

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- C#
- CSS
- Flex
- Go
- HTML
- Java
- JavaScript
- Kotlin
- Objective C
- PHP
- PL/I
- PL/SQL
- Python**
- RPG
- Ruby
- Scala
- Swift
- Terraform
- Text
- TypeScript
- T-SQL
- VB.NET
- VB6
- XML



Python static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your PYTHON code

All rules 216

Vulnerability 29

Bug 55

Security Hotspot 31

Code Smell 101

Tags ▾

Search by name...



Code Smell

A field should not duplicate the name of its containing class

Code Smell

Function names should comply with a naming convention

Code Smell

Functions and lambdas should not reference variables defined in enclosing loops

Code Smell

Sections of code should not be commented out

Code Smell

Unused function parameters should be removed

Code Smell

Unused class-private methods should be removed

Code Smell

Track uses of "FIXME" tags

Code Smell

"Exception" and "BaseException" should not be raised

Code Smell

Redundant pairs of parentheses should be removed

Code Smell

Nested blocks of code should not be left empty

Code Smell

Functions, methods and lambdas should not have too many parameters

Arguments given to functions should be of an expected type

Analyze your code

Code Smell Critical ? suspicious

The CPython interpreter does not check arguments type when functions are called. However a function can express the type it expects for each argument in its documentation or by using **Type Hints**. Calling such a function with an argument of a different type can easily create a bug. Even if it works right now it can fail later when APIs evolve or when type checks are added (ex: with `isinstance`).

This rule raises an issue when a function or method is called with an argument of a different type than the one described in its type annotations. It also checks argument types for builtin functions.

Noncompliant Code Example

```
def func(var: str):
    pass

func(42) # Noncompliant

len(1) # Noncompliant
```

Compliant Solution

```
def func(var: str):
    pass

func("42")


len("1")
```

See

- [Python documentation - builtins](#)
- [PEP 484 – Type Hints](#)
- [Python documentation - typing – Support for type hints](#)

Available In:


sonarlint sonarcloud sonarqube

 Code Smell

Collapsible "if" statements should be merged

 Code Smell

Logging should not be vulnerable to injection attacks

 Vulnerability

Repeated patterns in regular expressions should not match the empty string

 Bug

Function parameters initial values should not be ignored