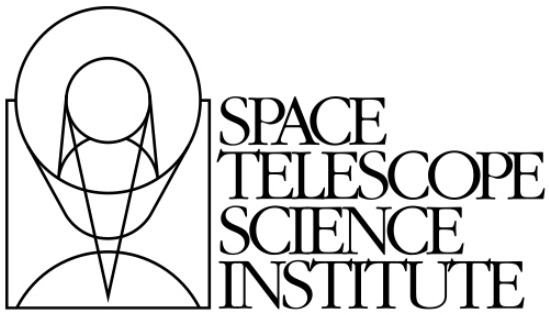
Version 1.0
January 2012

Basic PyRAF



Space Telescope Science Institute
3700 San Martin Drive
Baltimore, Maryland 21218

Operated by the Association of Universities for Research in Astronomy, Inc., for the National Aeronautics and Space Administration

Revision History

Version	Date	Editor and Contributors
1.0	January 2012	K. Azalee Bostroem

This is an unofficial document for internal RIAB training purposes only

Send comments or corrections to:
Space Telescope Science Institute
3700 San Martin Drive
Baltimore, Maryland 21218
e-mail: bostroem@stsci.edu

Contents

Chapter 1

Introduction to PyRAF

1.1 What is PyRAF?

PyRAF is a command language for running IRAF tasks in a Python like environment. It works very similar to IRAF, but has been updated to allow such things as importing Python modules, starting in any directory, GUI parameter editing and help. Most importantly, it can be imported into Python allowing you to run IRAF commands from within a larger script. PyRAF is part of the stsci_python package and is a product of the Science Software Branch right here at STScI. All of the STScI calibration pipelines can be run from PyRAF although they are now written in Python and more recent pipelines (e.g. *Calcos*) can be run as stand-alone programs.

1.1.1 IRAF

IRAF stands for Image Reduction and Analysis Facility and is written and supported by the National Optical Astronomy Observatories (NOAO). It has many tools for both spectroscopic and image analysis. Scripting in IRAF can be done using the CL scripting, however that will not be covered in this document. Most information covered in this document applies to both IRAF and PyRAF but is generally more user friendly in PyRAF.

1.2 Getting Started

1.2.1 login.cl file

All of your PyRAF preferences are held in a file with the name login.cl. If you are running IRAF, then your login.cl file must be in the directory in which you start IRAF. PyRAF can be called from any directory as long as your login.cl file is located in an directory called iraf in your home directory. When you start PyRAF it will first look for a login.cl file in the directory in which you started it (this allows you to have different configurations for different projects). If it doesn't find a login.cl file there it will then look in your /home/iraf directory (where home is your login name). You can use your login.cl file to set all kinds of preferences (look inside it, most things are pretty self explanatory, but ask if you have questions). In here you can also add packages and set paths to commonly used directories (such as oref) you can see some are already set up.

1.2.2 IRAF, IRAFX, and IRAFDEV

There are three environment which you can use which get updated at various times. These set the versions of STSDAS, STSCI_PYTHON, PyRAF, and PyFITS. The IRAF environment contains packages which have been released to the public. This generally happens a few times a year. This is the most stable (but least up to date) environment. The IRAFX environment is updated weekly and may contain some bugs but is generally pretty stable. This is the environment chosen by most people at the institute. The IRAFDEV environment is updated daily. More details can be found at: http://stsdas.stsci.edu/configuration.html#mac

All Macs have a local copy of IRAF, IRAFX, and IRAFDEV installed. This is not automatically updated. If you would like it to be, please email support@stsci.edu and specify when you would like your packages updated. Most people choose very early Monday morning.

You can set the default environment in your .scienv file in your home directory. If you open this you will see iraf, irafx, irafdev each on a new line with 2 commented out. The uncommented one is your default environment. You can also set an environment temporarily in a terminal window by typing IRAF, IRAFX, or IRAFDEV in the command line.

When you start up PyRAF, the environment will be printed prominently in the startup text.

1.2.3 Exercises

1. Find and open your login.cl file. See if you can find which editor is used by default. Are any packages loaded automatically?

