Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра «Информатика и вычислительной техники»

**ОТЧЕТ ПО ПРАКТИКЕ №5**

Иструментальные средства анализа кода программ

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

подпись, датаинициалы, фамилия

Студент КИ22-08Б, 032214719 \_\_\_\_\_\_\_\_\_\_ E.А. Озеров

номер группы, зачетной книжкиподпись, датаинициалы, фамилия

Красноярск 2023

Оглавление

[Статический анализатор кода: 3](#_Toc153618159)

[Профайлинг: 3](#_Toc153618160)

[Профайл памяти: 3](#_Toc153618161)

[Профайл времени: 4](#_Toc153618162)

[Итог: 5](#_Toc153618163)

# Статический анализатор кода:

Для статического анализа я использовал библиотеку на питоне – pylint.

Данный анализатор показал следующие ошибки:

1) дописал docstring к классам и функциям

2) добавил в конец пустую строчку в main.py

3) разделил вывод в консоль на две строки в функции information в main.py, так как предел 79 символов

4) в цикле была переменная i, которая не использовалась в теле цикла, я i заменил на \_i

5) добавил docstring к файлам

6) добавил кодировку, с которой следует читать файл в main.py

7) заменил list() на [] в main.py

8) добавил sys.exit

9) сделал строки кароче в data.py

10) привел LeapYear и FullData к формату snake\_case

11) убрал некоторые if после raise в data.py

12) заменил first\_rate == False на not first\_rate и другие похожу конструкции

13) привел FirstRate и SecondRate к формату snake\_case

14) добавил кодировку, с которой следует читать файл в currency.py

15) редактировал некоторые условие в currency.py

16) сделал более явную ошибку (так говорит pylint) в check\_rate в файле currency.py

17) везде добавил docstring в test\_program.py

18) привел значение к формату snake\_case

После анализа кода я исправил все вышеперечисленные ошибки.

И теперь pulint показывает следующие сообщение.

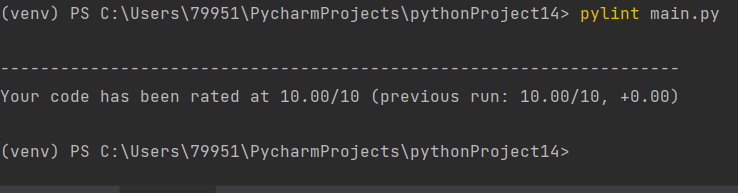


Рисунок 1 - Вывод pylin, после рефакторинга кода

# Профайлинг:

# Профайл памяти:

Для профайлинга памяти на питоне, я использовал библиотеку memory\_profiler.

После запуска кода данный профайл показал следующие:

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 2 - Профайлер памяти

# Профайл времени:

Для профайлинга времени на питоне, я использовал встроенную в питон библиотеку сProfile.

После запуска кода данный профайл показал следующие:

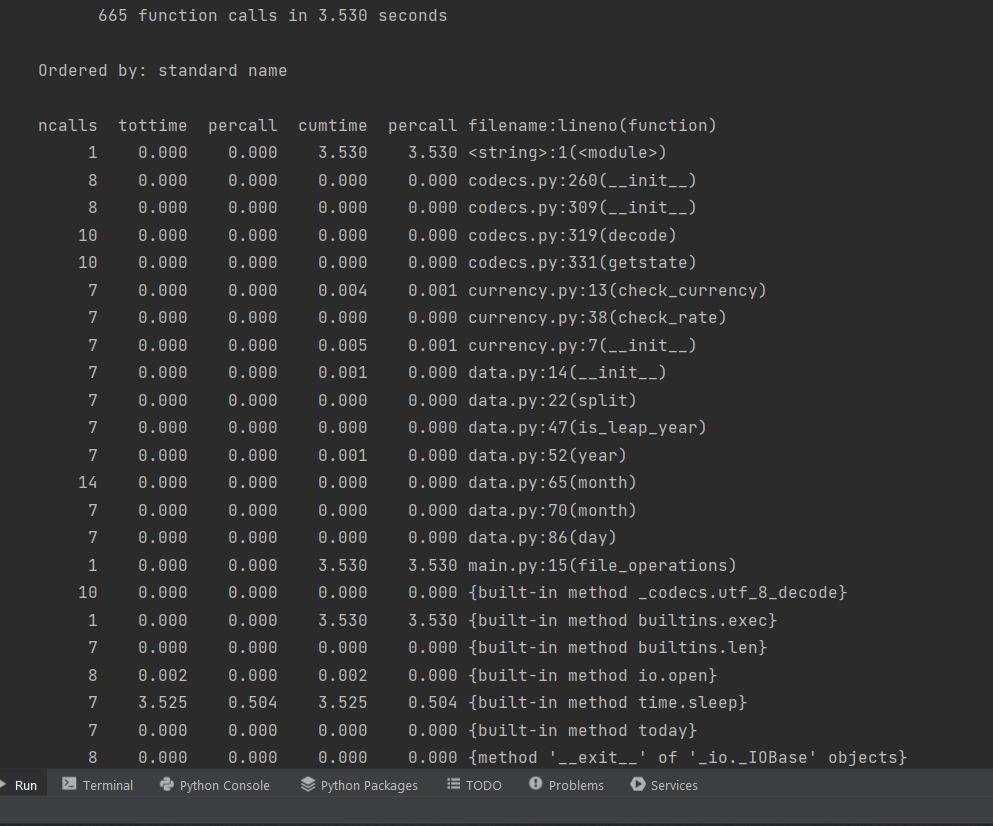


Рисунок 3 - Профайлер времени

# Итог:

Подводя итог, можно отметить, что данная программа покрыта тестами, использует профайлы, проверена статическим анализатором, исходя из этого можно сказать, что данная программа безопасна и легко дополняема.