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

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 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
Code Smell

Unused function parameters should be removed
Code Smell

Unused class-private methods should

Bare "raise" statements should not be used in "finally" blocks

Analyze your code

Code Smell | Critical | error-handling unpredictable confusing

A bare `raise` statement, i.e. a `raise` with no exception provided, will re-raise the last active exception in the current scope. If no exception is active a `RuntimeError` is raised instead.

If the bare "raise" statement is in a `finally` block, it will only have an active exception to re-raise when an exception from the `try` block is not caught or when an exception is raised by an `except` or `else` block. Thus bare `raise` statements should not be relied upon in `finally` blocks. It is simpler to let the exception raise automatically.

This rule raises an issue when a bare `raise` statements is in a `finally` block.

Noncompliant Code Example

```
def foo(param):
    result = 0
    try:
        print("foo")
    except ValueError as e:
        pass
    else:
        if param:
            raise ValueError()
    finally:
        if param:
            raise  # Noncompliant. This will fail in so
        else:
            result = 1
    return result
```

Compliant Solution

```
def foo(param):
    result = 0
    try:
        print("foo")
    except ValueError as e:
        pass
    else:
        if param:
            raise ValueError()
    finally:
        if not param:
            result = 1
```

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

```
        # the exception will raise automatically
    return result
```

See

- Python Documentation - The `raise` statement

Available In:

sonarlint | sonarcloud | sonarqube