2. Try switching between IRAF, IRAFX, and IRAFDEV. Open PyRAF in each to confirm that you switched.

3. Open your .scienv file and find which IRAF environment is set.

1.3 The Basics

1.3.1 Navigation

PyRAF contains many packages which have to be imported. You import a package by typing the name in the command line. For example, to use the **Calstis** function I need to have the **stis** packge loaded which is in the **hst_calib** package, which is in the **stsdas** package, I type the following in a PyRAF window:

```
–> stsdas
–> hst_calib
–> stis
```

To find out the packages available in your current package type *?*. For example:

```
–> stsdas
–> hst_calib
–> ?
```

This will tell you all of the packages in **hst_calib**. Functions with an @ symbol are parameter tables, not functions.

1.3.2 Help

You get help information on any package or function by typing **help function**. This not only tells you about a function, it also tells you the packages that you have to call to get to it. For instance, if you want to use the command **splot**:

```
–> help splot
```

This will tell you that **splot** is in the **onedspec** in the **noao** package. This can also be entered in GUI form from the epar GUI.

1.3.3 epar

All functions have input parameters most of which can either be entered in the command line or can be set in the parameter editor using the epar GUI. If you type: *epar function* you will get a table of all of the parameters. There are 5 buttons at the top of the epar editor: Execute, Save, Unlearn, Cancel, and Help. Execute runs the function with the parameters given, Save exits the epar editor and saves the parameters but doesn't execute the function, Unlearn set all of the parameters to their default values, Cancel exits the epar editor without saving, and Help displays the help file for the function.

1.3.4 Using a List as Input

Many PyRAF functions cannot use wildcards but can take a file which is a list of files. A file which is a list is distinguished from an input file for processing by using an @ sign at the beginning of the file name. For example if you have 3 files, file1.fits, file2.fits, and file3.fits, you can first make a file which is a list of the file names:

> ls *.fits > filelist.txt

Then use this as the input to the **catfits** function:

–> catfits @filelist.txt

1.3.5 importing function

Because PyRAF is Python based, you can import any Python module. This is particularly useful when calibrating a group of datasets as many instrument pipelines (Cal*) will not take wild cards as inputs. If you are mixing PyRAF and Python commands you often have to put an iraf. infront of the commands to let PyRAF know that your are executing a PyRAF command. In the following example I will import the **glob** module from python and use it to select all of the raw files in a directory. I will then calibrate each raw file (note I use iraf.calstis rather than just calstis).

```
> pyraf
–> import glob
–> flist = glob.glob('*raw.fits')
–> for ifile in flist:
     iraf.calstis(ifile)
```

1.3.6 Exercises

1. In PyRAF located the package **apextract**. What tasks are in this package? What does the **aptrace** task do? Are there any parameter tasks?

Chapter 2

Useful IRAF Tasks

2.1 General

2.1.1 Institute Specific Packages

2.1.1.1 stsdas

The **stsdas** package (Space Telescope Science Data Analysis Software Package) is a package put together by the institute. It includes all of the instrument specific calibration packages as well as other helpful tasks. You will most likely load it every time you use PyRAF and may want to consider adding it to your login.cl file.

2.1.1.2 hst_calib

The **hst_calib** package contains all of the HST science instrument calibration packages. These packages contain all of the tasks which are used to calibrate data from each instrument as well as tasks to model the combined response of the various detectors and optical elements. It also contains tasks to derive the instrument calibrations and construct the calibration reference files.

2.1.2 catfits

catfits gives you the details of a file. For each extension it tells you the type (table or image), name, dimension, and pixel type.

2.1.3 Header Information

There is a lot of information contained in the header of each file. These tasks will help you view and edit them. I will only describe these briefly as I encourage you to use python or IDL for these tasks. Note a version of **hedit** and **imheader** have been written in Python. Please see the help sections of these tasks for additional information on the parameters and examples of their use.
WARNING: **imheader** and **hselect** only work on the 0th extension of fits binary tables. **thselect** can be used instead of **hselect**.

2.1.3.1 imheader

imhead displays the header information for a given image. If the parameter LONGHEADER = no, the image name, dimensions, pixel type, and title are printed. If LONGHEADER = yes all header data for a given extension is printed. If no extension is specified, the 0th extension is used.

2.1.3.2 hselect

hselect displays a single header keyword.

2.1.3.3 hedit

hedit updates or adds a given header keyword.

