

# 代码静态检查

---

清华大学软件学院 刘强、陈光



# 代码评审技术

代码审查 ( Code Review ) 是一种用来确认方案设计和代码实现的质量保证机制，它通过阅读代码来检查源代码与编码规范的符合性以及代码的质量。

## 代码审查的作用

- 检查设计的合理性
- 互为 Backup
- 分享知识、设计、技术
- 增加代码可读性
- 处理代码中的“地雷区”



# 缺陷检查表

## 编程规范

- 按照具体编程语言的编码规范进行检查，包括命名规则、程序注释、缩进排版、声明与初始化、语句格式等。

## 面向对象设计

- 类的设计和抽象是否合适
- 是否符合面向接口编程的思想
- 是否使用合适的设计模式

## 性能方面

- 在出现海量数据时，队列、表、文件在传输、上载等方面是否会出现问题，是否控制如分配的内存块大小、队列长度等
- 对 Hashtable、Vector 等集合类数据结构的选择和设置是否合适
- 有无滥用 String 对象的现象
- 是否采用通用的线程池、对象池等高速缓存技术以提高性能
- 类的接口是否定义良好，如参数类型等应避免内部转换

## 性能方面（续）

- 是否采用内存或硬盘缓冲机制以提高效率？
- 并发访问时的应对策略
- I/O 方面是否使用了合适的类或采用良好的方法以提高性能（如减少序列化、使用 buffer 类封装流等）
- 同步方法的使用是否得当，是否过度使用？
- 递归方法中的迭代次数是否合适（应保证在合理的栈空间范围内）
- 如果调用了阻塞方法，是否考虑了保证性能的措施
- 避免过度优化，对性能要求高的代码是否使用profile工具

## 资源释放处理

- 分配的内存是否释放，尤其在错误处理路径上（如 C/C++）
- 错误发生时是否所有对象被释放，如数据库连接、Socket、文件等
- 是否同一个对象被释放多次（如 C/C++）
- 代码是否保存准确的对象引用计数

# 缺陷检查表

## 程序流程

- 循环结束条件是否准确
- 是否避免了死循环的产生
- 对循环的处理是否合适，应考虑到性能方面的影响

## 线程安全

- 代码中所有的全局变量是否是线程安全的
- 需要被多个线程访问的对象是否线程安全，检查有无通过同步方法保护
- 同步对象上的锁是否按相同的顺序获得和释放以避免死锁，注意错误处理代码
- 是否存在可能的死锁或是竞争，当用到多个锁时，避免出现类似情况：线程A获得锁1，然后锁2，线程B获得锁2，然后锁1
- 在保证线程安全的同时，注意避免过度使用同步，导致性能降低

## 数据库处理

- 数据库设计或SQL语句是否便于移植（注意与性能会存在冲突）
- 数据库资源是否正常关闭和释放

## 数据库处理（续）

- 数据库访问模块是否正确封装，便于管理和提高性能
- 是否采用合适的事务隔离级别
- 是否采用存储过程以提高性能
- 是否采用 PreparedStatement 以提高性能

## 通讯方面

- Socket 通讯是否存在长期阻塞问题
- 发送接收的数据流是否采用缓冲机制
- Socket 超时处理和异常处理
- 数据传输的流量控制问题

## JAVA对象处理

- 对象生命周期的处理，是否对象引用已失效可设置 null 并被回收
- 在对象传值和传参方面有无问题，对象的 clone 方法使用是否过度
- 是否大量经常地创建临时对象
- 是否尽量使用局部对象（堆栈对象）
- 在只需要对象引用的地方是否创建了新的对象实例

# 缺陷检查表

## 异常处理

- 每次当方法返回时是否正确处理了异常，如最简单的处理是记录日志到日志文件中
- 是否对数据的值和范围是否合法进行校验，包括使用断言
- 在出错路径上是否所有的资源和内存都已经释放
- 所有抛出的异常是否都得到正确的处理，特别是对子方法抛出的异常，在整个调用栈中必须能够被捕捉并处理
- 当调用导致错误发生时，方法的调用者应该得到一个通知
- 不要忘了对错误处理部分的代码进行测试，很多代码在正常情况下执行良好，而一旦出错整个系统就崩溃了？

