2015-03-12_intro_to_python

March 17, 2015

```
In [1]: %matplotlib inline
        from matplotlib.pyplot import plot
        import numpy
```

1

An Introduction to:
Ben Rusholme IPAC 2015-03-12 https://caltech.box.com/intro-to-python

1.1 Table of Contents

2 Introduction

2.1 Purpose

- 90 min introduction to Python, O(50) people
- You will ***not*** learn Python in this session
- To learn work through tutorials such as:
  - Software Carpentry
  - Think Python
  - Python for Scientific Computing
- Hopefully this session will smooth your journey
- I will cover a lot of material at high speed
- I am assuming you currently use IDL

2.2 Questions to Answer?

- What software is available?
- How do I get everything I need everywhere I work?
- What are the basics?
- How do I read in my data?
- How do I make plots?
- What are the differences from IDL?
- What are the potential problems?

2.3 About this Presentation

- This presentation is written in the IPython notebook
- This allows me to interleave live code and text

```
In [2]: print 'Hello Wonderful World!'
```

```
Hello Wonderful World!
```

- I have never given a notebook presentation before
- I have tried to optimize for readibility in this room
- This notebook will be available later on Box
- NB: This is not how I regularly work in Python!

2.4 About Me

- Physics background, no formal computer science training
- Past projects have used FORTRAN, C, IDL
- Joined IPAC in 2006 to work on Planck. Picked up Python as gone along
- Day-to-day I do a lot of batch parallel data processing

2.5 Why Python?

- It's ***free***! (install everywhere, run processing in parallel)
- Everyone else is using it (libraries, local help...)
- Reference CPython integrates well with C code
- Highly extensible, object orientated
- Designed for clean, concise, readable syntax

2.6 Disadvantages

- Managing modules
- Modules can make (unexpectedly) significant changes
- Performance (vs compiled languages)
- Only single thread can run at once (GIL)

3 Installation

3.1 TL;DR

Install Anaconda Python v3

3.2 The Python Ecosystem

3.3 Distributions

- Anaconda - Continuum Analytics (conda, mini-conda...)
- Canopy - Enthought
- Ureka - STSci & Gemini

Can also install via OS tools: * OSX - Macports, Homebrew * Linux - Yum, apt-get ...

3.4 Interpreter: v2 vs v3?

Python v3 (released December, 2008) is not backward compatible with v2. Two major changes: * `print` statement became function `print()` * All `ints` are automatically `long` * float division (`3/2=1` became `3/2=1.5`, use `3//2=1` for integer/floor)

Python v2 no longer getting new features, only bug/security updates until 2020.

It has taken (lots of) time for the external packages to catch up. IMHO we are finally at the tipping point for v2 -> v3 migration. * If starting fresh, and external packages compliant, I recommend v3 * Else stick with v2

This notebook works in both v2 & v3

NB: If you need to convert look into the standard library modules: `__future__` and `2to3`

3.5 Interpreter: Language Implementations

The reference interpreter is CPython, coded in C. There are other implementations:

- Stackless - concurrency using tasklets and channels
- PyPy - Python in Python
- Jython - Python in Java
- IronPython - Python in C# (.NET)

You should know that these exist, but shouldn't need them

NB: CPython != Cython

3.6 Standard Library

Contains O(100) modules. For full list see: https://docs.python.org/2/library/

Most useful:

Category	Name
Datatypes	collections, copy, datetime, queue
Numeric	math, random
Files	csv, (c)pickle
OS	glob, os, subprocess, time
Language	`__future__`, 2to3, itertools, multiprocessing, pdb, (c)profile

3.7 PyPI - The Python Package Index

```
In [2]: from IPython.display import HTML
        HTML('<iframe src=https://pypi.python.org/pypi width=600 height=350></iframe>')
```

```
Out[2]: <IPython.core.display.HTML at 0x1036de5d0>
```

3.8 External: The Big 3

NB: All part of SciPy project

Name	Logo	Description
IPython		Enhanced interactive console
Numpy		Base N-dimensional array package

Name	Logo	Description
Matplotlib		Comprehensive 2D plotting

3.9 External: Other Important

Name	Description
astropy	Astronomy
cython	C extensions
healpy	HEALPix
h5py	HDF5
mpi4py	Message-passing parallelism
nose	Unit testing
pandas	Data structures and analysis (competitor to R)
pep8	Style checking
pip	Package installation
pyfits	FITs files
pyraf	IRAF
scikit-learn	Machine learning
scikit-image	Image processing
scipy	Scientific codes
sympy	Symbolic mathematics
virtualenv	Environment virtualization
yt	Visualization

3.10 Installing Modules

Package manager: * Conda: `conda install <package>` * Macports: `port install <package>`
pip: * `pip install <package>`
By hand: * `python setup.py install`
Installed to: `<sys.prefix>/lib/python<sys.version_info>/site-packages`

3.11 Repeatable Installations

- `pip freeze > requirements.txt`
- `pip install -r requirements.txt`

pip freeze — head altgraph==0.12 APLpy==0.9.14 astropy==0.4.4 ATpy==0.9.7 backports==1.0 backports.ssl-match-hostname==3.4.0.2 Beaker==1.6.4 certifi==14.5.14 Cython==0.21.2 docutils==0.12

3.12 Virtual Environments

If virtualenv module installed:

cd my$_{p}roject_{f}oldervirtualenvmy_{p}rojectsourcemy_{p}roject/bin/activatepipinstall - rrequirements.txtdeactivaterm - rfmy_{p}roject$

Can be run from source if neither installed nor install permissions:

curl -O https://pypi.python.org/packages/source/v/virtualenv/virtualenv.X.X.tar.gz tar -xzf virtualenv-X.X.tar.gz cd virtualenv-X.X python virtualenv.py my$_{p}rojectsourcemy_{p}roject/bin/activatedeactivaterm - rfmy_{p}roject$

Conda has similiar functionality:

conda create -n ENV anaconda source activate ENV source deactivate conda remove –all -n ENV

I use conda's virtual environments to switch between Python v2 & v3 on my laptop, see (http://continuum.io/blog/anaconda-python-3)

4 Running Python

4.1 Direct Interpreter

$Interactiveinterpreter$ python Python 2.7.9 (default, Dec 11 2014, 02:36:08) [GCC 4.2.1 Compatible Apple LLVM 5.1 (clang-503.0.40)] on darwin Type "help", "copyright", "credits" or "license" for more information. ¿¿¿ print 'hello' hello ¿¿¿ exit Use exit() or Ctrl-D (i.e. EOF) to exit ¿¿¿ exit()

Running a script $catscript.pyprint'hello'$ python script.py hello

$Runcommand(s)$ VAR=GOODBYE $python - c"print`hello';print'$VAR"'" hello GOODBYE

$Shellredirection$ cat script.py — python hello

4.2 IPython

- Command shell wrapper around the interpreter
- Enhanced REPL
- Can embed in scripts (== IDL STOP)
- Browser-based notebook with inline plots
- Parallel execution of commands and scripts

Notes: * –pylab option deprecated * Non-Python parts recently spun off into Jupyter

4.2.1 REPL

- Command history/searching
- Tab completion
- ?/??/help() for info/help
- ! system escape
- Input/output caching (_#)
- % "magic" methods:

Command	Description
%run (-t, -d, -p) <`script.py`>	Run script
%timeit	Time command
%cpaste	Indented cut and paste
%who(s)	Examine namespace

```
In [3]: %lsmagic
```

```
Out[3]: Available line magics:
```

```
%alias  %alias_magic  %autocall  %automagic  %autosave  %bookmark  %cat  %cd  %clear  %colors

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%capture  %%debug  %%file  %%html  %%javascript  %%latex  %%perl

Automagic is ON, % prefix IS NOT needed for line magics.
```

4.2.2 Notebook

"Web-based interactive computational environment where you can combine code execution, text, mathematics, plots and rich media into a single document"

