Astronomical Data Analysis Software and Systems VIII
ASP Conference Series, Vol. 172, 1999
D. M. Mehringer, R. L. Plante, and D. A. Roberts, eds.

Python in Astronomy

Norbert Pirzkal, Richard N. Hook

ST-ECF, Karl-Schwarschild Str.2, Garching bei Munchen D-85748, Germany

Abstract. We report on the use of Python to perform basic astronomical tasks. Python is a powerful, object-oriented scripting language that is easy to extend, free, and available for most computer platforms. Python, together with its Numerical module, and with the pFitsio module that is being developed by one of the authors, provides an attractive alternative to other languages such as Perl, IDL, and the IRAF CL, which are already widely used by the community. Simple examples of how Python can be used to explore and manipulate real astronomical data are given.

1. Why Python?

Python is an interpreted, interactive, object-oriented programming language. Invented in 1990 by Guido van Rossum, it shares many of its features with other modern languages such as Tcl, Perl and Java. Python has a simple, easy to learn syntax which emphasizes readability, and it draws many of its features from other popular languages such as C, C++, Modula-3, ABD, and others. We have found that Python satisfies the need for a high level programming language which fills the gap between regular shell scripting languages and compiled languages such as C. For tasks that are too complicated and too large to implement with regular shell scripts, but for which the extra overhead of coding everything in C is not warranted, Python offers a quick and elegant option. Some of the attractive features of Python include the use of transparent byte-code compilation for speed, automatic memory management and garbage collection, and a very powerful object oriented and modular design. Python ships with a large number of modules (See the Python standard documentation), and can be easily extended in C. With modules such as pFitsio or PyFits and the Numerical modules, astronomical images and tables can be easily accessed and manipulated as numerical arrays. We present small Python examples as illustrations of the language's power and encourage the reader to learn more about Python from the book by Lutz (1996).

2. Examples

2.1. Reading a List from a File and building Dictionaries

In this example (Fig. 1), we assume that we have a file called *assoc.clear*. This file contains a list of datasets from the HST STIS camera which have been previously

determined to be combinable into longer integration time images that we wish to refer to as "Associations". The file is as follow:

```
ID       sub dataset   xoff  yoff   ra     dec  r d  exp
046P3X010 1 046P3X010  0.00  0.00 271.37 -19.90 0 1 300.00 ...
046P3X010 1 046P3Y010 -0.07 -0.02 271.37 -19.90 0 1 300.00 ...
...
```

We would like to be able to easily query all the members of a given association ("ID" column), and to be able to access and potentially modify the information of the individual datasets that make up an association. We therefore build a Dictionary of datasets where each dataset's name serves as a key in this dictionary. Each entry of the dataset dictionary is in fact itself a dictionary which uses the original ASCII file column name as keys to hold the ASCII file original data. Hence, dataset['O46P47010']['xoff'] would return the value of 0.0. Another dictionary is then created and uses the names of the associations (Column 1 in the ASCII file) as a key. Each association key is then assigned a list of datasets from the dataset dictionary. Once the dictionaries are set up, which only takes a few lines (grey overlay region), one has the ability to manipulate and "query" those dictionaries in a very high level, natural, way.

2.2. Reading FITS data into Arrays and manipulating those Arrays

Starting from the set of dictionaries we have built in our first example, we now proceed to read in each individual FITS file (Fig. 2). Notice that we now load both the pFitsio and the Numerical modules and how, once an image is stored in a two dimensional array, we can compute statistics on any part of this array. Our second example also shows the use of the powerful Numerical module's where() command which the IDL[1] user will recognize.

3. Availability and References

Python runs on virtually all computer platforms, including the JAVA virtual machine (JPython). One can download the compiled Python binaries or the Python source code from the Python Homepage[2]. Exhaustive Python documentation, tutorials, JPython, and third party modules can also be found on the Python homepage. Information concerning the pFitsio and PyFits modules should be directed to the authors of this paper, and Paul Barrett (see Barrett & Bridgman 1999), respectively.

References

Barrett, P. E. & Bridgman, W. T. 1999, this volume, 483

Lutz, M. 1996, Programming Python, (Sebastopol: O'Reilly)

[1]IDL is the trademark of Research Systems, Inc.

