Antelope and Python

Kent Lindquist

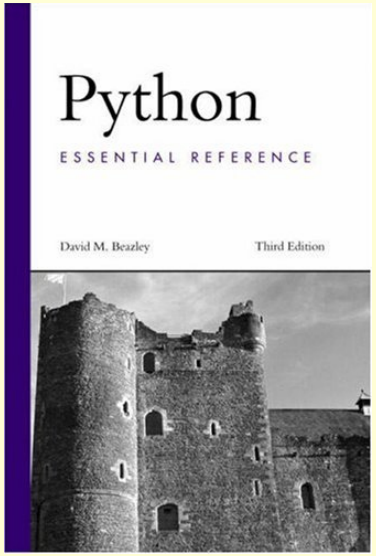
June, 2013

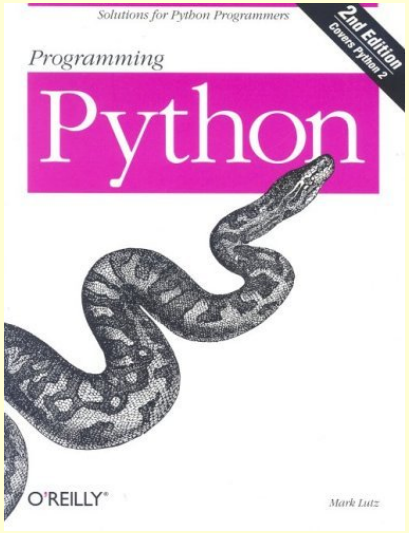
Brisbane, Australia AUG

BRTT

Python

- Python: Object-oriented scripting language
  - http://www.python.org
  - Dynamic
  - Powerful
  - Extensible
  - Fast





BRTT

About Python

- 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

Python in Antelope: History I

- Initial impetus: PASSCAL Instrument Center
  - Some pieces; Not a generic interface
- 2007: First open-source version, IRIS/ANF
  - Datascope; waveform plotting, orbtopo
  - Good proof-of-concept; lots of routines missing
  - Advice from Alex Clemesha, Rob Newman
- 2008: GA Consulting on Python
  - Ole Nielsen, Nariman Habili, Phil Cummins, Spiro Spiliopoulos, Michael Potter
  - Thin C layer with Python intelligence in script
  - Better architecture; warts and missing pieces

