Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра «Информатика и вычислительной техники»

**ОТЧЕТ ПО ПРАКТИКЕ №5**

Инструментальные средства анализа кода программ

Преподаватель \_\_\_\_\_\_\_\_\_\_ В.С. Васильев

подпись, датаинициалы, фамилия

Студент КИ22-08Б, 032210274 \_\_\_\_\_\_\_\_\_\_ К.А. Астафьева

номер группы, зачетной книжкиподпись, датаинициалы, фамилия

Красноярск 2023

**Оглавление**

[**Статический анализ кода**: 3](#_Toc153742581)

[**Профайлинг:** 6](#_Toc153742582)

[**Профайлинг времени:** 6](#_Toc153742583)

[**Профайлинг памяти:** 6](#_Toc153742584)

[**Заключение:** 7](#_Toc153742585)

# 

# **Статический анализ кода**

Статический анализ будем проводить с помощью программного обеспечения Pylint.

Изначально в файлах DateClass.py, TeacherClass.py, function.py, main.py, test1.py, test2.py pylint выводил следующие ошибки:

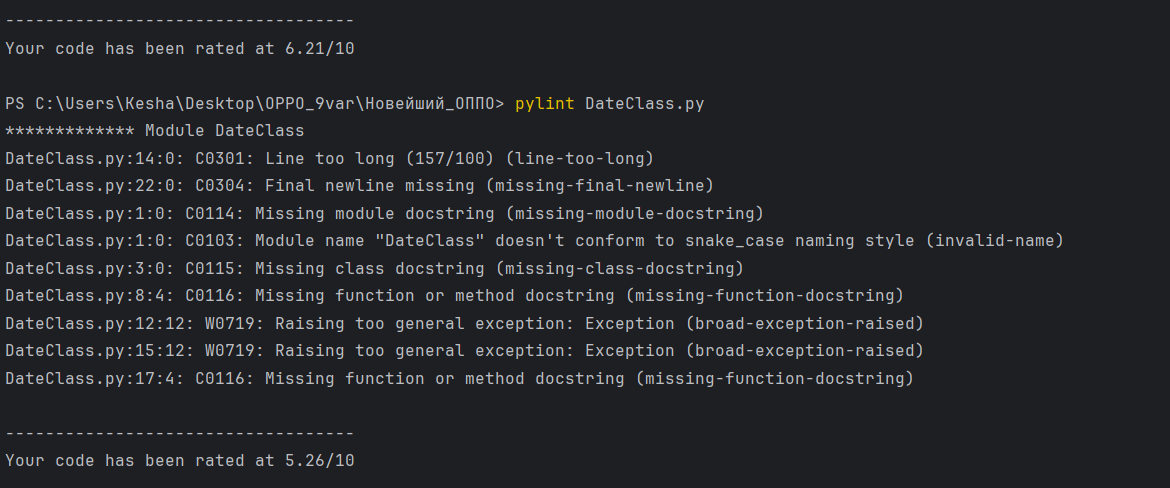


Рисунок 1 - ошибки в файле DateClass.py

