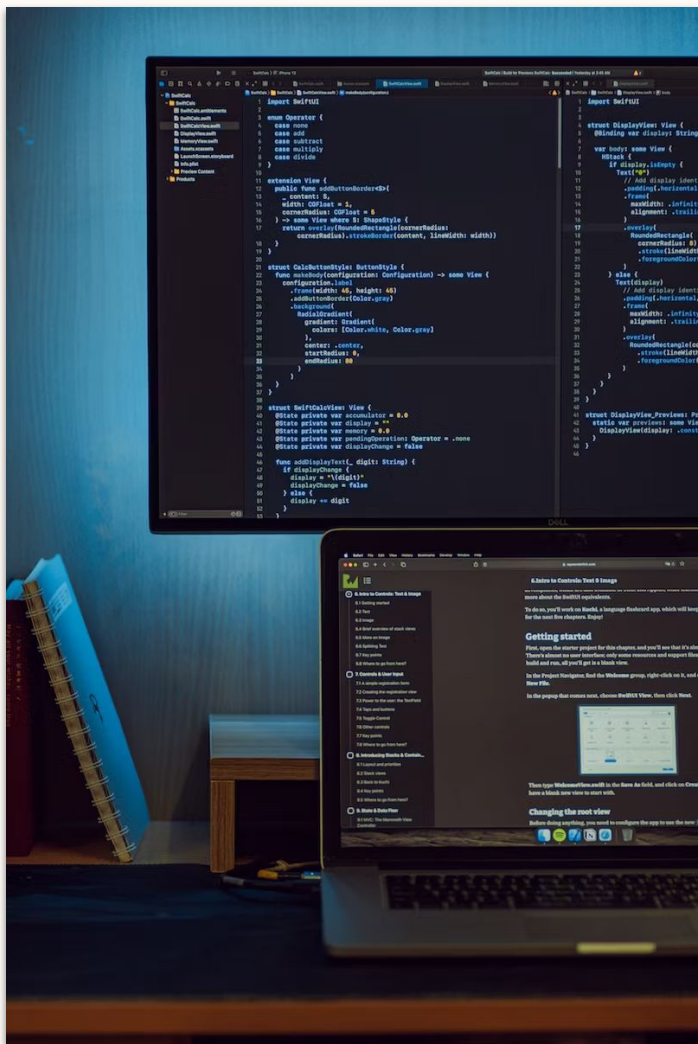


Continue

Clean Code Principles

For Data Science & Machine Learning

DCA0305
ivanovitch.silva@ufrn.br

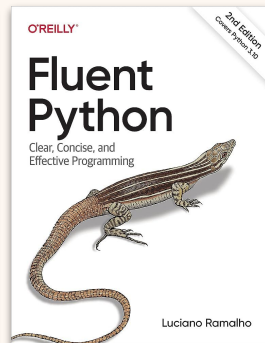
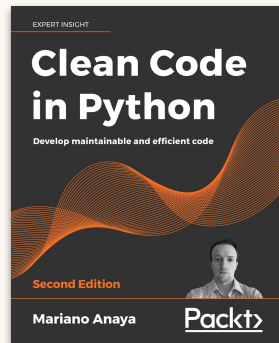


