Python & CASA Scripting

Steven T. Myers
(NRAO-Socorro)

ALMA

JVLA

CASA Scripting - Data Reduction Workshop Oct 2014
CASA Scripting – Why?

• a durable and executable record of your processing
  – should contain annotation!
• transporatable and distributable
  – can send to your colleagues or post online
• efficient for long reduction sequences
  – datasets often too large to “archive” by the user
• reproducible (for a given version of CASA)
  – important for debugging and error finding
• build up a custom library of useful functions and tasks
  – importable, reusable, tradeable
• session logs: casapy-<…>.log and ipython-<…>.log not complete
Python

• CASA uses Python for standard scripting interface, with IPython extensions for interactive command line use

• Many online (and book) sources for information
  – Easy for users to “build their own” scripts/tasks

• Public Documentation:

  • Python:
    - [http://python.org/doc](http://python.org/doc) (e.g. Tutorial for novices)
  • IPython:
    - [http://ipython.org](http://ipython.org)
  • matplotlib:
    - [http://matplotlib.sourceforge.net](http://matplotlib.sourceforge.net)
  • [http://matplotlib.sourceforge.net](http://matplotlib.sourceforge.net)
Python

- **CASA-specific Documentation:**

- **Casaguides:**
  
  [http://casaguides.nrao.edu](http://casaguides.nrao.edu) (Getting Started in CASA)

- **CASA User Reference and Cookbook:**
  
  [http://casa.nrao.edu/ref_cookbook.shtml](http://casa.nrao.edu/ref_cookbook.shtml) (Appendix B)

- **Example scripts:**
  
  [http://www.aoc.nrao.edu/~smyers/casa/scripts/4.2.2/run_b1608_demo_v4.2.2.py](http://www.aoc.nrao.edu/~smyers/casa/scripts/4.2.2/run_b1608_demo_v4.2.2.py)
  
  [http://www.aoc.nrao.edu/~smyers/casa/scripts/4.2.2/scaleweights.py](http://www.aoc.nrao.edu/~smyers/casa/scripts/4.2.2/scaleweights.py)
Python Basics

- Setting variables:
  - Assignment `<parameter>`=<value>
  - Testing `<parameter>`==<value> (or >,<,>={},<={})
  - Tasks use a “standard” set of global variables
- Watch out for mis-spellings (e.g. correlation vs. corellation), you just create a new variable

- Lists:
  - Assignment: `antlist = ['ea04', 'ea05', 'ea13']`
  - Append: `antlist.append('ea28')`
  - 0-based indices: `antlist[0]` (returns value ‘ea04’)
Python – Strings, Files, Output

- **Strings split operator (and tuples)**
  
  ```python
  # break string into key=val sets
  keyvlist = cmdstr.split()
  if keyvlist.__len__() > 0:
      for keyv in keyvlist:
          (xkey, val) = keyv.split('=')
  ```

- **File creation and access**
  
  ```
  logfile = open(outfile, 'w')
  ```

- **Output: print**
  
  ```
  print 'Cleaning MFS continuum image SPW '+spw+' '+instokes
  print 'Field %s P/I = %10.4f RLPD = %8.3f deg' % (infield, mflx, rlpd)
  print >>logfile, 'Field %s P/I = %10.4f RLPD = %8.3f deg' % (infield, mflx, rlpd)
  ```
Python Basics – Ranges, Loops

• Range function
  – Assignment `antlist = range(4,8)`
  • Equivalent to `antlist = [4, 5, 6, 7]`

• Blocks, Loops, and Indentation :
  – Indentation matters, sets apart levels of blocks/loops
  – Conditional blocks: if-elif-else
  – Loops: for, while
    • for i in range(5)
    • for ant in antlist
    • while <boolean>
Example Script – Conditional, Assignment

- Conditional blocks: if – elif – else

  if obsconfig=='C':
    # C-config FOV/beam = 128
    myimsize = 400
  if obsband=='L':
    # L-band beam 30' FOV at 1.5 GHz
    # C-config resolution: 1.5 GHz = 14"
    mycell = '4.0arcsec'
  elif obsband=='S':
    # S-band beam 15' FOV at 3 GHz
    # C-config resolution: 3 GHz = 7"
    mycell = '2.0arcsec'
  else:
    print 'ERROR: unknown band '+obsband
Example Script - Loop

- Loops (with some string construction thrown in)

```python
for field in myfieldlist:
    splitfile = prefix + '.field' + field + '.split.ms'
    outputvis = splitfile
    saveinputs(taskname,splitfile+'.saved')
    print 'Splitting field '+field+' to '+splitfile
    split()
```
Python Dictionaries

- A nested associative (hashed) list of \{ <key> : <value> \}

```
polname = 'J1331+3030'
polsrc = {}
polsrc[polname] = {}
polsrc[polname]['0'] = { 'I' : 14.61,
                           'F' : 0.094,
                           'X' : 66.0 }
polsrc[polname]['1'] = { 'I' : 13.09,
                           'F' : 0.094,
                           'X' : 66.0 }
```

- Access

```
polsrc [polname] ['0'] ['I']
```

Dictionaries can be saved to files using pickle module:

```
import pickle
...
```

You can also use text files containing dictionary, e.g.

```
polsrc = { '0': { 'I': 14.61, ... }, ... }
```
Python / IPython Basics

• Toolkit return values (and some math)

```python
if instokes.count('QU')>0:
    qval = imval(mfsimage,stokes='Q')
    uval = imval(mfsimage,stokes='U')
    qflx = qval['data'][0]
    uflx = uval['data'][0]
    rlpd = atan2(uflx,qflx)*180.0/pi
    pflx = sqrt(qflx*qflx + uflx*uflx)
```