## 方法（函数）

- 方法的参数是否都做了校验
- 数组类结构是否做了边界校验

## 方法（续）

- 变量在使用前是否做了初始化
- 返回堆对象的引用，不要返回栈对象的引用
- 方法的 API 是否被良好定义，即是否尽量面向接口编程，以便于维护和重构

## 安全方面

- 对命令行执行的代码，需要详细检查命令行参数
- WEB 类程序检查是否对访问参数进行合法性验证
- 重要信息的保存是否选用合适的加密算法
- 通讯时考虑是否选用安全的通讯方式

## 其他

- 日志是否正常输出和控制
- 配置信息如何获得，是否有硬编码

# Python代码分析工具

---



<http://www.pylint.org/>

**Pylint** 是一个 Python 代码分析工具，它用于分析 Python 代码的错误，查找不符合代码风格标准（Pylint 默认使用的代码风格是 PEP 8）和有潜在问题的代码。

# Python代码分析工具

---



Pylint功能

检查代码风格是否符合PEP8规范

检查代码是否存在常见的错误和违反最佳实践

<http://pylint-messages.wikidot.com/all-messages>

检查重复的代码

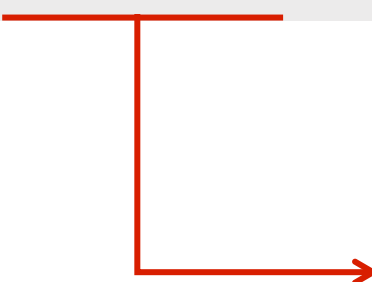
# PyLint 安装与使用

---

PyLint 安装 : `pip install -U pylint` # 安装最新版的PyLint

PyLint 使用 :

`pylint [options] module_or_package_or_file` # 对模块/包/文件运行pylint



- `--rcfile=<file>` 指定检查的配置文件
- `--ignore=<file>` 不进行检查的文件列表
- `--disable=<msg ids>` 关闭某种类型的检查
- `-f <format>` 报告类型, 如html



# PyLint 安装与使用

`pylint --help-msg <msg-id>` 查看某种类型问题的帮助

例如：`pylint --help-msg invalid-name`

```
No config file found, using default configuration
:invalid-name (C0103): *Invalid %s name "%s"%s*
    Used when the name doesn't match the regular expression associated to its type
    (constant, variable, class...). This message belongs to the basic checker.
```

`pylint --generate-rcfile` 根据当前配置生成配置文件

- 通常结合 `--disable` 使用
- Pylint 默认首先读取当前目录下的`pylintrc`文件作为配置
- 生成`pylintrc`后可以将配置加入版本控制系统供团队使用

## Pylint使用示例 simplecaeser.py

pylint simplecaeser.py

```
# coding=utf-8
import string

shift = 3
choice = input("would you like to encode or decode?")
word = (input("Please enter text"))
letters = string.ascii_letters + string.punctuation + string.digits
encoded = ""
if choice == "encode":
    for letter in word:
        if letter == ' ':
            encoded = encoded + ' '
        else:
            x = letters.index(letter) + shift
            encoded = encoded + letters[x]
    encoded = encoded + choice
else:
    for letter in word:
        if letter == ' ':
            encoded = encoded + ' '
        else:
            x = letters.index(letter) - shift
            encoded = encoded + letters[x]

print(encoded)
```

No config file found, using default configuration

\*\*\*\*\* Module simplecaeser

C: 1, 0: Missing module docstring (missing-docstring)  
 C: 4, 0: Invalid constant name "shift" (invalid-name)  
 C: 5, 0: Invalid constant name "choice" (invalid-name)  
 C: 6, 0: Invalid constant name "word" (invalid-name)  
 C: 7, 0: Invalid constant name "letters" (invalid-name)  
 C: 8, 0: Invalid constant name "encoded" (invalid-name)

类型：

- C ( Convention, 约定 )
- R ( Refactor, 重构 )
- W ( Warning, 警告 )
- E ( Error, 错误 )

错误类型：出现位置（行，列）：说明信息（问题标识）

Report

=====

20 statements analysed.

