Python Coding Guidelines

Devi Prasad, Chandan

2013-11-20

Contents

1 Purpose of the document

This document provides the coding guidelines for Python programming language based on http://www.python.org/dev/peps/pep-0008/.

2 Modification Log

Rev	Description	Author	Date
0.01	Initial draft	Avinash	*2013-11-20 Wed*

3 Code lay-out

3.1 Indentation

- Use 4 spaces per indentation level (no tabs).

3.2 Maximum Line Length

- Limit all lines to a maximum of 79 characters.

3.3 Blank Lines

- Separate top-level function and class definitions with two blank lines.
- Method definitions inside a class are separated by a single blank line.

- Use blank lines in functions, sparingly, to indicate logical sections.

3.4 Imports

- Imports should usually be on separate lines
- Imports are always put at the top of the file, just after any module comments and docstrings, and before module globals and constants.
- Imports should be grouped in the following order:
    1. standard library imports
    2. related third party imports
    3. local application/library specific imports

You should put a blank line between each group of imports.

Put any relevant __all__ specification after the imports.

3.5 Whitespace in Expressions and Statements

- http://www.python.org/dev/peps/pep-0008/#id17 (stick to all of the points)

3.6 Comments

- State why and now how'

3.7 Documentation Strings

- Write docstrings for all public modules, functions, classes, and methods. Docstrings are not necessary for non-public methods, but you should have a comment that describes what the method does. This comment should appear after the def line.

4 Naming Conventions

4.1 Overriding Principle

- Names that are visible to the user as public parts of the API should follow conventions that reflect usage rather than implementation.

4.2 Descriptive: Naming Styles

- http://www.python.org/dev/peps/pep-0008/#descriptive-naming-styles

4.3 Package and Module Names

- Modules should have short, all-lowercase names. Underscores can be used in the module name if it improves readability. Python packages should also have short, all-lowercase names, although the use of underscores is discouraged.

4.4 Class Names

- Class names should normally use the CapWords convention.

4.5 Exception Names

- Because exceptions should be classes, the class naming convention applies here. However, you should use the suffix "Error" on your exception names (if the exception actually is an error).

4.6 Global Variable Names

- Modules that are designed for use via from M import * should use the __all__ mechanism to prevent exporting globals.

4.7 Function Names

- Function names should be lowercase, with words separated by underscores as necessary to improve readability.

4.8 Function and method arguments

- Always use self for the first argument to instance methods.
- Always use cls for the first argument to class methods.

4.9 Method Names and Instance Variables

- Use the function naming rules: lowercase with words separated by underscores as necessary to improve readability.
- Use one leading underscore only for non-public methods and instance variables.

4.10 Constants

- Constants are usually defined on a module level and written in all capital letters with underscores separating words. Examples include $\text{MAX}_{\text{OVERFLOW}}$ and TOTAL.

4.11 Designing for inheritance

- http://www.python.org/dev/peps/pep-0008/#id38 (read the complete section)

4.12 Public and internal interfaces

- Any backwards compatibility guarantees apply only to public interfaces. Accordingly, it is important that users be able to clearly distinguish between public and internal interfaces.

- To better support introspection, modules should explicitly declare the names in their public API using the __all__ attribute. Setting __all__ to an empty list indicates that the module has no public API.

- Even with __all__ set appropriately, internal interfaces (packages, modules, classes, functions, attributes or other names) should still be prefixed with a single leading underscore.

5 Programming Recommendations

- http://www.python.org/dev/peps/pep-0008/#id40