

Programming with Python 4

Python for non-programmers

Babar Ali

Topics

- Input from text files
- Output to text files and screen.
- Try, except blocks and error handling
- Functions & Libraries

INPUT

Files on disk

- We will focus on ASCII (as opposed to binary) files here.
- Libraries (such as numpy) add significantly easier to use functions to handle files than core python.
 - Try to use them.
- The following lines of code show one way to read the file. We will expand on each command.

Basic code:

```
>>> file = "/Users/babar/file.txt" # Tell python which file.
>>> ou = open(file,"r")           # Have python "open" the file for access.
>>> line=ou.readline()           # Read the first line from our file.
>>> ou.close()                   # Close the file. No more access.
>>> print line                   # Print what we just read.
```

A detailed look

```
>>> file = "/Users/babar/file.txt"
```

- The string variable 'file' is used to define the full directory path and name of the ASCII text file.
- If you don't tell python / spyder the full pathname, it will use the current working directory.
- Since, you may not always remember or wish to use a directory different than the current working directory, it is best
- Always define the full path name so there is no ambiguity.
- If you plan to use the same directory for access to many files and for output, also define the directory in a separate variable.

```
>>> dirname = "/Users/babar/"  
>>> file1 = dirname+"Jphot.txt"  
>>> file2 = dirname+"Kphot.txt"
```

A detailed look (cont.)

```
>>> ou = open(file,"r")
```

- This line calls a core python function 'open' and gives it two arguments:
 - "file" is the variable that is pointing to the file we wish to use.
 - "r" tells python to read from the file.
- You can specify the file name directly instead of first defining it as a string.
 - But, take care to put quotes around the name.
- Here is what else you can tell python to do with files instead of "r"ead:
 - "w" tells python you wish to **w**rite to a file. If the file already exists, it will be deleted first and recreated. So, be careful. Its an easy mistake to wipe out existing file.
 - "a" tells python to **a**ppend to an existing file.
- Once opened, the file will stay opened until closed.
- The variable 'ou' contains the link to the opened file. You need it to access the linked file.

What to do with 'line'?

- Recall: we stored the first line in a variable called 'line'.
- This is a string variable and can be manipulated just like any other string variables.
- For file I/O in particular, the following provide some useful functions:

```
>>> line.strip()      # Remove the trailing '\n' end of line character as well as spaces.
>>> line.split()      # Split the contents of the line using space ' ' as the delimiter.
>>> line.split(',')    # Same as above, except use comma ',' as delimiter.
```

Example:

```
>>> line = "100 34.2345 -5.2344 1"    # This is our line.
>>> line.split()      # produces the following output
['100', '34.2345', '-5.2344', '1']
>>> # A string vector with four values – the original 4 numbers in the line separated by
space ' '
```

For more help

- <http://python4astronomers.github.io/files/asciifiles.html>

A more complete example

- Read and parse a text file containing 5 columns and two header line.

```
>>> # Our input file looks like this:
>>> # ID, ra, dec, flux, comments
>>> # , (deg), (deg), (mJy),
>>> # 1 , 35.2345 , -5.1234 , 100.0 , There is a bright filament nearby.
>>> # 2 , 35.5436 , -5.5567 , 120.0 ,
>>> # 3 , 35.8934 , -5.9832 , 150.0 , Part of a binary pair.
>>> # 4 , 36.52 , -6.1654 , 102.0 , Value is from Scott.
>>> iu=open(myFile,"r")
>>> header1=ou.readline()
>>> header2=ou.readline()
>>> id = []
>>> ra = []
>>> dec = []
>>> flux = []
>>> comment = []
>>> line=ou.readline()
```

Example continued.

```
>>> while line!="":
>>>     words = line.split(',')
>>>     id.append( int(words[0]) )
>>>     ra.append( float(words[1]) )
>>>     dec.append( float(words[2]) )
>>>     flux.append( float(words[3]) )
>>>     comment.append( words[4] )
>>>     line=iu.readline()
>>> iu.close()
```

Using numpy

- numpy provides two functions to read ASCII files. We will use `genfromtxt`
- Functions automatically perform a number of the steps.
- This makes programming simpler to understand and less prone to errors.

