**PEP 8 -- Style Guide for Python Code**

[Code Lay-out](https://www.python.org/dev/peps/pep-0008/#id16)

[**Indentation**](https://www.python.org/dev/peps/pep-0008/#id17)

Use 4 spaces per indentation level.

[**Tabs or Spaces?**](https://www.python.org/dev/peps/pep-0008/#id18)

Spaces are the preferred indentation method.

Tabs should be used solely to remain consistent with code that is already indented with tabs.

Python 3 disallows mixing the use of tabs and spaces for indentation.

Python 2 code indented with a mixture of tabs and spaces should be converted to using spaces exclusively.

[**Maximum Line Length**](https://www.python.org/dev/peps/pep-0008/#id19)

Limit all lines to a maximum of 79 characters.

For flowing long blocks of text with fewer structural restrictions (docstrings or comments), the line length should be limited to 72 characters.

[**Should a Line Break Before or After a Binary Operator?**](https://www.python.org/dev/peps/pep-0008/#id20)

In Python code, it is permissible to break before or after a binary operator, as long as the convention is consistent locally.

Not Readable

# No: operators sit far away from their operands

income = (gross\_wages +

taxable\_interest +

(dividends - qualified\_dividends) -

ira\_deduction -

student\_loan\_interest)

More Readable

# Yes: easy to match operators with operands

income = (gross\_wages

+ taxable\_interest

+ (dividends - qualified\_dividends)

- ira\_deduction

- student\_loan\_interest)

[**Blank Lines**](https://www.python.org/dev/peps/pep-0008/#id21)

Surround top-level function and class definitions with two blank lines.

Method definitions inside a class are surrounded by a single blank line.

[**Imports**](https://www.python.org/dev/peps/pep-0008/#id23)

* Imports should usually be on separate lines:

Yes: import os

import sys

No: import sys, os

* It's okay to say this though:

from subprocess import Popen, PIPEWildcard

* imports (from <module> import \*) should be avoided, as they make it unclear which names are present in the namespace, confusing both readers and many automated tools.

[**Comments**](https://www.python.org/dev/peps/pep-0008/#id30)

Comments that contradict the code are worse than no comments. Always make a priority of keeping the comments up-to-date when the code changes!

Comments should be complete sentences. The first word should be capitalized, unless it is an identifier that begins with a lower case letter (never alter the case of identifiers!).

[**Inline Comments**](https://www.python.org/dev/peps/pep-0008/#id32)

Use inline comments sparingly.

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.

* Inline comments are unnecessary and in fact distracting if they state the obvious. Don't do this:

x = x + 1 # Increment x

* But sometimes, this is useful:

x = x + 1 # Compensate for border

[**Documentation Strings**](https://www.python.org/dev/peps/pep-0008/#id33) **(docstrings)**

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.

that ends a multiline docstring should be on a line by itself:

"""Return a foobang

Optional plotz says to frobnicate the bizbaz first.

"""

For one liner docstrings, please keep the closing """ on the same line.

[**Descriptive: Naming Styles**](https://www.python.org/dev/peps/pep-0008/#id36)

There are a lot of different naming styles. It helps to be able to recognize what naming style is being used, independently from what they are used for.

The following naming styles are commonly distinguished:

* b (single lowercase letter)
* B (single uppercase letter)
* lowercase
* lower\_case\_with\_underscores
* UPPERCASE
* UPPER\_CASE\_WITH\_UNDERSCORES
* CapitalizedWords (or CapWords, or CamelCase -- so named because of the bumpy look of its letters ). This is also sometimes known as StudlyCaps.
* mixedCase (differs from CapitalizedWords by initial lowercase character!)
* Capitalized\_Words\_With\_Underscores (ugly!)

[**Package and Module Names**](https://www.python.org/dev/peps/pep-0008/#id40)

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

[**Class Names**](https://www.python.org/dev/peps/pep-0008/#id41)

Class names should normally use the CapWords convention.

[**Function and Variable Names**](https://www.python.org/dev/peps/pep-0008/#id45)

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

Variable names follow the same convention as function names.

[**Constants**](https://www.python.org/dev/peps/pep-0008/#id48)

Constants are usually defined on a module level and written in all capital letters with underscores separating words. Examples include MAX\_OVERFLOW and TOTAL.