Lab2：代码评审与程序性能优化

实验要求

- 针对**Lab1**所完成的代码，进行代码评审**(**走查**)**和性能分析，从时间性能角度对代码进行优化；
- 练习代码评审的两个方面：静态分析、动态分析**(profiling)**；
- 使用以下三个工具完成实验：
  - Pylint
  - PyChecker
  - Profile
- 按**Lab1**的分组方式，两人一组，随机分配另一组的代码作为本组评审和分析的对象，实验期间不能与原作者进行沟通。

Pylint

- **Pylint是一个Python代码检查工具，可以帮助程序员方便地检查程序代码的语法和风格http://www.pylint.org/**
- **Coding Standard**
  - checking line-code's length,
  - checking if variable names are well-formed according to your coding standard
  - checking if imported modules are used
- **Error detection**
  - checking if declared interfaces are truly implemented
  - checking if modules are imported
- **Refactoring help**
  - Pylint detects duplicated code
- **Fully customizable**
  - Modify your pylint to customize which errors or conventions are important to you. The big advantage with Pylint is that it is configurable, customizable, and you can easily write a small plugin to add a personal feature.
- **IDE integration**

Install Pylint

- setuptools
- Download ez_install.py
- pip install pylint (windows)

Pylint Results

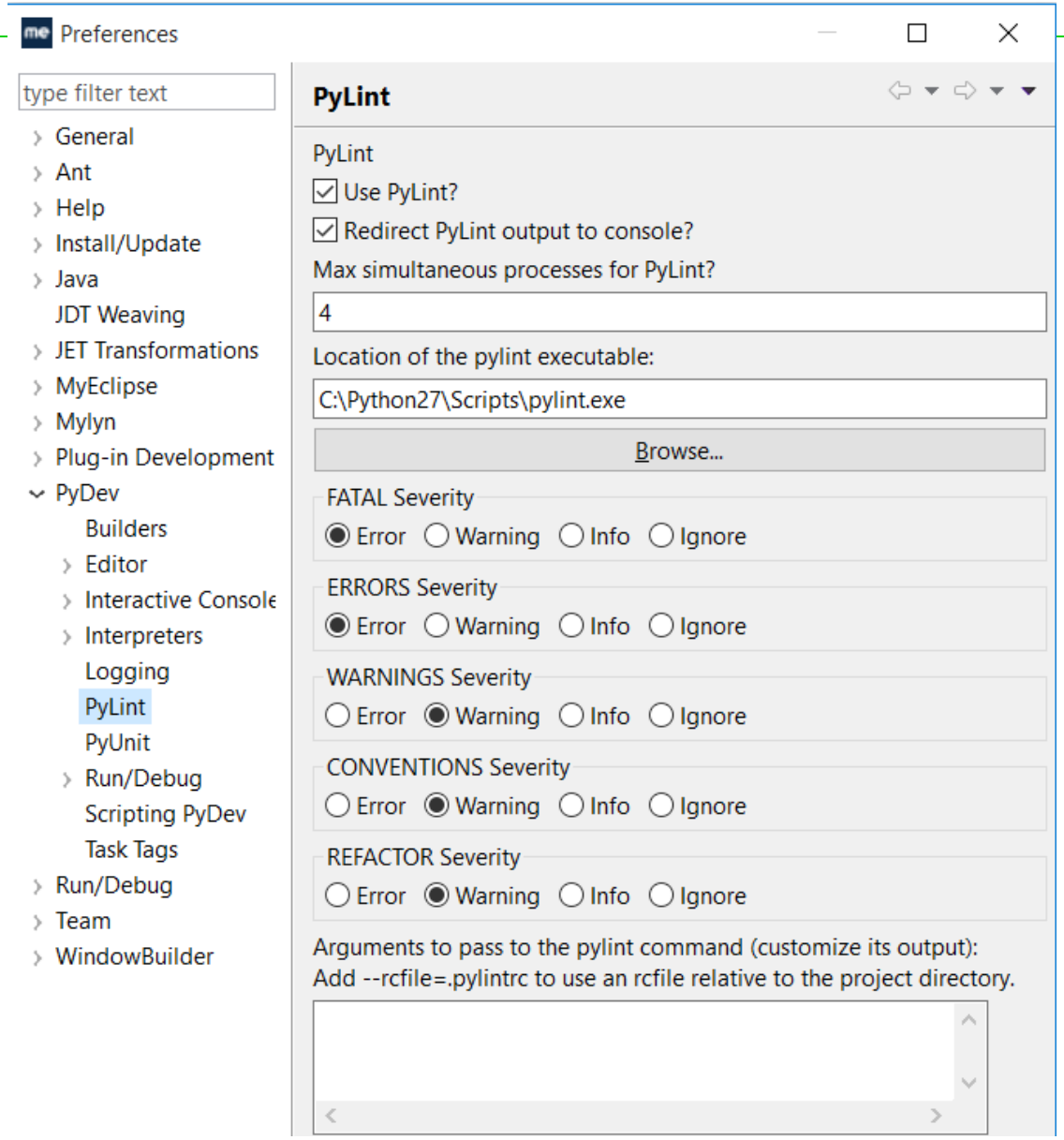
C:\WINDOWS\system32\cmd.exe

```
                    information. See doc for all details

C:\Python27>pylint c:\lab1.py
No config file found, using default configuration
************* Module lab1
C:  9, 0: Line too long (113/100) (line-too-long)
C: 10, 0: Line too long (108/100) (line-too-long)
C: 11, 0: Exactly one space required around assignment
reqStability=[0.0 for col in range(0,processNum,1)]  #每个流程的稳定
            ^ (bad-whitespace)
C: 11, 0: Exactly one space required after comma
reqStability=[0.0 for col in range(0,processNum,1)]  #每个流程的稳定
                                    ^ (bad-whitespace)
C: 11, 0: Exactly one space required after comma
reqStability=[0.0 for col in range(0,processNum,1)]  #每个流程的稳定
                                              ^ (bad-whitespace)
C: 12, 0: Exactly one space required around assignment
reqCost=[0.0 for col in range(processNum)]  #每个流程成本的要求
       ^ (bad-whitespace)
C: 14, 0: Exactly one space required around assignment
searchNode=[[False for col in range(serviceTypeNumber)] for row in r
           ^ (bad-whitespace)
C: 15, 0: Exactly one space required around assignment
processMark=[0 for col in range(processNum)]  #每个流程中的活动数
            ^ (bad-whitespace)
C: 16, 0: Exactly one space required around assignment
finalNode=[0 for col in range(processNum)]  #每个流程中最大编号的服务
         ^ (bad-whitespace)
C: 18, 0: Exactly one space required around assignment
