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...

Python parser failure
Code Smell

Files should not be too complex
Code Smell

Docstrings should be defined
Code Smell

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

All "except" blocks should be able to catch exceptions

Analyze your code

Bug Major

Exceptions handlers (`except:`) are evaluated in the order they are written. Once a match is found, the evaluation stops.

In some contexts an except block is dead code as it will never catch any exception:

- If there is a handler for a base class followed by a handler for class derived from that base class, the second handler will never trigger: The handler for the base class will match the derived class, and will be the only executed handler.
- When multiple `except` statements try to catch the same exception class, only the first one will be executed.
- In python 3, `BaseException` is the parent of every exception class. When `BaseException` is caught and the same try-except block has a bare `except:` statement, i.e. an `except` with no expression, the bare except will never catch anything.

This rule raises an issue when an `except` block catches every exception before a later `except` block could catch it.

Noncompliant Code Example

```
def foo():
    try:
        raise FloatingPointError()
    except (ArithmeticError, RuntimeError) as e:
        print(e)
    except FloatingPointError as e: # Noncompliant. Floating
        print("Never executed")
    except OverflowError as e: # Noncompliant. OverflowError
        print("Never executed")

    try:
        raise TypeError()
    except TypeError as e:
        print(e)
    except TypeError as e: # Noncompliant. Duplicate Except.
        print("Never executed")

    try:
        raise ValueError()
    except BaseException as e:
        print(e)
    except: # Noncompliant. This is equivalent to "except Ba
        print("Never executed")
```

Compliant Solution

```
def foo():
    try:
        raise FloatingPointError()
    except FloatingPointError as e:
        print("Executed")
```

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

Comments should not be located at the end of lines of code

Code Smell

```
except OverflowError as e:
    print("Executed")
except (ArithmeticError, RuntimeError) as e:
    print(e)

try:
    raise TypeError()
except TypeError as e:
    print(e)

try:
    raise ValueError()
except BaseException as e:
    print(e)
```

See

- Python Documentation - The `try` statement

Available In:

sonarlint | sonarcloud | sonarqube