Python Coding Standards

PEP 8, sometimes spelled PEP8 or PEP-8, is a document that provides guidelines and best practices on how to write Python code. It was written in 2001 by Guido van Rossum, Barry Warsaw, and Nick Coghlan. The primary focus of PEP 8 is to improve the readability and consistency of Python code.

PEP stands for Python Enhancement Proposal, and there are several of them. A PEP is a document that describes new features proposed for Python and documents aspects of Python, like design and style, for the community.

Table of Contents

Naming Conventions

The table below outlines some of the common naming styles in Python code and when you should use them:

Type	Naming Convention	Examples
Function	Use a lowercase word or words. Separate words by underscores to improve readability.	function, my_function
Variable	Use a lowercase single letter, word, or words. Separate words with underscores to improve readability.	name, var, my_variable
Class	Start each word with a capital letter. Do not separate words with underscores. This style is called camel case.	Model, MyClass
Method	Use a lowercase word or words. Separate words with underscores to improve readability.	class_method, method
Constant	Use an uppercase single letter, word, or words. Separate words with underscores to improve readability.	CONSTANT, MY_CONSTANT, MY_LONG_CONSTANT
Module	Use a short, lowercase word or words. Separate words with underscores to improve readability.	module.py, my_module.py
Package	Use a short, lowercase word or words. Do not separate words with underscores.	package, mypackage

Note: Never use l, O, or I single letter names as these can be mistaken for 1 and 0, depending on typeface.
Example: N = 2 # This may look like you're trying to reassign 2 to zero

Code Layout

- Surround top-level functions and classes with two blank lines.
- Surround method definitions inside classes with a single blank line.
- Use blank lines sparingly inside functions to show clear steps

Indentation

- Use 4 consecutive spaces to indicate indentation.
- Prefer spaces over tabs

Indentation Following Line Breaks

- There are two styles of indentation you can use. The first of these is to align the indented block with the opening delimiter.
- An alternative style of indentation following a line break is a hanging indent. This is a typographical term meaning that every line but the first in a paragraph or statement is indented.

Where to Put the Closing Brace

PEP 8 provides two options for the position of the closing brace in implied line continuations:

- Line up the closing brace with the first non whitespace character of the previous line.
- Line up the closing brace with the first character of the line that starts the construct.

Maximum Line Length and Line Breaking

- PEP 8 suggests lines should be limited to 79 characters. This is because it allows you to have multiple files open next to one another, while also avoiding line wrapping. Python will assume line continuation if code is contained within parentheses, brackets, or braces.
- If it is impossible to use implied continuation, then you can use

backslashes to break lines instead.

Comments

- **Limit the line length of comments and docstrings to 72 characters**
- **Use complete sentences, starting with a capital letter.**
- **Make sure to update comments if you change your code.**

Block comments

- Indent block comments to the same level as the code they describe.
- Start each line with a # followed by a single space.
- Separate paragraphs by a line containing a single # .

Inline Comments

- Use inline comments sparingly.
- Write inline comments on the same line as the statement they refer to.
- Separate inline comments by two or more spaces from the statement.
- Start inline comments with a # and a single space, like block comments.
- Don't use them to explain the obvious.

Documentation Strings

- Surround docstrings with three double quotes on either side, as in """This is a docstring""".
- Write them for all public modules, functions, classes, and methods. Put the """ that ends a multi line docstring on a line by itself.
- For one-line docstrings, keep the """ on the same line.

Imports

- Should always come at the top of the Python script
- Separate libraries should be imported on separate lines

- Imports should be grouped in the following order:
    - Standard library imports
    - Related third party imports
    - Local application/library specific imports
- Include a blank line after each group of imports
- Can import multiple classes from the same module in a single line

Whitespace in Expressions and Statements

Whitespace Around Binary Operators

- Surround the following binary operators with a single space on either side:
    - Assignment operators (=, +=, -=, and so forth)
    - Comparisons (==, !=, >, <. >=, <=) and (is, is not, in, not in) Booleans (and, not, or) .
- When = is used to assign a default value to a function argument, do not surround it with spaces.

When to Avoid Adding Whitespace

- The most important place to avoid adding whitespace is at the end of a line. This is known as trailing whitespace.
- Immediately inside parentheses, brackets, or braces.
- Before a comma, semicolon, or colon.
- Before the open parenthesis that starts the argument list of a function call: Before the open bracket that starts an index or slice.
- Between a trailing comma and a closing parenthesis.
- To align assignment operators.
- immediately inside brackets, as well as before commas and colons.

Programming Recommendations

- Don't compare boolean values to True or False using the equivalence operator. Use the fact that empty sequences are falsy in if statements.
- Use is not rather than not ... is in if statements.
- Use **.startswith()** and **.endswith()** instead of slicing.

When to Ignore PEP 8

- If complying with PEP 8 would break compatibility with existing software.
- If code surrounding what you're working on is inconsistent with PEP 8.
- If code needs to remain compatible with older versions of Python