



5

Oii

HTML

Java

Kotlin

PHP

PL/I

PL/SQL

**Python** 

**RPG** 

Ruby

Scala

Swift

Text

T-SQL

**VB.NET** 

VB<sub>6</sub>

**XML** 

Terraform

**TypeScript** 

JavaScript

Objective C

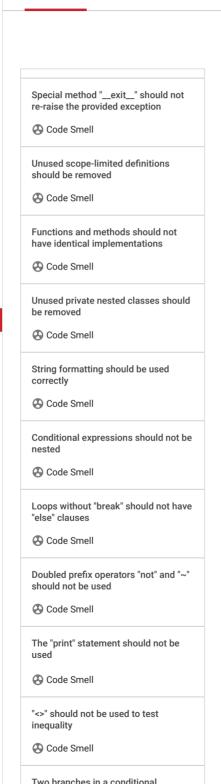


# Python static code analysis

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



Tags



structure should not have exactly the

Unused assignments should be

same implementation

Code Smell

```
"self" should be the first argument
                                               Analyze your code
to instance methods
convention confusing suspicious
Instance methods, i.e. methods not annotated with @classmethod or
@staticmethod, are expected to have at least one parameter. This parameter will
reference the object instance on which the method is called. By convention, this first
parameter is named "self".
Naming the "self" parameter differently is confusing. It might also indicate that the
```

Search by name...

Note also that creating methods which are used as static methods without the @staticmethod decorator is a bad practice because calling these methods on an

"self" parameter was forgotten, in which case calling the method will most probably

instance will raise a TypeError. Either move the method out of the class or decorate it with @staticmethod.

This rule raises an issue when the first parameter of an instance method is not called "self".

## **Noncompliant Code Example**

```
class MyClass:
   def send_request(request): # Noncompliant. "self" was p
        print("send_request")
class ClassWithStaticMethod:
   def static_method(param): # Noncompliant
       print(param)
ClassWithStaticMethod().static method(42) # Method is avail
```

# **Compliant Solution**

```
class MyClass:
    def send_request(self, request):
        print("send_request")
class ClassWithStaticMethod:
    @staticmethod
    def static_method(param):
        print(param)
ClassWithStaticMethod().static_method(42)
```

## Exceptions

This rule will also accept "cls" or "mcs" as first parameter's name for metaclasses'

No issue will be raised for methods called \_\_init\_subclass\_\_, \_class\_getitem\_\_ or \_\_new\_\_ as these methods' first parameter is a class.

You can also disable issues on methods decorated with a specific decorator. Add these decorators to this rule's "ignoreDecorators" parameter.

a assignments snould be removed Code Smell

A field should not duplicate the name of its containing class

A Code Smell

Function names should comply with a naming convention

Code Smell

Functions and lambdas should not reference variables defined in enclosing loops

Code Smell

With "ignoredDecorators" set to "abstractmethod"

from abc import abstractmethod, ABC class MyClass(ABC): @abstractmethod def method(): # No issue, even if it is better in this

- Python documentation Method Objects
- PEP8 Function and Method Arguments

Available In:

sonarlint ⊕ | sonarcloud 👌 | sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

Privacy Policy