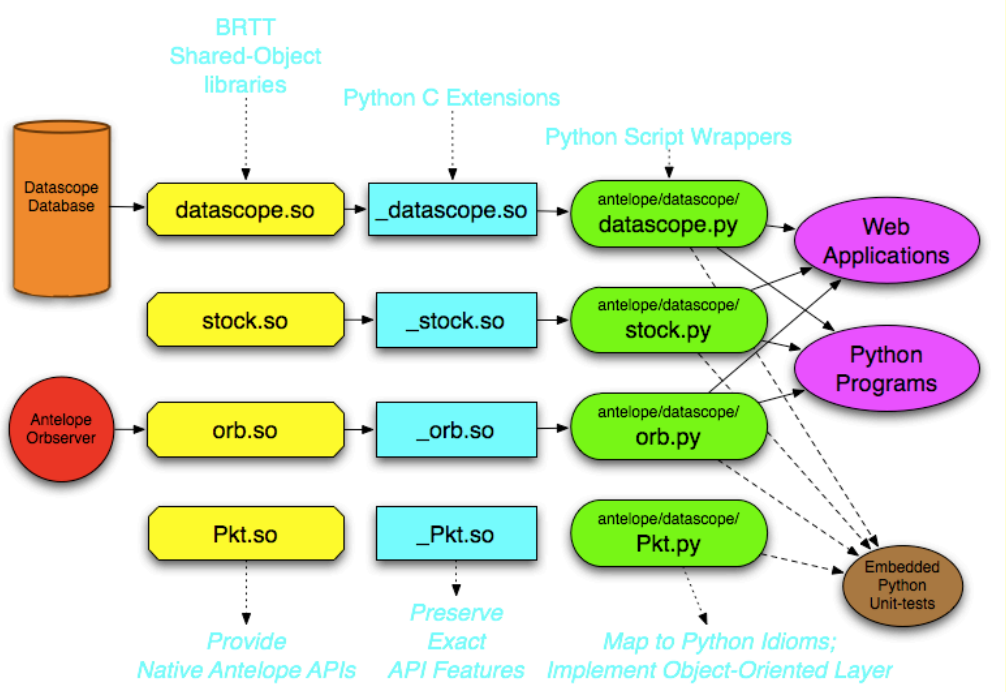
Python in Antelope: History II

- 2009: Added python orb, Pkt functions for UCSD
  - Experiment with AMQP for OOI
  - Filling out interfaces
  - open-source and integration issues
  - Discussions of heavy rewrite / expansion through GA
- 2010-2011: pre-release *Oryx*
  - Rtwebserver, rtcache
  - Headed towards Lindquist Consulting, Inc. Product
  - Never materialized as independent product: KL->BRTT

Python in Antelope: History III

- 2012: BRTT, first commercial version.
  - Python interpreter shipped with Antelope
  - raw, scripted layers separate
  - Docs; functional basic toolkit
  - Peregrine
