|  |
| --- |
|  |
| Я МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных технологий (МОСИТ)

|  |
| --- |
| **ПРАКТИЧЕСКАЯ РАБОТА №5** |
| **по дисциплине** |
| **«Тестирование и верификация программного обеспечения»** |

|  |  |
| --- | --- |
| Выполнил студент группы ИКБО-08-21 | Шварев М.К.  Зеневич Д.С. |
| Принял Новичков Д.Е. | \_\_\_\_\_\_\_\_\_\_\_\_\_ |

|  |  |  |
| --- | --- | --- |
| Практическая работа  выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2023

1. **Статические анализаторы**

Pylint — это статический анализатор кода на языке Python, разработанный для помощи в обнаружении и исправлении проблем в коде. Он анализирует исходный код и выводит отчет о потенциальных проблемах, таких как неиспользуемый код, несовместимость типов, слишком много аргументов в функциях и т.д.

Плюсы Pylint:

1. Обнаруживает потенциальные проблемы в коде, такие как неиспользуемые импорты, неиспользуемые переменные и т. д.
2. Может быть интегрирован с различными инструментами разработки, такими как PyCharm, Visual Studio Code и т. д., через плагины.
3. Поддерживает большое количество правил для обнаружения потенциальных проблем.
4. Имеет открытый исходный код и бесплатен для использования.
5. Есть возможность настроить правила под свои нужды.

Минусы Pylint:

1. Может быть сложным в настройке для начинающих пользователей.
2. Некоторые пользователи считают его вывод слишком подробным и отвлекающим от основной работы.
3. Не поддерживает анализ кода на других языках программирования, кроме Python.
4. В некоторых случаях может выдавать ложные срабатывания.

Результат тестирования программы с помощью Pylint представлен на рисунке 1.



Рисунок 1 – Результат тестирования используя Pylint

Pycodestyle — это инструмент для проверки стиля кода на Python. Он проверяет код на соответствие стандарту PEP 8 — общепризнанному соглашению о стиле кода в Python-сообществе. Pycodestyle помогает поддерживать единый стиль кода и находить потенциальные ошибки, связанные с форматированием кода.

Плюсы Pycodestyle:

1. Бесплатная лицензия
2. Открытый исходный код
3. Поддержка большого количества правил форматирования кода
4. Возможность интеграции с другими инструментами, такими как Flake8 или Black

Минусы Pycodestyle:

1. Может быть сложен в использовании для начинающих
2. Не проверяет все возможные ошибки в коде (например, синтаксические ошибки)

Результат тестирования с использованием Pycodestyle представлен на рисунке 2.