Excels for prototyping, data examination, remote work & presentations

Cells can be: * Code * Markdown * Header * Raw

Press 'h' for list of keyboard commands

```
In [5]: from IPython.display import Latex
        Latex(r"""\begin{eqnarray}
        \nabla \times \vec{\mathbf{B}} -\, \frac1c\, \frac{\partial\vec{\mathbf{E}}}{\partial t} & = \f
        \nabla \cdot \vec{\mathbf{E}} & = 4 \pi \rho \\
        \nabla \times \vec{\mathbf{E}}\, +\, \frac1c\, \frac{\partial\vec{\mathbf{B}}}{\partial t} & =
        \nabla \cdot \vec{\mathbf{B}} & = 0
        \end{eqnarray}""")
```

```
Out[5]:
```

$$\nabla \times \vec{\mathbf{B}} - \frac{1}{c}\frac{\partial\vec{\mathbf{E}}}{\partial t} = \tfrac{4\pi}{c}\vec{\mathbf{j}} \tag{1}$$

$$\nabla \cdot \vec{\mathbf{E}} = 4\pi\rho \tag{2}$$

$$\nabla \times \vec{\mathbf{E}} + \frac{1}{c}\frac{\partial\vec{\mathbf{B}}}{\partial t} = \vec{\mathbf{0}} \tag{3}$$

$$\nabla \cdot \vec{\mathbf{B}} = 0 \tag{4}$$

5 Basic Syntax

5.1 Operators

5.1.1 Arithmetic & Assignment

Operator		Assignment	
		=	
+		+=	increment
-		-=	decrement
`*`		`*=`	
/		/=	
//	floor	//=	
%	modulo	%=	
`**`	power (**NOT** ^)	`**=`	
		a = b = 1	multiple

Operator	Assignment
	a, b = 1, 2

```
In [6]: a = 3
        a += 27
        print(a)
```

```
30
```

```
In [7]: q, m = divmod(10,3)
        print(q, m)
```

```
(3, 1)
```

5.1.2 Logical

and or not

5.1.3 Membership

in not in

```
In [8]: 'i' in 'team'
```

```
Out[8]: False
```

5.1.4 Identity

is is not

```
In [9]: 2 is 2.0
```

```
Out[9]: False
```

5.1.5 Comparison

== != ¡ ¿ ¡= ¿=

5.1.6 Bitwise/Binary

Symbol	Operation
&	AND
|	OR
ˆ	XOR
˜	ones complement
<<	left shift
>>	right shift

```
In [10]: 9 & 8
```

```
Out[10]: 8
```

```
In [11]: print(bin(9))
         print(bin(8))
```

```
0b1001
0b1000
```

5.2 Datatypes

5.2.1 Constants

True False None

5.2.2 Numeric

Type	Assignment	Note
`int`	`a = 1`	
`long`	`a = 1L`	Python2 only - all Python3 ints are long
`float`	`a = 1.`	
`complex`	`a = 1j`	j can be either lower or upper case

```
In [12]: a = 1j
         print(a, type(a))
```

```
(1j, <type ’complex’>)
```

5.2.3 Sequences

String Denoted by either single or double quotes, just match consistently:

```
In [13]: print(’these’, "are", ’"some"’, "’strings’")
```

```
(’these’, ’are’, ’"some"’, "’strings’")
```

List General container, denoted by square brackets `[...]`, can contain ***anything***

```
In [14]: a = [1, 3.0, 4+5j, ’six’ ]
         print(a)
         a[0]=’blah’
         print(a)
```

```
[1, 3.0, (4+5j), ’six’]
[’blah’, 3.0, (4+5j), ’six’]
```

Tuple An ***immutable*** list, that is state cannot be changed after creation. Denoted by round brackets `(...)`

```
In [15]: source = (’M56’, 289.1479411, 30.1845005)
         some_string = ’abcde’
         source[0]=’m31’
```

```
    ---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)

    <ipython-input-15-16d8d816dcac> in <module>()
      1 source = ('M56', 289.1479411, 30.1845005)
      2 some_string = 'abcde'
----> 3 source[0]='m31'


    TypeError: 'tuple' object does not support item assignment
```

5.2.4 Other

Sets There are 2 set types: `set` and `frozenset` (mutable/immutable). Sets are created with curly brackets {...} or the function `set()`:

```
In [16]: a = {1,2,3}
         type(a)

Out[16]: set
```

The classic set methods (intersection/union) are available:
a.add a.intersection a.remove a.clear a.intersection$_{u}pdatea.symmetric_{d}ifferencea.copya.isdisjointa.symmetric_{d}ifferen$
The common use of sets is to remove duplicates:

```
In [17]: a = [1,1,2,9,2,3,4,5,6,7,6,5,3,2,1]
         set(a)

Out[17]: {1, 2, 3, 4, 5, 6, 7, 9}
```

Note there is also a numpy routine to do the same:

```
In [18]: numpy.unique(a)

Out[18]: array([1, 2, 3, 4, 5, 6, 7, 9])
```

Dictionaries A dictionary contains values accessed by keys:

```
In [19]: d = {'key':'value'}
```

They are used for mapping one quantity to something else:

```
In [20]: translate = {'red':'rouge',
                      'green':'vert',
                      'blue':'bleu',
                      'black':'noir',
                      'white':'blanc'}
         print(translate['red'])
         print(translate.keys())

rouge
['blue', 'black', 'white', 'green', 'red']
```

Note: * order is not conserved! Use collections.ordereddict if important * Mapping is one way only

```
In [21]: print(translate['noir'])
```

```
        ---------------------------------------------------------------------------
    KeyError                                  Traceback (most recent call last)

        <ipython-input-21-f8850c1b94bd> in <module>()
    ----> 1 print(translate['noir'])


        KeyError: 'noir'
```

Example of dict in action:

```
In [22]: hfi = { '100': [ '100-1a','100-1b','100-2a','100-2b',
                          '100-3a','100-3b','100-4a','100-4b'  ],
                 '143': [ '143-1a','143-1b','143-2a','143-2b',
                          '143-3a','143-3b','143-4a','143-4b',
                          '143-5' ,'143-6' ,'143-7'            ],
                 '217': [ '217-1' ,'217-2' ,'217-3' ,'217-4' ,
                          '217-5a','217-5b','217-6a','217-6b',
                          '217-7a','217-7b','217-8a','217-8b'  ],
                 '353': [ '353-1' ,'353-2' ,'353-3a','353-3b',
                          '353-4a','353-4b','353-5a','353-5b',
                          '353-6a','353-6b','353-7' ,'353-8'   ],
                 '545': [ '545-1' ,'545-2' ,         '545-4'   ],
                 '857': [ '857-1' ,'857-2' ,'857-3' ,'857-4'   ] }

         print( sorted( hfi.keys() ))
         print( hfi['857'] )
```

```
['100', '143', '217', '353', '545', '857']
['857-1', '857-2', '857-3', '857-4']
```

5.3 Indexing

- First element = 0
- Initialize index list with range(start, stop, step)
- Index with [start: stop: step]
- Negative indices count back from end

```
In [24]: a = range(10)
         print(a)  # Python3 print(*a) to expand range
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [25]: a[0]
```

```
Out[25]: 0
```

```
In [26]: print( a[3:6] )
         print( a[3:] )
```

```
[3, 4, 5]
[3, 4, 5, 6, 7, 8, 9]
```

```
In [45]: print( a[-1] )
         print( a[-2] )
```

```
9
8
```

```
In [27]: print( a[::2]  )
         print( a[1::2] )
         print( a[::-1] )
```

```
[0, 2, 4, 6, 8]
[1, 3, 5, 7, 9]
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

5.4 Numpy (Arrays)

Python has array module in standard library but just use numpy for numerical work. There are multiple array creation routines. The most important are:

Routine	Description
`numpy.array()`	Convert existing array
`numpy.arange()`	Evenly-spaced values
`numpy.zeros()`	Array filled with zeros
`numpy.full()`	Array filled with specified value

Numpy can parse statements inside indexing brackets (fancy indexing):