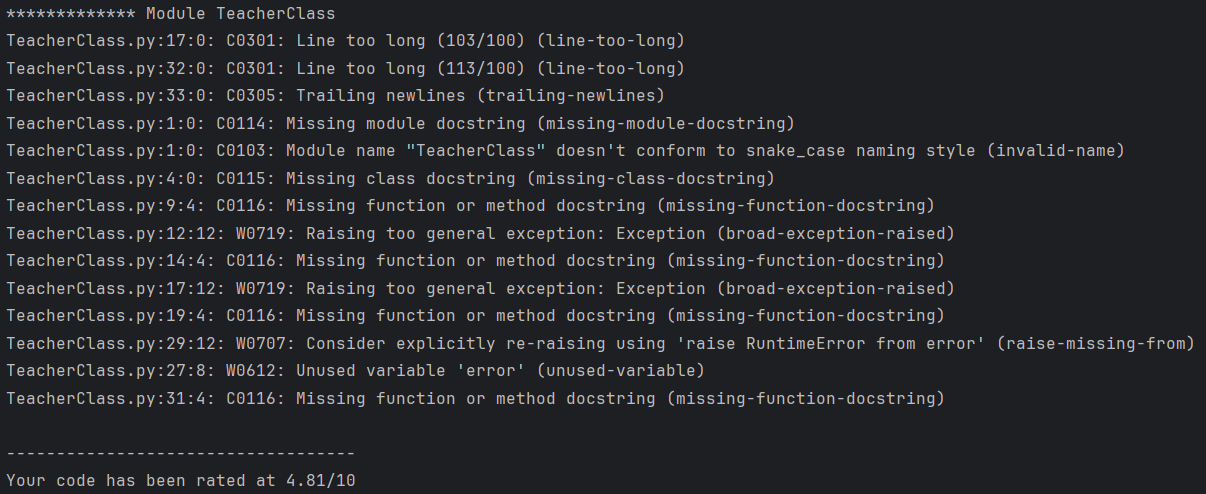


Рисунок 2 - ошибки в файле TeacherClass.py

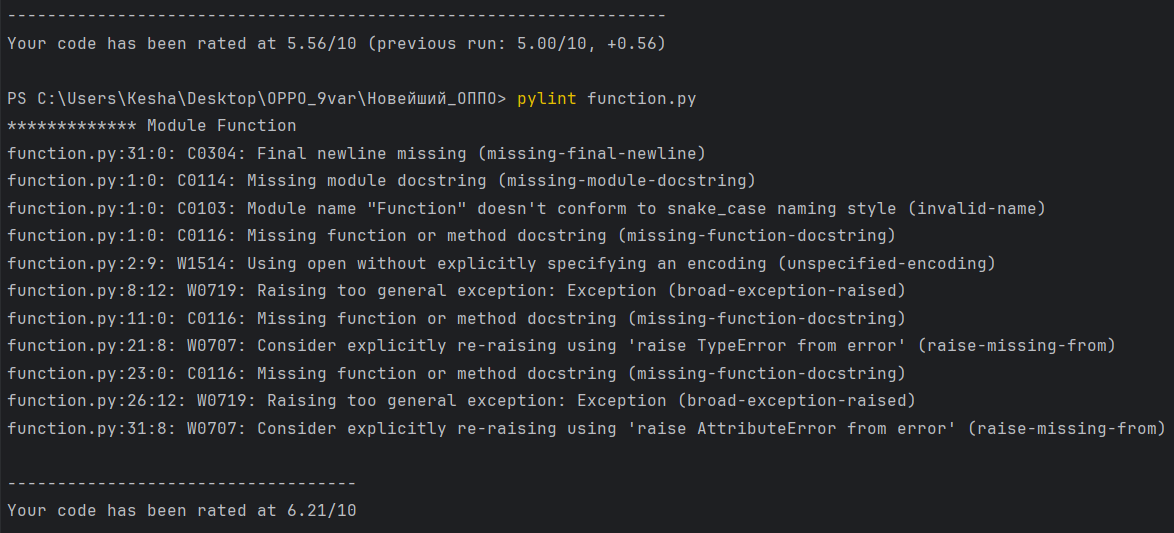


Рисунок 3 - ошибки в файле Function.py

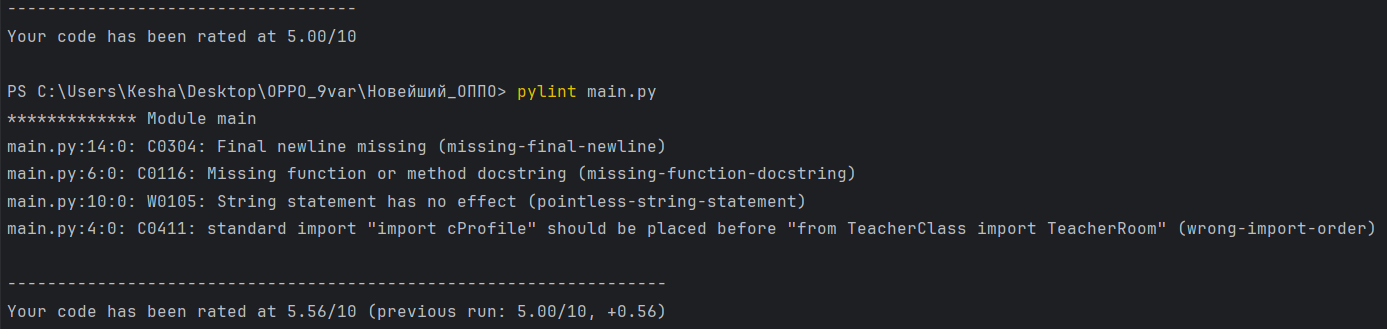


Рисунок 4 - ошибки в файле main.py

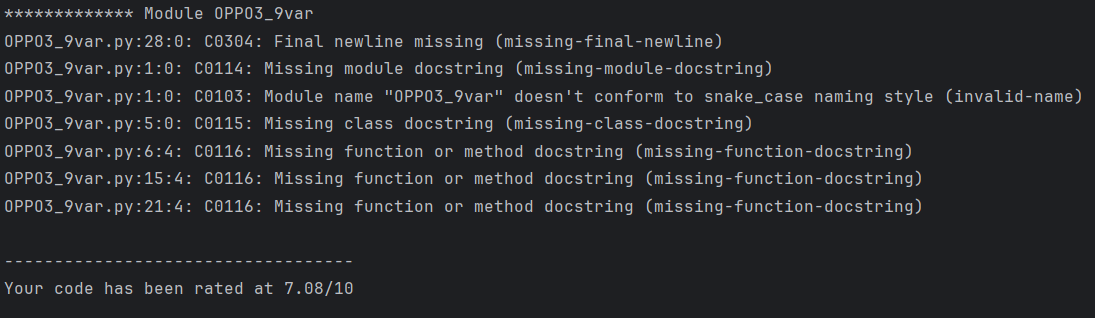


Рисунок 5 - ошибки в файле OPPO3\_9var.py

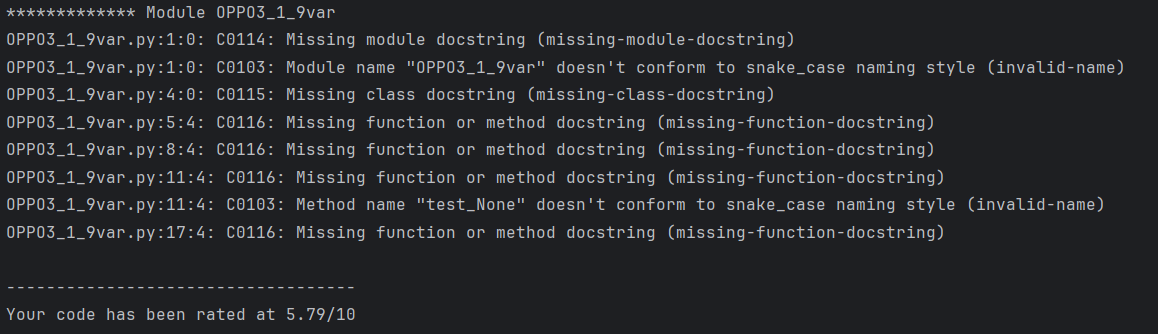


Рисунок 6 - ошибки в файле OPPO3\_1\_9var.py

После всех исправлений pylint выводит:

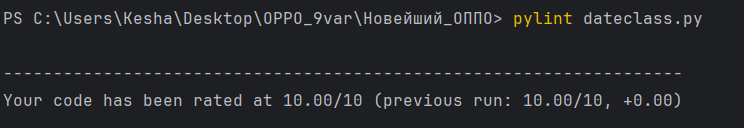


Рисунок 6 - dateclass.py после исправлений

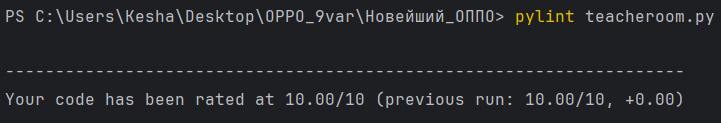


Рисунок 7 - teacherclass.py после исправлений

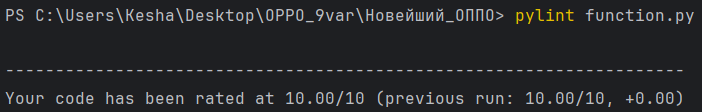


Рисунок 8 - function.py после исправлений

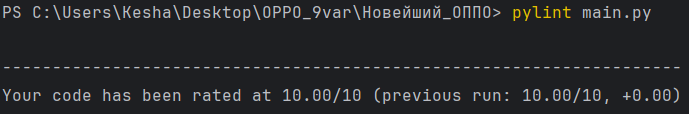


Рисунок 9 - main.py после исправлений

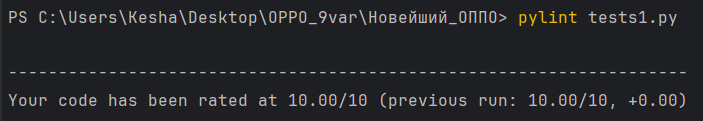


Рисунок 10 – tests1.py после исправлений

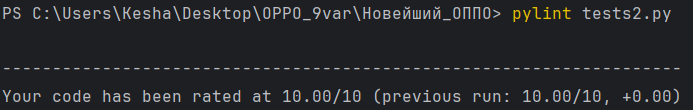


Рисунок 11 – tests2.py после исправлений

# **Профайлинг**

## **Профайлинг времени**

На питоне для этого используем библиотеку cProfile.

