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

Implicit string and byte concatenations should not be confusing

Analyze your code

Code Smell | Major | confusing suspicious

Python concatenates adjacent string or byte literals at compile time. It means that `"a" "b"` is equivalent to `"ab"`. This is sometimes used to split a long string on multiple lines. However an implicit string concatenation can also be very confusing. In the following contexts it might indicate that a comma was forgotten:

- when the two strings are on the same line it looks like a badly formatted tuple. Parenthesises are not mandatory to create a tuple, only the comma is.
- when the strings are in a list, set or tuple.

Noncompliant Code Example

```
def func():
    return "item1" "item2"  # Noncompliant

["1",
 "2"  # Noncompliant
 "3",
 "a very very very"  # Noncompliant
 "very very long string",
 "4"]
```

Compliant Solution

```
def func():
    return "item1", "item2"

["1",
 "2",
 "3",
 "a very very very" +
 "very very long string",
 "4"]
```

Exceptions

No issue will be raised when there is a visible reason for the string concatenation:

- when the quotes used for both strings are different. This can be used to avoid escaping quotes
- when the strings or bytes have different prefixes, i.e. "f" for f-strings, "r" for raw, "u" for unicode and no prefix for normal strings.
- when strings are visibly split to avoid long lines of code. (Example: the first string ends with a space, punctuation or `\n`).

Code Smell

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

Available In:

sonarlint | sonarcloud | sonarqube