```
In [28]: import numpy
         a = numpy.arange(10)
         print( a )
         print( a[a > 6] )
```

```
[0 1 2 3 4 5 6 7 8 9]
[7 8 9]
```

Use record arrays for structured data; can be indexed by field name:

```
In [29]: img = numpy.zeros((2,2), {'names': ('r','g','b'), 'formats': (numpy.float32, numpy.float32, num
         print( img['r'] )
         print() # Empty line
         print( img.__repr__() )
```

```
[[ 0.  0.]
 [ 0.  0.]]
()
array([[(0.0, 0.0, 0.0), (0.0, 0.0, 0.0)],
       [(0.0, 0.0, 0.0), (0.0, 0.0, 0.0)]],
      dtype=[('r', '<f4'), ('g', '<f4'), ('b', '<f4')])
```

5.5 Functions vs Methods

Function is independent of explicitly passed data:

IDL¿ a='abcde' IDL¿ strupcase(a) ABCDE IDL¿ strlen(a) 5

```
In [30]: a='abcde'
         len(a) # Function

Out[30]: 5
```

Method is associated with implicitly passed object:

```
In [31]: a.upper() # Method

Out[31]: 'ABCDE'
```

Tab-completion will show you all methods associated with an object!

```
In [63]: a.


          File "<ipython-input-63-a0d310e2b5e6>", line 1
        a.
          ^
    SyntaxError: invalid syntax
```

a.capitalize a.format a.isupper a.rindex a.strip a.center a.index a.join a.rjust a.swapcase a.count a.isalnum a.ljust a.rpartition a.title a.decode a.isalpha a.lower a.rsplit a.translate a.encode a.isdigit a.lstrip a.rstrip a.upper a.endswith a.islower a.partition a.split a.zfill a.expandtabs a.isspace a.replace a.splitlines a.find a.istitle a.rfind a.startswith

Methods change with object type!

a = [1,2,3]

a. a.append a.extend a.insert a.remove a.sort a.count a.index a.pop a.reverse

Everything in Python is an object!

In [1]: import math

In [2]: math. math.acos math.degrees math.fsum math.pi math.acosh math.e math.gamma math.pow math.asin math.erf math.hypot math.radians math.asinh math.erfc math.isinf math.sin math.atan math.exp math.isnan math.sinh math.atan2 math.expm1 math.ldexp math.sqrt math.atanh math.fabs math.lgamma math.tan math.ceil math.factorial math.log math.tanh math.copysign math.floor math.log10 math.trunc math.cos math.fmod math.log1p math.cosh math.frexp math.modf

Note methods can: * accept arguments * stack

```
In [32]: 'eleven'.count('e')

Out[32]: 3

In [33]: filename = 'HFI_SkyMap_353-1_2048_R2.00_year-1.fits'
         filename.strip('.fits').split('_')

Out[33]: ['HFI', 'SkyMap', '353-1', '2048', 'R2.00', 'year-1']

In [34]: instrument, product, detector, nside, version, period = filename.strip('.fits').split('_')
         print( detector, period )

('353-1', 'year-1')
```

6 Namespaces

6.1 Import Statements

Most programs start with import statements of the form:

import module import module as mod from module import * from module import function1, function2

These set different mapping of names to objects:

```
In [35]: %reset -f
         %whos
```

```
Interactive namespace is empty.
```

```
In [36]: import math
         %whos
         print()
         print( math.pi )
```

```
Variable   Type      Data/Info
------------------------------
math       module    <module 'math' from '/Use<...>2.7/lib-dynload/math.so'>
()
3.14159265359
```

```
In [37]: %reset -f
         import math as m
         %whos
         print()
         print( m.pi )
```

```
Variable   Type      Data/Info
------------------------------
m          module    <module 'math' from '/Use<...>2.7/lib-dynload/math.so'>
()
3.14159265359
```

```
In [38]: %reset -f
         from math import *
         %whos
         print()
         print( pi )
```

```
Variable    Type                          Data/Info
---------------------------------------------------
acos        builtin_function_or_method    <built-in function acos>
acosh       builtin_function_or_method    <built-in function acosh>
asin        builtin_function_or_method    <built-in function asin>
asinh       builtin_function_or_method    <built-in function asinh>
atan        builtin_function_or_method    <built-in function atan>
atan2       builtin_function_or_method    <built-in function atan2>
atanh       builtin_function_or_method    <built-in function atanh>
ceil        builtin_function_or_method    <built-in function ceil>
copysign    builtin_function_or_method    <built-in function copysign>
cos         builtin_function_or_method    <built-in function cos>
cosh        builtin_function_or_method    <built-in function cosh>
degrees     builtin_function_or_method    <built-in function degrees>
```

```
e           float                          2.71828182846
erf         builtin_function_or_method    <built-in function erf>
erfc        builtin_function_or_method    <built-in function erfc>
exp         builtin_function_or_method    <built-in function exp>
expm1       builtin_function_or_method    <built-in function expm1>
fabs        builtin_function_or_method    <built-in function fabs>
factorial   builtin_function_or_method    <built-in function factorial>
floor       builtin_function_or_method    <built-in function floor>
fmod        builtin_function_or_method    <built-in function fmod>
frexp       builtin_function_or_method    <built-in function frexp>
fsum        builtin_function_or_method    <built-in function fsum>
gamma       builtin_function_or_method    <built-in function gamma>
hypot       builtin_function_or_method    <built-in function hypot>
isinf       builtin_function_or_method    <built-in function isinf>
isnan       builtin_function_or_method    <built-in function isnan>
ldexp       builtin_function_or_method    <built-in function ldexp>
lgamma      builtin_function_or_method    <built-in function lgamma>
log         builtin_function_or_method    <built-in function log>
log10       builtin_function_or_method    <built-in function log10>
log1p       builtin_function_or_method    <built-in function log1p>
modf        builtin_function_or_method    <built-in function modf>
pi          float                          3.14159265359
pow         builtin_function_or_method    <built-in function pow>
radians     builtin_function_or_method    <built-in function radians>
sin         builtin_function_or_method    <built-in function sin>
sinh        builtin_function_or_method    <built-in function sinh>
sqrt        builtin_function_or_method    <built-in function sqrt>
tan         builtin_function_or_method    <built-in function tan>
tanh        builtin_function_or_method    <built-in function tanh>
trunc       builtin_function_or_method    <built-in function trunc>
()
3.14159265359
```

```
In [39]: %reset -f
         from math import pi, e
         %whos
         print()
         print( pi )
```

```
Variable   Type     Data/Info
-----------------------------
e          float    2.71828182846
pi         float    3.14159265359
()
3.14159265359
```

Renaming useful for mixing/matching modules but beware of overwriting!
import pyfits
replace with:
import astropy.io.fits as pyfits

6.2 Reusing Your Own Code

Put common code into a file and import into your namespace (needs to be in same directory or locatable via environment variable PYTHONPATH):

```
In [40]: %%bash
         cat my_lib.py

hfi = { ’100’: [ ’100-1a’,’100-1b’,’100-2a’,’100-2b’,
                 ’100-3a’,’100-3b’,’100-4a’,’100-4b’  ],
        ’143’: [ ’143-1a’,’143-1b’,’143-2a’,’143-2b’,
                 ’143-3a’,’143-3b’,’143-4a’,’143-4b’,
                 ’143-5’ ,’143-6’ ,’143-7’            ],
        ’217’: [ ’217-1’ ,’217-2’ ,’217-3’ ,’217-4’ ,
                 ’217-5a’,’217-5b’,’217-6a’,’217-6b’,
                 ’217-7a’,’217-7b’,’217-8a’,’217-8b’  ],
        ’353’: [ ’353-1’ ,’353-2’ ,’353-3a’,’353-3b’,
                 ’353-4a’,’353-4b’,’353-5a’,’353-5b’,
                 ’353-6a’,’353-6b’,’353-7’ ,’353-8’   ],
        ’545’: [ ’545-1’ ,’545-2’ ,         ’545-4’   ],
        ’857’: [ ’857-1’ ,’857-2’ ,’857-3’ ,’857-4’   ] }
```

