МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра ИС

ОТЧЕТ

по практической работе № 5

по дисциплине «Автоматизация тестирования»

Тема: «Тестирование на основе Pytest»

Студент гр. 2372 _____ Соколовский В.Д.

Преподаватель _____ Турнецкая Е.Л.

Санкт-Петербург

2024

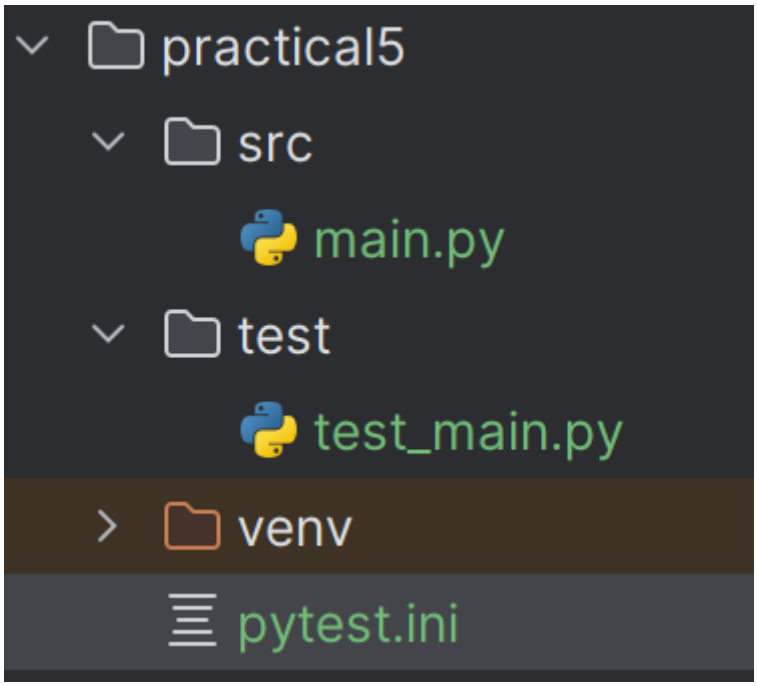
Цель работы: Получение практических навыков по тестированию с помощью Pytest.

Поставленные задачи:

1. Установить программное окружение проекта по автоматизированному тестированию с фреймворком Pytest.
2. Реализовать тестовые функции для проведения модульного тестирования на основе программных инструментов Pytest.
3. Зафиксировать результаты тестирования в отчете.

Выполнение работы:

Структура проекта согласно рекомендациям разработчиками Pytest:



Установленные компоненты программного окружения:

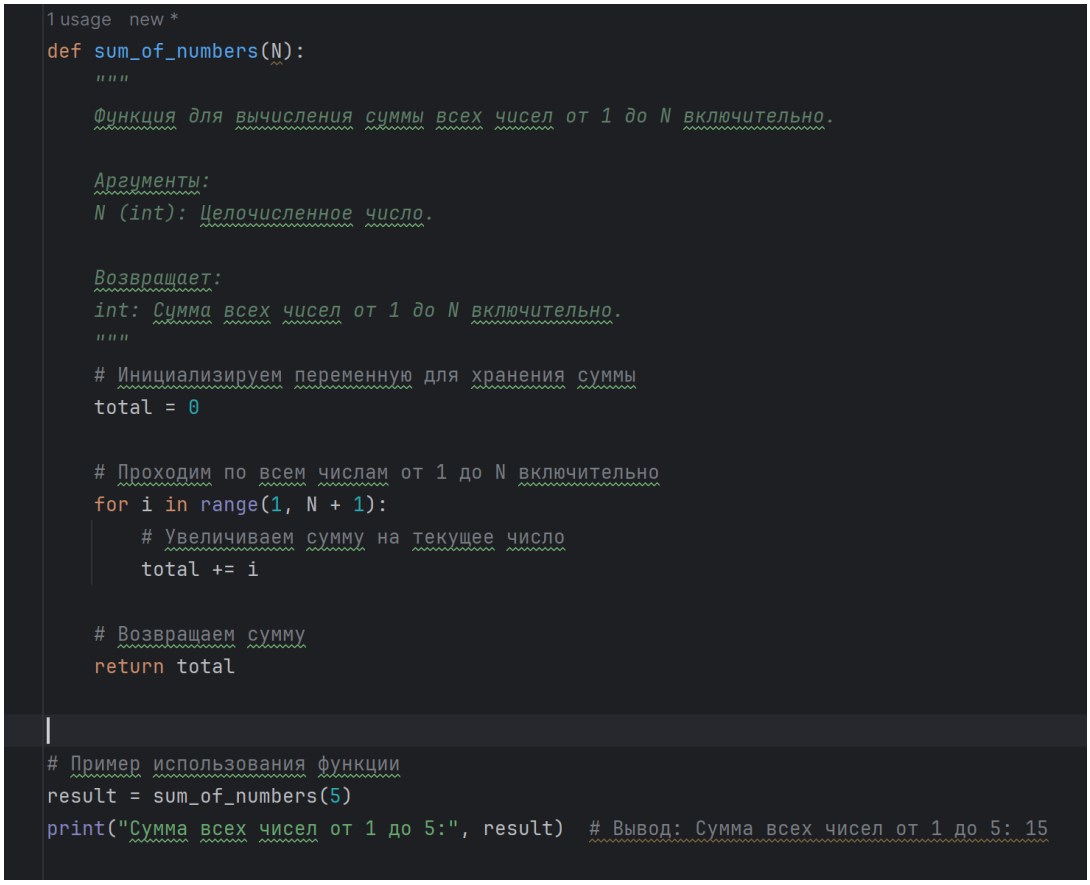
```
pytest                    8.1.1
pytest-html               4.1.1
```

