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

Security Hotspot

Using regular expressions is security-sensitive
Security Hotspot

Dynamically executing code is security-sensitive
Security Hotspot

Cyclomatic Complexity of functions should not be too high
Code Smell

Control flow statements "if", "for", "while", "try" and "with" should not be nested too deeply
Code Smell

Cyclomatic Complexity of classes should not be too high
Code Smell

"\" should only be used as an escape character outside of raw strings
Bug

Using shell interpreter when executing OS commands is security-sensitive
Security Hotspot

Functions should use "return" consistently
Code Smell

Python parser failure
Code Smell

Files should not be too complex
Code Smell

Docstrings should be defined
Code Smell

## Identical expressions should not be used on both sides of a binary operator

Analyze your code

Bug Major

Using the same value on either side of a binary operator is almost always a mistake. In the case of logical operators, it is either a copy/paste error and therefore a bug, or it is simply wasted code, and should be simplified. In the case of bitwise operators and most binary mathematical operators, having the same value on both sides of an operator yields predictable results, and should be simplified.

Note that this rule will raise issues on `a == a` and `a != a` expressions which are sometime used to detect `NaN` values. It is recommended to use instead `math.isnan` or an equivalent function. This will improve code readability.

### Noncompliant Code Example

```
if a == a: # Noncompliant
    work()

if  a != a: # Noncompliant
    work()

if  a == b and a == b: # Noncompliant
    work()

if a == b or a == b: # Noncompliant
    work()

j = 5 / 5 # Noncompliant
k = 5 - 5 # Noncompliant
```

### Exceptions

The following are ignored:

- The expression `1 << 1`

### See

- {rule:python:S1656} - Implements a check on =.

Available In:

sonarlint sonarcloud sonarqube

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