Our complete example, now in numpy:

```
>>> import numpy as np
>>> data = np.genfromtxt(myFile, dtype=('i','f','f','f','S20'),\
                        names="id,ra,dec,flux,comment", \
                        skip_header=3, usecols=(0,1,2,3,4),delimiter=",")
```

A closer look at numpy genfromtxt

- *myFile* is the name of the file to read from.
- *dtype* specifies the type of data. One per column, in parenthesis and quotes, as shown.
- *names* tells python what name to use for each column.
 - NOTE: Not related to names in file itself.
- *skip_header* = tells how many header lines to skip.
- *usecols* says read data from these columns.
- *delimiter*= tells what separates data columns

The output

- data contains data on all columns and accessed by the assigned name.
- data["id"] # All elements of id
- data["ra"][0] # The first element of ra
- data["dec"][0:2] # The first & 2nd element
- data["flux"][:] # All elements of flux
- data["comments"]

The output (cont.)

- If names was omitted during the call, the default name of the columns is used.
 - The default name is 'f#', where # stands for 0, 1, 2, 3
- The extracted columns in the output are numpy arrays of the type specified in dtype.
 - For example, data["id"] is a numpy integer array.

What else can you do with genfromtxt?

Option	
comment="#"	Treat the row/line as a comment If the line begins with whatever character is specified in quotes. In this case, '#'. In this case, the line is treated as a comment if it starts with '#'. The rest of the line is ignored.
autostrip=0 or autostrip=1	Whether to strip white spaces from the variables. 1=yes, 0=no. If autostrip=1, leading and trailing white spaces are removed from each variable in the data array.
skip_footer=	The number of lines to skip at the end of the file. This option is useful for skipping the footer of a file.

OUTPUT

Output is just as easy

We have already seen
print

Which prints to the screen.

Now we will use it to print to a file.

Basic code:

```
>>> file = "/Users/babar/output.txt" # Tell python which file.
>>> ou = open(file,"w")             # Have python "open" the file for access.
>>> print >>ou, "Hello File"        # Read the first line from our file.
>>> ou.close()                       # Close the file. No more access.
```

A detailed look

```
>>> file = "/Users/babar/output.txt"
```

- As for input, the string variable 'file' is used to define the full directory path and name of the ASCII text file.
- If you don't tell python / spyder the full pathname, it will use the current working directory.
- Since, you may not always remember or wish to use a directory different than the current working directory, it is best
- Always define the full path name so there is no ambiguity.

A detailed look (cont.)

```
>>> ou = open(file,"w")
```

- This line calls a core python function 'open' and gives it two arguments:
 - "file" is the variable that is pointing to the file we wish to use.
 - "w" tells python to **w**rite to the file.
- You can specify the file name directly instead of first defining it as a string.
 - But, take care to put quotes around the name.
- You can also use:
 - "a" to **a**ppend to an existing file.
- Once opened, the file will stay opened until closed.
- The variable 'ou' contains the link to the opened file. You need it to access the linked file.

Writing to the file

```
>>> print >>ou, "Hello File"
```

- Use the normal python print statement to write to the file.
- The **>>ou** construct added to the print simply tells it to redirect the result of the printing to **ou**.
- All formatting rules for printing apply here as well.

Full example

```
>>> # Using data read earlier in variable data with numpy's genfromtxt
>>> ou=open(myOutputFile,"w")
>>> print >>ou, "This is a header line"
>>> print >>ou, "This is another header line"
>>> nlines = len( data["id"] )
>>> for i in range(nlines):
...     print >>ou, "%4i, %8.3f, %8.3f, %8.3f, %s" % ( data["id"][i], data["ra"][i], \
...           data["dec"][i], data["flux"][i], data["comment"][i] )
>>> ou.close()
```

Using numpy:

```
>>> import numpy as np
>>> np.savetxt(myOutputFile,data,fmt="%4i %8.3f %8.3f %8.3f %s",delimiter=",",\
...           header="This is first line\nThis is 2nd line", comments="#")
```

CATCHING ERRORS

Try and if fail then do something else

```
>>> # python allows a way to catch errors and/or problems:
>>> try:
...     command1 # Run some commands.
...     command2
>>> except Exception, e:
...     print "Your commands did not work. Here is why, maybe"
...     print e.message
```

The **try, except** syntax allows you to test the execution of any command. If any of the commands in the **try:** block generates a python error, then the statements under the **except:** block are executed.

Generally, you put "safe" print statements in the except block to let you figure out what may have happened, but there are always exceptions.

YES: if you have an error in the except block, you may crash the program.

FUNCTIONS

The basic building blocks of code

- Functions:
 - Execute a set of instructions when called.
 - Tidy up code by modularizing it.
 - Should always be used when the same algorithms (with slight parameter differences) are repeated.
- A library is a set of themed functions. E.g. numpy for numerics-related python functions

def:ining Functions.

```
>>> # Creating a function is easy.  
>>> def myFunction(input,optionA=1,optionB=2):  
...     command1 # Run some commands.  
...     command2  
...     return result
```

The **def** statement tell python we are defining a function.