  - Solid raw layer; glitches in scripted layer, divergent open-source developments
- Beg. 2013: Script-layer rewrite by Jeff Laughlin, Laughlin Consulting
  - Pkt, stock, orb, brttpkt, elog
  - In Antelope 5.3
- Summer 2013:
  - More Jeff Laughlin rewrites: Datascope, coords
  - Advanced Tk utilities, buplot

Python Interface Structure



BRTT

Multiple Layers

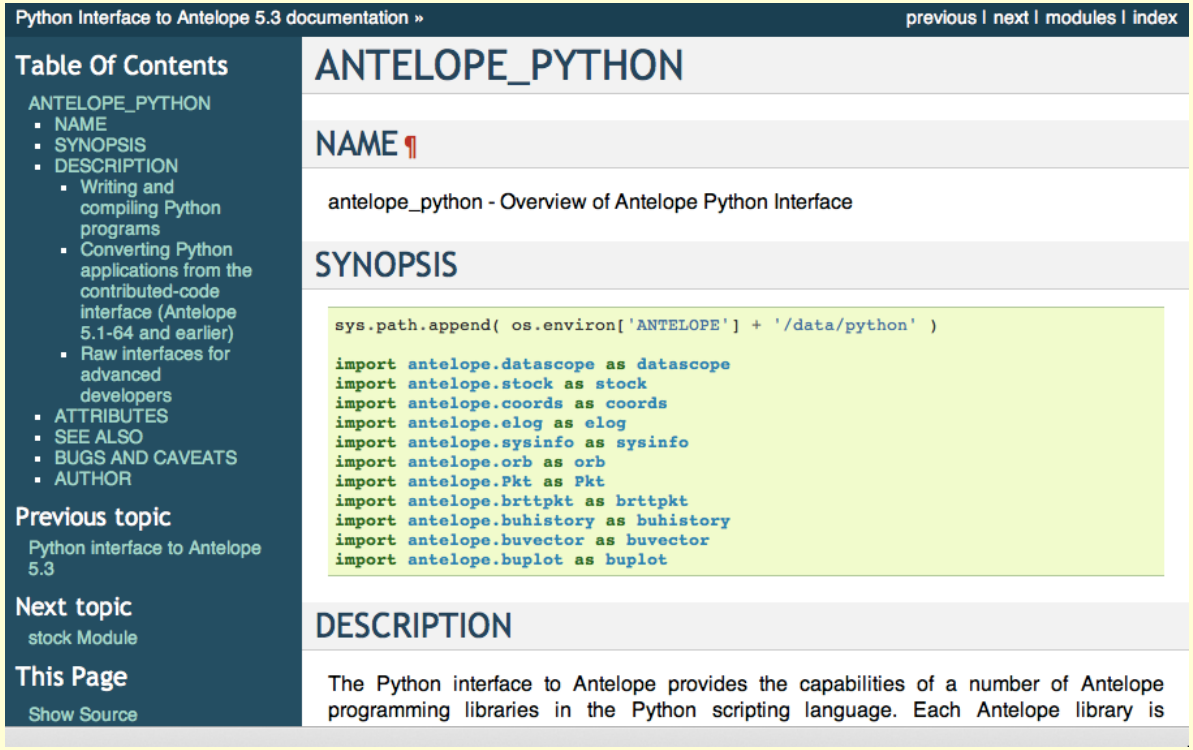
- Raw layer
  - _function naming convention: not for general use
  - Slavish adherence to C return values and structure
  - No Python intelligence
- Scripted layer
  - Intended for general user
  - On top of raw layer
  - Implements the ‘feel’ of Python

