Python Coding Standards

Best Practices Workshop, March 25-26 2019, Hampton VA

Carlos Cruz
Jules Kouatchou
Brent Smith

NASA GSFC Code 606/610 (ASTG/GMAO)
Greenbelt, Maryland 20771

Goals

- We use the Python Enhancement Proposal 8 (PEP8) document to provide guidelines and best practices on how to write Python code.
- PEP8 addresses topics such as name conventions, code layout, indentation, comments, etc..
- The goal is to make the code more readable, maintainable and sharable.

Indentation - I

- Use 4 spaces per indentation level.

```python
# YES
def long_function_name(
        var_one, var_two, var_three,
        var_four):
    print(var_one)

# NO
def long_function_name(
    var_one, var_two, var_three,
    var_four):
    print(var_one)

```

Indentation - II

- The closing brace/bracket/parenthesis on multiline constructs may either line up under the first non-whitespace character of the last line of list

```
my_list = [
    1, 2, 3,
    4, 5, 6,
    ]
result = some_function_that_takes_arguments(
    'a', 'b', 'c',
    'd', 'e', 'f',
    )

```

Tabs or Spaces

- Spaces are the preferred indentation method.
- Tabs should be used solely to remain consistent with code that is already indented with tabs.

Maximum Line Length

- Limit all lines to a maximum of 79 characters.
- For docstrings or comments, the line length should be limited to 72 characters.

Blank Lines

- Surround top-level function and class definitions with two blank lines.
- Method definitions inside a class are surrounded by a single blank line.
- Use blank lines in functions, sparingly, to indicate logical sections.

Imports

- One import statement per line
- 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.

Module Level Dunder Names

Module level "dunders" (i.e. names with two leading and two trailing underscores) should be placed after the module docstring but before any import statements except from __future__ imports.

```
"""
    This is the example module.
    This module does stuff.
"""

from __future__ import barry_as_FLUFL

__all__ = ['a', 'b', 'c']
__version__ = '0.1'
__author__ = 'Cardinal Biggles'

import os
import sys
```

Comments

- Use complete sentences.
- Use two spaces after a sentence-ending period in multi-sentence comments, except after the final sentence.
- Each line of a block comment starts with a # and a single space

Inline Comments

- An inline comment is a comment on the same line as a statement.
- Inline comments should be separated by at least two spaces from the statement.
- They should start with a # and a single space

Documentation String

- Write docstrings for all public modules, functions, classes, and methods.
- Docstrings should have comments that describe what the method does.
- Then triple quote that ends a multiline docstring should be on a line by itself.

```
"""This is an example of docstring.

The ending triple quote should be on a line by itself.
"""
```

Descriptive: Naming Styles

- lowercase
- lower_case_with_underscores.
- UPPERCASE
- UPPER_CASE_WITH_UNDERSCORES
- CapitalizedWords
- mixedCase
- Capitalized_Words_With_Underscores

Packages and Modules 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.

Class Names

- Class names should normally use the CapWords convention.

Function and Variable Names

- Function names should be lowercase, with words separated by underscores as necessary to improve readability.
- Variable names follow the same convention as function names.

Function and Method Arguments

- Always use *self*for the first argument to instance methods.
- Always use *cls* for the first argument to class methods.
- If a function argument's name clashes with a reserved keyword, it is generally better to append a single trailing underscore rather than use an abbreviation or spelling corruption.
  *class_* is better than *clss*.

Constants

- Constants are usually defined on a module level and written in all capital letters with underscores separating words.

```
MINUTES_PER_HOUR  =  60
HOURS_PER_DAY     =  24
DAYS_PER_WEEK     =  7
MONTH_PER_YEAR    =  12
```

Return Statements

- Either all return statements in a function should return an expression, or none of them should.
- If any return statement returns an expression, any return statements where no value is returned should explicitly state this as *return None*, and an explicit return statement should be present at the end of the function.

```python
def bar(x):
    if x < 0:
        return None
    return math.sqrt(x)
```

Listing 1: YES

```python
def bar(x):
    if x < 0:
        return
    return math.sqrt(x)
```

Listing 2: NO

Sequences

- For strings, lists, tuples, use the fact that empty sequences are false.

```
Yes:  if not seq:
      if seq:

No:   if len(seq):
      if not len(seq):
```

Boolean Comparison

- Don't compare boolean values to True or False using ==.

```
Yes:    if greeting:
No:     if greeting == True:
Worse:  if greeting is True:
```