# Python Coding Style (selected items from https://peps.python.org/pep-0008/)

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

Use 4 spaces per indentation level.

[**Maximum Line Length**](https://peps.python.org/pep-0008/#maximum-line-length)

Limit all lines to a maximum of 72 characters.

[**Blank Lines**](https://peps.python.org/pep-0008/#blank-lines)

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

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

[**Whitespace in Expressions and Statements**](https://peps.python.org/pep-0008/#whitespace-in-expressions-and-statements)

[**Pet Peeves**](https://peps.python.org/pep-0008/#pet-peeves)

Avoid extraneous whitespace in the following situations:

Immediately inside parentheses, brackets or braces:

# Correct:

* spam(ham[1], {eggs: 2})

# Wrong:

* spam( ham[ 1 ], { eggs: 2 } )

Between a trailing comma and a following close parenthesis:

# Correct:

* foo = (0,)

# Wrong:

* bar = (0, )

Immediately before a comma, semicolon, or colon:

# Correct:

* if x == 4: print(x, y); x, y = y, x

# Wrong:

* if x == 4 : print(x , y) ; x , y = y , x

**Spacing:**

 Always surround these binary operators with a single space on either side: assignment (=), augmented assignment (+=, -= etc.), comparisons (==, <, >, !=, <>, <=, >=, in, not in, is, is not), Booleans (and, or, not).

 If operators with different priorities are used, consider adding whitespace around the operators with the lowest priority(ies). Use your own judgment; however, never use more than one space, and always have the same amount of whitespace on both sides of a binary operator:

# Correct:

i = i + 1

submitted += 1

x = x\*2 - 1

hypot2 = x\*x + y\*y

c = (a+b) \* (a-b)

#Wrong:

i=i+1

submitted +=1

x = x \* 2 - 1

hypot2 = x \* x + y \* y

c = (a + b) \* (a - b)

[**Inline Comments**](https://peps.python.org/pep-0008/#inline-comments)

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://peps.python.org/pep-0008/#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.
* [PEP 257](https://peps.python.org/pep-0257) describes good docstring conventions. Note that most importantly, the """ that ends a multiline docstring should be on a line by itself:

"""Return a foobang

Optional plotz says to frobnicate the bizbaz first.

"""

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

mixedCase is allowed only in contexts where that’s already the prevailing style (e.g. threading.py), to retain backwards compatibility.

#### [Constants](https://peps.python.org/pep-0008/#constants)

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

x = x + 1 # Compensate for border