Agenda

1. Documentation & Following PEP8

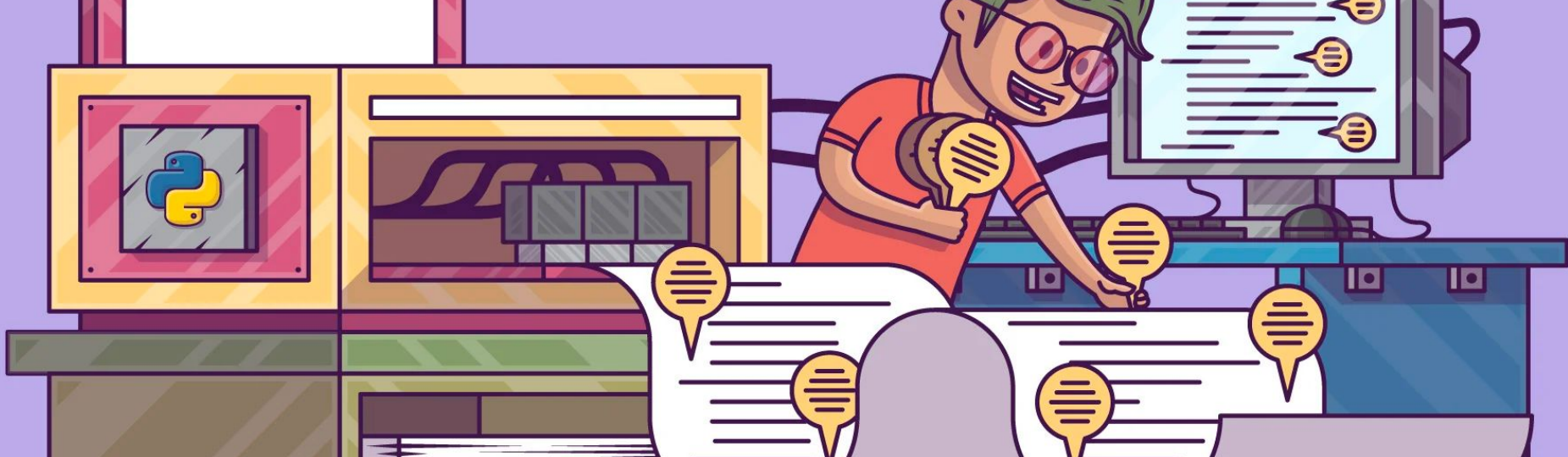
2. Production Ready Code

- Handling Errors
- Writing Tests and Logs



"Debugging is hard, testing is easy"

*"Far better an approximate answer to the right question, which is often vague,
than an exact answer to the wrong question"*



Documentation

Understand tricky bits, find your way, explain what parts do and why they're important.



Line Level



Function or Module Level



Project Level

```
class MyClass:
    """
    This is a demonstration class.

    Attributes:
    attribute1 (type): Description of
    attribute1
    """
    def __init__(self, attribute1):
        self.attribute1 = attribute1
```

```
def add(a, b):
    """
    This function adds two numbers and returns the sum.

    Parameters:
    a (int or float): The first number
    b (int or float): The second number

    Returns:
    int or float: The sum of a and b
    """
    return a + b
```

```
# This is a line comment
print("Hello, World!") # This comment explains what the line does
```

Documentation

```
"""
This module provides utility functions for basic arithmetic
operations.

Functions:
- add(a, b): Returns the sum of a and b
- subtract(a, b): Returns the difference between a and b
"""

# rest of the code
```

```

from typing import List

def add(a: int, b: int) -> int:
    """
    This function adds two integers and returns the sum.

    Parameters:
    a (int): The first number
    b (int): The second number

    Returns:
    int: The sum of a and b
    """
    return a + b

def get_even_numbers(numbers: List[int]) -> List[int]:
    """
    Given a list of integers, return a list containing only the even numbers.

    Parameters:
    numbers (List[int]): A list of integers

    Returns:
    List[int]: A list of even integers
    """
    return [x for x in numbers if x % 2 == 0]

```

Type Annotations

Annotations can also be added for class attributes and instance variables

```
class Point:
    x: float
    y: float

    def __init__(self, x: float, y: float) -> None:
        self.x = x
        self.y = y
```

Type
Annotations



Contents

- [Introduction](#)
- [A Foolish Consistency is the Hobgoblin of Little Minds](#)
- [Code Lay-out](#)
 - [Indentation](#)
 - [Tabs or Spaces?](#)
 - [Maximum Line Length](#)
 - [Should a Line Break Before or After a Binary Operator?](#)
 - [Blank Lines](#)
 - [Source File Encoding](#)
 - [Imports](#)
 - [Module Level Dunder Names](#)
- [String Quotes](#)
- [Whitespace in Expressions and Statements](#)
 - [Pet Peeves](#)
 - [Other Recommendations](#)
- [When to Use Trailing Commas](#)
- [Comments](#)
 - [Block Comments](#)
 - [Inline Comments](#)
 - [Documentation Strings](#)

PEP 8 – Style Guide for Python Code

Author: Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>

Status: [Active](#)

Type: [Process](#)

Created: 05-Jul-2001

Post-History: 05-Jul-2001, 01-Aug-2013

► [Table of Contents](#)

[Introduction](#)

This document gives coding conventions for the Python code comprising the standard library in the main Python distribution. Please see the companion informational PEP describing [style guidelines for the C code in the C implementation of Python](#).

