Python “Cheat Sheet”

Nicholas P. Ross

July 21, 2016

Abstract

The is my (NPR’s) version of a ∼Python “Cheat Sheet”, (including some parts for folks that want to migrate form IDL).

1 Real basics

1.1 iPython

$> conda update ipython

1.2 Versions

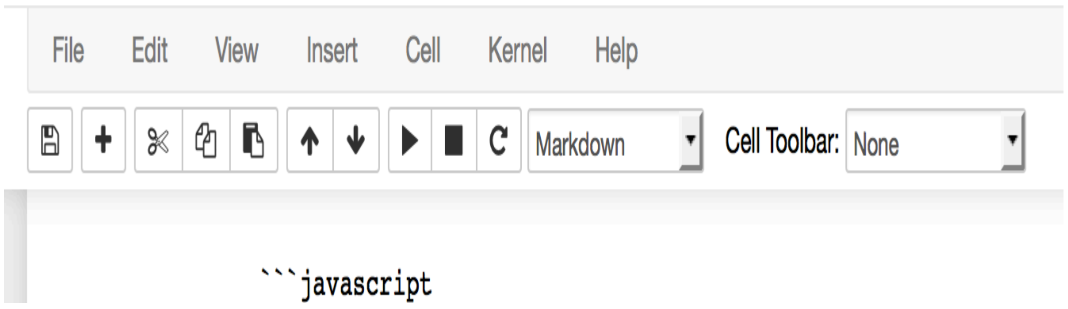
$> python

```
    Python 2.7.6 |AnacondaCE 1.3.1 (x86\_64)| (default, Jan 10
        2014, 11:23:15)
      [GCC 4.0.1 (Apple Inc. build 5493)] on darwin
      Type "help", "copyright", "credits" or "license" for more
          information.
      >>> import numpy
      >>> print numpy.__version__

      1.8.2
      >>> import astropy
      >>> print astropy.__version__

      0.4.1 \\
      \noindent
     >>> import sys\\
     >>> print (sys.version) \\

     2.7.10 (default, Oct 23 2015, 18:05:06) \\
     \lbrack GCC 4.2.1 Compatible Apple LLVM 7.0.0
         (clang-700.0.59.5)]
```



Cell Toolbar: None

~~~
```javascript
~~~

Figure 1: Clicking on the Cell Toolbar "Code", "Markdown" etc. will power what happens in the Clells!!!

1.3 iPython from Fernando Perez

Try: tmpnb.org **VERY USEFUL**
http://www.pythonforbeginners.com/basics/ipython-a-short-introduction

1.4 Notebook

Click on the NBviewer...
Then you can see the e.g. html of the notebook.
But to change/execute it, then all you have to do is click the download button...
Then put it on gitHub/Dropbox etc...
(I need to learn about "Tmox" and "SCreen" Terminal emulators...)

Run a code cell using Shift-Enter or
Alt-Enter runs the current cell and inserts a new one below. Ctrl-Enter run the current cell and enters command mode.
Google: "ipython beyond plain python"

http://nbviewer.ipython.org/github/fperez/cit2013/blob/master/06-IPython%20-%20beyond%20plain%20Python.ipynb
iPython NB power = power of python + power of the command line with "!" + "%" and "%%" "magics"...
http://nbviewer.ipython.org/github/ipython/ipython/blob/1.x/examples/notebooks/Part%204%
%20Markdown%20Cells.ipynb
https://github.com/profjsb/python-bootcamp

2 Britton's Classes :-)

2.1 "If lost in the desert..."

>>> dir(thing)
>>> dir(thing)

2.2 Lists

```
$>>>$ super_list = [0, [3,4,5], "Hello World!", range(5)]
$>>>$ print super\_list

[0, [3, 4, 5], 'Hello World!', [0, 1, 2, 3, 4]]
$>>>$ print super\_list[1]
%+ - = ! / ( ) [ ] < > | ' :
\lbrack3, 4, 5]
$>>>$ print super\_list[-1]
\lbrack0, 1, 2, 3, 4]
$>>>$ print super\_list[1[0]]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not subscriptable
$>>>$ print super\_list[1][0]
  3

$>>>$ c = range(10) \\
$>>>$ print c \\
\lbrack0, 1, 2, 3, 4, 5, 6, 7, 8, 9]\\
$>>>$ c.append(range(3)) \\
$>>>$ print c \\
\lbrack0, 1, 2, 3, 4, 5, 6, 7, 8, 9, [0, 1, 2]]\\
$>>>$ c.extend(range(3)) \\
$>>>$ print c \\
\lbrack0, 1, 2, 3, 4, 5, 6, 7, 8, 9, [0, 1, 2], 0, 1, 2]\\
$>>>$ del c[4]\\
$>>>$ print c\\
%[0, 1, 2, 3, 5, 6, 7, 8, 9, [0, 1, 2], 0, 1, 2]\\


