Google Python Style Guide

Revision 2.59

Amit Patel Antoine Picard Eugene Jhong Jeremy Hylton Matt Smart Mike Shields

Each style point has a summary for which additional information is available by toggling the accompanying arrow button that looks this way:

You may toggle all summaries with the big arrow button:

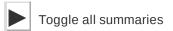


Table of Contents

Python Language Rules	Lint Imports Packages Exceptions Global variables Nested/Local/Inner Classes and Functions List Comprehensions Default Iterators and Operators Generators Lambda Functions Conditional Expressions Default Argument Values Properties True/False evaluations Deprecated Language Features Lexical Scoping Function and Method Decorators Threading Power Features
Python Style Rules	Semicolons Line length Parentheses Indentation Blank Lines Whitespace Shebang Line Comments Classes Strings Files and Sockets TODO Comments Imports formatting Statements Access Control Naming Main

Important Note

Displaying Hidden Details in this Guide

This style guide contains many details that are initially hidden from view. They are marked by the triangle icon, which you see here on your left. Click it now. You should see "Hooray" appear below.

Background

Python is the main scripting language used at Google. This style guide is a list of dos and don'ts for Python programs.

To help you format code correctly, we've created a <u>settings file for Vim</u>. For Emacs, the default settings should be fine.

Python Language Rules

Lint

Run pylint over your code.

Imports

Use imports for packages and modules only.

Packages

Import each module using the full pathname location of the module.

Exceptions

Exceptions are allowed but must be used carefully.

Global variables

Avoid global variables.

Nested/Local/Inner Classes and Functions

Nested/local/inner classes and functions are fine.

List Comprehensions

Okay to use for simple cases.

Default Iterators and Operators

Use default iterators and operators for types that support them, like lists, dictionaries, and files.

Generators

Use generators as needed.

Lambda Functions

▶ Okay for one-liners.

Conditional Expressions

▶ Okay for one-liners.

Default Argument Values

Okay in most cases.

Properties

Use properties for accessing or setting data where you would normally have used simple, lightweight accessor or setter methods.

True/False evaluations

Use the "implicit" false if at all possible.

Deprecated Language Features

Use string methods instead of the string module where possible. Use function call syntax instead of apply. Use list comprehensions and for loops instead of filter and map when the function argument would have been an inlined lambda anyway. Use for loops instead of reduce.

Lexical Scoping

Okay to use.

Function and Method Decorators

Use decorators judiciously when there is a clear advantage.

Threading

Do not rely on the atomicity of built-in types.

Power Features

Avoid these features.

Python Style Rules

Semicolons

Do not terminate your lines with semi-colons and do not use semi-colons to put two commands on the same line.

Line length

Maximum line length is 80 characters.

Parentheses

▶ Use parentheses sparingly.

Indentation

Indent your code blocks with 4 spaces.

Blank Lines

Two blank lines between top-level definitions, one blank line between method definitions.

Whitespace

Follow standard typographic rules for the use of spaces around punctuation.

Shebang Line

Most .py files do not need to start with a #! line. Start the main file of a program with #!/usr/bin/python with an optional single digit 2 or 3 suffix per PEP-394.

Comments

Be sure to use the right style for module, function, method and in-line comments

Classes

If a class inherits from no other base classes, explicitly inherit from object. This also applies to nested classes.

Strings

Use the format method or the % operator for formatting strings, even when the parameters are all strings. Use your best judgement to decide between + and % (or format) though.

Files and Sockets

Explicitly close files and sockets when done with them.

TODO Comments

Use T0D0 comments for code that is temporary, a short-term solution, or good-enough but not perfect.

Imports formatting

Imports should be on separate lines.

Statements

Generally only one statement per line.

Access Control

If an accessor function would be trivial you should use public variables instead of accessor functions to avoid the extra cost of function calls in Python. When more functionality is added you can use property to keep the syntax consistent.

Naming

module_name, package_name, ClassName, method_name, ExceptionName, function_name, GLOBAL_CONSTANT_NAME, global_var_name, instance_var_name, function parameter name, local var name.

Main

Even a file meant to be used as a script should be importable and a mere import should not have the side effect of executing the script's main

functionality. The main functionality should be in a main() function.

Parting Words

BE CONSISTENT.

If you're editing code, take a few minutes to look at the code around you and determine its style. If they use spaces around all their arithmetic operators, you should too. If their comments have little boxes of hash marks around them, make your comments have little boxes of hash marks around them too.

The point of having style guidelines is to have a common vocabulary of coding so people can concentrate on what you're saying rather than on how you're saying it. We present global style rules here so people know the vocabulary, but local style is also important. If code you add to a file looks drastically different from the existing code around it, it throws readers out of their rhythm when they go to read it. Avoid this.

Revision 2.59

Amit Patel
Antoine Picard
Eugene Jhong
Gregory P. Smith
Jeremy Hylton
Matt Smart
Mike Shields
Shane Liebling