
































-  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

Track uses of "NOSONAR" comments

Track comments matching a regular expression

Statements should be on separate lines

Functions should not contain too many return statements

Files should not have too many lines of code

Lines should not be too long

Methods and properties that don't access instance data should be static

New-style classes should be used

Parentheses should not be used after certain keywords

Track "TODO" and "FIXME" comments that do not contain a reference to a person

Module names should comply with a naming convention

Functions and methods should not have identical implementations

Analyze your code

 Code Smell

 Major 

 confusing duplicate suspicious

When two functions or methods have the same implementation, either it was a mistake - something else was intended - or the duplication was intentional, but may be confusing to maintainers. In the latter case, one implementation should invoke the other. Numerical and string literals are not taken into account.

Noncompliant Code Example

```
class MyClass:
    code = "bounteous"

    def calculate_code(self):
        self.do_the_thing()
        return self.__class__.code

    def get_name(self): # Noncompliant
        self.do_the_thing()
        return self.__class__.code

    def do_the_thing(self):
        pass # on purpose
```

Compliant Solution

```
class MyClass:
    code = "bounteous"

    def calculate_code(self):
        self.do_the_thing()
        return self.__class__.code

    def get_name(self):
        return self.calculate_code()


    def do_the_thing(self):
        pass # on purpose
```

Exceptions


No issue will be raised on empty methods/functions and methods/functions with only one line of code.

Available In:


sonarlint  | sonarcloud  | sonarqube 

 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

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
[Privacy Policy](#)