```
In [41]: %reset -f
         from my_lib import hfi
         %whos
         hfi

Variable   Type    Data/Info
----------------------------
hfi        dict    n=6

Out[41]: {’100’: [’100-1a’,
           ’100-1b’,
           ’100-2a’,
           ’100-2b’,
           ’100-3a’,
           ’100-3b’,
           ’100-4a’,
           ’100-4b’],
          ’143’: [’143-1a’,
           ’143-1b’,
           ’143-2a’,
           ’143-2b’,
           ’143-3a’,
           ’143-3b’,
           ’143-4a’,
           ’143-4b’,
           ’143-5’,
           ’143-6’,
           ’143-7’],
          ’217’: [’217-1’,
           ’217-2’,
           ’217-3’,
           ’217-4’,
           ’217-5a’,
           ’217-5b’,
           ’217-6a’,
           ’217-6b’,
           ’217-7a’,
           ’217-7b’,
           '217-8a',
           '217-8b'],
          '353': ['353-1',
           '353-2',
           '353-3a',
           '353-3b',
           '353-4a',
           '353-4b',
           '353-5a',
           '353-5b',
           '353-6a',
           '353-6b',
           '353-7',
           '353-8'],
          '545': ['545-1', '545-2', '545-4'],
          '857': ['857-1', '857-2', '857-3', '857-4']}
```

7 Compound Statements

7.1 Whitespace

- Python does not use brackets or begin/end/do/done to enclose compound statements.
- Blocks are marked by colon ':' then relative indentation.
- Either tabs or spaces can be used (but not mixed). PEP8 recommends 4 x space:

statement: blah blah statement: blah blah blah blah statement: blah One liner blah blah blah blah

Get comfortable with selecting & manipulating text blocks in your favorite editor (CTRL-V in VIM)! Use `%cpaste` magic to copy/paste into IPython.

7.2 Functions

```
In [42]: def do_nothing(input):
             return input

         do_nothing('a')

Out[42]: 'a'
```

7.3 Control Flow

- while
- for
- if

but no case/switch. Can be modified by:

- break
- continue
- pass

7.3.1 While. . .

```
In [43]: count = 0
         while count < 9:
            print( 'The count is:', count )
            count += 1

         print( "Good bye!" )

('The count is:', 0)
('The count is:', 1)
('The count is:', 2)
('The count is:', 3)
('The count is:', 4)
('The count is:', 5)
('The count is:', 6)
('The count is:', 7)
('The count is:', 8)
Good bye!
```

7.3.2 For...

Collection-controlled, similar to IDL `FOREACH`. To count give range.

```
In [44]: start = 0
         stop = 10
         step = 1

         for index in range(start, stop, step):
             print( index )

0
1
2
3
4
5
6
7
8
9
```

```
In [45]: for filter in ['u', 'g', 'r', 'i', 'z']:
             print( filter )

u
g
r
i
z
```

```
In [46]: filters = ['u', 'g', 'r', 'i', 'z']
         for filter in filters:
             print( filter )

u
g
r
i
z
```

```
In [47]: for index, filter in enumerate(filters):
             print( index, filter )
```

```
(0, 'u')
(1, 'g')
(2, 'r')
(3, 'i')
(4, 'z')
```

7.3.3 If. . .

```
In [49]: age = int(input('How old are you? '))

         if age <= 2:
             print(' free')
         elif 2 < age < 13:
             print(' child fare')
         else:
             print(' adult fare')
```

```
How old are you? 6
 child fare
```

8 File I/O

8.1 ASCII

Python can natively open/read/write but there are more powerful tools in many modules.

```
In [50]: ! cat ascii.txt

         f = open('ascii.txt','r')
         data = f.readlines()
         f.close()

         print()
         print( data )
```

```
#id,      ra,               dec
M56,      289.147941100,   30.184500500
ic4710,   277.158208330,   -66.982277780
ngc4552,  188.915863750,   12.556341390
# Appending comment to file# Appending comment to file# Appending comment to file# Appending comment to
['#id,      ra,               dec\n', 'M56,      289.147941100,   30.184500500\n', 'ic4710,   277.158208
```

```
In [51]: f = open('ascii.txt','a')
         f.write('# Appending comment to file')
         f.close()

         ! cat ascii.txt
```

```
#id,      ra,               dec
M56,      289.147941100,   30.184500500
ic4710,   277.158208330,   -66.982277780
ngc4552,  188.915863750,   12.556341390
# Appending comment to file# Appending comment to file# Appending comment to file# Appending comment to
```

```
In [52]: import numpy
         data = numpy.genfromtxt('ascii.txt', delimiter=',', dtype=None, names=True)
         data
```

```
Out[52]: array([('M56', 289.1479411, 30.1845005),
                ('ic4710', 277.15820833, -66.98227778),
                ('ngc4552', 188.91586375, 12.55634139)],
               dtype=[('id', 'S7'), ('ra', '<f8'), ('dec', '<f8')])
```

```
In [53]: data['id']
```

```
Out[53]: array(['M56', 'ic4710', 'ngc4552'],
               dtype='|S7')
```

```
In [54]: data['id' == 'M56']['ra']
```

```
Out[54]: 289.14794110000003
```

8.2 FITs

PyFITS has been incorporated into astropy.io.fits, but the API remains the same.

The three most useful commands for reading in data are: * pyfits.info(<filename>) * pyfits.getheader(<filename>,<extension>) * pyfits.getdata(<filename>,<extension>)

```
In [55]: import astropy.io.fits as pyfits
         filename = 'hst_wfpc2_downsized_example.fits'
         pyfits.info(filename)
```

```
Filename: hst_wfpc2_downsized_example.fits
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU     262   (200, 200, 4)   float32
1    u5780205r_cvt.c0h.tab  TableHDU       353   4R x 49C     [D25.17, D25.17, E15.7, E15.7, E15.7, E15.
```

```
In [56]: hdr = pyfits.getheader(filename,0)
         hdr
```

```
Out[56]: SIMPLE  =                    T / file does conform to FITS standard
         BITPIX  =                  -32 / number of bits per data pixel
         NAXIS   =                    3 / number of data axes
         NAXIS1  =                  200 / length of data axis 1
         NAXIS2  =                  200 / length of data axis 2
         NAXIS3  =                    4 / length of data axis 3
         EXTEND  =                    T / FITS dataset may contain extensions
         COMMENT   FITS (Flexible Image Transport System) format is defined in 'Astronomy
         COMMENT   and Astrophysics', volume 376, page 359; bibcode: 2001A&A...376..359H
         BSCALE  =                1.0E0 / REAL = TAPE*BSCALE + BZERO
         BZERO   =                0.0E0 /
         OPSIZE  =                 2112 / PSIZE of original image
         ORIGIN  = 'STScI-STSDAS'       / Fitsio version 21-Feb-1996
         FITSDATE= '2004-01-09'         / Date FITS file was created
         FILENAME= 'u5780205r_cvt.c0h'  / Original filename
         ALLG-MAX=          3.777701E3 / Data max in all groups
         ALLG-MIN=         -7.319537E1 / Data min in all groups
         ODATTYPE= 'FLOATING'           / Original datatype: Single precision real
         SDASMGNU=                    4 / Number of groups in original image
         CRVAL1  =       182.6311886308
```

