sonar RULES

Products

- 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...

Functions should not have too many lines of code
Code Smell

Track uses of "NOSONAR" comments
Code Smell

Track comments matching a regular expression
Code Smell

Statements should be on separate lines
Code Smell

Functions should not contain too many return statements
Code Smell

Files should not have too many lines of code
Code Smell

Lines should not be too long
Code Smell

Methods and properties that don't access instance data should be static
Code Smell

New-style classes should be used
Code Smell

Parentheses should not be used after certain keywords
Code Smell

Track "TODO" and "FIXME" comments that do not contain a reference to a person
Code Smell

Module names should comply with a naming convention

Builtins should not be shadowed by local variables

Analyze your code

Code Smell | Major | bad-practice confusing pitfall

Shadowing a builtin makes your code more difficult to read and maintain. It may also be a source of bugs as you can reference the builtin by mistake.

It is sometimes ok to shadow a builtin to improve the readability of a public API or to support multiple versions of a library. In these cases the value is higher than the maintainability cost. Just be careful when you do it.

It is not ok to shadow builtins with variables which are local to a function or method. These variables are not public and can be easily renamed, thus reducing the confusion and making the code less error-prone.

This rule raises an issue when the name of a local variable matches the name of a builtin.

Noncompliant Code Example

```
def a_function():
    int = 42  # Noncompliant; int is a builtin
```

Compliant Solution

```
def a_function():
    value = 42
```

See

- Python documentation - Built-in Types
- Python documentation - Built-in Functions

Available In:
sonarlint | sonarcloud | sonarqube

Code Smell

Comments should not be located at the end of lines of code

Code Smell

Lines should not end with trailing whitespaces

Code Smell

Files should contain an empty newline at the end

Code Smell

Long suffix "L" should be upper case

Code Smell