2.1.4 Exercises

1. Use **catfits** to determine how many extensions the file /grp/hst/cdbs/oref/t9a1003so_tds.fits has.

2. Use imhead to determine the dimensions of the file /grp/hst/cdbs/oref/v8918108o_bia.fits.

3. Use thselect to find the value of the DETECTOR keyword in the file /grp/hst/cdbs/oref/ub920085o_tds.fits.

4. Copy the file /grp/hst/cdbs/oref/ub920085o_tds.fits to your computer. Modify the DATE keyword to be today's date

2.2 Images

2.2.1 display

This task is used to display images in DS9. You must make sure a DS9 window is open before you use the **display** task. **display** has many options which are detailed in the help section, in general you need to specify an extension and a DS9 frame for the image to be displayed in. When you use **display** to bring up an image in DS9 you cannot use the DS9 options for scale.

For this reason you may sometimes find it helpful to open a file directly in DS9 rather than using PyRAF.

EXAMPLE

> ds9 &	Open DS9
> pyraf	Open PyRAF
–> display /grp/hst/cdbs/oref/u211310do_bia.fits[1] 2	Display the first extension of the file in the second DS9 frame.
z1=-0.7425513 z2=15.1016	This is output from **display** which tells you about your contrast and scaling.

2.2.2 imexamine

imexamine allows you to display your image in a similar manner to display, however it has additional image analysis capabilities. By placing your cursor over the image and typing certain commands, various types of plots and text output can be created. A complete list tools can be found in the help to the **imexamine** task. I most often use 'e' to create a contour plot, 'c' to create a column plot, 'l' to create a line plot, and 'r' to create a radial profile. Type 'q' when you are in the image window to exit **imexamine** and continue your work.

EXAMPLE

> ds9 &	Open DS9
> pyraf	Open PyRAF
–> imexamine /grp/hst/cdbs/oref/u211310do_bia.fits[1] 2	Display the first extension of the file in the second DS9 frame.

You should now have a circular cursor on top of the DS9 window. Place the cursor over a part of the image and type e. You should see a new window pop up with a contour plot of the area where your cursor is on the image. Press q to exit **imexamine**.

2.2.3 imstatistics

imstatistics is a task which computes statistics of the pixel values of images. **imstatistics** can compute the number of pixels, mean, median (called midst), mode, standard deviation, skew, kurtosis, min and max pixel values. Upper and lower sigma clipping can also per performed for a set number of iterations.

EXAMPLE

```
–>imstatistics /grp/hst/cdbs/oref/u211310do_bia.fits[1]
#   IMAGE                                NPIX      MEAN    STDDEV    MIN      MAX
    /grp/hst/cdbs/oref/u211310do_bia.fits[1] 1048576  1.487   3.747     -18.41   2224.
```

2.2.4 imcalc

imcalc is used to combine images arithmetically pixel by pixel (sum, multiply, divide, etc). A list of images is used in the input field, and then each image is referred to as im1, im2, im3, etc. in the calculation.

EXAMPLE

Find the average of the first extensions of j8c0a1011_drz.fits and j8c0c1011_drz.fits in their over-lapping regions [1:4211, 1:4232] and call the output out.fits

```
–> imcalc j8c0a1011_drz.fits[1][1:4211,1:4232],j8c0c1011_drz.fits[1][1:4211,1:4232] out.fits "(im1+im2)/2."
Percent done: 10 20 30 40 50 60 70 80 90 100
```

2.2.5 Exercises

1. Download the following datasets from the MAST archive: j8c0a1011, j8c0c1011, j8c0d1011
2. Display the first extension of j8c0a1011_drz.fits in the first frame of DS9
3. Use **imexamine** to make a radial plot of one of the stars in the first extension of j8c0c1011_drz.fits
4. What are the maximum and minimum pixel values of the j8c0d1011_drz.fits? What if you use an upper sigma boundary of 3 and a lower of 2 and repeat the clipping 5 times? How many pixels are eliminated using this sigma clipping?
5. Find the dimensions of each image. Create a weighted average of the 3 images, weighting by their exposure times.

2.3 Photometry

2.3.1 daofind

daofind is a commonly used tool to locate a bunch of point sources in an image. **daofind** searches images for local density maxima, which have a full-width half-maximum of datapars.fwhmpsf and a peak amplitude greater than findpars.threshold * datapars.sigma above the

local background, and writes a list of detected objects in the file output. For this reason, the values that you use in these parameters will greatly affect the number of objects missed and the number of object with false detections. Each image is different and usually requires some parameter tweaking.