```
CRVAL2  =       39.39633673411
CRPIX1  =                 420.
CRPIX2  =                424.5
CD1_1   =        -1.067040E-6
CD1_2   =        -1.259580E-5
CD2_1   =        -1.260160E-5
CD2_2   =         1.066550E-6
DATAMIN =         -7.319537E1 / DATA MIN
DATAMAX =          3.777701E3 / DATA MAX
MIR_REVR=                    T
ORIENTAT=               -85.16
FILLCNT =                    0
ERRCNT  =                    0
FPKTTIME=         51229.798574
LPKTTIME=         51229.798742
CTYPE1  = 'RA---TAN'
CTYPE2  = 'DEC--TAN'
DETECTOR=                    1
DEZERO  =             316.6452
BIASEVEN=             316.6715
BIASODD =             316.6189
GOODMIN =            -5.064006
GOODMAX =              2552.17
DATAMEAN=            0.4182382
GPIXELS =               632387
SOFTERRS=                    0
CALIBDEF=                 1466
STATICD =                    0
ATODSAT =                   16
DATALOST=                    0
BADPIXEL=                    0
OVERLAP =                    0
PHOTMODE= 'WFPC2,1,A2D7,LRF#4877.0,,CAL'
PHOTFLAM=         3.447460E-16
PHOTZPT =                -21.1
PHOTPLAM=             4884.258
PHOTBW  =             20.20996
MEDIAN  =            -0.175651
MEDSHADO=            -0.121681
HISTWIDE=             1.033711
SKEWNESS=            -1.983727
MEANC10 =              0.12958
MEANC25 =            0.3129676
MEANC50 =            0.4577668
MEANC100=            0.3916293
MEANC200=            0.3115222
MEANC300=            0.3295493
BACKGRND=           -0.3676353
ORIGIN  = 'NOAO-IRAF FITS Image Kernel December 2001' / FITS file originator
DATE    = '2004-01-09T03:26:36'
IRAF-TLM= '03:26:36 (09/01/2004)'
FILETYPE= 'SCI      '           / type of data found in data file

TELESCOP= 'HST'                 / telescope used to acquire data
```

```
INSTRUME= 'WFPC2 '             / identifier for instrument used to acquire data
EQUINOX =               2000.0 / equinox of celestial coord. system

              / WFPC-II DATA DESCRIPTOR KEYWORDS

ROOTNAME= 'u5780205r'             / rootname of the observation set
PROCTIME=   5.301314019676E+04 / Pipeline processing time (MJD)
OPUS_VER= 'OPUS 14.5a        ' / OPUS software system version number
CAL_VER = '                        ' / CALWP2 code version

              / SCIENCE INSTRUMENT CONFIGURATION

MODE    = 'FULL'               / instr. mode: FULL (full res.), AREA (area int.)
SERIALS = 'OFF'                / serial clocks: ON, OFF

              / IMAGE TYPE CHARACTERISTICS

IMAGETYP= 'EXT               ' / DARK/BIAS/IFLAT/UFLAT/VFLAT/KSPOT/EXT/ECAL
CDBSFILE= 'NO                ' / GENERIC/BIAS/DARK/PREF/FLAT/MASK/ATOD/NO
PKTFMT  =                   96 / packet format code

              / FILTER CONFIGURATION

FILTNAM1= 'FR533P15'           / first filter name
FILTNAM2= '        '           / second filter name
FILTER1 =                   69 / first filter number (0-48)
FILTER2 =                    0 / second filter number (0-48)
FILTROT =                 15.0 / partial filter rotation angle (degrees)
LRFWAVE =          4877.000000 / linear ramp filter wavelength

              / INSTRUMENT STATUS USED IN DATA PROCESSING

UCH1CJTM=             -88.2569 / TEC cold junction #1 temperature (Celsius)
UCH2CJTM=             -88.6697 / TEC cold junction #2 temperature (Celsius)
UCH3CJTM=             -88.3028 / TEC cold junction #3 temperature (Celsius)
UCH4CJTM=             -88.7671 / TEC cold junction #4 temperature (Celsius)
UBAY3TMP=              13.2302 / bay 3 A1 temperature (deg C)
KSPOTS  = 'OFF'                / Status of Kelsall spot lamps: ON, OFF
SHUTTER = 'A'                  / Shutter in place at beginning of the exposure
ATODGAIN=                  7.0 / Analog to Digital Gain (Electrons/DN)

              / RSDP CONTROL KEYWORDS

MASKCORR= 'COMPLETE'           / Do mask correction: PERFORM, OMIT, COMPLETE
ATODCORR= 'COMPLETE'           / Do A-to-D correction: PERFORM, OMIT, COMPLETE
BLEVCORR= 'COMPLETE'           / Do bias level correction
BIASCORR= 'COMPLETE'           / Do bias correction: PERFORM, OMIT, COMPLETE
DARKCORR= 'COMPLETE'           / Do dark correction: PERFORM, OMIT, COMPLETE
FLATCORR= 'SKIPPED '           / Do flat field correction
SHADCORR= 'OMIT    '           / Do shaded shutter correction
DOSATMAP= 'OMIT    '           / Output saturated pixel map
DOPHOTOM= 'COMPLETE'           / Fill photometry keywords
DOHISTOS= 'OMIT    '           / Make histograms: PERFORM, OMIT, COMPLETE
OUTDTYPE= 'REAL  '             / Output image datatype: REAL, LONG, SHORT
```

```
              / CALIBRATION REFERENCE FILES

MASKFILE= 'uref$f8213081u.r0h     ' / name of the input DQF of known bad pixels
ATODFILE= 'uref$dbu1405iu.r1h'      / name of the A-to-D conversion file
BLEVFILE= 'ucal$u5780205r.x0h     ' / Engineering file with extended register da
BLEVDFIL= 'ucal$u5780205r.q1h     ' / Engineering file DQF
BIASFILE= 'uref$j9a1612mu.r2h'      / name of the bias frame reference file
BIASDFIL= 'uref$j9a1612mu.b2h'      / name of the bias frame reference DQF
DARKFILE= 'uref$j2g1549cu.r3h'      / name of the dark reference file
DARKDFIL= 'uref$j2g1549cu.b3h'      / name of the dark reference DQF
FLATFILE= 'uref$f4i1559cu.r4h'      / name of the flat field reference file
FLATDFIL= 'uref$f4i1559cu.b4h'      / name of the flat field reference DQF
SHADFILE= 'uref$e371355eu.r5h'      / name of the reference file for shutter sha
PHOTTAB = 'u5780205r_c3t.fits'      / name of the photometry calibration table
GRAPHTAB= 'mtab$n9i1408hm_tmg.fits' / the HST graph table
COMPTAB = 'mtab$nc809508m_tmc.fits' / the HST components table

              / DEFAULT KEYWORDS SET BY STSCI

SATURATE=                 4095 / Data value at which saturation occurs
USCALE  =                  1.0 / Scale factor for output image
UZERO   =                  0.0 / Zero point for output image

              / READOUT DURATION INFORMATION

READTIME=                  464 / Length of time for CCD readout in clock ticks

              / PLANETARY SCIENCE KEYWORDS

PA_V3   =            49.936909 / position angle of V3-axis of HST (deg)
RA_SUN  =   3.337194516616E+02 / right ascension of the sun (deg)
DEC_SUN =  -1.086675160382E+01 / declination of the sun (deg)
EQNX_SUN=               2000.0 / equinox of the sun
MTFLAG  =                    F / moving target flag; T if it is a moving target
EQRADTRG=             0.000000 / equatorial radius of target (km)
FLATNTRG=             0.000000 / flattening of target
NPDECTRG=             0.000000 / north pole declination of target (deg)
NPRATRG =             0.000000 / north pole right ascension of target (deg)
ROTRTTRG=             0.000000 / rotation rate of target
LONGPMER=             0.000000 / longitude of prime meridian (deg)
EPLONGPM=             0.000000 / epoch of longitude of prime meridian (sec)
SURFLATD=             0.000000 / surface feature latitude (deg)
SURFLONG=             0.000000 / surface feature longitude (deg)
SURFALTD=             0.000000 / surface feature altitude (km)

