
































-  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

Character classes in regular expressions should not contain the same character twice

Analyze your code

 Code Smell

 Major 

 regex

Character classes in regular expressions are a convenient way to match one of several possible characters by listing the allowed characters or ranges of characters. If the same character is listed twice in the same character class or if the character class contains overlapping ranges, this has no effect.

Thus duplicate characters in a character class are either a simple oversight or a sign that a range in the character class matches more than is intended or that the author misunderstood how character classes work and wanted to match more than one character. A common example of the latter mistake is trying to use a range like `[0-99]` to match numbers of up to two digits, when in fact it is equivalent to `[0-9]`. Another common cause is forgetting to escape the `-` character, creating an unintended range that overlaps with other characters in the character class.

Noncompliant Code Example

```
r"[0-99]" # Noncompliant, this won't actually match strings
r"[0-9.-_]" # Noncompliant, .-_ is a range that already contains
```

Compliant Solution

```
r"[0-9]{1,2}"
r"[0-9.\_\-\_]"
```


Available In:

 sonarlint

 sonarcloud

 sonarqube

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