Вариант 7

Напишите функцию, которая принимает на вход целочисленное число N и возвращает сумму всех чисел от 1 до N включительно.

Листинг 1 – Программный код в файле main.py

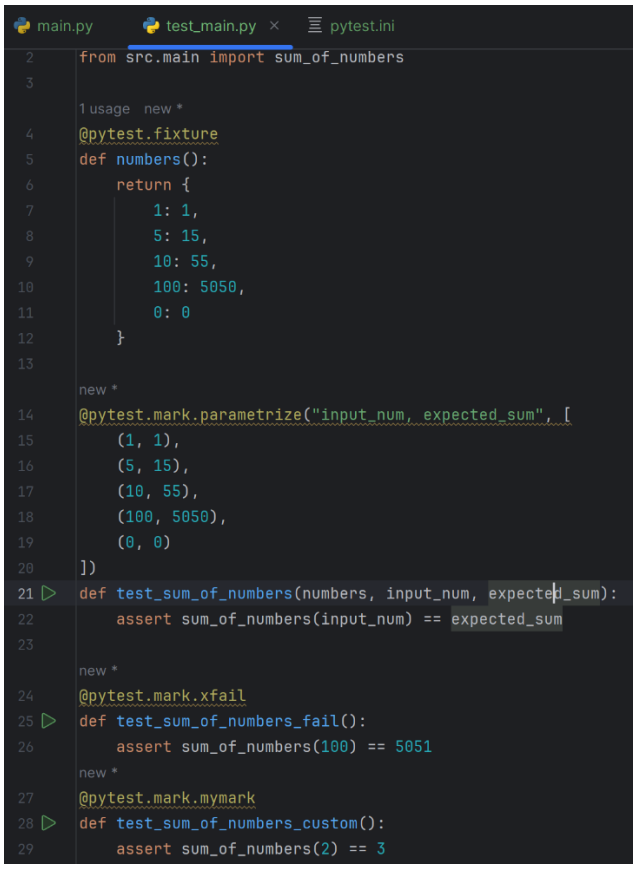
```
def sum_of_numbers(N):
    """
    Функция для вычисления суммы всех чисел от 1 до N включительно.

    Аргументы:
    N (int): Целочисленное число.

    Возвращает:
    int: Сумма всех чисел от 1 до N включительно.
    """
    # Инициализируем переменную для хранения суммы
    total = 0

    # Проходим по всем числам от 1 до N включительно
    for i in range(1, N + 1):
        # Увеличиваем сумму на текущее число
        total += i

    # Возвращаем сумму
    return total

# Пример использования функции
result = sum_of_numbers(5)
print("Сумма всех чисел от 1 до 5:", result)  # Вывод: Сумма всех чисел от 1 до 5: 15
```