Statistics by type ← 列出所查文件的模块、类、方法和函数的数量

-----

+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
type	number	old number	difference	%documented	%badname
+=====+	+=====+	+=====+	+=====+	+=====+	+=====+
module	1	1	=	0.00	0.00
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
class	0	0	=	0	0
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
method	0	0	=	0	0
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
function	0	0	=	0	0
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+

### Raw metrics

type	number	%	previous	difference
code	22	95.65	22	=
docstring	0	0.00	0	=
comment	1	4.35	1	=
empty	0	0.00	0	=

显示前一次结果，可以做对比

### Duplication

	now	previous	difference
nb duplicated lines	0	0	=
percent duplicated lines	0.000	0.000	=

### Global evaluation

Your code has been rated at 7.00/10 (previous run: 7.00/10, +0.00)

### Messages by category

type	number	previous	difference
convention	6	6	=
refactor	0	0	=
warning	0	0	=
error	0	0	=

### Messages

message id	occurrences
invalid-name	5
missing-docstring	1

分析实际的代码、文档字符串、注释、空行等分别有多少行以及所占百分比

对前面源代码分析信息进行汇总，分别列出R, W, E, C各是多少

显示是否有重复的行，由此可以判断是否有可以抽离出来的部分进行重构。

根据问题标识进行汇总

对分析文件的一个评分，10分为满分

# 案例：生命游戏

---



```
***** Module game_map
C:108, 0: Line too long (108/100) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
C: 97, 12: Invalid variable name "d" (invalid-name)
```

```
game_map.py x
107 def get_neighbor_count_map(self):
108     return [[self.get_neighbor_count(row, col) for col in range(self.cols)] for row in range(self.rows)]
```

```
C:124, 4: Missing method docstring (missing-docstring)
W:132,27: Used builtin function 'map' (bad-builtin)
```

```
***** Module game_timer
C: 1, 0: Missing module docstring (missing-docstring)
C: 9, 0: Missing class docstring (missing-docstring)
C: 22, 4: Missing method docstring (missing-docstring)
C: 29, 4: Missing method docstring (missing-docstring)
```

```
***** Module life_game
C: 1, 0: Missing module docstring (missing-docstring)
C: 9, 0: Missing class docstring (missing-docstring)
C: 15, 4: Missing method docstring (missing-docstring)
C: 19, 4: Missing method docstring (missing-docstring)
C: 23,16: Invalid variable name "nc" (invalid-name)
```

```
***** Module main
C: 13, 0: Line too long (120/100) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
```

**line-too-long** : 行过长  
PEP8规定python中每行的宽度不能超过100字符

```
main.py x
12 def print_usage():
13     print("Usage:", sys.argv[0], '<map_rows=10>', '<map_cols=10>', '<life_init_possibility=0.5>', '<time_int
```

```
***** Module game_map
C:108, 0: Line too long (108/100) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
C: 97,12: Invalid variable name "d" (invalid-name)
C:107, 4: Missing method docstring (missing-docstring)
C:110, 4: Missing method docstring (missing-docstring)
C:124, 4: Missing method docstring (missing-docstring)
W:132,27: Used builtin function 'map' (bad-builtin)
***** Module game_timer
C: 1, 0: Missing module docstring (missing-docstring)
C: 9, 0: Missing class docstring (missing-docstring)
C: 22, 4: Missing method docstring (missing-docstring)
C: 29, 4: Missing method docstring (missing-docstring)
***** Module life_game
C: 1, 0: Missing module docstring (missing-docstring)
C: 9, 0: Missing class docstring (missing-docstring)
C: 15, 4: Missing method docstring (missing-docstring)
C: 19, 4: Missing method docstring (missing-docstring)
C: 23,16: Invalid variable name "nc" (invalid-name)
***** Module main
C: 13, 0: Line too long (120/100) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
C: 12, 0: Missing function docstring (missing-docstring)
W: 46, 4: No exception type(s) specified (bare-except)
```