              / PODPS FILL VALUES

PODPSFF =                    0 / 0=(no  podps fill); 1=(podps fill present)
STDCFFF =                    0 / 0=(no st dcf fill); 1=(st dcf fill present)
STDCFFP = '0x5569'             / st dcf fill pattern (hex)
RSDPFILL=                 -100 / bad data fill value for calibrated images

              / EXPOSURE TIME AND RELATED INFORMATION
```

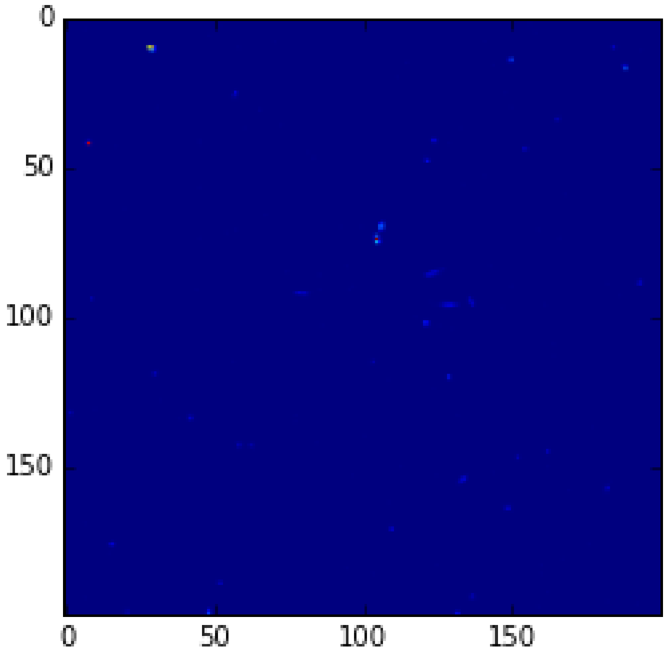
```
UEXPODUR=                  300 / commanded duration of exposure (sec)
NSHUTA17=                    1 / Number of AP17 shutter B closes
DARKTIME=   3.000000000000E+02 / Dark time (seconds)
UEXPOTIM=                16880 / Major frame pulse time preceding exposure start
PSTRTIME= '1999.051:19:08:37 ' / predicted obs. start time (yyyy.ddd:hh:mm:ss)
PSTPTIME= '1999.051:19:16:37 ' / predicted obs. stop time (yyyy.ddd:hh:mm:ss)

              / EXPOSURE INFORMATION

SUNANGLE=           141.618347 / angle between sun and V1 axis
MOONANGL=           126.698997 / angle between moon and V1 axis
SUN_ALT =           -31.523479 / altitude of the sun above Earth's limb
FGSLOCK = 'FINE              ' / commanded FGS lock (FINE,COARSE,GYROS,UNKNOWN)

DATE-OBS= '1999-02-20'         / UT date of start of observation (yyyy-mm-dd)
TIME-OBS= '19:03:13'           / UT time of start of observation (hh:mm:ss)
EXPSTART=   5.122979390428E+04 / exposure start time (Modified Julian Date)
EXPEND  =   5.122979737650E+04 / exposure end time (Modified Julian Date)
EXPTIME =   3.000000000000E+02 / exposure duration (seconds)--calculated
EXPFLAG = 'NORMAL       '      / Exposure interruption indicator

              / TARGET & PROPOSAL ID
TARGNAME= 'NGC4151                         ' / proposer's target name
RA_TARG =   1.826355000000E+02 / right ascension of the target (deg) (J2000)
DEC_TARG=   3.940576666667E+01 / declination of the target (deg) (J2000)
ECL_LONG=           164.096619 / ecliptic longitude of the target (deg) (J2000)
ECL_LAT =            36.623709 / ecliptic latitude of the target (deg) (J2000)
GAL_LONG=           155.079532 / galactic longitude of the target (deg) (J2000)
GAL_LAT =            75.062679 / galactic latitude of the target (deg) (J2000)

PROPOSID=                 8019 / PEP proposal identifier
PEP_EXPO= '02-030         '    / PEP exposure identifier including sequence
LINENUM = '02.030         '    / PEP proposal line number
SEQLINE = '               '    / PEP line number of defined sequence
SEQNAME = '               '    / PEP define/use sequence name
HISTORY   MASKFILE=uref$f8213081u.r0h  MASKCORR=COMPLETED
HISTORY   PEDIGREE=INFLIGHT 01/01/1994 - 15/05/1995
HISTORY   DESCRIP=STATIC MASK - INCLUDES CHARGE TRANSFER TRAPS
HISTORY   BIASFILE=uref$j9a1612mu.r2h  BIASCORR=COMPLETED
HISTORY   PEDIGREE=INFLIGHT 29/08/98 - 21/08/99
HISTORY   DESCRIP=not significantly different from j6e16008u.
HISTORY   DARKFILE=uref$j2g1549cu.r3h  DARKCORR=COMPLETED
HISTORY   PEDIGREE=INFLIGHT 16/02/1999 - 16/02/1999
HISTORY   DESCRIP=Pipeline dark: 120 frame superdark with hotpixels from
HISTORY   16/02/99
HISTORY   FLATFILE=uref$f4i1559cu.r4h  FLATCORR=SKIPPED
HISTORY   PEDIGREE=DUMMY  18/04/1995
HISTORY   DESCRIP=All pixels set to value of 1. Not flat-fielded.
HISTORY   PC1: bias jump level ~0.100 DN.
HISTORY   The following throughput tables were used:
HISTORY   crotacomp$hst_ota_007_syn.fits, crwfpc2comp$wfpc2_optics_006_syn.fits,
HISTORY   crwfpc2comp$wfpc2_lrf_004_syn.fits[wave#],
HISTORY   crwfpc2comp$wfpc2_dqepc1_005_syn.fits,
HISTORY   crwfpc2comp$wfpc2_a2d7pc1_004_syn.fits,
HISTORY   crwfpc2comp$wfpc2_flatpc1_003_syn.fits
HISTORY   The following throughput tables were used:
HISTORY   crotacomp$hst_ota_007_syn.fits, crwfpc2comp$wfpc2_optics_006_syn.fits,
HISTORY   crwfpc2comp$wfpc2_lrf_004_syn.fits[wave#],
HISTORY   crwfpc2comp$wfpc2_dqewfc2_005_syn.fits,
HISTORY   crwfpc2comp$wfpc2_a2d7wf2_004_syn.fits,
HISTORY   crwfpc2comp$wfpc2_flatwf2_003_syn.fits
HISTORY   The following throughput tables were used:
HISTORY   crotacomp$hst_ota_007_syn.fits, crwfpc2comp$wfpc2_optics_006_syn.fits,
HISTORY   crwfpc2comp$wfpc2_lrf_004_syn.fits[wave#],
HISTORY   crwfpc2comp$wfpc2_dqewfc3_005_syn.fits,
HISTORY   crwfpc2comp$wfpc2_a2d7wf3_004_syn.fits,
HISTORY   crwfpc2comp$wfpc2_flatwf3_003_syn.fits
HISTORY   The following throughput tables were used:
HISTORY   crotacomp$hst_ota_007_syn.fits, crwfpc2comp$wfpc2_optics_006_syn.fits,
HISTORY   crwfpc2comp$wfpc2_lrf_004_syn.fits[wave#],
HISTORY   crwfpc2comp$wfpc2_dqewfc4_005_syn.fits,
HISTORY   crwfpc2comp$wfpc2_a2d7wf4_004_syn.fits,
HISTORY   crwfpc2comp$wfpc2_flatwf4_003_syn.fits
CTYPE3  = 'GROUP_NUMBER'       / Extra dimension axis name
CD3_3   =                    1 /
CD3_1   =                    0 /
CD1_3   =                    0 /
CD2_3   =                    0 /
CD3_2   =                    0 /
```