This document and [PEP 257](#) (Docstring Conventions) were adapted from Guido's original Python Style Guide essay, with some additions from Barry's style guide [\[2\]](#).

Python Enhancement Proposals | Python » PEP Index » PEP 3107 ⓘ

Contents

- Abstract
- Rationale
- Fundamentals of Function Annotations
- Syntax
 - Parameters
 - Return Values
 - Lambda
- Accessing Function Annotations
- Use Cases
- Standard Library
 - pydoc and inspect
- Relation to Other PEPs
 - Function Signature Objects (PEP 362)
- Implementation
- Rejected Proposals
- References and Footnotes
- Copyright

[Page Source \(GitHub\)](#)

PEP 3107 – Function Annotations

Author: Collin Winter <collinwinter at google.com>, Tony Lownds <tony at lownds.com>
Status: Final
Type: Standards Track
Created: 02-Dec-2006
Python-Version: 3.0
Post-History:

► [Table of Contents](#)

Abstract

This PEP introduces a syntax for adding arbitrary metadata annotations to Python functions [1].

Rationale

Because Python's 2.x series lacks a standard way of annotating a function's parameters and return

Python Enhancement Proposals | Python » PEP Index » PEP 257 ⓘ

Contents

- Abstract
- Rationale
- Specification
 - What is a Docstring?
 - One-line Docstrings
 - Multi-line Docstrings
 - Handling Docstring Indentation
- Copyright
- Acknowledgements

[Page Source \(GitHub\)](#)

PEP 257 – Docstring Conventions

Author: David Goodger <goodger at python.org>, Guido van Rossum <guido at python.org>
Discussions-To: [Doc-SIG list](#)
Status: Active
Type: Informational
Created: 29-May-2001
Post-History: 13-Jun-2001

► [Table of Contents](#)