cutCost=[0.0 for col in range(serviceTypeNumber)]  #剪枝的成本边界,
       ^ (bad-whitespace)
C: 19, 0: Exactly one space required around assignment
cutStability=[0.0 for col in range(serviceTypeNumber)]  #剪枝的stabi
            ^ (bad-whitespace)
C: 20, 0: Exactly one space required around assignment
lowestCost=[0.0 for col in range(serviceTypeNumber)]  #最小成本
           ^ (bad-whitespace)
C: 21, 0: Exactly one space required around assignment
highestStability=[0.0 for col in range(serviceTypeNumber)]  #最大稳定
                ^ (bad-whitespace)
C: 22, 0: Line too long (109/100) (line-too-long)
C: 25, 0: Exactly one space required around assignment
answerTR=[0.0 for col in range(processNum)]  #每个流程中解的TR
        ^ (bad-whitespace)
```

C:\WINDOWS\system32\cmd.exe

```
Statistics by type
------------------

+---------+-------+-----------+-----------+-------------+---------+
|type     |number |old number |difference |%documented |%badname |
+=========+=======+===========+===========+=============+=========+
|module   |1      |NC         |NC         |0.00         |0.00     |
+---------+-------+-----------+-----------+-------------+---------+
|class    |0      |NC         |NC         |0            |0        |
+---------+-------+-----------+-----------+-------------+---------+
|method   |0      |NC         |NC         |0            |0        |
+---------+-------+-----------+-----------+-------------+---------+
|function |4      |NC         |NC         |0.00         |100.00   |
+---------+-------+-----------+-----------+-------------+---------+


Raw metrics
-----------

+----------+-------+------+---------+-----------+
|type      |number |%     |previous |difference |
+==========+=======+======+=========+===========+
|code      |118    |84.89 |NC       |NC         |
+----------+-------+------+---------+-----------+
|docstring |4      |2.88  |NC       |NC         |
+----------+-------+------+---------+-----------+
|comment   |7      |5.04  |NC       |NC         |
+----------+-------+------+---------+-----------+
|empty     |10     |7.19  |NC       |NC         |
+----------+-------+------+---------+-----------+


Duplication
-----------

+-------------------------+------+---------+-----------+
|                         |now   |previous |difference |
+=========================+======+=========+===========+
|nb duplicated lines      |0     |NC       |NC         |
+-------------------------+------+---------+-----------+
|percent duplicated lines |0.000 |NC       |NC         |
+-------------------------+------+---------+-----------+
```

