*Astronomical Data Analysis Software and Systems XV*
*ASP Conference Series, Vol. 351, 2006*
*C. Gabriel, C. Arviset, D. Ponz and E. Solano, eds.*

# ParselTongue: AIPS Talking Python

Mark Kettenis, Huib Jan van Langevelde, Cormac Reynolds

*Joint Institute for VLBI in Europe, Dwingeloo, The Netherlands*

Bill Cotton

*National Radio Astronomy Observatory,Charlottesville VA, USA*

**Abstract.** After more than 20 years of service, classic AIPS still is the data reduction package of choice for many radio-interferometry projects, especially for VLBI. Its age shows, most prominently in the limited scripting capabilities of its user interface: POPS. ParselTongue is an attempt to make the trusted AIPS algorithms and AIPS data structures available in a modern dynamic programming language: Python. It also provides an environment to do distributed computing to take advantage of modern computing clusters. This makes it suitable for use as a scripting interface for doing complicated data reduction on large data sets. It is also used as a coding platform for the new calibration algorithms that are being developed for the European VLBI Network as part of the ALBUS project. Here we hope to take advantage of Python's extensive support for web-based technologies to automate things like collecting calibration data.

## 1. ParselTongue

ParselTongue[1] provides a Python interface to classic AIPS[2] and Obit[3]. This package makes it possible to run AIPS (and Obit) tasks, and manipulate AIPS (meta)data from a modern dynamic programming language. It will serve as the infrastructure for RadioNet's Advance Long Baseline User Software (ALBUS)[4] project, but is useful in its own right for things like complex automated data reduction and implementing pipelines. Its features are best illustrated by a couple of examples.

[1] `http://www.radionet-eu.org/rnwiki/ParselTongue`

[2] `http://www.aoc.nrao.edu/aips/`

[3] `http://www.cv.nrao.edu/~bcotton/Obit.html`

[4] `http://www.radionet-eu.org/jra/albus.php`

## 2. Examples

The first example shows how to run FITLD to load an image into AIPS and then run IMEAN to calculate the RMS noise of the image:

```
from AIPSTask import AIPSTask
from AIPSData import AIPSImage

image = AIPSImage('4C39.25', 'ICLN', 1, 1)

fitld = AIPSTask('fitld')
fitld.infile = '/home/potter/4C39.25.FITS'
fitld.outdata = image
fitld.go()                          # Run FITLD

imean = AIPSTask('imean')
imean.indata = image
imean.go()                          # Run IMEAN

print 'RMS noise:', imean.pixstd  # Print output from IMEAN
```

Note that ParselTongue allows you to use full path names to specify your input files, even if they are longer than 48 characters. The AIPSTask objects used to run FITLD and IMEAN in the example are constructed dynamically from AIPS help files. This means that all AIPS tasks for which a standard AIPS documentation file can be found by ParselTongue, are supported; even ones that don't exist yet. Changes to the existing AIPS installation are not required.

Creating pipeline-like scripts of course requires more than just running tasks. A truly useful script will also need to make decisions based on the data being processed. To make this possible, ParselTongue provides almost full access to image and UV data. The next example shows how to read keywords from headers and collect information from extension tables:

```
from AIPSData import AIPSUVData

uvdata = AIPSUVData('N05L2', 'UVDATA', 1, 1)

print 'Date:', uvdata.header.date
print 'Telescope', uvdata.header.telescop
print 'Antennas:', uvdata.antennas
print 'Sources:', uvdata.sources
print 'Stokes:', uvdata.stokes
```

Dull read and write access to extension tables is also possible:

```
from Wizardry.AIPSData import AIPSUVData

uvdata = AIPSUVData('N05L2', 'UVDATA', 1, 1)
oldcl = uvdata.table('CL', 1)
newcl = uvdata.attach_table('CL', 2, no_term=6)
```

