The FITS format

The file format used to store most Astronomy data was developed in 1981 and therefore has a lot of quirks resulting from the technology available at the time. To read or write FITS files, you should use the package astropy. There are packages such as pyfits and cfitsio, but they were absorbed into the tools provided by astropy and are therefore no longer maintained.

HDUs

The FITS format is built of Header-Data Units (HDUs). A header-data unit contains a header and data. Headers are composed of 80-character cards, each of which contains an 8-character name, a value, and an optional comment. Headers must be a multiple of 2880 characters and terminate with the keyword `END` . The data in an HDU can be a table of ascii characters, a binary table, or an image. The type of data (if any) is specified in the header. FITS files must start with a primary HDU which must be an image, and not every HDU must have data. There are a couple of other quirks in the specification you may run in to.

Reading FITS files

Reading from FITS files is quite easy with astropy. First off, we need get access to the part of astropy we want to use. Astropy is organized into modules, and the particular module we want is `astropy.io.fits` . The common import statement here is `from astropy.io import fits` . This asks Python to load astropy, but gives us a nice shortcut, instead of having to type out `astropy.io.fits` , we can just say `fits` . The module we've just imported has a lot of attributes (components). There one we want to read files is called `fits.open` . You can read a complete description online, but for now this function expects a file path (or name) and will return a list of HDUs.

```
from astropy.io import fits

hdulist = fits.open('some_fits_file.fits')
```

You may notice that astropy doesn't care at all what the name of the file is. It's easy to mistakenly open something that isn't a FITS file. Try that and see what happens.

Now that we have our list of HDUs, we want to grab the first element. Indexing in C-based languages starts at 0, so in this case we want `hdu = hdulist[0]` . Now we have a header data unit, which has two important attributes, the header and data. We access these like every other attribute. Try printing `hdu.header` . What you got probably looks like a mess. I don't know why astropy does this to us, but to actually get a useful display of the header, we need to use `print(repr(header))` . `repr` is a Python built-in function that asks for a nice printable representation of something.

This HDU also has some data, so let's print `hdu.data` too. Astropy will always produce a numpy ndarray for image HDUs, and some sort of astropy table for table HDUs.

Writing to FITS files

Writing to fits files is much harder than reading. At the simplest level, `fits.writeto` has you covered if you just want to save a file with a single HDU and a minimal header, or if you already have the header prepared.

If you want to create your own entries in a header, you need to first construct the header you want then append new cards to it. For example,

```
primary = fits.PrimaryHDU()
primary.header.append(fits.Card('MJD-OBS', time))
hdulist = fits.HDUList([primary])
hdulist.writeto('some_file_name.fits')
```

Astropy provides very good instructions and examples on building FITS files in their official documentation.