Pylint+Pydev



Pylint+Pydev

PyDev - PythonLab1/Search.py - MyEclipse Enterprise Workbench

File Edit Source Refactoring Navigate Search Project MyEclipse Pydev Run Window Help

PyDev Package

- PythonLab1
  - Search.py
  - python1 (C:\Python2
- JavaLab1
- Lab1Java
- Servers

Search.java | Search.java | Search | MyEclipse Dashboard

```
1  #coding=utf-8
2
3  import os
4
5  import datetime
6  processNum = 3            #多少个过程
7  serviceTypeNumber = 12     #多少种服务
8  eachTypeServiceNumber = 500  #每种服务的候选服务数目
9
10 serviceStability = [[0.0000 for col in range(eachTypeServiceNumber)] for row in range(
```

Outline

type filter text

- os
- datetime
- processNum
- serviceTypeN
- eachTypeSer
- serviceStabili
- serviceCost
- reqStability
- reqCost
- searchNode
- processMark
- finalNode

Console | Console

PyLint

```
PyLint: Executing command line: C:\Python27\Scripts\pylint.exe C:\Users\Andre.Xu\Workspaces\MyEclipse 201
PyLint: The stdout of the command line is: ************* Module Search
C:  4, 0: Trailing whitespace (trailing-whitespace)
C:  9, 0: Line too long (116/100) (line-too-long)
C: 10, 0: Line too long (111/100) (line-too-long)
C: 11, 0: Exactly one space required around assignment
reqStability=[0.0000 for col in range(0,processNum,1)]  #每个流程的稳定性要求

            ^ (bad-whitespace)
C: 11, 0: Exactly one space required after comma
reqStability=[0.0000 for col in range(0,processNum,1)]  #每个流程的稳定性要求

                                    ^ (bad-whitespace)
C: 11, 0: Exactly one space required after comma
reqStability=[0.0000 for col in range(0,processNum,1)]  #每个流程的稳定性要求
```

PyChecker

- **PyChecker is a tool for finding bugs in python source code. It finds problems that are typically caught by a compiler for less dynamic languages, like C and C++. It is similar to lint. Because of the dynamic nature of python, some warnings may be incorrect; however, spurious warnings should be fairly infrequent.** **http://pychecker.sourceforge.net/**
- **Types of problems that can be found include:**
  - No global found (e.g., using a module without importing it)
  - Passing the wrong number of parameters to functions/methods/constructors
  - Passing the wrong number of parameters to builtin functions & methods
  - Using format strings that don't match arguments
  - Using class methods and attributes that don't exist
  - Changing signature when overriding a method
  - Redefining a function/class/method in the same scope
  - Using a variable before setting it
  - self is not the first parameter defined for a method
  - Unused globals and locals (module or variable)
  - Unused function/method arguments (can ignore self)
  - No doc strings in modules, classes, functions, and methods