Requirements for project pyext:update;1
General goals:

- State-of-the-art Python interface for Antelope that hides C ugliness from Python programmer
- Appropriate object structure and behaviors
- Appropriate exception hierarchy and behavior
- Seamlessly handle memory management issues
- Seamlessly hide C-structure interaction, copying, passing, allocation/ deallocation
- Succeeds at adoption by Python / Antelope community (inspires app development, not interface rewrites)
- Result must be straightforward to maintain and extend by BRTT (i.e. passes BRTT acceptance)
- Provides demonstration template, model for how to write wrappers for remaining Antelope libraries

Hyperlinked Sphinx Docs

- file:///opt/antelope/5.3/html/antelope_python_overview.html



Python Interface to Antelope 5.3 documentation »

previous | next | modules | index

Table Of Contents

ANTELOPE_PYTHON

NAME ¶

antelope_python - Overview of Antelope Python Interface

SYNOPSIS

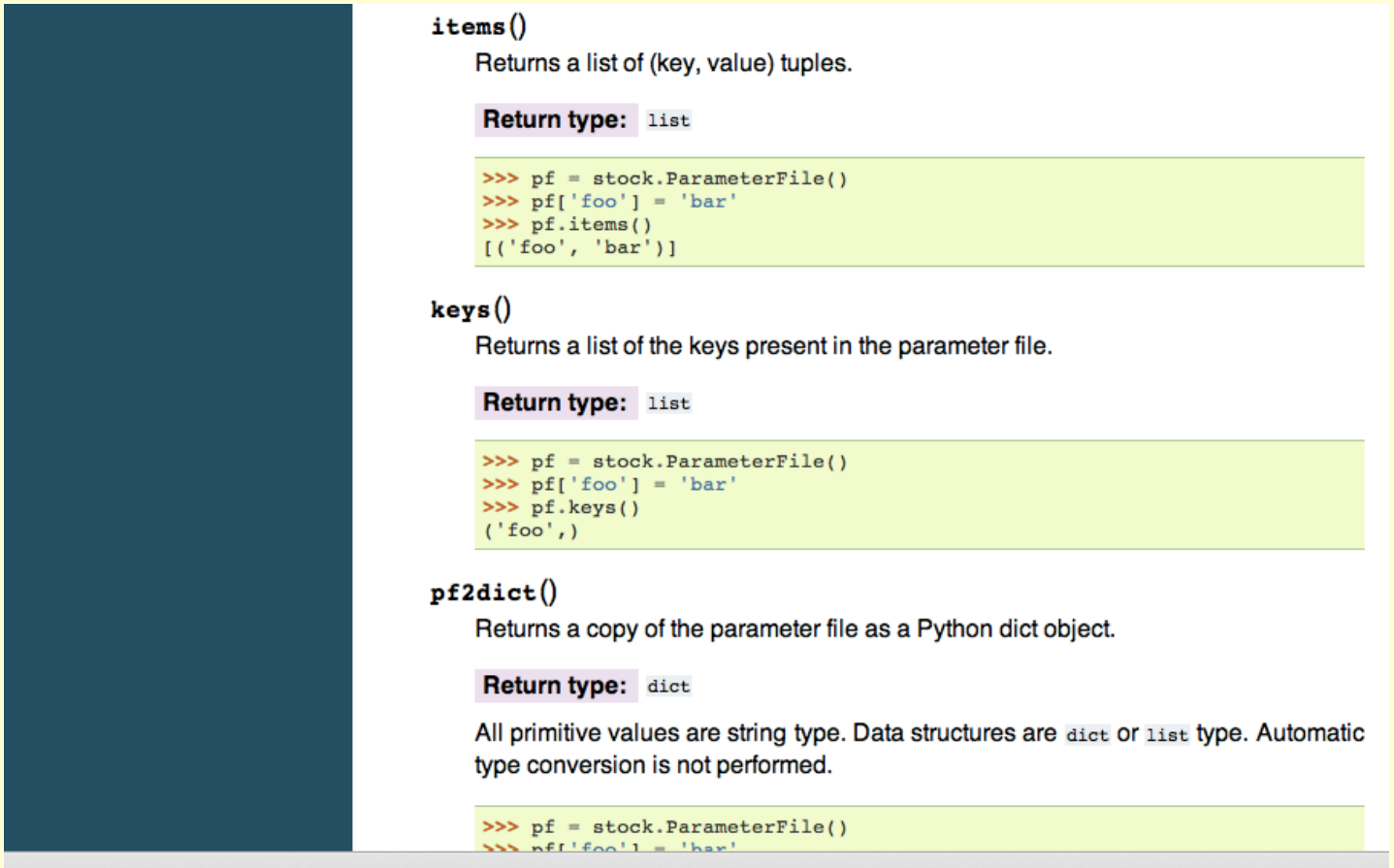
```
sys.path.append( os.environ['ANTELOPE'] + '/data/python' )

import antelope.datascope as datascope
import antelope.stock as stock
import antelope.coords as coords
import antelope.elog as elog
import antelope.sysinfo as sysinfo
import antelope.orb as orb
import antelope.Pkt as Pkt
import antelope.brttpkt as brttpkt
import antelope.buhistory as buhistory
import antelope.buvector as buvector
import antelope.buplot as buplot
```

