Python Coding Style

Most languages can be written (or more concise, *formatted*) in different styles; some are more readable than others. Making it easy for others to read your code is always a good idea, and adopting a nice coding style helps tremendously for that.

For Python, “Python Enhancement Proposal 8”, also known as “PEP 8”, [see https://www.python.org/dev/peps/pep-0008/] has emerged as the style guide that most projects adhere to; it promotes a very readable and eye-pleasing coding style. Every Python developer should read it at some point; here are the most important points extracted for you:

- Use 4-space indentation, and no tabs.

  4 spaces are a good compromise between small indentation (allows greater nesting depth) and large indentation (easier to read). Tabs introduce confusion, and are best left out.

- Wrap lines so that they don’t exceed 79 characters.

  This helps users with small displays and makes it possible to have several code files side-by-side on larger displays.

- Use blank lines to separate functions and classes, and larger blocks of code inside functions.

- When possible, put comments on a line of their own.

- Use docstrings: string line(s) that occurs as the first statement in a module, function, class, or method definition, enclosed between """.

- Use spaces around operators and after commas, but not directly inside bracketing constructs. Example: a = f(1, 2) + g(3, 4).

- Name your classes and functions consistently; the convention is to use

  - CamelCase for classes, and
  - lower_case_with_underscores for functions and methods.

  Always use “self” as the name for the first method argument.
  For more on classes and methods, see “A First Look at Classes” at
  https://docs.python.org/release/2.6.8/tutorial/classes.html#tut-firstclasses

- Don’t use fancy encodings if your code is meant to be used in international environments. Plain ASCII works best in any case.