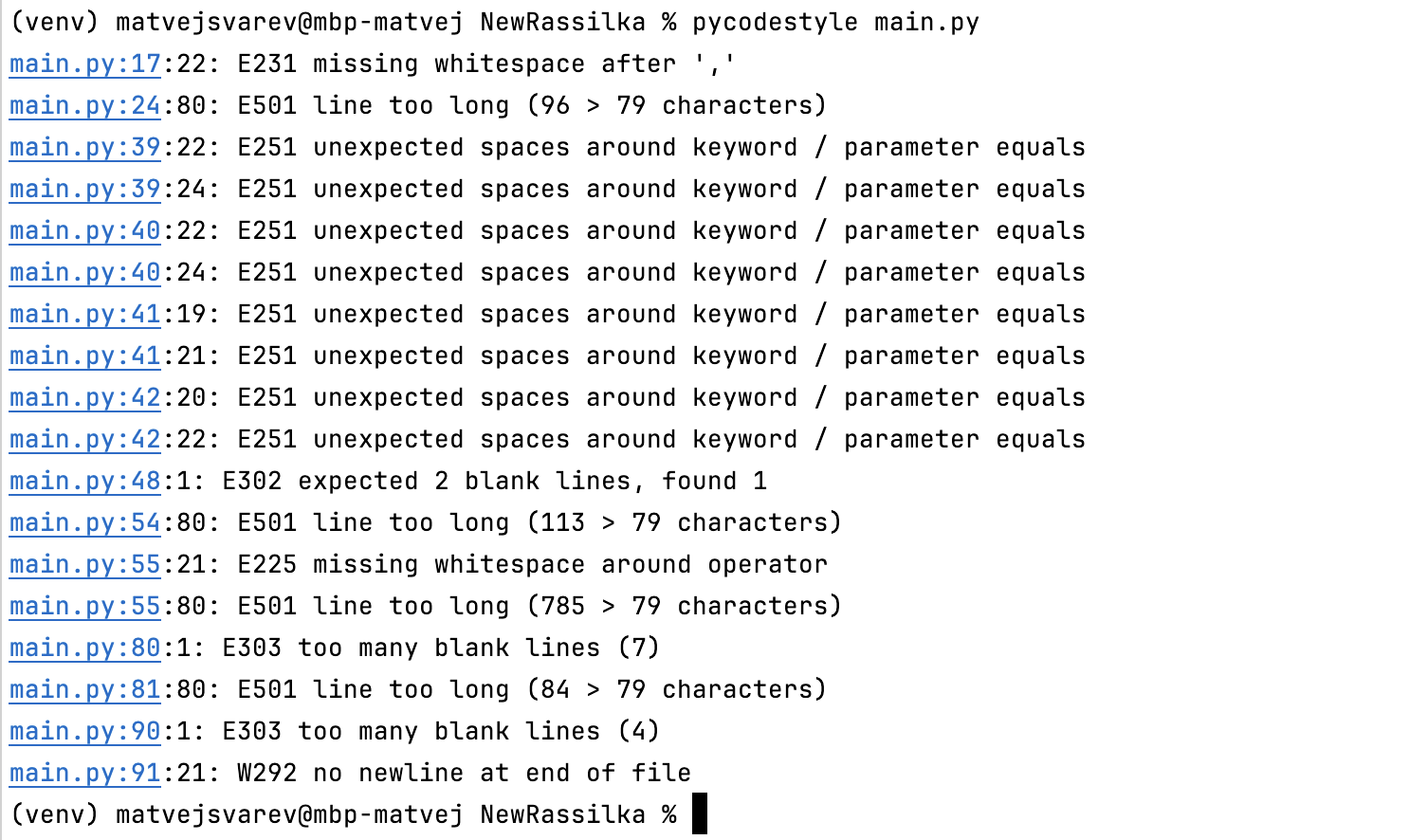


Рисунок 2 – Результат работы Pycodestyle

Mypy — это инструмент статического анализа кода на языке программирования Python. Он использует информацию о типах данных, чтобы проверить, соответствует ли код правилам типизации. mypy может обнаружить многие ошибки ещё до запуска программы, что ускоряет процесс разработки.

Плюсы Mypy:

1. mypy помогает находить ошибки на ранних этапах разработки, что экономит время и усилия
2. mypy предоставляет информацию о типах переменных, что может помочь в понимании кода
3. mipy может использоваться в сочетании с другими инструментами для проверки кода, такими как Pylint

Минусы Mypy:

1. Использование mypy может замедлить процесс разработки, так как требует времени на анализ кода
2. mypy не всегда может обнаружить все ошибки, особенно те, которые связаны с динамической типизацией в Python
3. Для использования mypy необходимо знать типы данных и правила типизации в Python, что может быть сложно для новичков

Результат тестирования с использованием Mypy представлен на рисунке 3.