The name of the function follows **def** call.

Mandatory input parameters are specified without the = syntax.

Optional parameters use the = syntax, where the value to the right of = is the default value, if no other value is specified.

Commands are indented.

The **return** statement returns the result. Please return only one variable or value. We will encounter how to return multiple values later.

A simple example.

```
>>> # A function to perform .
>>> def myMathTool(a,b,operation="+"):
...     if operation=="+":
...         result = a+b
...     elif operation=="-":
...         result = a-b
...     elif operation=="*":
...         result = a*b
...     elif operation=="/":
...         if b!=0.:
...             result = a/b
...         else:
...             print "I will not divide by 0. You should know better"
...             print "returned value is 0"
...             result = 0
...     else:
...         print "Operation not understood "+operation
...         print "returned value is 0"
...         result = 0
...     return result
```

Executing functions

- In spyder:
 - Read or edit your function in the editor,
 - Then choose 'Run' to define it.
- Command line python
 - `execfile("/path/fileWithFunction.py")` defines the function.
- simply call the function by its name and the parameters, if needed.

```
>>> # Creating a function is easy.  
>>> myMathTool(10.,20.,operation="/")  
>>> c = myMathTool(10.,201.,operation="*")
```

IPAC ASCII TABLE EXAMPLE

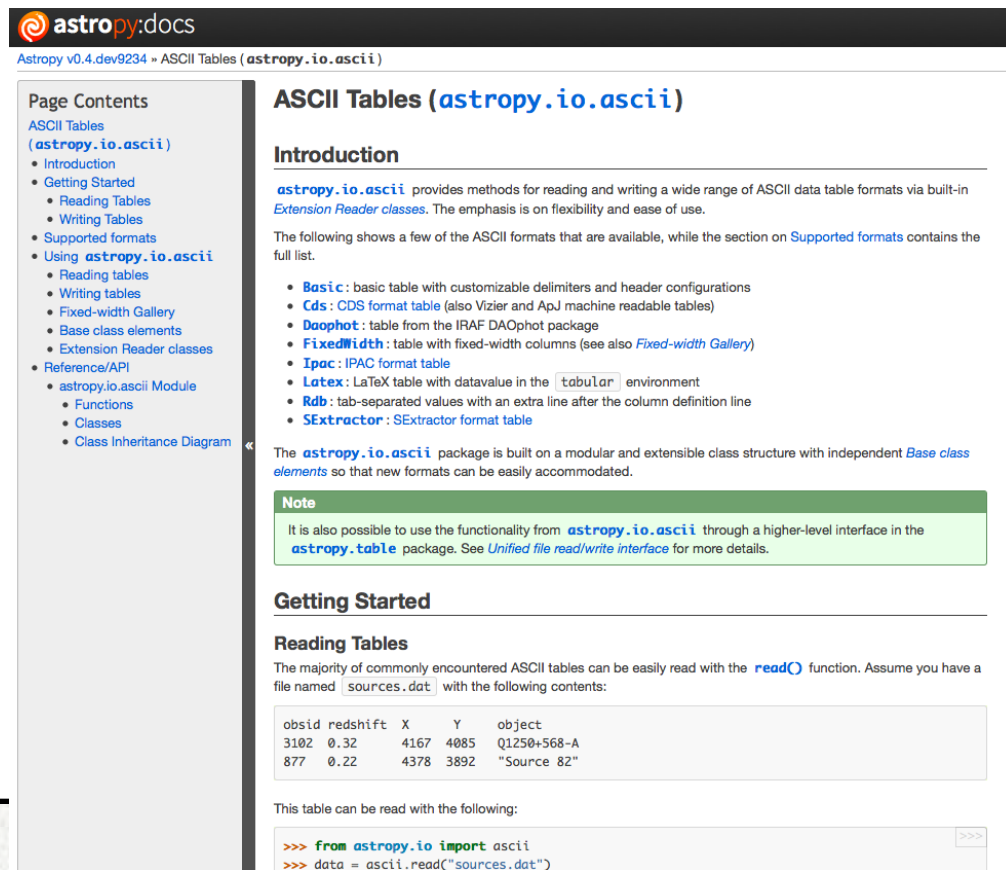
IPAC tables

____ H photometric uncertainty of the associated 2MASS All-Sky PSC source
 k_m_2mass (mag)
 ____ Ks magnitude entry of the associated 2MASS All-Sky PSC source
 k_msig_2mass (mag)
 ____ Ks photometric uncertainty of the associated 2MASS All-Sky PSC source
 angle (deg)
 ____ Position Angle in degree.
 dist (arcsec)
 ____ Distance between the target position and each source in arcsec.