Скриншот 1 - Программный код файла main.py

Листинг 3 – Программный код с тестами в файле test_main.py

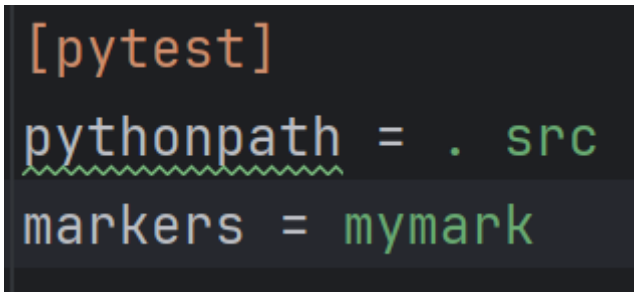
```
import pytest
from src.main import sum_of_numbers

@pytest.fixture
def numbers():
    return {
        1: 1,
        5: 15,
        10: 55,
        100: 5050,
        0: 0
    }

@pytest.mark.parametrize("input_num, expected_sum", [
    (1, 1),
    (5, 15),
    (10, 55),
    (100, 5050),
    (0, 0)
])
def test_sum_of_numbers(numbers, input_num, expected_sum):
    assert sum_of_numbers(input_num) == expected_sum

@pytest.mark.xfail
def test_sum_of_numbers_fail():
    assert sum_of_numbers(100) == 5051
@pytest.mark.mymark
def test_sum_of_numbers_custom():
    assert sum_of_numbers(2) == 3
```



Скриншот 2 – Программный код с тестами в файле test_main.py

Листинг 3 – Программный код в конфигурационном файле pytest.ini

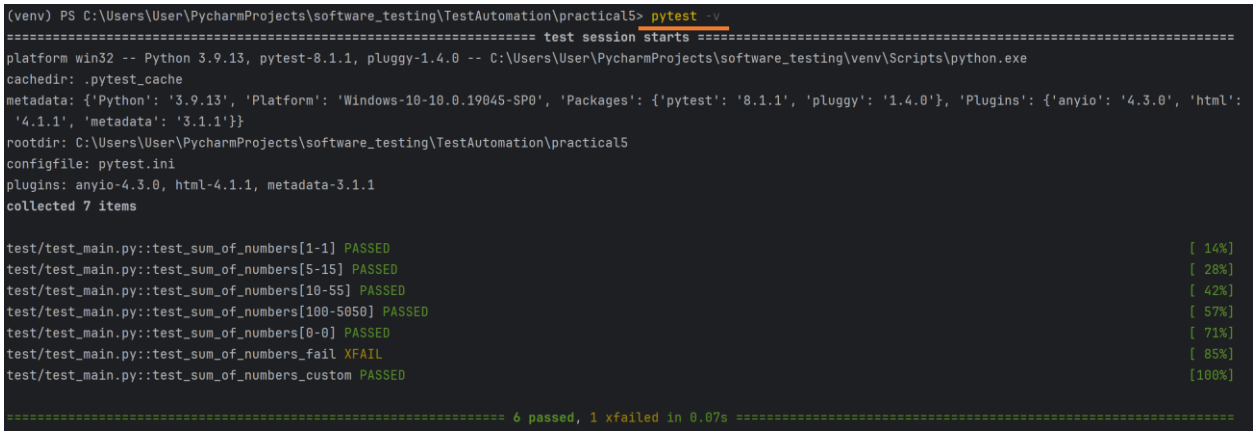
```
[pytest]
pythonpath = . src
markers = mymark
```



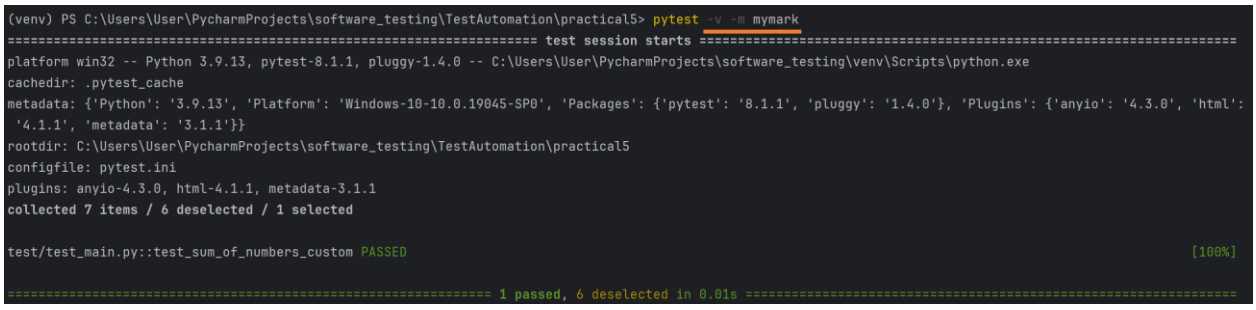
Скриншот 3 – Программный код в конфигурационном файле pytest.ini

