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
Code Smell

Special method "__exit__" should not re-raise the provided exception

Analyze your code

Code Smell | Major | error-handling bad-practice

The special method `__exit__` should only raise an exception when it fails. It should never raise the provided exception, it is the caller's responsibility.

Raising this exception will make the stack trace difficult to understand.

The `__exit__` method can filter passed-in exceptions by simply returning True or False.

This rule raises an issue when:

- an `__exit__` method has a bare `raise` outside of an `except` block.
- an `__exit__` method raises the exception provided as parameter.

Noncompliant Code Example

```
class MyContextManager:
    def __enter__(self):
        return self
    def __exit__(self, *args):
        raise  # Noncompliant
        raise args[2]  # Noncompliant

class MyContextManager:
    def __enter__(self):
        return self
    def __exit__(self, exc_type, exc_value, traceback):
        raise exc_value # Noncompliant
```

Compliant Solution

```
class MyContextManager:
    def __enter__(self):
        return self
    def __exit__(self, exc_type, exc_value, traceback):
        # by default the function will return None, which is
        pass

class MyContextManager:
    def __enter__(self, stop_exceptions):
        return self
    def __exit__(self, *args):
        try:
            print("42")
        except:
            print("exception")
            raise  # No issue when raising another exception
        raise MemoryError("No more memory")  # This is ok to
```

See

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

- Python documentation – The `__exit__` special method
- PEP 343 – The "with" Statement

Available In:

sonarlint | sonarcloud | sonarqube