PyChecker

- 下载地址：http://sourceforge.net/projects/pychecker/
- 从网页上下载pychecker-0.8.19.tar.gz，解压。
- 执行命令行python setup.py install 执行安装。这也是在Windows下python软件的通用安装方式。
- 使用方法： Pychecker 模块名

```
C:\WINDOWS\system32\cmd.exe

C:\>pychecker test.py

C:\>C:\Python27\python.exe C:\Python27\Lib\site-packages\pychecker\checker.py test.py
Processing module test (test.py)...
100

Warnings...

None

C:\>pychecker setup.py

C:\>C:\Python27\python.exe C:\Python27\Lib\site-packages\pychecker\checker.py setup.py
Processing module setup (setup.py)...

Warnings...

[system path]\distutils\command\bdist_wininst.py:277: Statement appears to have no effect

[system path]\distutils\command\build_scripts.py:78: No class attribute (dry_run) found
[system path]\distutils\command\build_scripts.py:95: No class attribute (dry_run) found
[system path]\distutils\command\build_scripts.py:118: (file) shadows builtin
[system path]\distutils\command\build_scripts.py:119: No class attribute (dry_run) found

[system path]\distutils\command\install_data.py:58: (dir) shadows builtin
[system path]\distutils\command\install_data.py:60: (dir) shadows builtin
[system path]\distutils\command\install_data.py:62: (dir) shadows builtin

[system path]\distutils\command\install_scripts.py:50: (file) shadows builtin
[system path]\distutils\command\install_scripts.py:51: No class attribute (dry_run) found

19 errors suppressed, use -#/--limit to increase the number of errors displayed

C:\>
```

Python Profile

- **It is a standard package.**
- **Running time analysis.**
- **Performance analysis.**

- **The Python Profilers** https://docs.python.org/2/library/profile.html

- 参考网站：**http://blog.csdn.net/gzlaiyonghao/article/details/1483728**

Profile example

- Source code

```
def foo():
    sum = 0
    for i in range(100):
        sum += i
    return sum

if __name__ == "__main__":
    foo()
```

- Run profile

```
def foo():
    sum = 0
    for i in range(100):
        sum += i
    return sum

if __name__ == "__main__":
    import profile
    profile.run("foo()")
```

- 方法2：直接运行python -m profile 模块名.py

Results

```
5 function calls in 0.143 CPU seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)
     1    0.000    0.000    0.000    0.000 :0(range)
     1    0.143    0.143    0.143    0.143 :0(setprofile)
     1    0.000    0.000    0.000    0.000 <string>:1(?)
     1    0.000    0.000    0.000    0.000 prof1.py:1(foo)
     1    0.000    0.000    0.143    0.143 profile:0(foo())
     0    0.000             0.000          profile:0(profiler)
```

ncalls	函数的被调用次数
tottime	函数总计运行时间，除去函数中调用的函数运行时间
percall	函数运行一次的平均时间，等于tottime/ncalls
cumtime	函数总计运行时间，含调用的函数运行时间
percall	函数运行一次的平均时间，等于cumtime/ncalls
filename:lineno(function)	函数所在的文件名，函数的行号，函数名

Memory_profiler

- **Use memory_profiler to analysis memory occupation.**
- **https://pypi.python.org/pypi/memory_profiler/**

```
@profile
def my_func():
a = [1] * (10 ** 6)
b = [2] * (2 * 10 ** 7)
del b
return a

if __name__ == '__main__':
my_func()
```

```
$ python -m memory_profiler example.py
```

```
Line # Mem usage Increment Line Contents
================================================
3 @profile
4 5.97 MB 0.00 MB def my_func():
5 13.61 MB 7.64 MB a = [1] * (10 ** 6)
6 166.20 MB 152.59 MB b = [2] * (2 * 10 ** 7)
7 13.61 MB -152.59 MB del b
8 13.61 MB 0.00 MB return a
```

实验过程

- 配置Pylint、Pychecker、Profile/Memory_profiler三个工具；
- 分别使用三种工具对Lab1原始代码进行评审和性能分析，记录结果，期间不要有任何修改：
- 对工具执行的结果进行人工分析
  - 对三者的结果进行对比，看它们发现问题的能力差异；
- 根据结果对源代码进行修正(代码规范、性能)；
- 重新使用工具进行评审和性能分析，直到无法再改进为止。

评判标准

- 是否可顺利配置三种工具；
- 能够充分理解这三种工具所检测的不符合规范的常见代码问题；
- 对代码做了充分的改进。

提交方式

- 请遵循实验报告模板撰写。

- 提交日期：**2015年10月24日 23:55**
- 提交两个文件到**CMS**：
  - 实验报告：命名规则“学号-Lab2-report.doc”
  - 优化之后的代码：命名规则“学号-Lab2-code.py”
  - 注意：提交的文件不要采用PDF格式，不要进行压缩。
- 同组的两人需分别提交上述文件。

结束