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...

re raise the provided exception
Code Smell

Unused scope-limited definitions should be removed
Code Smell

Functions and methods should not have identical implementations
Code Smell

Unused private nested classes should be removed
Code Smell

String formatting should be used correctly
Code Smell

Conditional expressions should not be nested
Code Smell

Loops without "break" should not have "else" clauses
Code Smell

Doubled prefix operators "not" and "~" should not be used
Code Smell

The "print" statement should not be used
Code Smell

"<>" should not be used to test inequality
Code Smell

Two branches in a conditional structure should not have exactly the same implementation
Code Smell

Unused assignments should be

Comparison to None should not be constant

Analyze your code

Code Smell | Critical | suspicious

Checking if a variable or parameter is `None` should only be done when you expect that it can be `None`. Doing so when the variable is always None or never None is confusing at best. At worse, there is a bug and the variable is not updated properly.

This rule raises an issue when expressions `X is None`, `X is not None`, `X == None` or `X != None` are constant, i.e. `X` is always None or never None.

Noncompliant Code Example

```
mynone = None
result = mynone is None: # Noncompliant. Always True.

if mynone == None: # Noncompliant. Always True.
    pass

if mynone is not None:  # Noncompliant. Always False.
    pass

if mynone == None:  # Noncompliant. Always False.
    pass

myint = 42
result = myint is None: # Noncompliant. Always False.

if myint == None: # Noncompliant. Always False.
    pass

if myint is not None:  # Noncompliant. Always True.
    pass

if myint == None:  # Noncompliant. Always True.
    pass
```

See

- Python documentation - Identity comparisons
- Python documentation - __eq__ operator

Available In:

sonarlint | sonarcloud | sonarqube

Unused assignments should be removed

☢ 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