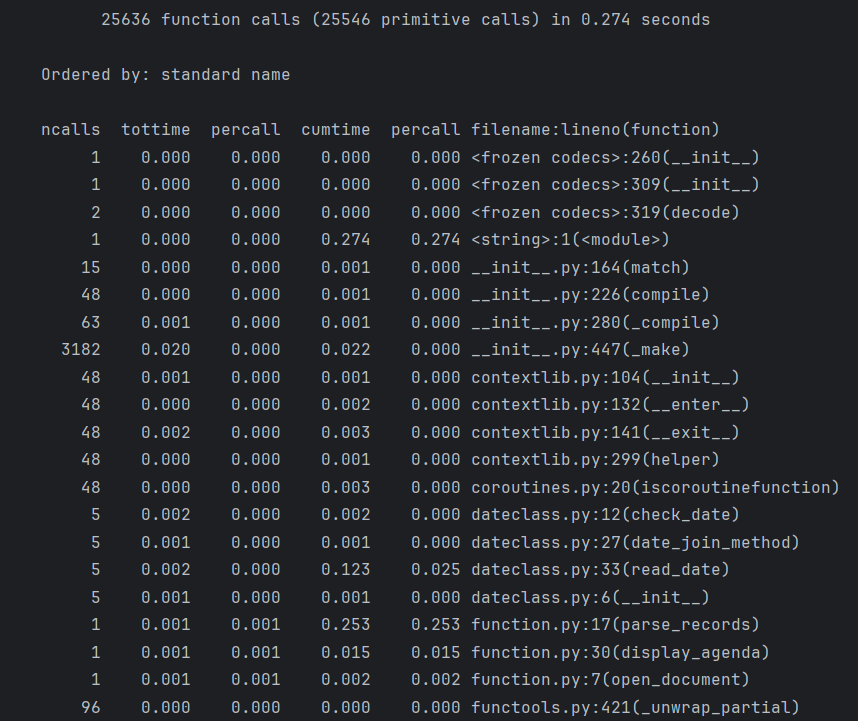


Рисунок 11 - Профайлинг через cProfile

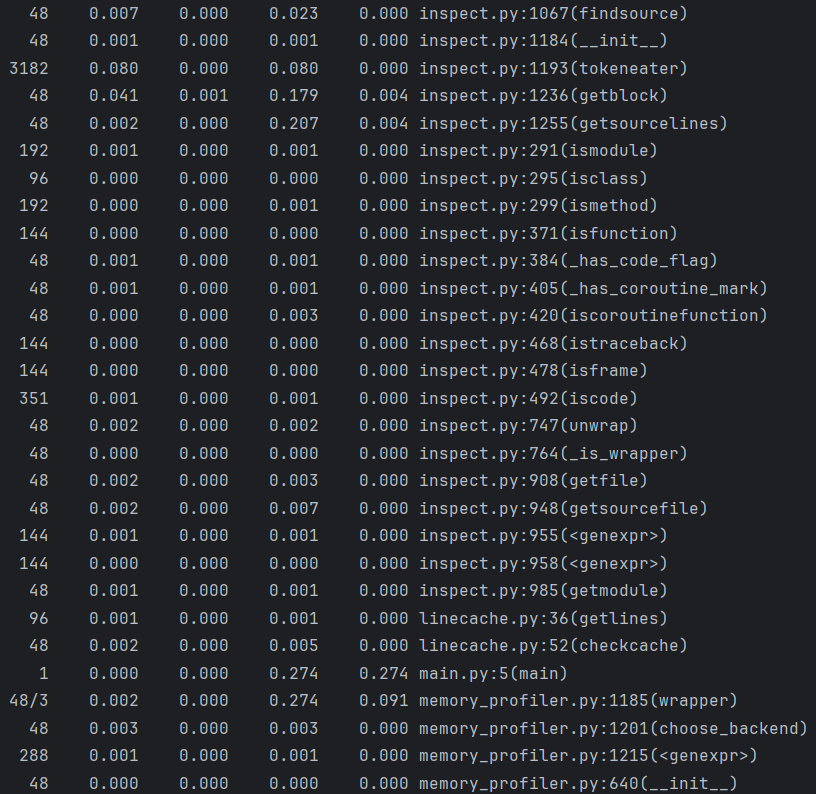


Рисунок 12 - Профайлинг через cProfile

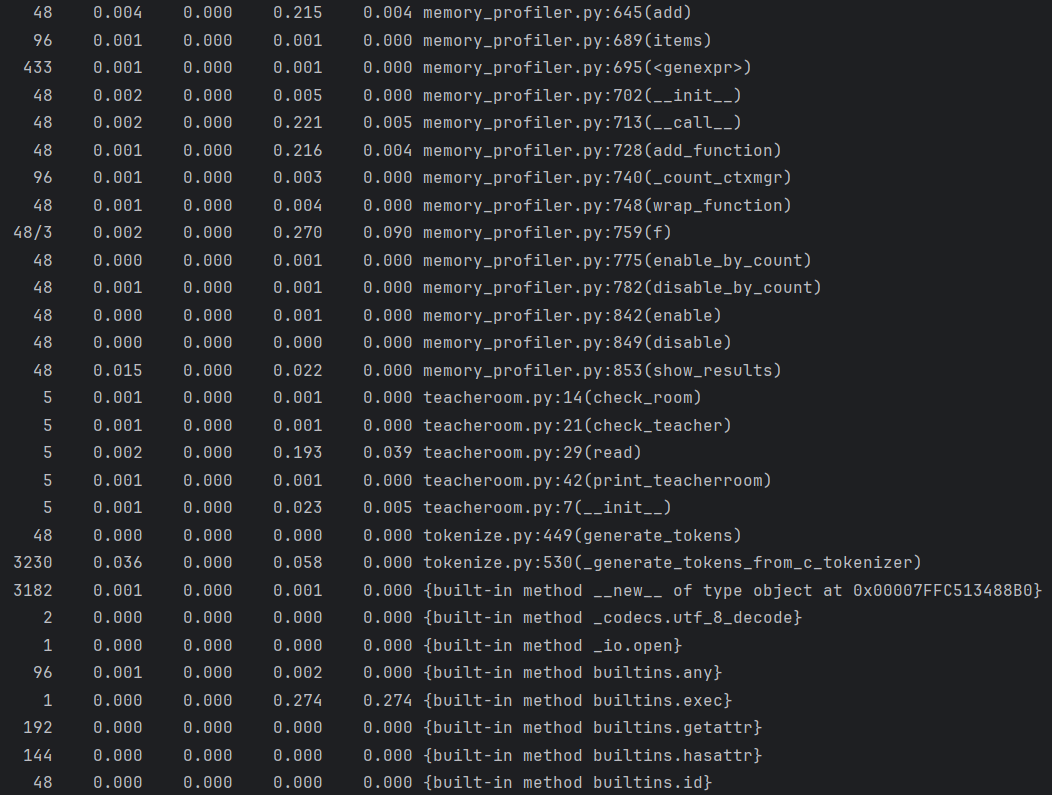


Рисунок 13 - Профайлинг через cProfile

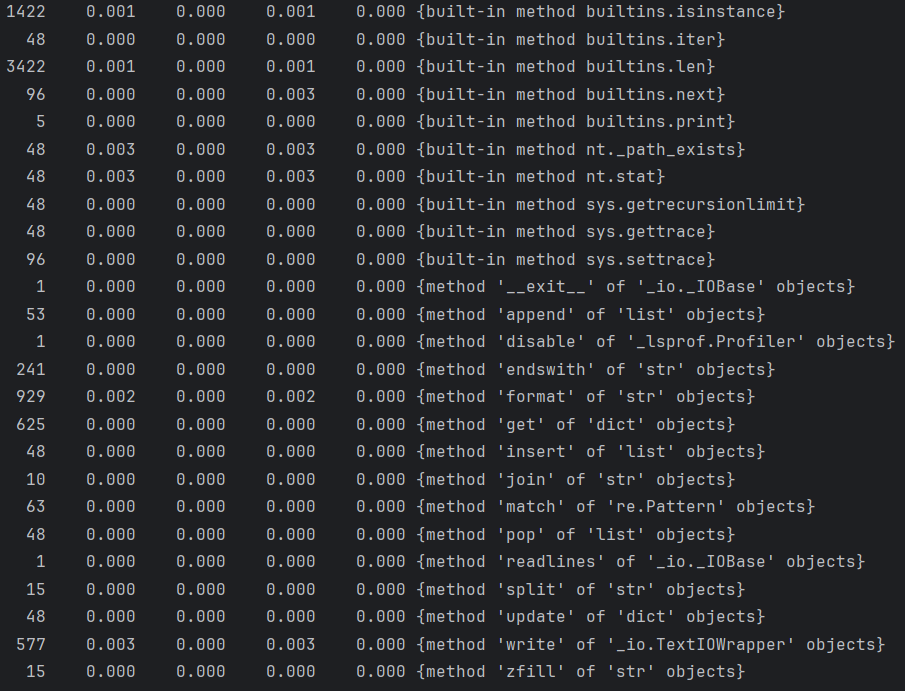


Рисунок 14 - Профайлинг через cProfile