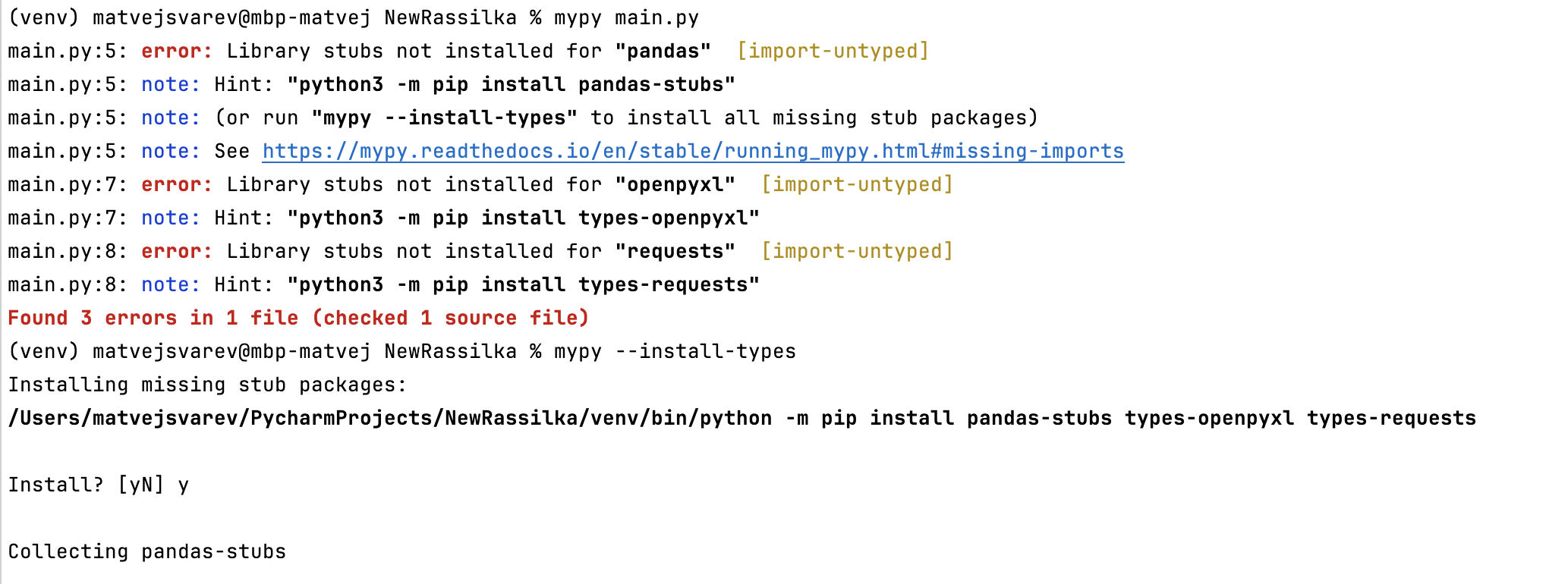


Рисунок 3 – Результат тестирования с использованием Mypy

**Вывод о целесообразности статического анализа**

Пользуясь статическими анализаторами, можно сделать вывод о том, что статический анализ кода является эффективным инструментом для обнаружения ошибок, улучшения качества и читаемости кода, а также для соблюдения стандартов и лучших практик программирования. Он позволяет выявить потенциальные проблемы на ранних стадиях разработки, до компиляции и выполнения программы, что снижает затраты на отладку и упрощает процесс исправления ошибок.

Однако, применение статического анализа требует определенных усилий и затрат на настройку и обучение использованию инструментов анализа, а также на интеграцию его результатов в процесс разработки.

**Предложение по собственному статическому анализатору**

Из того, что можно было бы добавить к уже существующему анализатору, было бы полезным внедрить:

- анализ безопасности кода на основе машинного обучения: использование алгоритмов машинного обучения для выявления потенциальных уязвимостей и угроз безопасности в коде;

- автоматическое исправление ошибок: возможность автоматического исправления обнаруженных проблем в коде с минимальным вмешательством разработчика.

- интеграция с облачными платформами: возможность проведения анализа кода непосредственно в облачной среде разработки, что позволит упростить процесс разработки и улучшить производительность.

**2. Динамические анализаторы**

cProfile — это модуль Python, предназначенный для профилирования кода. Он предоставляет информацию о том, как долго выполняется каждый участок кода, а также о том, сколько раз каждый участок кода был выполнен. Это позволяет разработчикам понять, где в коде есть проблемы с производительностью, и оптимизировать эти участки.