DESCRIPTION

The Python interface to Antelope provides the capabilities of a number of Antelope programming libraries in the Python scripting language. Each Antelope library is

Hyperlinked Sphinx Docs



items()

Returns a list of (key, value) tuples.

Return type: `list`

```
>>> pf = stock.ParameterFile()
>>> pf['foo'] = 'bar'
>>> pf.items()
[('foo', 'bar')]
```

keys()

Returns a list of the keys present in the parameter file.

Return type: `list`

```
>>> pf = stock.ParameterFile()
>>> pf['foo'] = 'bar'
>>> pf.keys()
('foo',)
```

pf2dict()

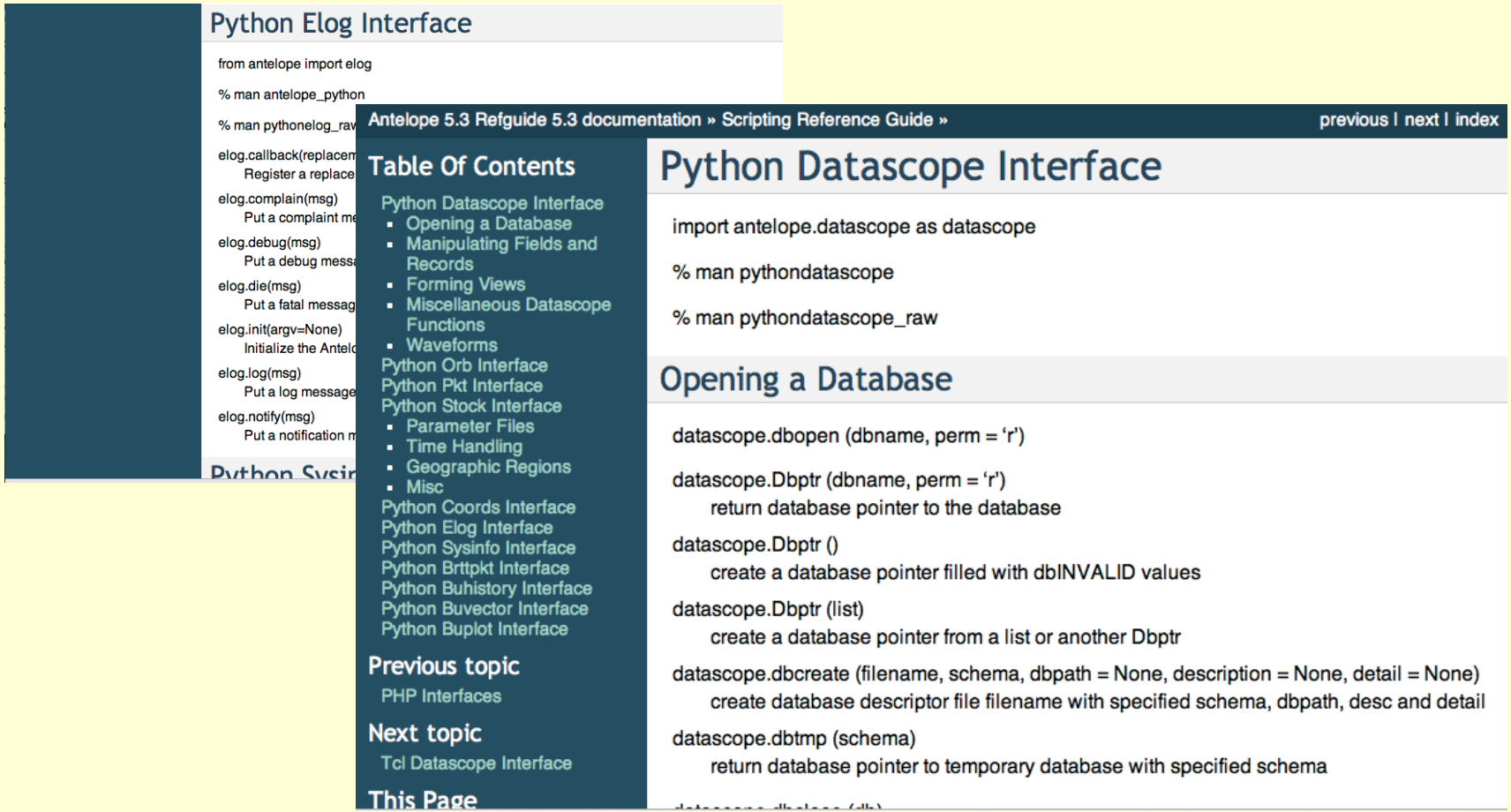
Returns a copy of the parameter file as a Python dict object.

Return type: `dict`

All primitive values are string type. Data structures are `dict` or `list` type. Automatic type conversion is not performed.

```
>>> pf = stock.ParameterFile()
```

Online Refguides



Python Elog Interface

from antelope import elog

% man antelope_python

% man pythonelog_raw

elog.callback(replacem
Register a replace

elog.complain(msg)
Put a complaint me

elog.debug(msg)
Put a debug messa

elog.die(msg)
Put a fatal messag

elog.init(argv=None)
Initialize the Antelo

elog.log(msg)
Put a log message

elog.notify(msg)
Put a notification m

Python Sysin

Antelope 5.3 Refguide 5.3 documentation » Scripting Reference Guide »

previous | next | index

Table Of Contents

Previous topic

PHP Interfaces

Next topic

Tcl Datascope Interface

This Page

Python Datascope Interface

import antelope.datascope as datascope

% man pythondatascope

% man pythondatascope_raw

Opening a Database

datascope.dbopen (dbname, perm = 'r')

datascope.Dbptr (dbname, perm = 'r')
return database pointer to the database

datascope.Dbptr ()
create a database pointer filled with dbINVALID values

datascope.Dbptr (list)
create a database pointer from a list or another Dbptr

datascope.dbcreate (filename, schema, dbpath = None, description = None, detail = None)
create database descriptor file filename with specified schema, dbpath, desc and detail

datascope.dbtmp (schema)
return database pointer to temporary database with specified schema

BRTT

Recommendations

- Old recommendation:
  - Use our scripted layer …
  - or write your own on top of the raw interface
  - Divergent interfaces threatening to take value out of interface to community
  - Messes in contrib
- New recommendation:
  - Use our scripted layer …
  - Or tell us what's wrong with it so we can fix
  - Leverage community resources

Thank You

- Feedback welcome