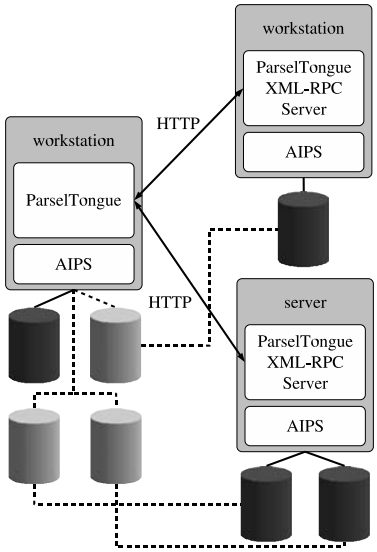


Figure 1. Remote execution of AIPS tasks using ParselTongue.

```
for row in oldcl:
    row.real_2 = row.real_1
    row.imag_2 = row.imag_1
    newcl.append(row)
```

This little script will transfer the gain calibration from the first polarization to the second polarization, and put it in a new calibration table.

In addition, it is even possible to access raw visibilities in UV data sets in an efficient way. All data access is implemented using the Obit C Library.

## 3. Features

ParselTongue can be used interactively from an interactive Python session. It customizes the standard Python environment to provide TAB-completion, minimal match and command line history. It also provides help on all documented AIPS and Obit tasks.

ParselTongue also allows remote execution of tasks using Python's built-in XML-RPC support. This can be used to run tasks in parallel on a cluster. The architecture is shown in figure 1. AIPS disks on remote machines appear as additional disks on the machine used to run the ParselTongue script. Tasks that use data on such a remote disk execute on the machine where this disk actually resides.

## 4. Science & Pipelines

Within JIVE, ParselTongue is already being used for complex data reduction tasks. Figure 2 shows methanol masers in Cep. A at 6.7 GHz. The data

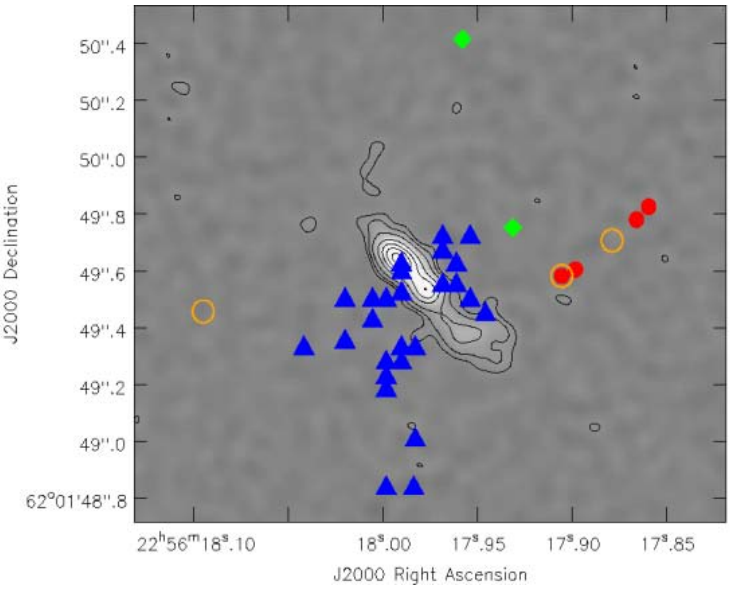


Figure 2. The source HW2 in star forming region Cep A. The background represents the 22 GHz continuum and the triangles are the locations of $H_2O$ masers. The filled circles indicate the 6.7 GHz methanol masers positions, the open circles the 12 GHz methanol masers and the diamonds the methanol maser emission at 107 GHz. (van Langevelde & Philips, in preparation).

was correlated in wide field astrometric mode at the EVN correlator at JIVE. ParselTongue scripts were used to process the 320 GB data set.

ParselTongue's ability to write AIPS extension tables has also been exercised successfully in applying calibration obtained from data correlated in wide band mode to narrow band spectral line data.

At JIVE we are also working on converting the EVN pipeline to ParselTongue. The current prototype is already using some of the possibilities that Python has to offer, for example to access the EVN archive remotely. This is something that is simply impossible from an AIPS run file.

## 5. Future development

Development of ParselTongue is currently focused on reliability and usability, but there are a few ideas for adding new functionality such as support for other task-based data reduction packages (MIRIAD, aips++, etc.) and an interface to the AIPS TV for interactive use. There are plans to add a work-flow manager to create and control ParselTongue scripts. ParselTongue could also form the basis for a Graphical User Interface to AIPS.