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...

"pass" should not be used needlessly
Code Smell

"except" clauses should do more than raise the same issue
Code Smell

Boolean checks should not be inverted
Code Smell

Unused local variables should be removed
Code Smell

Local variable and function parameter names should comply with a naming convention
Code Smell

Field names should comply with a naming convention
Code Smell

Class names should comply with a naming convention
Code Smell

Method names should comply with a naming convention
Code Smell

Track uses of "TODO" tags
Code Smell

HTML autoescape mechanism should not be globally disabled
Vulnerability

Variables, classes and functions should be either defined or imported
Bug

New objects should not be created only to check their identity

Analyze your code

Bug Major

Identity operators `is` and `is not` check if the same object is on both sides, i.e. `a is b` returns `True` if `id(a) == id(b)`.

When a new object is created it will have its own identity. Thus if an object is created and used only in an identity check it is not possible for the other operand to be the same object. The comparison is always `False` or always `True` depending on the operator used, `is` or `is not`. To avoid this problem the identity operator could be replaced with an equality operator (== or !=), which will use __eq__ or __ne__ methods under the hood.

This rule raises an issue when at least one operand of an identity operator is a new object which has been created just for this check, i.e.:

- When it is a dict, list or set literal.
- When it is a call to `dict`, `set`, `list` or `complex` built-in functions.
- When such a new object is assigned to only one variable and this variable is used in an identity check.

Noncompliant Code Example

```
def func(param):
    param is {1: 2}  # Noncompliant; always False
    param is not {1, 2, 3}  # Noncompliant; always True
    param is [1, 2, 3]  # Noncompliant; always False

    param is dict(a=1)  # Noncompliant; always False

    mylist = []  # mylist is assigned a new object
    param is mylist  # Noncompliant; always False
```

Compliant Solution

```
def func(param):
    param == {1: 2}
    param != {1, 2, 3}
    param == [1, 2, 3]

    param == dict(a=1)

    mylist = []
    param == mylist
```

See

- Why does Python 3.8 log a SyntaxWarning for 'is' with literals? - Adam Johnson
- Equality vs identity - Trey Hunner

"__exit__" should accept type, value, and traceback arguments

Bug

"return" and "yield" should not be used in the same function

Bug

Track lack of copyright and license headers

Code Smell

HTTP response headers should not be vulnerable to injection attacks

Vulnerability

Regular expressions should be syntactically valid

Available In:

sonarlint | sonarcloud | sonarqube