Тестирование:

Позитивное тестирование:



Скриншот 4 - Результат запуска тестов с маркерами и результат запуска тестов в терминале после создания фикстуры



Скриншот 5 - Запуск тестов с пользовательским маркером mymark после изменения конфигурационного файла pytest.ini

Негативное тестирование:

```python
@pytest.mark.parametrize("input_num, expected_sum", [
    (1, 1),
    (5, 15),
    (10, 55),
    (100, 5050),
    (0, 9)
])
def test_sum_of_numbers(numbers, input_num, expected_sum):
    assert sum_of_numbers(input_num) == expected_sum
```

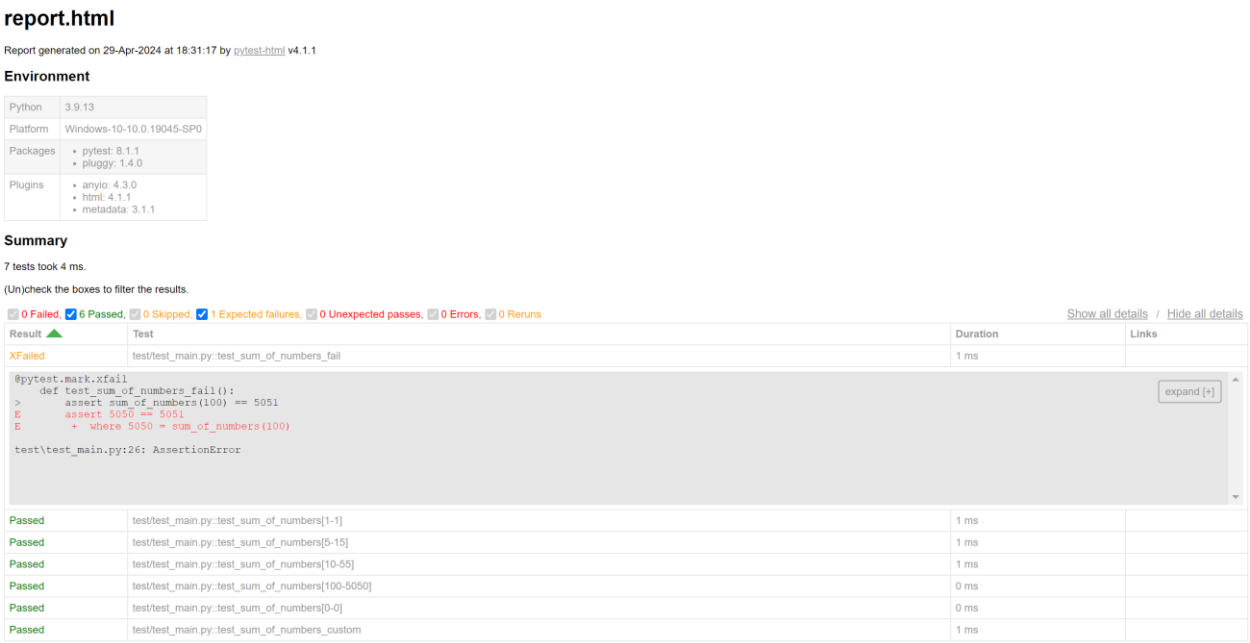
Скриншот 6 – Фикстура с измененными данными

```
(venv) PS C:\Users\User\PycharmProjects\software_testing\TestAutomation\practical5> pytest -v
============================= test session starts =============================
platform win32 -- Python 3.9.13, pytest-8.1.1, pluggy-1.4.0 -- C:\Users\User\PycharmProjects\software_testing\venv\Scripts\python.exe
cachedir: .pytest_cache
metadata: {'Python': '3.9.13', 'Platform': 'Windows-10-10.0.19045-SP0', 'Packages': {'pytest': '8.1.1', 'pluggy': '1.4.0'}, 'Plugins': {'anyio': '4.3.0', 'html':
 '4.1.1', 'metadata': '3.1.1'}}
rootdir: C:\Users\User\PycharmProjects\software_testing\TestAutomation\practical5
configfile: pytest.ini
plugins: anyio-4.3.0, html-4.1.1, metadata-3.1.1
collected 7 items

test/test_main.py::test_sum_of_numbers[1-1] PASSED                       [ 14%]
test/test_main.py::test_sum_of_numbers[5-15] PASSED                      [ 28%]
test/test_main.py::test_sum_of_numbers[10-55] PASSED                     [ 42%]
test/test_main.py::test_sum_of_numbers[100-5050] PASSED                  [ 57%]
test/test_main.py::test_sum_of_numbers[0-9] FAILED                       [ 71%]
test/test_main.py::test_sum_of_numbers_fail XFAIL                        [ 85%]
test/test_main.py::test_sum_of_numbers_custom PASSED                     [100%]

================================== FAILURES ===================================
```

```
_____________________ test_sum_of_numbers[0-9] _____________________

numbers = {0: 0, 1: 1, 5: 15, 10: 55, ...}, input_num = 0, expected_sum = 9

    @pytest.mark.parametrize("input_num, expected_sum", [
        (1, 1),
        (5, 15),
        (10, 55),
        (100, 5050),
        (0, 9)
    ])
    def test_sum_of_numbers(numbers, input_num, expected_sum):
>       assert sum_of_numbers(input_num) == expected_sum
E       assert 0 == 9
E        +  where 0 = sum_of_numbers(0)

test\test_main.py:22: AssertionError
=========================== short test summary info ===========================
FAILED test/test_main.py::test_sum_of_numbers[0-9] - assert 0 == 9
========== 1 failed, 5 passed, 1 xfailed in 0.12s ==========
```

