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...

Method overrides should not change contracts

Analyze your code

Code Smell Critical suspicious

Because a subclass instance may be used as an instance of the superclass, overriding methods should uphold the aspects of the superclass contract that relate to the Liskov Substitution Principle. Specifically, an overriding method should be callable with the same parameters as the overriden one.

The following modifications are ok:

- Adding an optional parameter, i.e. with a default value, as long as they don't change the order of positional parameters.
- Renaming a positional-only parameter.
- Reordering keyword-only parameters.
- Adding a default value to an existing parameter.
- Changing the default value of an existing parameter.
- Extend the ways a parameter can be provided, i.e. change a keyword-only or positional-only parameter to a keyword-or-positional parameter. This is only true if the order of positional parameters doesn't change. New positional parameters should be placed at the end.
- Adding a vararg parameter (`*args`).
- Adding a keywords parameter (`**kwargs`).

The following modifications are not ok:

- Removing parameters, even when they have default values.
- Adding mandatory parameters, i.e. without a default value.
- Removing the default value of a parameter.
- Reordering parameters, except when they are keyword-only parameters.
- Removing some ways of providing a parameter. If a parameter could be passed as keyword it should still be possible to pass it as keyword, and the same is true for positional parameters.
- Removing a vararg parameter (`*args`).
- Removing a keywords parameter (`**kwargs`).

This rule raises an issue when the signature of an overriding method does not accept the same parameters as the overriden one. Only instance methods are considered, class methods and static methods are ignored.

Noncompliant Code Example

```
class ParentClass(object):
    def mymethod(self, param1):
        pass

class ChildClassMore(ParentClass):
    def mymethod(self, param1, param2, param3): # Noncomplia
        # Remove parameter "param2" or provide a default val
        # Remove parameter "param3" or provide a default val
        pass

class ChildClassLess(ParentClass):
    def mymethod(self): # Noncompliant. Add missing paramete
        pass

class ChildClassReordered(ParentClass):
```

Using hardcoded IP addresses is security-sensitive

Security Hotspot

Regular expression quantifiers and character classes should be used concisely

Code Smell

Character classes should be preferred over reluctant quantifiers in regular expressions

Code Smell

A subclass should not be in the same "except" statement as a parent class

```
    def mymethod(self, inserted, param1): # Noncompliant
        # Remove parameters "inserted" or provide a default
        pass
```

Compliant Solution

```
class ParentClass(object):
    def mymethod(self, param1):
        pass

class ChildClassMore(ParentClass):
    def mymethod(self, param1, param2=None, param3=None):
        pass

class ChildClassLess(ParentClass):
    def mymethod(self, param1=None):
        pass

class ChildClassReordered(ParentClass):
    def mymethod(self, param1, inserted=None):
        pass
```

Exceptions

In theory renaming parameters also breaks Liskov Substitution Principle. Arguments can't be passed via keyword arguments anymore. However, as PEP-570 says, it is common to rename parameters when it improves code readability and when arguments are always passed by position.

"Positional-Only Parameters" were introduced in python 3.8 to solve this problem. As most programs will need to support older versions of python, this rule won't raise an issue on renamed parameters.

```
class ParentClass(object):
    def mymethod(self, param1):
        pass

class ChildClassRenamed(ParentClass):
    def mymethod(self, renamed): # No issue but this is susp
        pass
```

See

- Wikipedia - Liskov substitution principle
- Python Enhancement Proposal (PEP) 3102 - Keyword-Only Arguments
- Python Enhancement Proposal (PEP) 570 - Python Positional-Only Parameters

Available In:

sonarlint | sonarcloud | sonarqube