%\vspace{12pt}
$>>>$ z = [42]*5\\
$>>>$ [42, 42, 42, 42, 42]\\

%\vspace{12pt}
$>>>$ print super\_list
[0, [3, 4, 5], 'Hello World!', [0, 1, 2, 3, 4]]
$>>>$ print len(super\_list)
```

IDL code	Python code
data=READFITS('file',header)	data=pyfits.open('file')
tdata = mrdfits('SpIESch1ch2.fits',0, hdr)	tdata = data[0].data
tbdata = mrdfits('SpIESch1ch2.fits',1, hdr)	tdata = data[1].data
help, tbdata, /str	info(tbdata)
print, size(tbdata)	shape(tbdata)
print, tbdata[0].flux_aper_1	print tbdata.FLUX_APER_1[0]
help, tbdata.flux_aper_1	tbdata.FLUX_APER_1?
fluxaper = tbdata.flux_aper_1[2]	fluxaper = ???
(using fitsio)	d = fitsio.read('SpIESch1ch2.fits',1)

Table 1: IDL to Python

```
4
$>>>$ print len(super\_list[-1])
5
```

3 IDL to Python

Key links:
IDL to Numeric/numarray Mapping
NumPy for IDL users

4 INPUT

5 OUTPUT

For the "write" statement, I think you have to put everything into a string format, otherwise it just barfs...

```
  outfile = open('WISE\_spectra\_triples\_4wget\_temp.dat', 'w') \\
for i in range(len(ra)):

    print i, ra[i]

    plate\_out = str(plate[i])

    mjd\_out = str(mjd[i])
    fiberid\_out = str(fiberid[i])

    outfile.write(plate\_out+"/spec-"+plate\_out+"-"+mjd\_out+"-"+fiberid\_out.zfill(4)+".fi
        \textbackslash n")
```

6 IDL Where...

7 v2 vs. v3

https://docs.python.org/3/howto/pyporting.html

7.1 print

print a vs. print (a) Thus, just use () all the time!!

7.2 Division

/ = truncating (integer floor) division in P2.x when using ints; float division in P3.x // = truncating div in P2.x, P3.x

8 Linear Algebra

http://docs.scipy.org/doc/scipy/reference/tutorial/linalg.html

```
import numpy as np
from scipy import linalg
A = np.array([[1,2],[3,4]])
linalg.inv(A)
A.dot(linalg.inv(A)) #double check
```

9 Gotchas

"follow up: PYTHONPATH is a hazardous environment variable, and should never include one Python's site-packages"

See 429 in history_20150113.txt and onwards... :-)

10 A few General Notes

10.1 What's the difference between raw_input() and input()?

The difference is that `raw_input()` does not exist in Python 3.x, while `input()` does. Actually, the old `raw_input()` has been renamed to `input()`, and the old `input()` is gone (but can easily be simulated by using `eval(input())`).

Reference: http://stackoverflow.com/questions/4915361/whats-the-difference-between-raw-input-and-input-in-python3-x.

11 A few general notes and commands

11.1 join()

Description: The method `join()` returns a string in which the string elements of sequence have been joined by str separator.
Syntax: Following is the syntax for `join()` method: `str.join(sequence)`.
Parameters: sequence – This is a sequence of the elements to be joined.
Example:

```
s = "-";
seq = ("a", "b", "c"); # This is sequence of strings.
print s.join( seq )
a-b-c
```

11.2 eval()

The eval function lets a python program run python code within itself.

```
x = 1
eval('x + 1')
2
eval('x')
1
```

```
l
[5, 5]
cmd
'insert(0,5)'
eval("l."+cmd)
print l
[5, 5, 5]
```

11.3 map()

`map(function, iterable, ...)`
Return an iterator that applies function to every item of iterable, yielding the results.

```
>>> def cube(x): return x*x*x
...
>>> map(cube,range(1,11))
```

```
<map object at 0x101c182e8>
>>> list(map(cube,range(1,11)))
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
>>>
```

The `list()` is needed in Python 3.x.

```
def f(x): return x % 2 != 0 and x % 3 != 0
...
>>> filter(f,range(2,25))
<filter object at 0x101c18390>
>>> list(filter(f,range(2,25)))
[5, 7, 11, 13, 17, 19, 23]
>>>
```

11.4 strip()

```
>>> str = "0000000this is string example....wow!!!0000000";
>>> print (str.strip( '0' ))
this is string example....wow!!!
```

General Wee Tips

Need points that are evenly spaced on a log scale? Use `np.logscale(start, stop, base)`
By convention, matplotlib is imported as mpl. Also by convention, matplotlib.pyplot is imported as plt.

Useful Resources

Borrows, begs and steals from:
http://www.astro.umd.edu/*sim*mbk/idl-numpy.html
http://www.johnny-lin.com/cdat_tips/tips_array/idl2num.html
http://www.astrobetter.com/idl-vs-python/
http://www.astrobetter.com/wiki/tiki-index.php?page=Python+Switchers+Guide
http://mathesaurus.sourceforge.net/idl-numpy.html http://www.scicoder.org/mapping-idl-to-python/
Also, http://www.cv.nrao.edu/ aleroy/pytut/topic2/intro_fits_files.py