The Python Interface to Antelope

New Mexico Tech
August 25, 2009

Dr. Kent Lindquist
Lindquist Consulting, Inc.
Acknowledgments

- PASSCAL authors who shared their beginnings
- Bob Busby and IRIS
- BRTT
- Frank Vernon, Rob Newman, Alex Clemesha (Array Network Facility)
- Ole Nielsen, Duncan Gray, Nariman Habili, Phil Cummins, Spiro Spiliopoulos (Geoscience Australia)
Antelope Python Interface

Python: Object-oriented scripting language

- http://www.python.org
- Dynamic
- Powerful
- Extensible
- Fast
Antelope Programming

Create your own:

- Acquisition Programs
- Real-time Processes
- Utility programs
- Database applications
- Scientific projects
- GUI Interfaces
- Etc…

- BRTT Interfaces
  - C
  - Perl
  - TCL/Tk
  - Command-line

- Contributed Interfaces
  - Matlab
  - PHP
  - Java
“About Python”

- Scripting language
- Created by Guido van Rossum, 1990’s
- [http://www.python.org/about](http://www.python.org/about):
  - Very clear, readable syntax
  - Strong introspection capabilities
  - Intuitive object orientation
  - Natural expression of procedural code
  - Full modularity, supporting hierarchical packages
  - Exception-based error handling
  - Very high level dynamic data types
  - Extensive standard libraries and third-party modules for virtually every task
  - Extensions and modules easily written in C, C++ (or Java for Jython, or .NET languages for IronPython)
  - Embeddable within applications as a scripting interface
Why another Interface?

Specific problem: Web-based waveform plotting

- EarthScope Array Network Facility support (esp. field personnel)
- Responsive web GUI
- Interactive
- Flexible and Extensible
- Fastest robust path: accessibility of database data in Python
  - Powerful web capabilities
  - Matplotlib
  - Personnel experience
Model: dbpick emulation
Web-based Interactive Waveform Display (Python)

development prototype courtesy
Alex Clemesha, UCSD

August 25, 2009
Lindquist Consulting, Inc.
Waveform Explorer
single-station focus

development prototype courtesy
Alex Clemesha, UCSD
Waveform Explorer
interactive zoom

development prototype courtesy
Alex Clemesha, UCSD

August 25, 2009  Lindquist Consulting, Inc.
Waveform server current status

- dbwfserver executable, in Antelope contributed-code repository
- Project now under leadership of Rob Newman (and Alex Clemesha, Juan Reyes) of UCSD group
Python Interface Characteristics

- Fairly uniform ‘look and feel’ across languages for Antelope APIs
  - Database pointers
  - Familiar functions
- Behaviors natural to each language
  - E.g. perl arrays / hashes, Matlab matrices
- Other interfaces implement subset of full C API
- Easy to do something quick; hard to do robustly, thoroughly, and faithful to language and API without glitches
Approach

- **Pyrex**
  - Requires a separate ‘pyrex’ compiler

- **SWIG (Simplified Wrapper and Interface Generator)**
  - Requires external ‘swig’ tool to process interface files
  - Creates code hard to hand-tune

- **Ctypes**
  - Forces user to understand lots of the C interface

- **“The hard way”**
  - Properly balance C / Script interface boundary
  - Exact features of Antelope API (C)
  - “Look and Feel” of Python (Python wrappers)
First Efforts

SAGE

- http://www.sagemath.org
- Open-source Mathematics Software System
- Built around a python interpreter
- Large software collection, includes many Python packages esp. Matplotlib
SAGE examples

Algebra

```
sage: M = IntegerModRing(7)
sage: M(2) + M(8)
3
sage: M.list()
[0, 1, 2, 3, 4, 5, 6]

sage: A.<a,b,c> = AbelianGroup([2,2,3]); A
Multiplicative Abelian Group isomorphic to C2 x C2 x C3
sage: A.order()
12
sage: A.list()
[1, c, c^2, b, b*c, b*c^2, a, a*c, a*c^2, a*b, a*b*c, a*b*c^2]
sage: c^5*b*a^4*c
b
```

Sage is built around powerful new feature to describe algebras; are algebraic objects like groups, rings and fields.

On the left side you can see some examples how to construct and use them. The first one picks two integers out of the ring of integers modulo 7. The list() method lists all elements of that ring. Similar, the second example constructs an abelian group and assigns its generators to the letters a, b and c.
Python Interface Architecture

- Datascope Database
  - datascope.so
  - stock.so
  - orb.so
  - Pkt.so
  - BRTT Shared-Object libraries
  - Python C Extensions
  - Python Script Wrappers

- Web Applications
- Python Programs
- Embedded Python Unit-Tests

Provide Native Antelope APIs
Preserve Exact API Features
Map to Python Idioms; Implement Object-Oriented Layer
Switching to Python

- Design class hierarchies
- Code indentation has syntactic meaning
- Quit putting semicolons everywhere
Datascope Database Pointers

- db.database
- db.record
- db.table
- db.field
Python Database pointers

- Full-fledged Python ‘Type’ (sub-class of list) and Object
- Addressable attribute fields
  - `db.record`
  - Familiar to Matlab and C coders
- Implements sequence abstraction, subclasses Python lists
  - `db[3]`
  - Familiar to Perl coders
- Implements Python dictionary access
  - `db['record']`
  - Familiar to Python coders
Example function call: dbsubset of origin table

C:
- `sprintf( expr, “orid == 645” );`
- `db = dbsubset( db, expr, 0 );`

Perl:
- `@db = dbsubset( @db, “orid == 645” );`

TCL/Tk:
- `set db [dbsubset $db ‘orid == 645’]`

Shell:
- `dbsubset demo.origin ‘orid == 645’`

Matlab:
- `db = dbsubset( db, ‘orid == 645’ );`

PHP:
- `$db = dbsubset( $db, ‘orid == 645’ );`
Python Method calls

- **Procedural (class) methods**
  - `dbsubset(db, 'orid == 645')`

- **OO (instance) methods**
  - `db.subset('orid == 645')`

- **Basic calls with familiar arguments**
  - `datascope.dblookup(db,'', 'origin', '', '')`

- **Enhanced calls with Python idioms**
  - `dblookup(db, table='origin')`
Other advantages

- Lots of people use Python
- Scientific, Math, Computer-science and Graphics capabilities
- Object-oriented programming accessibility
  - Complementary to C++
  - Antelope object support in Python should surpass that in PHP, Perl
- unit testing capability
- It’s fun
Pforbstat connectivity monitoring
Pforbstat network overview
Directed-graph analysis of data-flow: ‘orbtogo’
Current status

- Datascope interface solidified
  - Includes trace-library commands
  - Includes response-file support
- Libstock interface
  - Basic time handling
  - Parameter files
- Orb interface just finished
  - All normal orb interactions
  - Packet unstuffing through separate Pkt module
- 155 unit tests
  - Python unittest module
  - Also serve as examples of each command in use
Future Directions

- Newly established interface
- Watch for problems
- Watch for opportunities