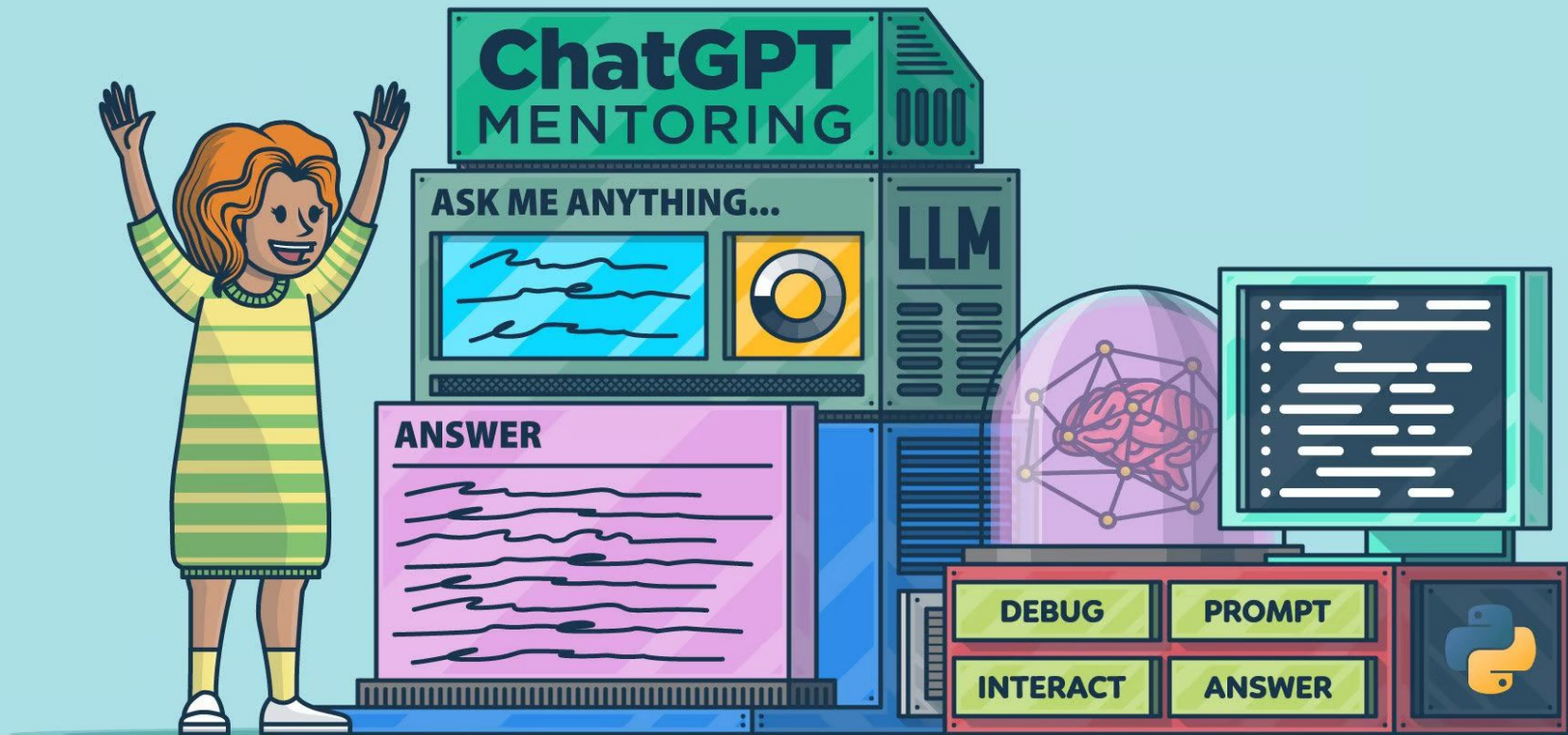
Abstract

This PEP documents the semantics and conventions associated with Python docstrings.

Rationale

The aim of this PEP is to standardize the high-level structure of docstrings: what they should contain,

- PEP-3107: <https://www.python.org/dev/peps/pep-3107/>
- PEP-484: <https://www.python.org/dev/peps/pep-0484/>
- PEP-526: <https://www.python.org/dev/peps/pep-0526/>
- PEP-557: <https://www.python.org/dev/peps/pep-0557/>
- PEP-585: <https://www.python.org/dev/peps/pep-0585/>



Real Python

Custom instructions

What would you like ChatGPT to know about you to provide better responses?

0/1500

How would you like ChatGPT to respond?

0/1500

Enable for new chats ☒

Cancel

Save

I am a data science professor for a University in Brazil. I teach technical ML and DL courses for students at undergraduate level in computer engineering. Most students attending my course are already familiar with basic ML terms and techniques and they attend my course to deep dive in technical concepts.

Respond in professional tone, deep dive into concepts, do not use too many filter words, always be straight to the point, and always at the beginning of your response you must have a abstract in exactly 3 sentences. At the end of your response you must suggest what other topics student must read as an optional reading to get more insight into the topic.

How to configure
some basic tools and
automatically
run checks on code.

#tooling



Real Python

Linters	Category	Description
Pylint	Logical & Stylistic	Checks for errors, tries to enforce a coding standard, looks for code smells
PyFlakes	Logical	Analyzes programs and detects various errors
pycodestyle	Stylistic	Checks against some of the style conventions in PEP 8
pydocstyle	Stylistic	Checks compliance with Python docstring conventions
Bandit	Logical	Analyzes code to find common security issues
MyPy	Logical	Checks for optionally-enforced static types
autopep8	Stylistic	Automatically formats Python code to conform to the PEP 8 style guide
nbQA	Logical & Stylistic	Linting and formatting Jupyter Notebooks with any command-line tool



Tool	Category	Description
Mccabe	Analytical	Checks McCabe complexity
Radon	Analytical	Analyzes code for various metrics (lines of code, complexity, and so on)
Black	Formatter	Formats Python code without compromise
Isort	Formatter	Formats imports by sorting alphabetically and separating into sections



PyCQA

Q Type to search

[Overview](#) [Repositories 28](#) [Projects 1](#) [Packages](#) [People 35](#)<https://github.com/PyCQA>

Python Code Quality Authority

Organization for code quality tools (and plugins) for the Python programming language

[580 followers](#) [Everywhere](#) <http://meta.pycqa.org> code-quality@python.org[Follow](#)

Pinned



meta

Public

Documentation about how the PyCQA organization works

[Python](#) [24](#) [23](#)

Repositories

Q Find a repository...