• Exception handling: try, except (catch stuff that fails)

```python
try:
    gaincal()
except:
    print 'ERROR: aborting script'
raise
```
Useful Python modules

• pickle – read/write python dictionaries

• os
  – os.system() # execute shell commands
  – os.access() # test existence of file and directories etc.
  – os.getenv() # get value of environment variables

• time
  – time.time() # return time, use for benchmarking

• datetime
  – datetime.today() # todays date
  – datetime.isoformat() # turn date into string

• xml.dom – read from xml file (others xml modules exist also)
CASA “User” Development Aspects

• CASA = Python + Toolkit + Applications
  – You can develop in CASA at C++ level
    • “hard” but clearly possible (become a CASA developer)
  – If it’s a Python (2.7x) module/script CASA can use it
  – If you can run it in casapy you can use the toolkit
    • write a CASA task or function (or simple script)
  – If you have an app with command interface you can call it from CASA if it works on standard data formats
    • MS, casa images, FITS images, some flavors of uvfits, text, …
  – Strive to conform to minimal common interface
    • e.g. Numpy arrays, matplotlib, use of other standard facilities
What do I do?

• Write a function in Python
  – learn Python (e.g. python.org)
  – write <function>.py (use Python aware editor, e.g. emacs)
  – bring into casapy
    • execfile(‘<function>.py’) OR
    • import <function>
  – call function in casapy
    • <function>_<method>_(<args>)
  – good for simple functionality
    • bypasses task parameter interface
    • see any Python reference on how to do this
Example Function

- See: http://www.aoc.nrao.edu/~smyers/casa/scripts/scaleweights.py
- Preamble (definition, import casac, set up tools)

```python
def scaleweights(msfile, wtfac=1.0):
    try:
        import casac
    except ImportError, e:
        print "failed to load casa:\n", e
        exit(1)
    #mstool = casac.homefinder.find_home_by_name('msHome')
    #ms = casac.ms = mstool.create()
    #tbtool = casac.homefinder.find_home_by_name('tableHome')
    #tb = casac.tb = tbtool.create()
```
Example Function

- Access subtables in the MS (ms tool, table tool)

```python
# Access the MS
try:
    ms.open(msfile,nomodify=False)
except:
    print "ERROR: failed to open ms tool on file "+msfile
    exit(1)
# Find number of data description IDs
tb.open(msfile+"/DATA_DESCRIPTION")
ddspwlist=tb.getcol("SPECTRAL_WINDOW_ID")
ddpollist=tb.getcol("POLARIZATION_ID")
tb.close()
```
Example Function

- Access data columns in the MS, modify, and put back

```python
# Go through and change the weights
for idd in range(ndd):
    # Find number of correlations in this DD
    pid = ddpollist[idd]
    ncorr = ncorlist[pid]
    # Select this DD (after reset if needed)
    if idd>0: ms.selectinit(reset=True)
    ms.selectinit(idd)
    recw = ms.getdata(['weight'])
    (nx, ni) = recw['weight'].shape
    for j in range(ni):
        for i in range(nx):
            recw['weight'][i,j]*=wtfac
    ms.putdata(recw)
```
Tasking Targets: A Rich Environment

- Possible targets for user development
  - operations on image (cube) data
    - extract pixels, manipulate, report results, possibly return to cube
    - examples: source/line fitting/extraction, filtering, statistics, transforms
    - also physical modelling (e.g. from spectral cube)
  - image visualization
    - interactive exploration, hardcopy, cross-matching, identification
    - possibly with built-in image operations
  - data-space operations
    - uv modelfitting, imaging, interference mitigation, data visualization
  - modelling
    - simulation-to-image, simulation-to-data
How does this work in practice?

- There have been contributed tasks, e.g.
  - importevla (wrapping asdm2ms)
  - flagcmd (wrapping table and flagger)
  - boxit (autoboxing, wrapping images)
  - autoclean (using autoboxing and imager)
  - plotweather (plot info from VLA WEATHER table)
How does this work in practice?

• Cool. How do I distribute/get stuff like this?
  – “insiders”: get CASA team to check into code base
  – “outsiders”: post somewhere (CASA Science Forum)
    • https://science.nrao.edu/forums/
  – “associates”: get put onto casaguides
    • http://casaguides.nrao.edu
  – future: better mechanism?
What else can I do?

• Write a CASA Task:
  – learn Python (e.g. python.org)
  – read (parts of) CASA User Reference & Cookbook
    • See Appendix H, includes example
  – put Python code into task_<task>.py
  – put params and help text into <task>.xml
  – use “buildmytasks task” from unix (outside casa)
    • compiles to .pyc and puts into mytasks.py
  – go into casapy and execfile(‘mytasks.py’)
    • to update task, need to restart casapy
Building Tasks - Documentation

• **Documentation:**

• **CASA User Reference and Cookbook:**
  [http://casa.nrao.edu/ref_cookbook.html](http://casa.nrao.edu/ref_cookbook.html) (Appendix H)

• **Casaguides:**
  [http://casaguides.nrao.edu](http://casaguides.nrao.edu) (**Writing a CASA Task**)
  
CASA Documentation

• Homepage: http://casa.nrao.edu → Using CASA
• CASA Reference Manual & Cookbook:
  http://casa.nrao.edu/docs/UserMan/UserMan.html
• CASA Task Reference (same as inline help):
  http://casa.nrao.edu/docs/TaskRef/TaskRef.html
• CASA Toolkit Manual:
  http://casa.nrao.edu/docs/casaref/CasaRef.html
• CASAguides Wiki:
  http://casaguides.nrao.edu
• Python:
  http://python.org/doc (e.g., see Tutorial for novices)
• IPython:
  http://ipython.org
• matplotlib:
  http://matplotlib.sourceforge.net/