EXAMPLE:

Parameter	Assignment Value	Maximum Objects Value
datapars.sigma	3.6	2.0
datapars.fwhmpsf	2.0	2.0
datapars.gain	ccdgain	ccdgain
findpars.thresh	50	20
findpars.sharplo	0.2	0.2
findpars.sharphi	0.8	0.8
findpars.roundlo	-1.0	-1.0
findpars.roundhi	1.0	1.0

Set the above values in the parameter files using epar.

2.3.2 tvmark

tvmark will mark the locations given in a text file on an image displayed in DS9. You can specify the shape marked, the size of the marker, and the marker color. This is most often used to check which objects were chosen when one uses a program to manually find stars (such as daofind or star find).

EXAMPLE

```
–> display f435w_drz.fits[1] 1
–> tvmark 1 cood_file.txt mark = circle radii = 8 color = 205
```

2.3.3 phot

phot performs aperture photometry on a list of stars based on a series of parameter files. The parameter files used are photpars@, centerpars@, and fitskypars@. Datapars describes instrument dependent parameters, photpars describes the functions used for computing the photometry, centerpars describes the centering algorithm parameters, and fitskypars describes the sky fitting parameters.

EXAMPLE

–>epar datapars
–> epar findpars
–>daofind f435w_drz.fits[1]
Input image(s) ('f435w_drz.fits[1]'):
FWHM of features in scale units (2.5) (CR or value):
New FWHM of features: 2.5 scale units 2.5 pixels
Standard deviation of background in counts (INDEF) (CR or value):
New standard deviation of background: INDEF counts
Detection threshold in sigma (6.) (CR or value):
New detection threshold: 6. sigma INDEF counts
Minimum good data value (INDEF) (CR or value):
New minimum good data value: INDEF counts
Maximum good data value (INDEF) (CR or value):
New maximum good data value: INDEF counts

Parameter	Value
datapars.exposure	EXPTIME
fitskypars.annulus	10.0
fitskypars.dannulus	5.0
fitskypars.salgoithm	mode
fitskypars.skyvalue	0.0
photpars.apertures	3.0, 10.0
centerpars.calgorithm	centroid
centerpars.cbox	10.
centerpars.maxshift	6

Set the above values in the parameter files using epar.
–> epar datapars
–> epar fitskypars
–> epar photpars
–> epar centerpars
Run **phot**

–> phot f435w_drz.fits[1] f435w_drz.fits1.coo.1

2.3.4 Exercises

1. Run daofind on j8c0a1011_drz.fits. Play with the parameters to try at least 3 runs. Choose your best run (most correctly marked stars and least incorrectly marked stars) for 2 and 3.

2. Mark the location of the stars found in 1 with green squares.

3. Perform aperture photometry on j8c0a1011_drz.fits using the same parameter values as were used in the example

2.4 Tables

2.4.1 tdump

tdump takes a table from PyRAF memory and writes it to an ASCII file. This can be especially helpful for global edits which are difficult using tedit.

EXAMPLE

–> tdump f435w_drz.fits1.mag.1 colum=c7,8,15,29,30,31,38,39,40 datafil=f435w.phota

2.4.2 tmerge

tmerge is used to merge two tables or to append one to the other.

EXAMPLE

–> tmerge f435w.phota,j8c0d101160_dao_ascii.tab phot_all.tab option="merge"

2.4.3 tcalc

Similar to **imcalc**, **tcalc** performs arithmetic operations on table columns.

EXAMPLE

–> tcalc phot_all.tab 435_apcor "mag3_435-mag10_435" colfmt=%6.5g

2.4.4 Exercises

1. Look in the help files for tcalc for all of the arithmetic options.

2.5 Spectroscopy

Note also that many IRAF spectral functions (including scopy and scombine) accept the optional parameters w1 and w2, where w1 refers to the minimum wavelength and w2 the maximum wavelength. Thus, to copy the wavelength range from 1700 - 3200 Å, the command would be

–> scopy input.fits output.fits w1=1700 w2=3200

The **sarith** task also has many of the same functions as **imarith**, except that, by default, it will apply those functions in wavelength rather than pixel space.

Most of these tasks require a one-dimensional image rather than a fits binary table. Therefore you will need to use **tomultispec** to convert your table into an image before using **scombine**, **splot**, and **continuum**.