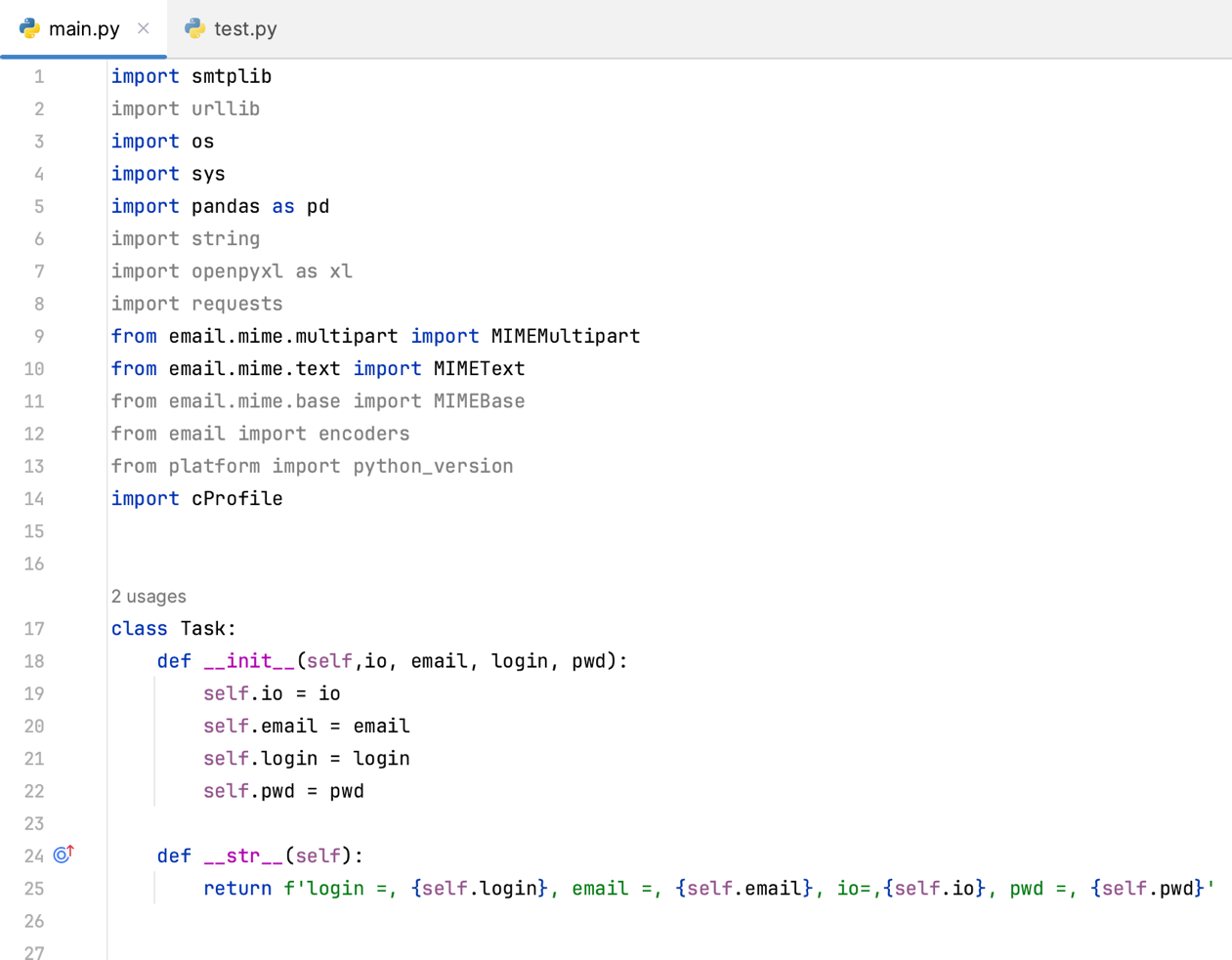
Плюсы использования cProfile

1. Простота использования: cProfile легко интегрируется в код и может быть использован для профилирования как отдельных функций, так и целых скриптов
2. Широкий спектр информации: cProfile предоставляет информацию не только о времени выполнения каждого участка кода, но и о том, сколько раз каждый участок был выполнен, что может быть полезно для понимания того, какие участки кода наиболее ресурсоемкие
3. Интеграция с другими инструментами: cProfile может быть интегрирован с другими инструментами для анализа производительности, такими как PyCallGraph

