Лабораторная работа №7:

Цель работы

1. Проверка с помощью создания контекста функций по считыванию настроек калькулятора и запись в файл результатов выполнения операций
2. Используя параметризация тестов написать тесты функции two_sum.
3. Используя гипотезы с помощью hypothesis протестировать вычисление факториала.
   Используя параметрические тесты, проверить пограничные случаи, обработку некорректных данных при вычислении факториала.

Комментарии по выполнению

Задача 1

Тестирование должно учитывать **развертывание и свертывание контекста**: подготовка временных файлов перед тестами и их корректное удаление после завершения. Тестирование можно проводить или с помощью unittest или с помощью pytest. Использовать tempfile (или другую для unittest) или tmpdir (для pytest) в тестах.

Проверьте (как минимум):

- Корректное считывание настроек из файла.
- Обработку ошибки FileNotFoundError.
- Обработку ошибки ValueError при некорректном JSON.

Для функции записи в файл проверьте:

- Корректную запись строки в файл.
- Создание файла, если он отсутствует.
- Добавление строк в существующий файл.

Модульные тесты должны быть вынесены в отдельный файл (например, test_calculator_io.py).

Код должен быть оформлен в соответствии со стандартом **PEP8**. Используйте комментарии и docstrings для описания функций.

Задача 2

Используя параметризацию тестов с использованием библиотеки pytest, написать тесты функции two_sum, которую вы реализовали ранее.

Требования

1. **Покрытие тестами:**
    - Тесты должны быть реализованы с использованием **pytest**.
    - Использовать параметризацию (@pytest.mark.parametrize) для проверки функции на разных входных данных.
    - Обеспечить покрытие тестами следующих случаев:
        - **Базовый случай**: массив из нескольких элементов, включая положительные и отрицательные числа.
        - **Пограничные случаи**:
            - Минимальный размер массива (len(nums) == 2).
            - Числа в массиве с минимальными и максимальными значениями в пределах [-10^9, 10^9].
        - **Особые случаи**:
            - Числа в массиве повторяются.
            - Все числа в массиве одинаковые.
    - Тесты должны проверять как входные данные, так и корректность индексов (порядок не важен).
2. **Тестируемая функция:**
    - Функция должна находиться в отдельном модуле (например, solution.py).
    - Тесты должны быть реализованы в отдельном файле (например, test_solution.py).

Задача 3

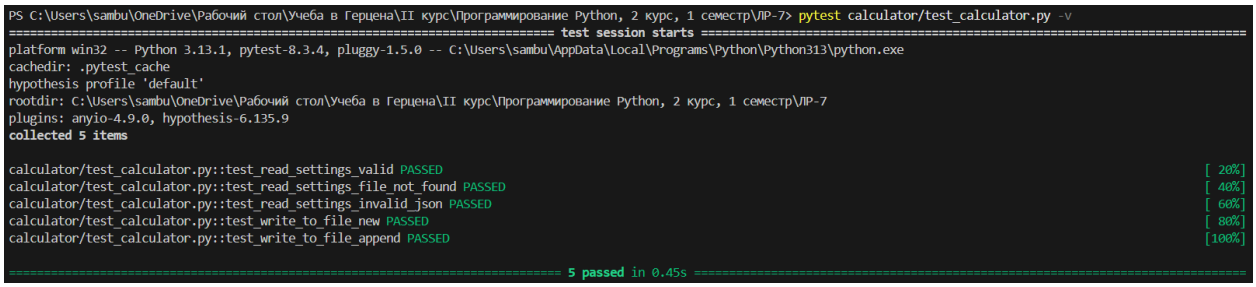
Автоматическая генерация входных данных с Hypothesis:

- Использовать стратегию @given для генерации входных данных n в диапазоне $0 \leqslant n \leqslant 100$.
- Проверить, что функция корректно обрабатывает все допустимые входные значения.
- Проверить свойства факториала:
    - $(n+1)! = (n+1) \times n!$
    - Функция не возвращает отрицательных значений.

- Обработка некорректных данных: функция должна выбрасывать исключение ValueError, если n не является натуральным числом (например, отрицательное число или дробь).

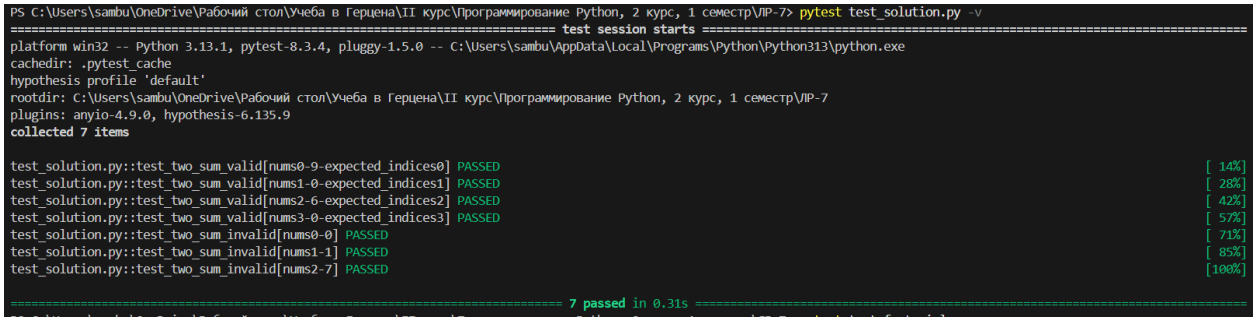
Задача 1:

Тесты для работы с файлами



Задача 2:

Тесты для функции two_sum



Задача 3:

Тесты для факториала