Type

Language

Sort

flake8

Public

flake8 is a python tool that glues together pycodestyle, pyflakes, mccabe, and third-party plugins to check the style and quality of some python code.

[Python](#) [2,989](#) [296](#) [25](#) [2](#) Updated yesterday

autoflake

Public

Removes unused imports and unused variables as reported by pyflakes

[Python](#) [802](#) [MIT](#) [77](#) [37 \(2 issues need help\)](#) [3](#) Updated 2 days ago

People

[View all](#)

Top languages

[Python](#) [Dockerfile](#)

Most used topics

[python](#)[linter](#)[flake8](#)[linter-flake8](#)[linter-plugin](#)[Report abuse](#)

pycodestyle Public[Watch 118](#)[Fork 760](#)[Star 4.9k](#)

main



11 branches



44 tags

[Go to file](#)[Add file](#)[Code](#)[sigmavirus24 Merge pull request #1186 from PyCQA/pre-commit-ci-update-...](#)

1c2147c on Aug 1



1,429 commits



.github/workflows

drop python 3.7 support

2 months ago



docs

fix some wrong renames from testsuite to testing/data vs test

2 months ago



testing

handle fstring continuation in 3.12

2 months ago



tests

move testsuite -> tests

2 months ago



.gitattributes

move testsuite -> tests

2 months ago



.gitignore

convert --doctest to pytest tests

2 months ago



.pre-commit-config.yaml

[pre-commit.ci] pre-commit autoupdate

2 months ago



.readthedocs.yaml

add minimal rtd configuration

2 months ago



CHANGES.txt

Release 2.11.0

2 months ago



CONTRIBUTING.rst

move testsuite -> tests

2 months ago



LICENSE

Bumped license year

3 years ago



Makefile

Simplified / cleaned up the Makefile to match Travis (just use tox)

3 years ago



README.rst

move testsuite -> tests

2 months ago

About

Simple Python style checker in one Python file

pycodestyle.pycqa.org[python](#)[styleguide](#)[style-guide](#)[linter-plugin](#)[pep8](#)[linter-flake8](#)[flake8-plugin](#)[Readme](#)[View license](#)[Activity](#)[4.9k stars](#)[118 watching](#)[760 forks](#)[Report repository](#)

Releases 4



1.7.1

[Latest](#)

on Oct 25, 2017

[+ 3 releases](#)



Public

Sponsor

Watch 76

Fork 1k

Star 4.8k

main

10 branches

184 tags

Go to file

Add file

<> Code

	Pierre-Sassoulas [doc tests] Better assertion message for multiple bad files (... cfa4c7f 12 hours ago 8,840 commits
	.github [primer] Fix conf following coverage's changes last week
	doc [doc tests] Better assertion message for multiple bad files (#9020) 12 hours ago
	examples Remove deprecated future argument 2 months ago
	pylint Fix Pyreverse optional annotation bug (#9016) yesterday
	script Bump pylint to 3.0.0a7, update changelog last month
	tests Fix Pyreverse optional annotation bug (#9016) yesterday
	.coveragerc Removing files from coveragerc since they have been deleted (#8417) 6 months ago
	.git-blame-ignore-revs Avoid blaming formatting commits 2 months ago
	.gitattributes [mixed-line-endings] Create the initial documentation (#8994) last week
	.gitignore Move pylint_primer_tests into tests last year
	.pre-commit-config.yaml [pre-commit.ci] pre-commit autoupdate (#9007) 4 days ago
	.pre-commit-hooks.yaml Add prettier to the pre-commit configuration 2 years ago
	.pyenchant_pylint_custom_dict.txt Add a new json2 reporter (#8929) last month
	.readthedocs.yaml [readthedoc] Upgrade the python version used during build 2 weeks ago
	CITATION.cff Add a CITATION.cff containing the necessary metadata to cite Pylint 3 months ago
	CODE_OF_CONDUCT.md Add a code of conduct from template (#5589) 2 years ago

About

It's not just a linter that annoys you!

pylint.readthedocs.io/en/latest/[static-code-analysis](#) [linter](#)
[static-analysis](#) [code-quality](#) [pep8](#)
[hacktoberfest](#) [closember](#)

