

- 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...



should be removed

Code Smell

Functions and methods should not have identical implementations

Code Smell

Unused private nested classes should be removed

Code Smell

String formatting should be used correctly

Code Smell

Conditional expressions should not be nested

Code Smell

Loops without "break" should not have "else" clauses

Code Smell

Doubled prefix operators "not" and "~" should not be used

Code Smell

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

Function parameters' default values should not be modified or assigned

Analyze your code

Code Smell Critical ?

In Python function parameters can have default values. These default values are expressions which are executed when the function is defined, i.e. only once. The same default value will be used every time the function is called, thus modifying it will have an effect on every subsequent call. This can create some very confusing bugs.

It is also a bad idea to store mutable default value in another object (ex: as an attribute). Multiple instances will then share the same value and modifying one object will modify all of them.

This rule raises an issue when:

- a default value is either modified in the function or assigned to anything else than a variable and it has one of the following types:
 - builtins: set, dict, list.
 - collections** module: deque, UserList, ChainMap, Counter, OrderedDict, defaultdict, UserDict.
- or when an attribute of a default value is assigned.

Noncompliant Code Example

In the following example, the parameter "param" has `list()` as a default value. This list is created only once and then reused in every call. Thus when it appends 'a' to this list, the next call will have ['a'] as a default value.

```
def myfunction(param=list()): # Noncompliant.
    param.append('a') # modification of the default value
    return param

print(myfunction()) # returns ['a']
print(myfunction()) # returns ['a', 'a']
print(myfunction()) # returns ['a', 'a', 'a']
```


In the following example the same list is used for multiple instances of `MyClass.param`.

```
class MyClass:
    def __init__(self, param=list()): # Noncompliant.
        self.param = param # The same list is used for all instances


    def process(self, value):
        self.param.append(value) # modifying the same list

a1, a2 = (MyClass(), MyClass())
a1.process("value")
print(a1.param) # ['value']
print(a2.param) # ['value']
```


A field should not duplicate the name of its containing class

 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

Sections of code should not be commented out

 Code Smell

Unused function parameters should

Compliant Solution

```
def myfunction(param=None):
    if param is None:
        param = list()
    param.append('a')
    return param

print(myfunction()) # returns ['a']
print(myfunction()) # returns ['a']
print(myfunction()) # returns ['a']
```

```
class MyClass:
    def __init__(self, param=None):
        if param is None:
            self.param = list()
        else:
            self.param = param

    def process(self, value):
        self.param.append(value)

a1, a2 = (MyClass(), MyClass())
a1.process("value")
print(a1.param) # ['value']
print(a2.param) # []
```

Exceptions

In some very rare cases modifying a default value is ok. For example, default values can be used as a cache.

No issue will be raised when the parameter's name contains "cache" or "memo" (as in memoization).

See

- [The Hitchhiker's Guide to Python - Common Gotchas](#)
- [Python documentation - Function definitions](#)

Available In:

sonarlint  | **sonarcloud**  | **sonarqube** 