2.5.1 tomultispec

The **tomultispec** command, available in the **hst calib.ctools** package, is the canonical way to move from STIS 1-dimensional spectra (sx1.fits or x1d.fits, which are stored as FITS binary tables) to standard spectra stored as one-dimensional images (which may then be plotted using splot or an equivalent tool). Note that tomultispec produces imh rather than FITS files, and these can be converted back into FITS files (e.g. so that they can be sent or copied elsewhere more easily) through the use of the **imcopy** command. Be sure that the imdir in your login.cl file is set to "$HDR"

EXAMPLE

–> tomultispec obmzl1020_sx1.fits obmzl1020_sx1.imh

2.5.2 scombine

scombine is a good tool for combining multiple spectra (in order to improve the final signal-to-noise ratio when several spectra were taken with the same settings). It allows any number of spectra to be combined by taking the average, the median, or the sum of the input spectra (using the combine parameter).

EXAMPLE

To combine "obmzl1020_sx1.imh" and "obmzl2020_sx1.imh" into "out.fits", using the median of the input spectra, the command would be:

–> scombine obmzl1020_sx1.imh,obmzl2020_sx1.imh out.imh comb=median

2.5.3 continuum

continuum is a task intended for normalizing spectra (it has many other functions, but that is the most important one when working with spectra here). **continuum** allows fit regions to be selected with the "s" key (pressing the first time to establish the start of the fit region, and the second time for the end, and using "z" to delete a fit region. Note that the continuum should not be fit across the line of interest, as this might distort the line's shape or equivalent width. Other useful commands in continuum include ":func", allowing the fitting function to be set (e.g. to "chebyshev", "legendre", "spline3"), ":o N", setting the order of the fitting function to N, and "f" to see the new fit after changing the order and fitting function.

EXAMPLE

–> continuum obmzl1020_sx1.imh obmzl1020_sub1.imh
Fit [1,1] of obmzl1020_sx1.imh w/ graph? (yes|no|skip|YES|NO|SKIP) ('yes'): yes

2.5.4 splot

splot is a tool that does many spectroscopy-related tasks. In addition to displaying spectra, it also allows you to fit spectral lines, determine signal-to-noise ratios, and even de-blend overlapping lines. splot has many other capabilities in addition to those described here, but these at least provide some information. Below are a very few of the extensive capabilities of splot. I encourage you to look at the help file to see what else is available.

WINDOWING: Press **w** to enter windowing mode. To autoscale the data between a range of values (basically zooming) move your cursor to one edge of the range you wish to use and press **a** then move your cursor to the other end of the range and press **a** again. Press **c** to clear all windowing and return to your original spectrum.

SIGNAL-TO-NOISE: Move your pointer to the leftmost part of the area you wish to check and press **m**, then move your pointer to the rightmost part of the area and press **m** again. In addition to SNR, you also get the average value and the rms.

LINE FITTING: Move your pointer to the leftmost edge of the line and press **k**. Then move your pointer to the rightmost edge of the line, and press **k** again for a simple fit, **g** for a Gaussian fit, **l** for a Lorentzian fit, and **v** for a Voigt profile fit. This gives you the line center, flux, equivalent width, and fwhm.

EQUIVALENT WIDTH: Move your pointer to the leftmost point of the area and press e, then move the pointer to the rightmost point of the area and press e again. This yields the line center,

equivalent widths, continuum, and flux. You can also use k for different profile fitting options and h for the most flexibility.

DEBLENDING: Move your pointer to the leftmost part of the line group and press **d**, then move your pointer to the rightmost part of the group and press **d** again. Now move your pointer to the centre of each line in the group, each time pressing **g** for a Gaussian fit, **l** for a Lorentzian fit, and **v** for a Voigt profile fit. Press **q** when all lines have been marked. Then press **s** to fit a single line, or a to fit all lines. Do the same for Gaussian, Lorentzian, and Voigt profile lines. Press **y** to fit and subtract the background, or **n** to leave it in place in the fit. You can now press **+** and **-** to move between fits, and **q** to exit the deblend fitting mode.

2.5.5 Exercises

1. Combine the datasets obmzl3030_sx1.fits and obmzl4030_sx1.fits .
2. Take your combined spectrum from 1 and remove the continuum
3. Display the continuum subtracted combined spectrum in splot. Fit a line with a Gaussian and a Voigt profile. What changes in the reported values?