Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

(МТУСИ)

Кафедра «Сетевых Информационных Технологий и Сервисов»

дисциплина «Рефакторинг баз данных и приложений»

Лабораторная работа № 2

**«Рефакторинг приложений с изучением модульного**

**тестирования»**

Выполнил: студент группы БПИ2303

Берездовец Илья Сергеевич

**Ссылка на GitHub: https://github.com/iberezaa**

**Цели лабораторной работы:**

1. Изучить основные принципы и задачи модульного тестирования.

2. Научиться разрабатывать модульные тесты для проверки функциональности отдельных

частей приложения.

3. Применить модульное тестирование для контроля качества кода в процессе рефакторинга.

4. Освоить использование инструментов для написания и запуска модульных тестов.

Ход работы:

Написал плохой код для реализации калькулятора.

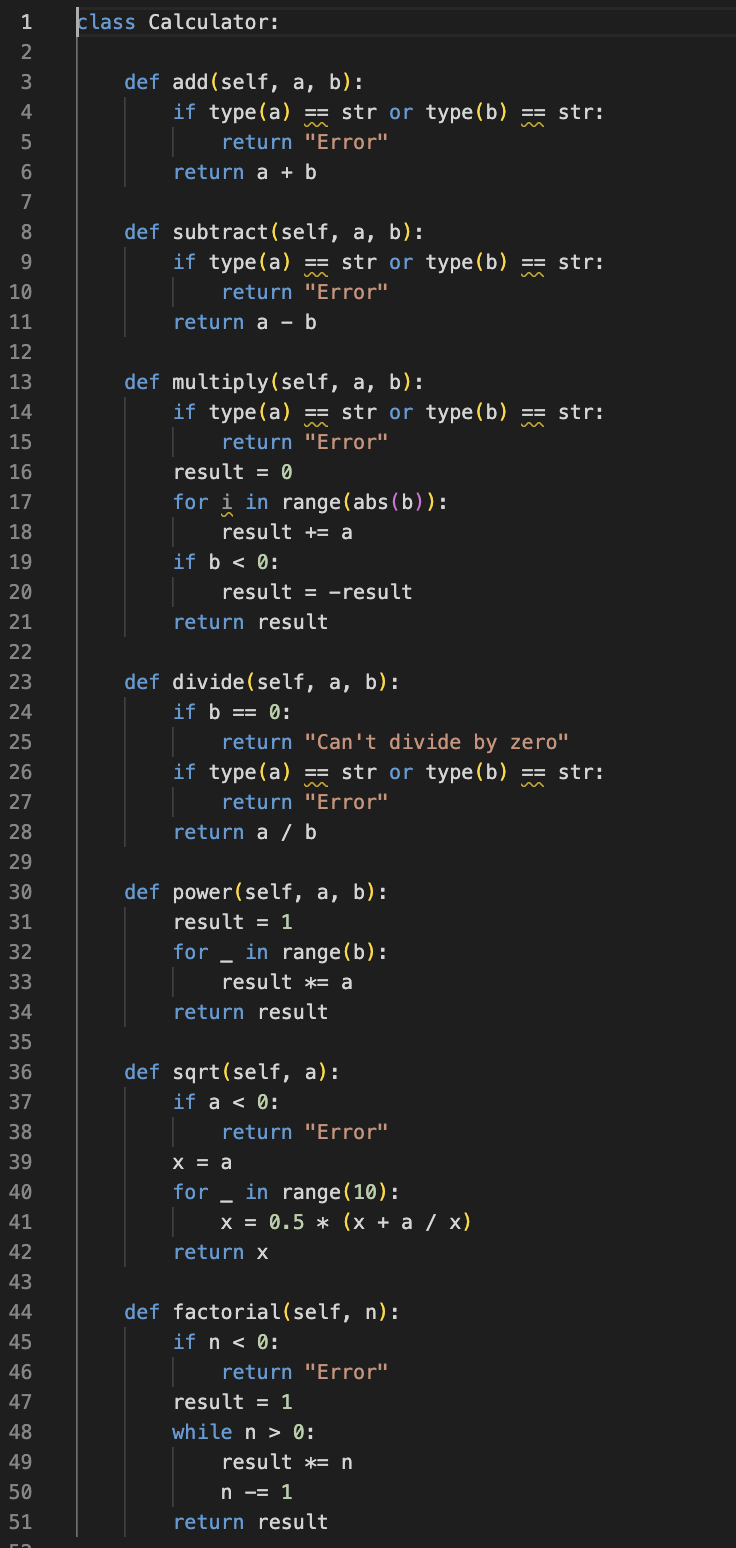


Рис.1 (Код до рефакторинга)

2) Провел рефакторинг кода:

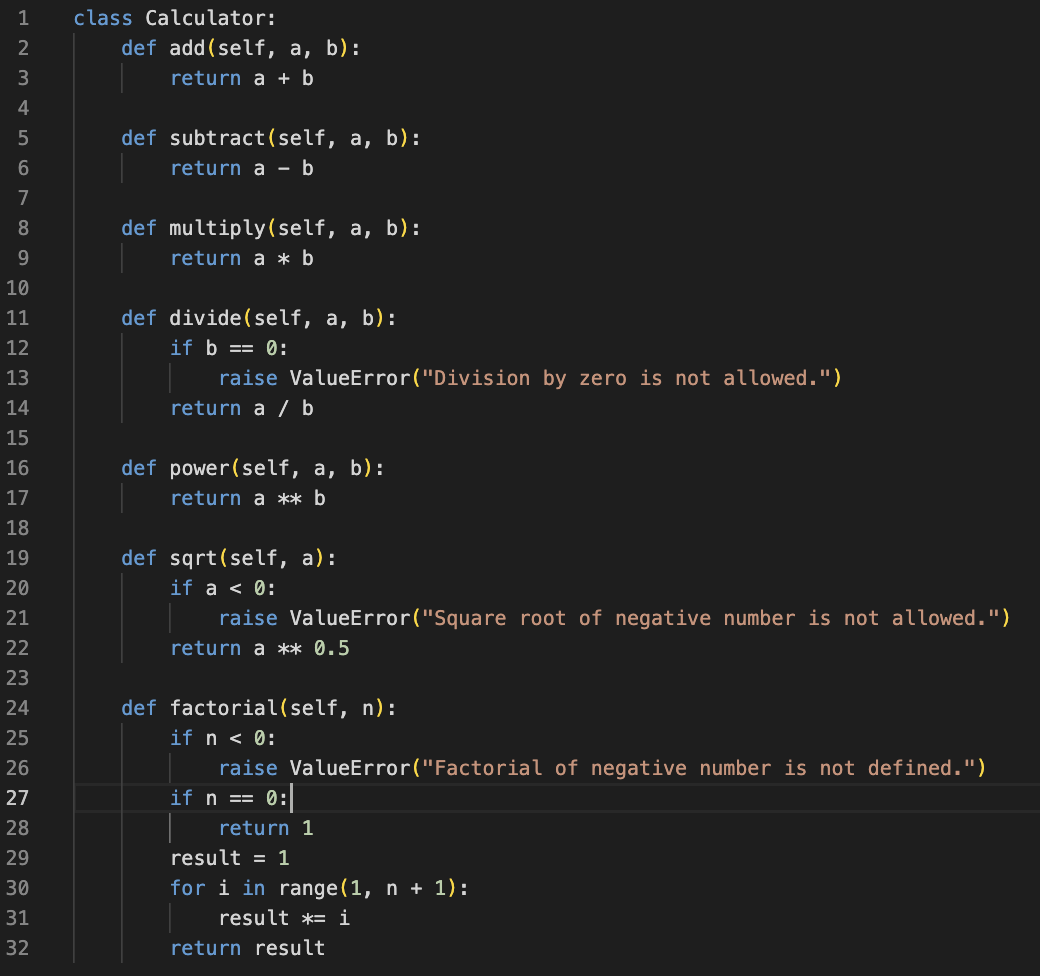


Рис. 2 (Код после рефакторинга)

Преимущества кода после рефакторинга:

1. Использует корректные исключения (raise ValueError) для критических ошибок (например, деление на ноль, вычисление корня из отрицательного числа)
2. Все функции реализованы максимально лаконично и читаемо
3. Использует встроенные операторы (\*, /, \*\*) для стандартных математических операций, что делает код быстрее и проще.
4. Прямые математические операции работают быстрее, поскольку используют оптимизированные механизмы Python.
5. Реализует факториал корректно с явным учетом n == 0

Вывод: Код после рефакторинга стал более чище, быстрее и проще для чтения.

3) Написал модульные тесты:



Рис. 3 (Модульные тесты)

4) Тесты прошли успешно.

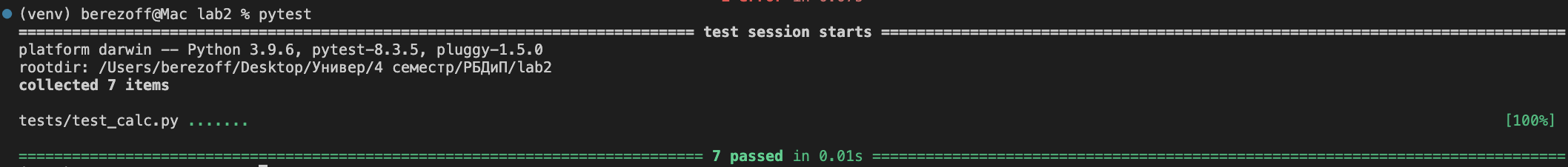


Рис. 4 (Успешно пройденные тесты)

Описание тестов:

1. Все основные методы класса Calculator проверяются: add(), subtract(), multiply(), divide(), power(), sqrt(), factorial().
2. Проверка на различные сценарии поведения программы: Проверяются стандартные случаи с корректными данными.
3. Каждый тест проверяет только одну конкретную функцию класса Calculator.
4. Код полностью совместим с pytest, что позволяет быстро и удобно запускать тесты.

Вывод:

Написан проект калькулятора. Произведен рефакторинг кода. Написаны модульный тесты. Тесты прошли успешно.