Readme
 GPL-2.0 license
 Code of conduct
 Security policy
 Cite this repository
 Activity
☆ 4.8k stars
👁 76 watching
🍴 1k forks
Report repository

Releases 76

 v2.17.5 Latest
on Jul 26

+ 75 releases



hhatto / autopep8

Type / to search

[Code](#) [Issues](#) [116](#) [Pull requests](#) [4](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)

autopep8

Public

[Watch](#) 72[Fork](#) 299[Star](#) 4.4k

main

7 branches

73 tags

[Go to file](#)[Add file](#)[Code](#)

hhatto Merge pull request #705 from hhatto/migrate-to-pyprojecttoml

4e869ad 2 weeks ago

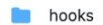
2,833 commits



.github

Add Python 3.11 to the test matrix

9 months ago



hooks

Remove unused variables

9 years ago



test

fix unit tests

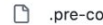
2 weeks ago



.gitignore

ignore mypy cache

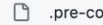
4 years ago



.pre-commit-config.yaml

add pre-commit-config

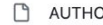
6 months ago



.pre-commit-hooks.yaml

add pre-commit configuration

6 months ago



AUTHORS.rst

many Thanksgit add AUTHORS.rst

3 years ago



CONTRIBUTING.rst

Rename "pep8" to "pycodestyle"

7 years ago



LICENSE

Update year

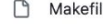
7 years ago



MANIFEST.in

Update MANIFEST.in

last month



Makefile

add test_ci

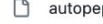
last year



README.rst

add pre-commit configuration

6 months ago



autopep8.py

version 2.0.4

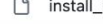
2 weeks ago



coveralls.bash

Move this mess into a separate file

10 years ago



install_hooks.bash

Use lowercase

9 years ago



pyproject.toml

migrate to pyproject.toml

2 weeks ago

About

A tool that automatically formats Python code to conform to the PEP 8 style guide.

pypi.org/project/autopep8/[python](#) [formatter](#) [pep8](#) [codeformatter](#)[Readme](#)[MIT license](#)[Activity](#)[4.4k stars](#)[72 watching](#)[299 forks](#)[Report repository](#)

Releases 23

[v2.0.4](#) [Latest](#)
2 weeks ago[+ 22 releases](#)

Packages

<https://github.com/hhatto/autopep8>



<https://code.visualstudio.com/docs/python/linting>

OVERVIEW

SETUP

GET STARTED

USER GUIDE

SOURCE CONTROL

TERMINAL

LANGUAGES

NODE.JS /

JAVASCRIPT

TYPESCRIPT

PYTHON

Tutorial

Editing Code

Linting

Formatting

Debugging

Environments

Testing

Python Interactive

Django Tutorial

Flask Tutorial

Create containers

Deploy Python Apps

Python in the Web

Settings Reference

Linting Python in Visual Studio Code

Edit

Linting highlights syntactical and stylistic problems in your Python source code, which often helps you identify and correct subtle programming errors or unconventional coding practices that can lead to errors. For example, linting detects use of an uninitialized or undefined variable, calls to undefined functions, missing parentheses, and even more subtle issues such as attempting to redefine built-in types or functions. Linting is distinct from [Formatting](#) because linting analyzes how the code runs and detects errors whereas formatting only restructures how code **appears**.

Note: Stylistic and syntactical code detection is enabled by the Language Server. To enable third-party linters for additional problem detection, you can enable them by using the **Python: Select Linter** command and selecting the appropriate linter.

Choose a linter

Install the linting tool of your choice from the VS Code [Marketplace](#).

Microsoft publishes the following linting extensions:

Linters	Extension
Pylint	https://marketplace.visualstudio.com/items?itemName=ms-python.pylint
flake8	https://marketplace.visualstudio.com/items?itemName=ms-python.flake8
mypy	https://marketplace.visualstudio.com/items?itemName=ms-python.mypy-type-checker

IN THIS ARTICLE

Choose a linter

General Settings

Disable linting

Run linting

Code Actions

Logging

Severity

Troubleshooting linting

Next steps

[Subscribe](#)

[Ask questions](#)

[Follow @code](#)

[Request features](#)

[Report issues](#)

[Watch videos](#)