```
In [94]: hdr['ORIGIN']
```

```
Out[94]: 'STScI-STSDAS'
```

```
In [57]: data = pyfits.getdata(filename,0)

         from matplotlib.pyplot import *
         imshow(data[0])
```

```
Out[57]: <matplotlib.image.AxesImage at 0x10cfc9c10>
```



8.3 Temporary files (pickling)

```
In [58]: # import cPickle as pickle  # Just pickle in Python3. Example below works in both
         import pickle

         data = range(10)
         pickle.dump(data, open('test.pkl', 'wb'))

         pickle.load(open('test.pkl', 'rb'))

Out[58]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

8.4 IDL savefiles

IDL¿ data = ['do','you','really','want','to','do','this','?'] IDL¿ save, data, filename='idl.sav'

```
In [60]: import scipy.io
         scipy.io.idl.readsav('idl.sav')

Out[60]: {'data': array(['do', 'you', 'really', 'want', 'to', 'do', 'this', '?'], dtype=object)}
```

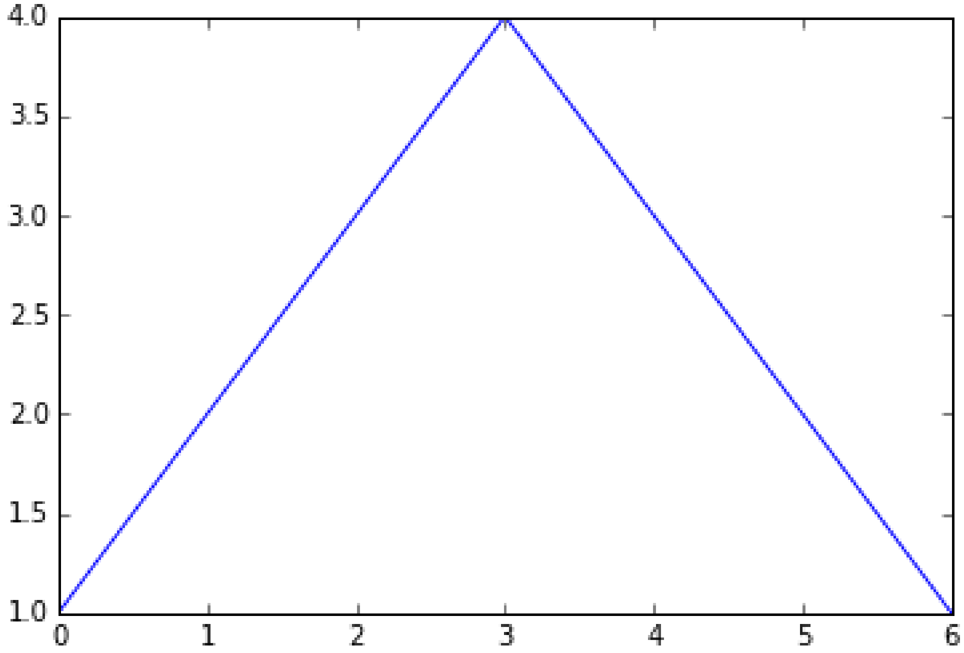
9 Plotting

9.1 Initialization

```
In [102]: %matplotlib inline
          from matplotlib.pyplot import *
          import numpy
```

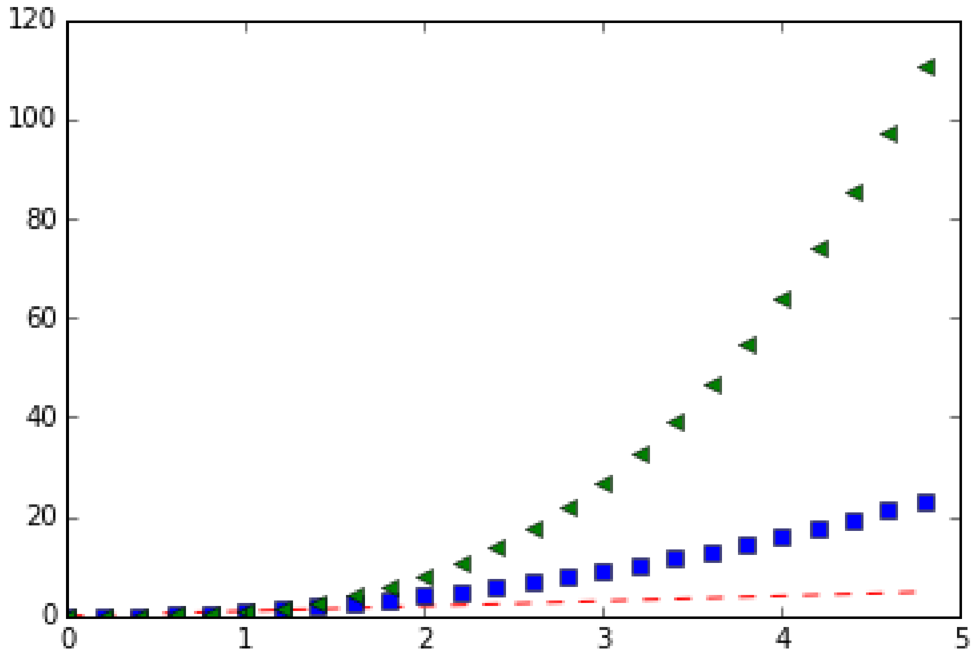
9.2 Simple Lines/Saving

```
In [103]: plot([1,2,3,4,3,2,1])
          #show()
          savefig('images/line_plot.png')
```



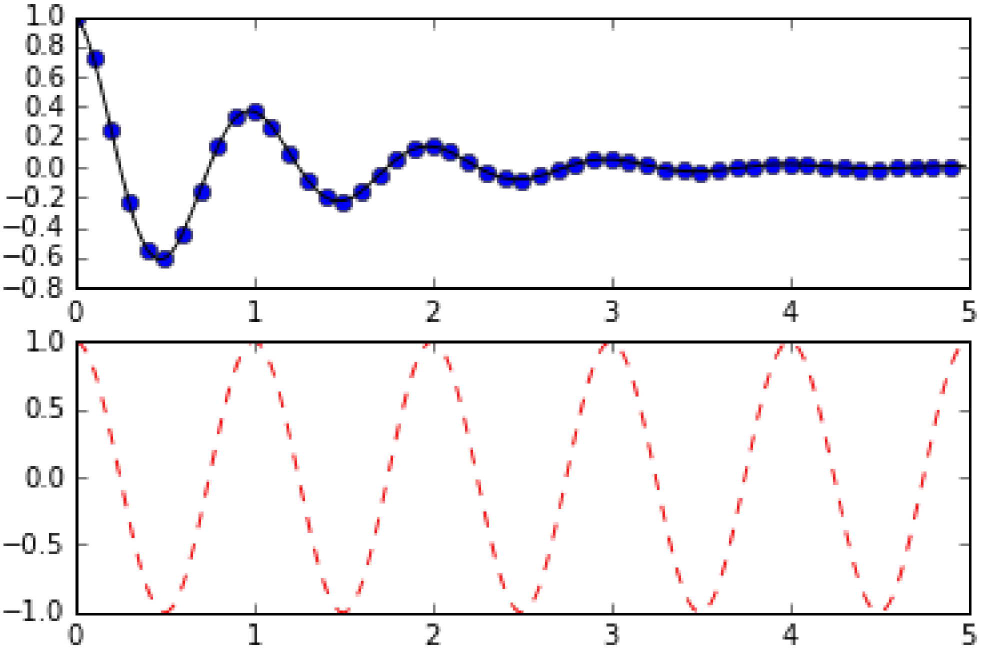
9.3 Multiple Lines

```
In [104]: # evenly sampled time at 200ms intervals
          t = numpy.arange(0., 5., 0.2)

          # red dashes, blue squares and green triangles
          plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g<')

Out[104]: [<matplotlib.lines.Line2D at 0x107fc91d0>,
           <matplotlib.lines.Line2D at 0x107fc9590>,
           <matplotlib.lines.Line2D at 0x107fc9e10>]
```



