documentation

(Note: this exercise was written by Jeannette Barnes of NOAO and modified by Eric Jensen to reflect the use of FITS files as the default image format and DS9 [rather than ximtool] as the image display program.)

Last update Feb 2014.