How Python Slithered Into Astronomy
Perry Greenfield
Space Telescope Science Institute
Outline

• What we do--Hubble Space Telescope and all that
• Where we were in 1998 (regarding scientific software)
• How Python rescued us
• Where we are going
• Random observations about scientific programming in astronomy
Space Telescope Science Institute

- Responsible for science operations of the Hubble Space Telescope (HST)
- And eventually the next large space Telescope: James Webb Space Telescope (JWST)
- Located in Baltimore on Johns Hopkins University campus (but not part of JHU)
- About 500 people
- Our group does the science software for the telescopes
HST

• 2.4 meter mirror
• Small compared to large ground telescopes
• The advantages of space:
  – No atmosphere to:
    • Blur images
    • Block UV and infrared wavelengths
    • Scatter light into the background
• Launched in 1990
  – Serious optical error--fixed by servicing mission
• 5 servicing missions by space shuttle
  – To bring new instruments
  – Replace failed or degrading hardware
HST Mirror
HST Launch

1990 April 24
M100, a Spiral Galaxy in the Virgo Cluster

Hubble Space Telescope / Wide Field Planetary Camera

WFPC-2 : Wide Field Camera

WFPC-1 : Wide Field Camera

WFPC-1 : deconvolved

Palomar 5m on a good night
M100 Galactic Nucleus

Hubble Space Telescope
Wide Field Planetary Camera 2

Wide Field Planetary Camera 1
Wide Field Planetary Camera 2
Jupiter Aurora
NASA and J. Clarke (University of Michigan) • STScI-PRC00-38
The Mice • Interacting Galaxies NGC 4676

NASA, H. Ford (JHU), G. Illingworth (UCSC/LO), M. Clampin (STScI), G. Hartig (STScI), the ACS Science Team and ESA • STScI-PRC02-11d
Jet from core of M87
Role of our science software

- Calibration: undo instrumental imperfections
  - Mostly automatically
- Combine and reduce data
- Tools to analyze data
- Tools to simulate observations
- And help observers plan observations
Our software history

• Pre-launch software written for IRAF system
• Image Reduction and Analysis Facility.
• Developed at National Optical Astronomy Observatory starting around 1980.
• Designed to be portable
  – And very successful at that
• Widely used by astronomers
  – And still is
• But…
IRAF

• All or nothing (like Java)
• Portability achieved through use of standard “Virtual Operating System” API
• And its own development language “SPP”
  – A hybrid between Fortran and C
  – Preprocessed into Fortran 66
  – Has annoying limitations
  – Used nowhere else
• And its own scripting language “CL”
  – Has really annoying limitations and behavior
  – Also used nowhere else
IRAF reconsidered

By 1995 serious concerns regarding choice of IRAF as basis of all software

- Developers see non-standard languages as bad for career
- Insufficient NOAO resources to keep VOS API modernized
  - NOAO unwilling to accept outside changes to system
- IRAF is the 1980 software world frozen in time
- Inability to link to other libraries
  - All must be re-implemented for IRAF
Attempt to evolve IRAF

• In 1995 STScI decides to write new calibration pipelines in C and use standard data format (“FITS”)
  – Writes new C interface for VOS
  – Writes new library to access FITS files from IRAF

• Backs effort to develop “OpenIRAF”
  – Incorporating CVOS into IRAF
  – Ability to link to external libraries
  – Ability to run IRAF tasks at host level

• OpenIRAF fails
Escape from IRAF

What to do?

• Fork IRAF?
  – Big political battle would result, unhappy user community.

• Rewrite software for a new system?
  – No support for rewriting all software in a new system
    (a very big effort, > 1 million lines of code)

• Something more subtle needed
  – Need access to old software but allow new software too
Solution: Alternate User Interface

- If users can run old and new through a familiar user interface we can hide that there are two different systems underneath
- No need to replace all old tasks right away
- Allows gradual transition
- Replace IRAF Command Language with our own version of Command Language
- User sees mostly the same interface
- In effect, we replace the IRAF scripting language
PyRAF is born (1998)

- Use Python as new scripting language
- Nontrivial to implement
  - Python must communicate with IRAF subprocesses through complex protocol.
  - Python must maintain environment that IRAF processes expect to see
  - Python must implement graphics subsystem to render IRAF plots (special metacode)
  - Python must emulate the weird CL language itself!
- Yet, Python made it doable!
2000: Goals Expanded

• Python was much more powerful than we expected.
• New desire to write applications themselves in Python.
  – IDL gave us faith it was possible
• But we needed new or improved libraries:
  – Array handling (Numeric not good enough)
  – Plotting (no good package available)
  – Module to read and write FITS files
• So we started on all 3
Python Arrays

- Numeric had a number of shortcomings:
  - Inefficient for large arrays
  - No support for memory mapping
  - Inconsistent/inefficient type handling
  - No support for records (structs)

- Rewrite appeared necessary, STScI began numarray
  - Followed Guido’s suggestion to do most in Python
    - At the time, no support for classes in C in Python
  - Fast for large arrays, slow for small
  - Hindered adoption by Numeric users

- Travis Oliphant redid in C resulting in numpy
  - Now good for large and small arrays!
Python 2-D Plotting

• Requirements:
  – Supports all platforms (Linux, Solaris, Mac, Windows)
  – GUI agnostic (must not be tied to one GUI)
  – Open Source
  – Image support
  – Support for publication quality

• Chaco effort started with enthought
  – Traits was initial byproduct
  – Ended up too complex for our needs
  – Trading interactive GUI features for simplicity

• Looked elsewhere, found John Hunter/matplotlib
  – Helped add what we needed.
PyFITS

- Standard data format in astronomy
- Starting point was Paul Barrett’s PyFITS module
- Adapted to use numarray, then numpy
- Needed record arrays to support table format
Example 1: Multidrizzle

- Need to combine multiple exposures taken at different pointings
- And reject cosmic rays simultaneously
- And deal with serious distortion
  - Simple shift and add of images won’t work
- And fractional pixel offsets
- Errors of 0.003 pixels in registration are noticeable.
What are we starting with?