Скриншот 7, 8 - Результат запуска тестов с измененными данными результат запуска тестов в терминале после создания фикстуры

Отчет о тестировании:



report.html

Report generated on 29-Apr-2024 at 18:31:17 by pytest-html v4.1.1

Environment

Python	3.9.13
Platform	Windows-10-10.0.19045-SP0
Packages	• pytest: 8.1.1 • pluggy: 1.4.0
Plugins	• anyio: 4.3.0 • html: 4.1.1 • metadata: 3.1.1

Summary

7 tests took 4 ms.

(Un)check the boxes to filter the results.

0 Failed, 6 Passed, 0 Skipped, 1 Expected failures, 0 Unexpected passes, 0 Errors, 0 Reruns — Show all details / Hide all details

Result	Test	Duration	Links
XFailed	test/test_main.py::test_sum_of_numbers_fail	1 ms	

```
@pytest.mark.xfail
    def test_sum_of_numbers_fail():
>       assert sum_of_numbers(100) == 5051
E       assert 5050 == 5051
E        +  where 5050 = sum_of_numbers(100)

test\test_main.py:26: AssertionError
```

expand [+]

Passed	test/test_main.py::test_sum_of_numbers[1-1]	1 ms	
Passed	test/test_main.py::test_sum_of_numbers[5-15]	1 ms	
Passed	test/test_main.py::test_sum_of_numbers[10-55]	1 ms	
Passed	test/test_main.py::test_sum_of_numbers[100-5050]	0 ms	
Passed	test/test_main.py::test_sum_of_numbers[0-0]	0 ms	
Passed	test/test_main.py::test_sum_of_numbers_custom	1 ms	

Скриншот 9 - Отображение HTML-отчета в браузере

Выводы:

В процессе выполнения работы были получены практические навыки по написанию и запуску тестов с использованием PyTest. Было изучено использование фикстур, параметризации тестов, а также маркировок для организации и запуска тестов с различными характеристиками.

Возникшие проблемы, такие как неожиданные ошибки в тестах или некорректное поведение приложения, были решены путем анализа кода и модификации тестов для правильного воспроизведения и проверки функционала.

В результате работы были успешно пройдены все поставленные задачи, что позволило получить практические навыки по тестированию с помощью PyTest и увереннее использовать данный инструмент для тестирования программного обеспечения.

Список использованных источников:

1. Фреймворки для тестирования. Образовательный блог компании Яндекс. URL: https://practicum.yandex.ru/blog/fraymvorki-dlya-testirovaniya-na-python/
2. Документация фреймворка Pytest URL: https://docs.pytest.org/en/8.0.x/