## **Профайлинг памяти**

На питоне для этого используем библиотеку memory\_profile.

Пример профайлинга памяти для одной из функций:

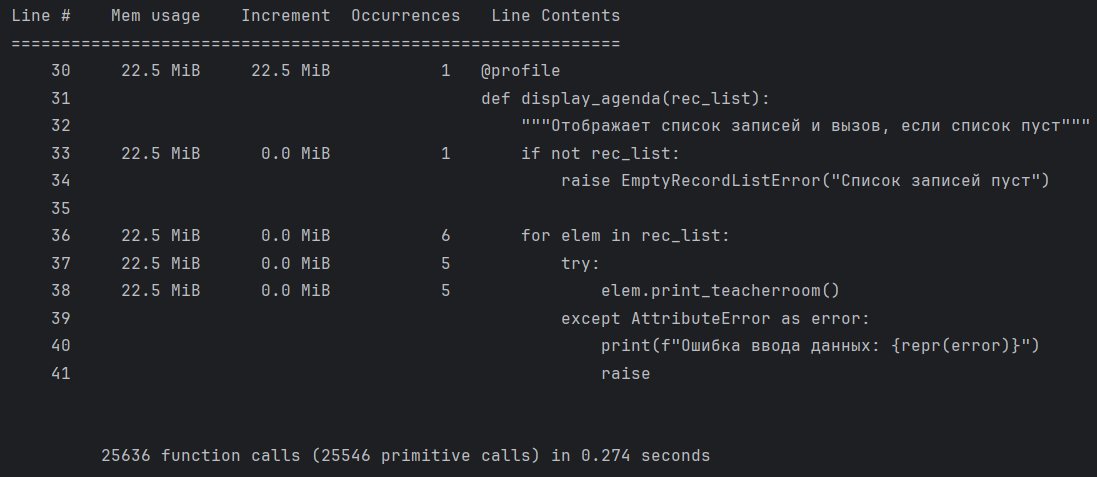


Рисунок 15 - Профайлинг через memory\_profile

# **Заключение**

Программа полностью соответствует всем современным необходимым требованиям по разработке программного обеспечения.