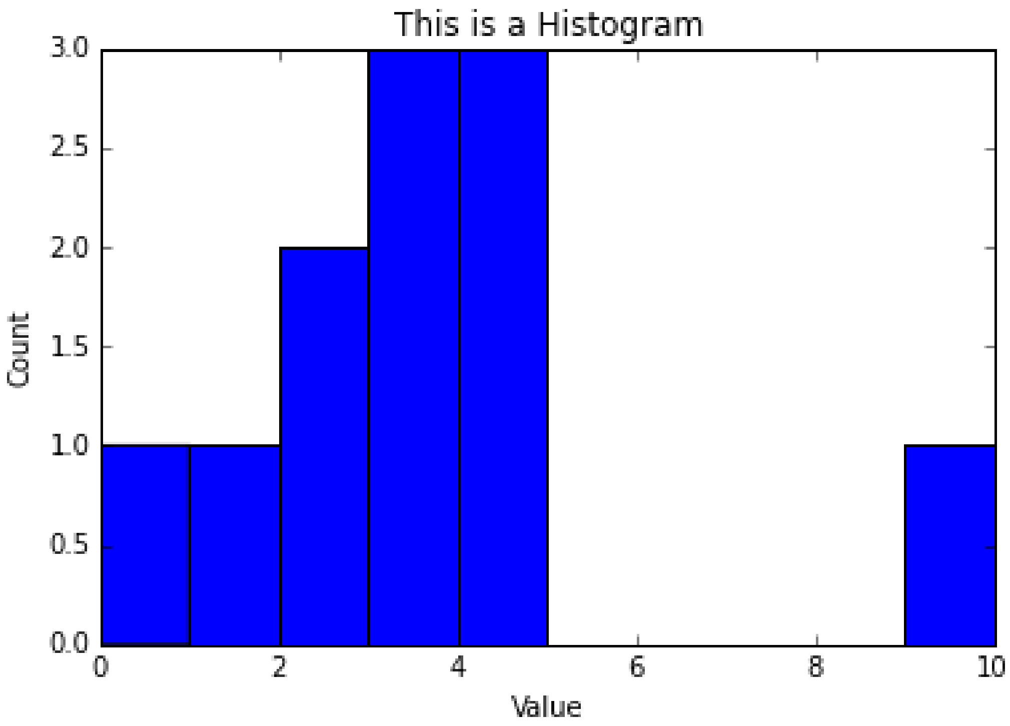
9.4 Subplots

```
In [105]: def f(t):
              return numpy.exp(-t) * numpy.cos(2*numpy.pi*t)

          t1 = numpy.arange(0.0, 5.0, 0.1)
          t2 = numpy.arange(0.0, 5.0, 0.02)

          figure(1)
          subplot(211)
          plot(t1, f(t1), 'bo', t2, f(t2), 'k')

          subplot(212)
          plot(t2, numpy.cos(2*np.pi*t2), 'r--')

Out[105]: [<matplotlib.lines.Line2D at 0x10822ec10>]
```



9.5 Histogram/Labels

```
In [106]: hist([0, 1, 2, 2, 3, 3, 3, 4, 4, 4, 10])
          xlabel('Value')
          ylabel('Count')
          title('This is a Histogram')

Out[106]: <matplotlib.text.Text at 0x1082b7b50>
```



9.6 Gallery

```
In [161]: from IPython.display import HTML
          HTML('<iframe src=http://matplotlib.org/gallery.html width=600 height=350></iframe>')

Out[161]: <IPython.core.display.HTML at 0x10ec3e550>
```

10 Tips, Tricks & Gotchas

10.1 Embedding (== IDL STOP)

```
In []: !cat ipython_stop.py
```

$pythonipython_stop.pyEnteringscript$

In [1]: Variable Type Data/Info ———————————— a int 1 b int 1 embed function ¡function embed at 0x102140cf8¿

In [2]: exit

In [3]: Variable Type Data/Info ———————————— a int 1 b int 2 embed function ¡function embed at 0x102140cf8¿

In [4]: a='one' In [5]: exit

In [6]: Variable Type Data/Info ———————————— a str one b int 3 embed function ¡function embed at 0x102140cf8¿

In [7]: exit

Exiting - goodbye!

10.2 Copying Variables

```
In [61]: a = 1
         b = a

         print( ’a = %s : %s\nb = %s : %s’ %(a, id(a), b, id(b)) )
```

```
a = 1 : 4298163880
b = 1 : 4298163880
```

```
In [62]: b = 2     # Reassign

         print( ’a = %s : %s\nb = %s : %s’ %(a, id(a), b, id(b)) )
```

```
a = 1 : 4298163880
b = 2 : 4298163856
```

```
In [63]: a = [1,2]
         b = a

         print( ’a = %s : %s\nb = %s : %s’ %(a, id(a), b, id(b)) )
```

```
a = [1, 2] : 4512399016
b = [1, 2] : 4512399016
```

```
In [64]: b[0] = 3     # Update

         print( ’a = %s : %s\nb = %s : %s’ %(a, id(a), b, id(b)) )
```

```
a = [3, 2] : 4512399016
b = [3, 2] : 4512399016
```

```
In [65]: import copy
         b = copy.copy(a)

         print( ’a = %s : %s\nb = %s : %s’ %(a, id(a), b, id(b)) )
```

```
a = [3, 2] : 4512399016
b = [3, 2] : 4557784920
```

```
In [66]: b[0] = 4

         print( ’a = %s : %s\nb = %s : %s’ %(a, id(a), b, id(b)) )
```

```
a = [3, 2] : 4512399016
b = [4, 2] : 4557784920
```

10.3 List != (Numpy) Array

- Lists can contain anything
- Arrays contain a single type
- Arrays are continuous in memory
- Operations are far more efficient

```
In [67]: a = [1,2,3]
         b = [4,5,6]

         a + b     # Appends
```

```
Out[67]: [1, 2, 3, 4, 5, 6]

In [68]: import numpy as np

         c = np.array([7,8,9])
         d = np.array([10,11,12])

         c + d    # Elementwise

Out[68]: array([17, 19, 21])

In [69]: a + c    # Elementwise !!!

Out[69]: array([ 8, 10, 12])

In [70]: [1,2,'three'] + c    # Cannot convert
```

```
        ---------------------------------------------------------------------------
    TypeError                                 Traceback (most recent call last)

        <ipython-input-70-593313b7eb3c> in <module>()
    ----> 1 [1,2,'three'] + c    # Cannot convert


        TypeError: can only concatenate list (not "numpy.ndarray") to list
```

11 To Conclude

- I hope that this has been a useful glimpse into Python
- Both the language itself …
- … and the scientific/technical ecosystem around it
- There is (of course) much, much more:
  - handling errors
  - OOP & classes
  - generators
  - …
- Plus it is still very much improving!

** Part II Accessing, Manipulating and Visualizing Astronomical Data March 19, 2015, 1:00-2:30 pm KS-410 David Shupe **

Astropy and affiliated packages for handling astronomical data: - Aims of the Astropy project - Astropy capabilities - FITS file handling - Table handling - Celestial Coordinates - Units and Quantities - Catalog queries - Astropy-affiliated packages for visualization - Example: APLPy for publication-quality display of images - Example: Glue for data exploration - How-to examples (e.g. spatially match catalogs, plot results, a simple image stacker)

https://caltech.box.com/intro-to-python