Atacama
Large
Millimeter
Array

Python Coding Standards

COMP-70.20.00.00-002-A-STD

Version: A

Status: *Released*

2003-07-21

Prepared By:		
Name(s) and Signature(s)	**Organization**	**Date**
Jeffrey A. Uphoff	NRAO	2003-07-21
Approved By:		
Name and Signature	**Organization**	**Date**
B.E. Glendenning G. Raffi	NRAO ESO	2003-09-12 2003-09-12
Released By:		
Name and Signature	**Organization**	**Date**
B.E. Glendenning G. Raffi	NRAO ESO	2003-09-12 2003-09-12

Change Record

Version	Date	Affected Section(s)	Change Request #	Reason/Initiation/Remarks

Table of Contents

1 Description

1.1 Purpose

This document, primarily through reference to established Python "community" documents, specifies standards and best practices for coding and documenting Python code within the ALMA software-development project.

1.2 Scope

All Python code checked into the ALMA software-development project's permanent CVS repository *shall* adhere to the standards and best practices specified in this document and its references.

This document assumes a working knowledge of the Python programming language.

2 Related Documents and Drawings

2.1 References

1. **Style Guide for Python Code**, Guido van Rossum and Barry Warsaw, http://www.python.org/peps/pep-0008.html
2. **Docstring Conventions**, David Goodger and Guido van Rossum, http://www.python.org/peps/pep-0257.html
3. **Docstring Processing System Framework**, David Goodger, http://www.python.org/peps/pep-0256.html
4. **Python Library Reference**, §5.1: "pydoc—Documentation generator and online help system," http://www.python.org/dev/doc/devel/lib/module-pydoc.html

2.2 Abbreviations and Acronyms

PEP: Python Enhancement Proposal, a standard for documentation within the Python community.

2.3 Glossary

docstring: a string literal that occurs as the first statement in a module, function, class, or method definition.

pydoc: extracts and displays docstrings from within the Python interactive interpreter, from the shell command line, and from a GUI window into a web browser

3 Code

3.1 Basic Standard

Except as noted in §3.2, code *shall* conform to PEP 8, the **Style Guide for Python Code** [1].

[1] addresses the following major subject areas: code layout and white-space, imports, comments, documentation strings, version bookkeeping, naming conventions, and programming recommendations.

3.2 Deviations from, and extensions to, the Basic Standard

3.2.1 Module Names

Module names *shall* use the "CapWords" naming convention.

[1] specifies either CapWords or all-lowercase. The general consensus within ALMA has been to use CapWords, thus it is so specified.

Module names map to file names, thus the module PositionsReader can be found in the file PositionsReader.py.

3.2.2 Package Names

Package names *shall* use the "CapWords" naming convention.

[1] specifies: "Python packages generally have a short all lowercase name." The ALMA convention of CapWords deviates from this generally accepted practice, though it does not conflict with [1].

Packages are considered an advanced tool in Python. In short, packages allow you to import modules using directory paths.

3.2.3 Method Names

Method names *shall* use the "mixedCase" naming convention.

[1] specifies: "Use lowercase for methods accessed by other classes or functions that are part of the implementation of an object type." The ALMA convention of

mixedCase conflicts with the letter of [1], though not with the spirit. The general consensus within ALMA has been to use mixedCase, thus it is so specified.

3.2.4 Comments

In addition to a blank line separating methods/functions, non-one-line functions *may* be separated by a line of –‘s, e.g.
--
to clearly delineate the functions.
(Please note the presence of a single space betwixt the leading # and the first -.)

This is an extension to [1], as [1] merely specifies conventions for the use of blank lines.

4 Documentation

4.1 Docstrings

Docstrings *shall* conform to PEP 257, the **Docstring Conventions** [2].

Additional docstrings information is available in PEP 256, **Docstring Processing System Framework** [3].

Scripts that can be run as standalone programs should provide a usage description in their docstrings.

4.2 Documentation Generator

The mechanism for generating Python documentation *shall* be pydoc.

Information on pydoc is available in PEP 256, **Docstring Processing System Framework** [3] and in **Python Library Reference**, §5.1 [4].