[2]`http://www.python.org`

```
import futils
```
Import the author's file utility module

```
d = futils.parseasciifile("assoc.clear"
```
Parse the ASCII file and store the result in an array d

```
d[0]
```
The first row of the file contained the column descriptions

```
['assoc_id', 'sub_assoc_id', 'dataset', 'xoff', 'yoff', 'ra_targ',
'dec_targ', 'ra_sd', 'dec_sd', 'actual_duration', 'ass_release_date_dmf']

d[2]
```
The association ID is in the first column, the dataset member's name in the third column

```
['O46P3X010', 1.0, 'O46P3X010', 0.0, 0.0, 271.378077, -19.90682, 0.0, 1.0,
300.0, 558543382.0]

len(d)
14
```
Get the length of the array d.

```
dataset={}
dic={}
```
Initialize an empty Dictionary to contain all the datasets (each line in the original ASCII file) and another one to contain the dataset associations

```
for i in d[2:]:
```
Loop over the rows of d

```
      tmp={}
```
Initialize an empty Dictionary to hold the associations

```
      for j in range(3,len(i)):
            tmp[d[0][j]]=i[j]
```
Assign each element of a row to a Dictionary using the column description of the original ASCII file as a key for this element

```
      dataset[i[2]] = tmp
```
In the dataset Dictionary, create a new element using the dataset's name as the key, and assign to it the Dictionary that was just created previously for this dataset

```
      if dic.has_key(i[0]):
            dic[i[0]][i[2]]=dataset[i[2]]
      else:
            dic[i[0]]={}
            dic[i[0]][i[2]]=dataset[i[2]]
```
If an entry for the current association of which the current row/dataset is a member exists then add this dataset to what is already in the association dictionary. Otherwise, create a new one.

```
dic.keys()
```
The association Dictionary should now contain a table of known associations, which we display here by querying the valid keys of this Dictionary

```
['O46P3X010', 'O46P46010', 'O46P49010', 'O46P40010', 'O46P47010',
'O46P4D010', 'O46P4A010']

dic['O46P49010'].keys()
```
We can now query the valid keys of a particular association, hence returning the list of datasets in this association

```
['O46P4C010', 'O46P4B010', 'O46P49010']

dic['O46P49010']['O46P4B010']['xoff']
```
Find out the x-offset of the dataset O46P4B010 member of the association O46P49010

```
0.04

dataset['O46P4B010']['xoff']=-1.0
dic['O46P49010']['O46P4B010']['xoff']
-1.0
dataset['O46P4B010']['xoff']
-1.0
```
Changing a value in one of the entries of the dataset dictionary properly affects the content of the association Dictionary

```
for item in dic['O46P49010'].keys():
    print item,dic['O46P49010'][item]['xoff'],\
              dic['O46P49010'][item]['yoff']

O46P4C010 0.06 10.76
O46P4B010 -1.0 10.78
O46P49010 0.0 0.0
```
Once the association dictionary is populated, we can easily query information about association, loop over its member datasets etc...

Figure 1. Dictionary and List creation and manipulation, (see text for more description).

```
import Numeric
import string
import pfitsio
import Statistics
```

Load the Numerical module to handle array algebra, the string module for string manipulation, the pfitsio module to access FITS files, and the Statistics module to provide some basic statistical functions.

```
for item in dic['O46P49010'].keys()
```

Following the previous example, iterate over the member datasets of the association O46P49010

Read the data from the second extension of the file "item"_crj.fits, and store the result in the array data

```
   data = pfitsio.read_data(string.lower(item)+"_crj.fits",2)
```

Get the x and y direction offsets from our dataset Dictionary, and convert those to integer values

```
   xoff = -int(dic['O46P49010'][item]['yoff'])
   yoff = -int(dic['O46P49010'][item]['xoff'])

   print "Cleaning the image",item
```

Compute the standard deviation and the median of the data, ignoring the borders of the image which are known to be noisy

```
   stdv =Statistics.standardDeviation(Numeric.ravel(data[50:950,50:950]))
   median = Statistics.median(Numeric.ravel(data[50:950,50:950]))
```

Replace all elements of data which have values higher than one sigma above the median of the entire image

```
   data = Numeric.where(Numeric.greater(data,median+1*stdv),median,data)
```

Get the shape of the data array and store the x and y sizes in xm and ym respectively

```
   xm,ym = Numeric.shape(data)


   print "Adding ",item," xoff=",xoff," yoff=",yoff
```

Compute the x and y coordinates of the overlap regions

```
   x1,x2 = max(0,xoff),min(xm-1,xm-1+xoff)
   y1,y2 = max(0,yoff),min(ym-1,ym-1+yoff)
   v1,v2 = max(0,-xoff),min(xm-1,xm-1-xoff)
   w1,w2 = max(0,-yoff),min(xm-1,xm-1-yoff)
```

Try to add the computed overlap region of the image that was just read to the current image sum. If this fail (if the image sum does not already exist, then create a new combined image of the appropriate size and store the current image in it

```
   try:
      comb[v1:v2,w1:w2] = comb[v1:v2,w1:w2] + data[x1:x2,y1:y2]
   except:
      comb = Numeric.zeros((xm,ym),Numeric.Float)
      comb[v1:v2,w1:w2] = data[x1:x2,y1:y2]
```

Once all the images have been summed up into the array comb, write this array to a file called test.fits

```
Cleaning the image O46P49010
Adding O46P49010 0 0
Cleaning the image O46P4B010
Adding O46P4B010 0 10
Cleaning the image O46P4C010
Adding O46P4C010 0 0


pfitsio.write_data(comb,"test.fits")
```

Figure 2. Array and FITS file access using Python, (see text for more description).