Минусы использования cProfile

1. Ограниченная поддержка Python 3: cProfile не полностью поддерживает все возможности Python 3, что может вызвать проблемы при использовании его в проектах, использующих последние версии Python
2. Не предоставляет информацию об утечках памяти: cProfile не предоставляет информацию об утечках памяти, что может затруднить обнаружение проблем с памятью в коде

Результат тестирования с помощью cProfile и код представлены на рисунках 4 и 5.



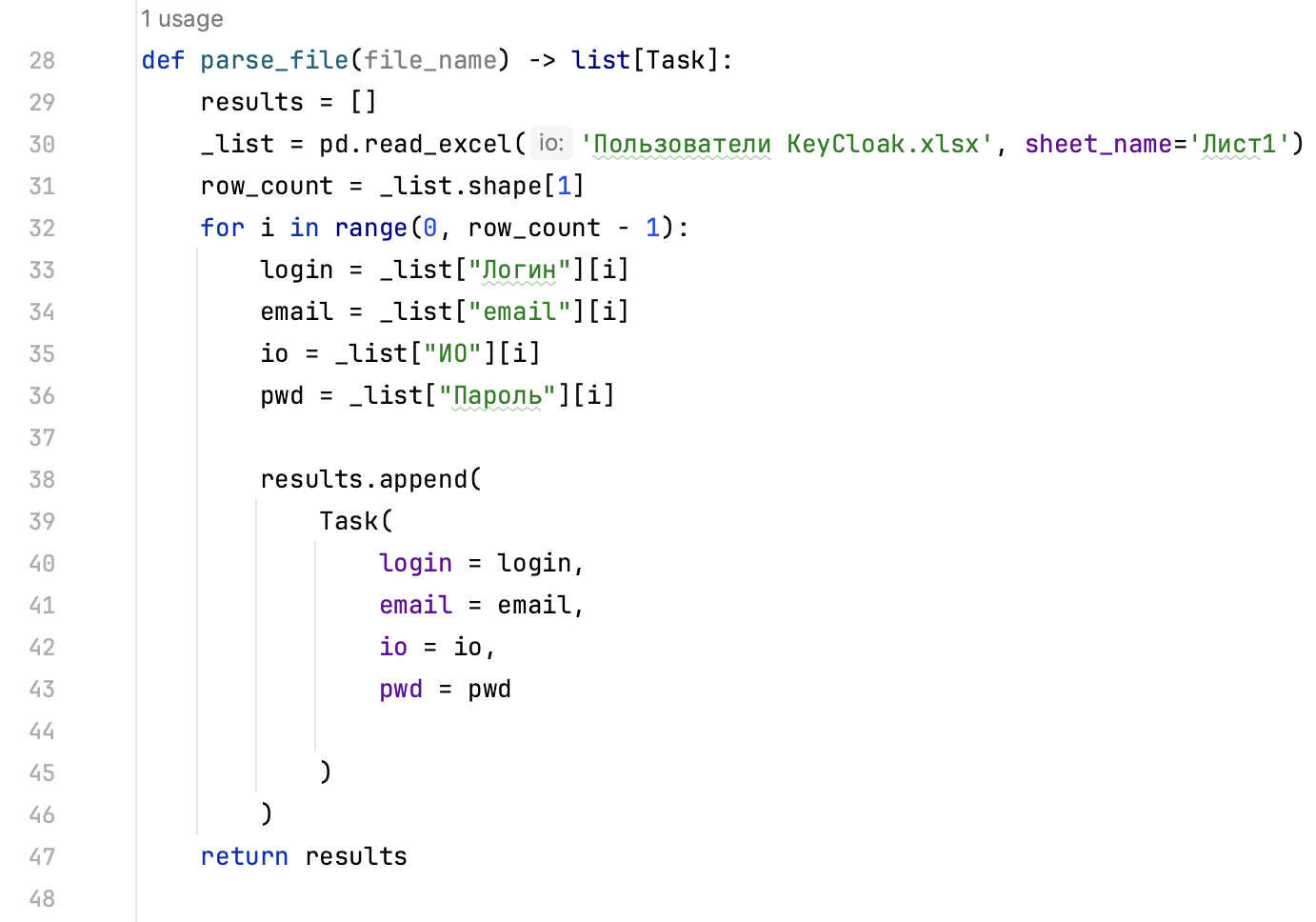






Рисунок 4 – Код

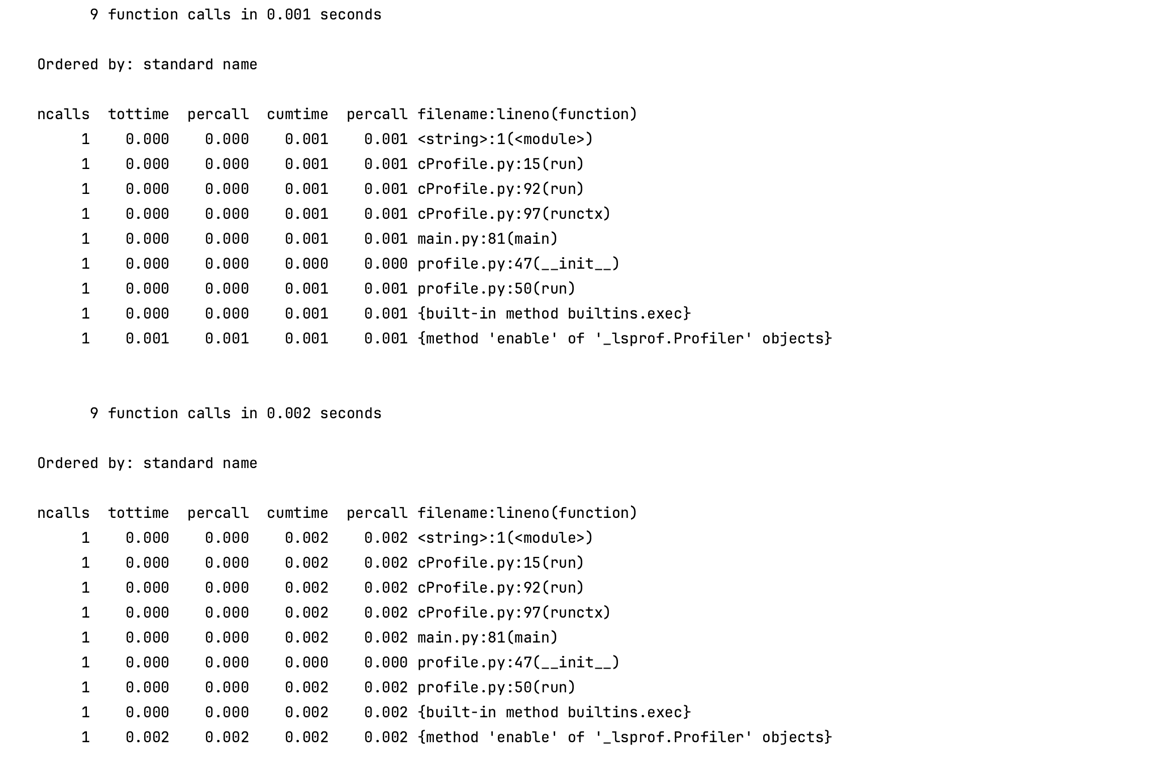


Рисунок 5 – Результат тестирования cProfile

**Вывод**

В ходе работы были рассмотрены основные принципы статического и динамического анализа исходного кода. Были описаны статические и динамические анализаторы, их преимущества и недостатки. В целом, работа позволяет получить представление о том, каким образом можно повысить качество программного обеспечения и уменьшить количество ошибок в коде.