**missing-docstring:**  
通常模块和方法都要添加docstring



```
***** Module game_map
C:108, 0: Line too long (108/100) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
C: 97,12: Invalid variable name "d" (invalid-name)
C:107, 4: Missing method docstring (missing-docstring)
C:110, 4: Missing method docstring (missing-docstring)
C:124, 4: Missing method docstring (missing-docstring)
W:132,27: Used builtin function 'map' (bad-builtin)
***** Module game_timer
C: 1, 0: Missing module docstring (missing-docstring)
C: 9, 0: Missing class docstring (missing-docstring)
C: 22, 4: Missing method docstring (missing-docstring)
C: 29, 4: Missing method docstring (missing-docstring)
***** Module life_game
C: 1, 0: Missing module docstring (missing-docstring)
C: 9, 0: Missing class docstring (missing-docstring)
C: 15, 4: Missing method docstring (missing-docstring)
C: 19, 4: Missing method docstring (missing-docstring)
C: 23,16: Invalid variable name "nc" (invalid-name)
***** Module main
C: 13, 0: Line too long (120/100) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
C: 12, 0: Missing function docstring (missing-docstring)
W: 46, 4: No exception type(s) specified (bare-except)
```

**invalid-name:**

不好的命名

默认配置中变量名  
至少需要3个字符



```

***** Module game_map
C:108, 0: Line too long (108/100) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
C: 97,12: Invalid variable name "d" (invalid-name)
C:107, 4: Missing method docstring (missing-docstring)
C:110, 4: Missing method docstring (missing-docstring)
C:124, 4: Missing method docstring (missing-docstring)
W:132,27: Used builtin function 'map' (bad-builtin)
***** Module game_timer
C: 1, 0: Missing module docstring (missing-docstring)
124 def print_map(self, cell_maps=None, sep=' '):
125     if not cell_maps:
126         cell_maps = ['0', '1']
127     if not isinstance(cell_maps, list) and not isinstance(cell_maps, dict):
128         raise TypeError("cell_maps should be list or dict")
129     if not isinstance(sep, str):
130         raise TypeError("sep should be string")
131     for row in self.cells:
132         print(sep.join(map(lambda cell: cell_maps[cell], row)))
C: 23,16: Invalid variable name "nc" (invalid-name)
***** Module main
C: 13, 0: Line too long (120/100) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
C: 12, 0: Missing function docstring (missing-docstring)
W: 46, 4: No exception type(s) specified (bare-except)

```

### bad-builtin:

尽可能不要使用map、filter这样的内置函数，而应该使用python特有的列表推导

```

***** Module game_map
C:108, 0: Line too long (108/100) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
C: 97,12: Invalid variable name "d" (invalid-name)
C:107, 4: Missing method docstring (missing-docstring)
C:110, 4: Missing method docstring (missing-docstring)
C:124, 4: Missing method docstring (missing-docstring)
W:132,27: Used builtin function 'map' (bad-builtin)
***** Module game_timer
C: 1, 0: Missing module docstring (missing-docstring)
C: 9, 0: Missing class docstring (missing-docstring)
C: 22, 4: Missing method docstring (missing-docstring)
C: 29, 4: Missing method docstring (missing-docstring)
***** Module life_game
C: 1, 0: Missing module docstring (missing-docstring)
C: 9, 0: Missing class docstring (missing-docstring)
43     if __name__ == '__main__':
44         try:
45             exit(main(sys.argv[1:]))
46         except:
47             traceback.print_exc()
48             print_usage()
C: 1, 0: Missing module docstring (missing-docstring)
C: 12, 0: Missing function docstring (missing-docstring)
W: 46, 4: No exception type(s) specified (bare-except)