designation	ra	dec	sigra	sigdec	sigradec	w1mpro	w1sigmpro	w1snr	w1rchi2	w2mpro	w2sigmpro	w2snr	w2rchi2	w3mpro	w3sigmpro	w3snr	w3rchi2
char	double	double	double	double	double	double	double	double	double	double	double	double	double	double	double	double	double
	deg	deg	arcsec	arcsec	arcsec	mag	mag			mag	mag			mag	mag		
null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null
J053530.49-051529.3	83.8770684	-5.2581562	1.0523	1.7555	-0.9508	8.364	null	1.8	8.591e+00	7.635	0.503	2.2	1.112e+01	-3.245			
J053539.34-051636.0	83.9139232	-5.2766884	0.0699	0.0666	-0.0292	9.817	0.023	46.8	1.689e+02	9.145	0.024	44.8	1.113e+02	4.037			
J053452.59-051655.2	83.7191573	-5.2820024	0.0670	0.0707	-0.0037	12.444	0.030	35.8	6.366e+01	12.448	0.118	9.2	1.032e+01	5.788			
J053518.81-051729.1	83.8283970	-5.2914413	0.0253	0.0252	-0.0109	6.032	0.042	25.9	5.246e+00	5.323	0.031	34.5	2.014e+00	1.727			
J053525.87-051831.7	83.8578189	-5.3088160	0.1755	0.1810	-0.0437	9.952	0.070	15.5	9.229e+00	9.328	0.080	13.6	6.816e+00	null			
J053537.20-051710.1	83.9050309	-5.2861632	0.0734	0.0744	-0.0485	7.654	0.037	29.1	1.334e+02	7.027	0.025	43.1	2.405e+02	1.549			
J053450.40-051604.0	83.7100413	-5.2677813	0.0459	0.0454	-0.0075	11.027	0.031	35.1	6.971e+01	10.111	0.026	42.3	5.271e+01	5.420			
J053518.52-051338.3	83.8271706	-5.2273263	0.0375	0.0355	-0.0062	7.895	0.023	46.4	4.054e+00	6.788	0.021	50.8	2.838e+00	2.740			
J053537.92-051701.5	83.9080061	-5.2837556	0.1267	0.1552	0.0327	9.560	0.041	26.7	5.575e+01	8.845	0.032	33.7	8.884e+01	3.942			
J053445.97-051710.3	83.6915582	-5.2862103	0.4911	0.5632	-0.0337	13.153	0.218	5.0	7.119e-01	12.495	0.297	3.7	4.186e-01	6.527			
J053540.20-051729.0	83.9175014	-5.2913937	0.0833	0.0837	-0.0182	8.373	0.025	43.4	5.163e+00	8.037	0.023	46.7	9.326e+00	4.908			
J053459.11-051658.4	83.7463179	-5.2829033	0.0778	0.0797	-0.0148	10.622	0.039	28.0	3.472e+01	9.838	0.045	24.0	1.680e+01	4.864			
J053458.21-051628.6	83.7425446	-5.2746263	0.0989	0.0896	0.0174	10.648	0.083	13.1	1.587e+01	9.702	0.077	14.1	1.345e+01	4.174			
J053504.99-051437.0	83.7708001	-5.2436324	0.0657	0.0735	-0.0147	11.421	0.043	25.3	1.712e+01	10.554	0.036	29.9	2.874e+01	5.232			
J053512.38-051448.6	83.8016153	-5.2468494	0.0278	0.0301	-0.0098	12.052	0.268	4.0	1.145e+00	10.465	0.062	17.4	9.438e+00	2.948			
J053458.97-051644.0	83.7457415	-5.2789042	0.0813	0.0916	0.0229	11.069	0.082	13.2	1.039e+01	10.253	0.186	5.8	1.357e+00	4.270			
J053506.03-051511.1	83.7751275	-5.2531046	0.0373	0.0508	-0.0061	13.208	null	1.3	8.463e-01	12.246	0.118	9.2	1.573e+00	7.360			
J053504.87-051356.1	83.7702991	-5.2322746	0.2134	0.1827	-0.0976	11.126	0.484	2.2	2.788e-01	9.990	0.293	3.7	6.851e-01	4.615			
J053514.83-051346.8	83.8118085	-5.2296894	0.0029	0.0030	0.0002	11.183	0.055	19.7	2.824e+01	9.953	0.028	38.4	5.837e+01	-3.061			
J053506.04-051502.0	83.7752013	-5.2505706	0.3602	0.3792	-0.0746	11.891	null	-0.2	2.274e-01	10.724	null	1.2	2.015e-01	6.372			
J053546.41-051655.1	83.9433765	-5.2819869	0.1424	0.1600	-0.0467	13.287	0.064	16.9	1.391e+01	12.055	0.084	12.9	7.548e+00	6.932			
J053512.56-051623.3	83.8023339	-5.2759337	0.0667	0.0681	-0.0171	8.662	0.032	33.6	7.157e-01	7.713	0.033	32.5	6.871e-01	6.419			
J053506.13-051424.7	83.7755789	-5.2402156	0.0480	0.0494	-0.0163	10.832	0.074	14.6	4.059e+00	9.859	0.037	29.3	1.654e+01	5.230			
J053505.20-051450.2	83.7716738	-5.2473053	0.0351	0.0335	0.0010	6.794	0.035	30.6	1.903e+00	6.625	0.019	56.1	1.346e+00	4.727			
J053532.44-051840.1	83.8851734	-5.3111535	0.1242	0.1110	-0.0408	9.914	0.039	27.8	1.154e+02	9.817	0.060	18.1	4.717e+01	3.749			
J053504.58-051427.8	83.7690990	-5.2410817	0.0554	0.0619	-0.0147	11.296	0.036	30.5	3.522e+01	10.571	0.034	31.8	2.437e+01	5.479			
J053511.55-051430.7	83.7981526	-5.2418722	0.1046	0.1116	-0.0355	12.473	0.263	4.1	1.437e+00	11.481	0.158	6.9	3.372e+00	4.096			
J053452.57-051537.5	83.7190737	-5.2604313	0.0614	0.0648	-0.0107	10.448	0.036	30.2	7.789e+00	10.027	0.035	30.8	7.782e+00	5.404			

