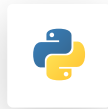


- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- C#
- CSS
- Flex
- Go
- HTML
- Java
- JavaScript
- Kotlin
- Objective C
- PHP
- PL/I
- PL/SQL
- Python**
- RPG
- Ruby
- Scala
- Swift
- Terraform
- Text
- TypeScript
- T-SQL
- VB.NET
- VB6
- XML



## 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

### Regular expressions should not be too complicated

Analyze your code

Code Smell Major ? regex

Overly complicated regular expressions are hard to read and to maintain and can easily cause hard-to-find bugs. If a regex is too complicated, you should consider replacing it or parts of it with regular code or splitting it apart into multiple patterns at least.

The complexity of a regular expression is determined as follows:

Each of the following operators increases the complexity by an amount equal to the current nesting level and also increases the current nesting level by one for its arguments:

- | - when multiple | operators are used together, the subsequent ones only increase the complexity by 1
- Quantifiers (\*, +, ?, {n,m}, {n,} or {n})
- Non-capturing groups that set flags (such as (?i:some\_pattern) or (?i)some\_pattern)
- Lookahead and lookbehind assertions

Additionally, each use of the following features increase the complexity by 1 regardless of nesting:

- character classes
- back references

#### Noncompliant Code Example

```
p = re.compile(r"^(?:(?:31(\w|-|\.)?(?:0?[13578]|1[02]))\1|(??:
```

```
if p.match($dateString):
    handleDate($dateString)
```


#### Compliant Solution

```
p = re.compile("^\d{1,2}([-/]\d{1,2})\1\d{1,4}$")
if p.match($dateString):
    dateParts = re.split(r"[-/]", dateString)
    day = intval(dateParts[0])
    month = intval(dateParts[1])
    year = intval($dateParts[2])
    // Put logic to validate and process the date based on i
```

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