```

**bare-except:**

在使用except关键字的时候没有指定异常类型

# 案例：生命游戏

line-too-long: 通过换行来使行变短

```
game_map.py x
107 def get_neighbor_count_map(self):
108     return [
109         [self.get_neighbor_count(row, col) for col in range(self.cols)]
110         for row in range(self.rows)
111     ]
```

```
main.py x
12 def print_usage():
13     print("Usage:", sys.argv[0], '<map_rows=10>', '<map_cols=10>', '<life_init_possibility=0.5>',
14         '<time_interval=1.0>')
```

# 案例：生命游戏



**missing-docstring:** 应该补充有意义的说明，这里暂时关闭这项检查

**invalid-name:**

- 默认的局部变量命名规则是[\[a-z\\_\]\[a-z0-9\\_\]{2,30}\\$](#)
- 可以通过**variable-rgx** 选项来重新设定
- 这里修改为[\[a-z\\_\]\[a-z0-9\\_\]{0,30}\\$](#)

```
(python3) [guangchen@Guangs-MacBook-Pro: life_game]$ \
> pylint --disable=missing-docstring \
>         --variable-rgx="[a-z_][a-z0-9_]{0,30}" \
>         --generate-rcfile > pylintrc
```

# 案例：生命游戏

bad-builtin: 修改map为列表推导

```
127 def print_map(self, cell_maps=None, sep=' '):
128     if not cell_maps:
129         cell_maps = ['0', '1']
130     if not isinstance(cell_maps, list) and not isinstance(cell_maps, dict):
131         raise TypeError("cell_maps should be list or dict")
132     if not isinstance(sep, str):
133         raise TypeError("sep should be string")
134     for row in self.cells:
135         print(sep.join([cell_maps[cell] for cell in row]))
136
```

bare-except: 这里我们对特定文件(main.py)  
关闭该检查，并在文件开头加注释

```
main.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  #
4  # @author Epsirom
5  # pylint: disable=bare-except
```

# 案例：生命游戏



```
report.txt x
1 ***** Module main
2 I: 5, 0: Locally disabling bare-except (W0702) (locally-disabled)
3
94 Global evaluation
95 -----
96 Your code has been rated at 10.00/10
```

# 代码静态分析工具



- **Checkstyle** : 通过对代码编码格式、命名约定、Javadoc、类设计等方面进行代码规范和风格的检查。
- **FindBugs** : 通过检查类文件或JAR文件，将字节码与一组缺陷模式进行对比从而发现代码缺陷，完成静态代码分析。
- **PMD** : 通过其内置的编码规则对Java代码进行静态检查，主要包括对潜在的Bug、未使用的代码、重复的代码、循环体创建新对象等问题的检验。
- **Jtest** : 能够按照其内置的超过800条的Java编码规范自动检查并纠正这些隐蔽且难以修复的编码错误，同时还支持用户自定义编码规则，帮助用户预防一些特殊用法的错误。



# 代码静态分析工具



- 微软Visual Studio中的代码分析工具可以检查代码中是否存在一些常见缺陷和违反良好编程习惯的情况。
- Lint是一种静态程序分析工具，目前已形成了一系列工具。它侧重于代码的逻辑分析，发现代码中一些潜在的错误，如数组访问越界、内存泄漏、使用未初始化变量等。
- JSHint ( [www.jshint.com](http://www.jshint.com) ) 是一款JavaScript代码验证工具，用于检测代码中的错误和潜在问题。
- CSSLint ( [csslint.net](http://csslint.net) ) 是一款CSS代码检查工具，可以进行基本语法检查以及使用一套预设的规则来检查代码中的问题。
- HTMLHint ( [htmlhint.com](http://htmlhint.com) ) 是一款基于JS开发的静态扫描组件，支持所有浏览器和Nodejs平台，可集成到IDE环境或编译系统中。





# 谢谢大家！

---

## THANKS

