
































-  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

## Doubled prefix operators "not" and "~" should not be used

Analyze your code

 Code Smell  Major 

Calling the not or ~ prefix operator twice might be redundant: the second invocation undoes the first. Such mistakes are typically caused by accidentally double-tapping the key in question without noticing. Either this is a bug, if the operator was actually meant to be called once, or misleading if done on purpose. Calling not twice is commonly done instead of using the dedicated "bool()" builtin function. However, the latter one increases the code readability and should be used.

### Noncompliant Code Example

```
a = 0
b = False

c = not not a # Noncompliant
d = ~~b # Noncompliant
```

### Compliant Solution

```
a = 0
b = False

c = bool(a)
d = ~b
```

### Exceptions

If the ~ function has been overloaded in a customised class and has been called twice, no warning is raised as it is assumed to be an expected usage.

Available In:

 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