Python static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your PYTHON code

Bare "raise" statements should not be used in "finally" blocks

Analyze your code

Code Smell | Critical | error-handling unpredictable confusing

A bare `raise` statement, i.e. a `raise` with no exception provided, will re-raise the last active exception in the current scope. If no exception is active a `RuntimeError` is raised instead.

If the bare "raise" statement is in a `finally` block, it will only have an active exception to re-raise when an exception from the `try` block is not caught or when an exception is raised by an `except` or `else` block. Thus bare `raise` statements should not be relied upon in `finally` blocks. It is simpler to let the exception raise automatically.

This rule raises an issue when a bare `raise` statements is in a `finally` block.

Noncompliant Code Example

```
def foo(param):
    result = 0
    try:
        print("foo")
    except ValueError as e:
        pass
    else:
        if param:
            raise ValueError()
    finally:
        if param:
            raise  # Noncompliant. This will fail in so
        else:
            result = 1
    return result
```

Compliant Solution

```
def foo(param):
    result = 0
    try:
        print("foo")
    except ValueError as e:
        pass
    else:
        if param:
            raise ValueError()
    finally:
        if not param:
            result = 1
```

be removed

Code Smell

Track uses of "FIXME" tags

Code Smell

"Exception" and "BaseException" should not be raised

Code Smell

Redundant pairs of parentheses should be removed

Code Smell

Nested blocks of code should not be left empty

Code Smell

```
        # the exception will raise automatically
    return result
```

See

- Python Documentation - The `raise` statement

Available In:

sonarlint | sonarcloud | sonarqube