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...

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

Custom Exception classes should inherit from "Exception" or one of its subclasses

Analyze your code

Code Smell   Critical

`SystemExit` is raised when `sys.exit()` is called. `KeyboardInterrupt` is raised when the user asks the program to stop by pressing interrupt keys. Both exceptions are expected to propagate up until the application stops.

In order to avoid catching `SystemExit` and `KeyboardInterrupt` by mistake PEP-352 created the root class `BaseException` from which `SystemExit`, `KeyboardInterrupt` and `Exception` derive. Thus developers can use `except Exception:` without preventing the software from stopping.

The `GeneratorExit` class also derives from `BaseException` as it is not really an error and is not supposed to be caught by user code.

As said in Python's documentation, user-defined exceptions are not supposed to inherit directly from `BaseException`. They should instead inherit from `Exception` or one of its subclasses.

This rule raises an issue when a class derives from one of the following exception classes: `BaseException`, `KeyboardInterrupt`, `SystemExit` or `GeneratorExit`.

Noncompliant Code Example

```
class MyException(BaseException):  # Noncompliant
    pass

class MyException(GeneratorExit):  # Noncompliant
    pass

class MyException(KeyboardInterrupt):  # Noncompliant
    pass

class MyException(SystemExit):  # Noncompliant
    pass
```

Compliant Solution

```
class MyException(Exception):
    pass
```

See

- PEP 352 – Required Superclass for Exceptions
- Python Documentation - BaseException class

Available In:

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