Wide Field Channel of ACS consists of two CCDs that don’t overlap.

Image Size
- $2 \times 4096 \times 2048$
Final Mosaic

- The image is oriented such that North is up.
Example 2: JWST metrology

- JWST is the next large space telescope
- To be launched in 2011 2013 2014 2015 2018?
- 6.5 meter telescope made of 18 hexagonal segments
- Infra-Red telescope
- Thus very cold (warm mirrors glow in the infra-red): 40º Kelvin
- Will be launched to L2 point (~1.6 Million km away)
- Mirror structure must be very stable
  - Needed tests to ensure that
Conclusions about Python’s Role at STScI

• Python was essential for our escape from IRAF
• Most new science applications at STScI written in Python now
• Python has made us much more productive and tackle problems we never would have before
Current STScI Focus

- Installation is biggest obstacle for our user community
- Lots of dependencies to install
  - Many inconsistent installation schemes used
  - Lots of things that can go wrong
- Working on a core release that supplies binaries
  - Using SAGE-like approach
  - Can update components or add new ones (at some risk)
- Will allow us to start using Scipy and Mayavi
  - Installation is the current barrier
Current STScI Focus (continued)

• Start replacing IRAF applications
  – To date we have focused more on complimentary applications
• Build more basic astronomy libraries
• Encourage an Open Source community
  – Astronomy has not been good at his
  – But reason to hope (more on this later)
What about the rest of Astronomy?

• Does Python have a greater role?
  – If so, how?
Progress so far

• PyRAF helped trigger increasing adoption of Python as the standard astronomy scripting language.
• Used by:
  – Chandra/Smithsonian Astrophysical Observatory
  – National Radio Astronomy Observatory
  – Gemini Observatory
  – European Space Observatory
  – And a number of others.
• Many contributed packages by individuals now.
Progress (cont.)

• Use as an applications language more limited
  – Partly conservatism: must use “real” language like C, C++, Fortran or Java
  – Partly lack of astronomy-specific libraries and tools (in comparison to IDL and IRAF)
• But now seeing increasing use
• Competes now with widely used IDL
  – Better at arrays, plotting, FITS manipulation
  – Worse at richness of astronomy tools
• Younger astronomers transitioning to Python instead
Why is Python good for Astronomy?

- Python is special because:
  - Its interactivity is essential for science
    - And programming too I argue
  - It is accessible to most scientists and engineers
    - Who don’t want to learn C, C++, or Java
    - Many think knowing Excel macros makes them programmers
    - So are matlab and IDL, though those are often unsuited for non-research code.
  - Its power for programmers
  - Tools and algorithms can be shared between the two groups
    - Communication is usually a problem
  - Few languages fit this role
  - But this all misses something important
Why has Python been successful?

• Technical superiority?
  – Sure, that was a necessary condition
  – But not a sufficient one
## Sociology: Astronomy vs Python Culture

<table>
<thead>
<tr>
<th>Traditional Astronomy Software</th>
<th>Python (and Open Source…)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Possessive/non-sharing</td>
<td>Cooperative/sharing</td>
</tr>
<tr>
<td>Fragmented but overlapping efforts</td>
<td>Many common projects</td>
</tr>
<tr>
<td>Top-down planning</td>
<td>Loosely organized</td>
</tr>
<tr>
<td>Design is committee-oriented</td>
<td>Design by the “doers”</td>
</tr>
<tr>
<td>Endless analysis and argument</td>
<td>Action-oriented/experimentation</td>
</tr>
<tr>
<td>Choose or unwilling to discard losing technologies</td>
<td>Good at rejecting or replacing obsolete technologies</td>
</tr>
<tr>
<td>No leader to resolve conflicts</td>
<td>BDFL resolves conflicts</td>
</tr>
<tr>
<td>Not fad resistant</td>
<td>Fad resistant</td>
</tr>
<tr>
<td>Backward compatibility is an absolute</td>
<td>Backward compatibility is important, but not an absolute</td>
</tr>
<tr>
<td>Overambitious goals…rarely met</td>
<td>Pragmatic</td>
</tr>
<tr>
<td>Progress is glacial</td>
<td>Very productive, high quality results</td>
</tr>
</tbody>
</table>
Possible Reasons

• Nature of funding?
• Nature of problems?
• Lack of humor, theme, or enthusiasm
• Academic culture?
• Lack of Guidos?
• Lack of unifying motivation?
Critical mass reached for integrating software efforts?

• Now attempting to unify existing astronomy packages to:
  • Reduce overlaps
  • Improve consistency of interfaces (user and software)
Credits

• **PyRAF:**
  – Rick White, Chris Sontag, Warren Hack, Todd Miller, Phil Hodge

• **PyFITS:**
  – Paul Barrett, Erik Bray, J C Hsu, James Taylor, Chris Hanley, Mike Droettboom

• **Matplotlib:**
  – Mike Droettboom, Paul Barrett, Nadia Dencheva, Todd Miller

• **Multidrizzle:**

• **JWST metrology:**
  – Warren Hack, Ivo Busko, Todd Miller

• **Numarray:**
  – Todd Miller

• **Exposure Time Calculators**
  – Vicki Laidler, Ivo Busko, Megan Sosey, Chris Hanley, Mark Sienkiewicz, Kevin Lindsay, Todd Miller

• **And many others on many other projects…**