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

String formatting should be used correctly

Analyze your code

Code Smell | Major | confusing

Formatting strings, either with the `%` operator or `str.format` method, requires a valid string and arguments matching this string's replacement fields.

This also applies to loggers from the `logging` module. Internally they use `%`-formatting. The only difference is that they will log an error instead of raising an exception when provided arguments are invalid.

Formatted string literals, also called "f-strings", are generally simpler to use, and any syntax mistake will fail at compile time. However it is easy to forget curly braces and it won't raise any error.

This rule raises an issue when:

- A string formatted with `%` will not return the expected string because some arguments are not used.
- A string formatted with `str.format` will not return the expected string because some arguments are not used.
- An "f-string" doesn't contain any replacement field, which probably means that some curly braces are missing.
- Loggers will log an error because their message is not formatted properly.

Rule {rule:python:S2275} covers cases where formatting a string will raise an exception.

Noncompliant Code Example

```
"Error %(message)s" % {"message": "something failed", "extra

"Error: User {} has not been able to access []".format("Alic

user = "Alice"
resource = "MyFile"
message = f"Error: User [user] has not been able to access [

import logging
logging.error("Error: User %s has not been able to access %s
```

Compliant Solution

```
"Error %(message)s" % {"message": "something failed"}

"Error: User {} has not been able to access {}".format("Alic

user = "Alice"
resource = "MyFile"
message = f"Error: User {user} has not been able to access {

import logging
logging.error("Error: User %s has not been able to access %s
```

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

See

- Python documentation - Format String Syntax
- Python documentation - printf-style String Formatting
- Python documentation - Loggers
- Python documentation - Using particular formatting styles throughout your application
- Python documentation - Formatted string literals

Available In:

sonarlint | sonarcloud | sonarqube