IPAC tables in python

- Use astropy and ascii table



astropy:docs
Astropy v0.4.dev9234 • ASCII Tables (`astropy.io.ascii`)

Page Contents

- ASCII Tables (`astropy.io.ascii`)
 - Introduction
 - Getting Started
 - Reading Tables
 - Writing Tables
 - Supported formats
 - Using `astropy.io.ascii`
 - Reading tables
 - Writing tables
 - Fixed-width Gallery
 - Base class elements
 - Extension Reader classes
 - Reference/API
 - `astropy.io.ascii` Module
 - Functions
 - Classes
 - Class Inheritance Diagram

ASCII Tables (`astropy.io.ascii`)

Introduction

`astropy.io.ascii` provides methods for reading and writing a wide range of ASCII data table formats via built-in *Extension Reader classes*. The emphasis is on flexibility and ease of use.

The following shows a few of the ASCII formats that are available, while the section on [Supported formats](#) contains the full list.

- **Basic**: basic table with customizable delimiters and header configurations
- **Cds**: CDS format table (also Vizier and ApJ machine readable tables)
- **Daophot**: table from the IRAF DAOPHOT package
- **FixedWidth**: table with fixed-width columns (see also [Fixed-width Gallery](#))
- **Ipac**: IPAC format table
- **Latex**: LaTeX table with data value in the `tabular` environment
- **Rdb**: tab-separated values with an extra line after the column definition line
- **SExtractor**: SEExtractor format table

The `astropy.io.ascii` package is built on a modular and extensible class structure with independent *Base class elements* so that new formats can be easily accommodated.

Note

It is also possible to use the functionality from `astropy.io.ascii` through a higher-level interface in the `astropy.table` package. See [Unified file read/write interface](#) for more details.

Getting Started

Reading Tables

The majority of commonly encountered ASCII tables can be easily read with the `read()` function. Assume you have a file named `sources.dat` with the following contents:

```
obsid redshift X Y object
3102 0.32 4167 4085 Q1250+568-A
877 0.22 4378 3892 "Source 82"
```

This table can be read with the following:

```
>>> from astropy.io import ascii
>>> data = ascii.read("sources.dat")
```

astropy ASCII tables

- <http://www.astropy.org/>
- Follow instructions on the website to install it for your OS and hardware combination

astropy tables

- The `ascii.read()` call will produce a data structure build upon the numpy ndarray.
- It is called astropy Table
- Very useful for astronomy work
- Can write to various formats (including LaTeX for publication tables).

DEMO READ WISE TABLE