Python static code analysis

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

Bare "raise" statements should only be used in "except" 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 the "raise" statement is not in an `except` or `finally` block, no exception is active and a `RuntimeError` is raised instead.

If the bare `raise` statement is in a function called in an `except` statement, the exception caught by the `except` will be raised. This works but is hard to understand and maintain. Nothing indicates in the parent `except` that the exception will be reraised, and nothing prevents a developer from calling the function in another context.

Note also that using a bare `raise` in a `finally` block only works when an exception is active, i.e. when an exception from the `try` block is not caught or when an exception is raised by an `except` block. It will fail in any other case and should not be relied upon. This code smell is handled by rule {rule:python:S5704}.

This rule raises an exception when a bare `raise` statement is not in an `except` or `finally` block.

Noncompliant Code Example

```
raise  # Noncompliant

def foo():
    raise  # Noncompliant
    try:
        raise  # Noncompliant
    except ValueError as e:
        handle_error()
    except:
        raise
    else:
        raise  # Noncompliant
    finally:
        raise

def handle_error():
    raise  # Noncompliant. This works but is hard to un
```

Compliant Solution

```
raise ValueError()

def foo():
    raise ValueError()
```

The "print" statement should not be used

Code Smell

"<>" should not be used to test inequality

Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Code Smell

Unused assignments should be removed

Code Smell

```
    try:
        raise ValueError()
    except:
        raise
    else:
        raise ValueError()
    finally:
        raise
```

See

- Python Documentation - The `raise` statement

Available In:

sonarlint | sonarcloud | sonarqube