



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

Отчет по практической работе

по дисциплине «Тестирование и верификация ПО»

Выполнили:

Студенты группы ИКБО-15-22

Ератин Н.В.

Проверил:

Доцент

Чернов Е.А.

2024 г.

СОЗДАНИЕ ПРОСТОГО МОДУЛЯ ПРОГРАММЫ

Была создана программа для работы с шифрованием строк на языке программирования Python. Программа включает в себя следующие функции: функцию для шифрования строки методом сдвига Цезаря, функцию для дешифрования строки, функцию для переворота строки, функцию для преобразования текста в верхний регистр и функцию для преобразования текста в нижний регистр. В одну из функций была заложена ошибка. Исходный код программы находится в Приложении А.

РАЗРАБОТКА ДОКУМЕНТАЦИИ МОДУЛЯ ПРОГРАММЫ

К созданной программе была написана следующая документация:

Данный модуль содержит набор функций для шифрования и дешифрования строк, а также выполнения операций с текстом, включая обратное отображение строк и изменение регистра символов. Эти функции полезны для обработки строковых данных в различных приложениях, связанных с шифрованием и анализом текста.

Функции:

encrypt(text, shift)

Описание функционала:

Шифрует строку с использованием метода сдвига Цезаря, где каждая буква смещается на заданное количество позиций.

Параметры:

- text (str): Исходная строка для шифрования.
- shift (int): Количество позиций для сдвига.

Возвращаемое значение:

- str: Зашифрованная строка.

Описание работы:

Функция преобразует каждую букву строки, сдвигая её на указанное количество позиций по алфавиту. Символы, не являющиеся буквами, не изменяются. Сдвиг применяется как к заглавным, так и к строчным буквам.

decrypt(text, shift)

Описание функционала:

Дешифрует строку, зашифрованную методом сдвига Цезаря с использованием того же значения сдвига.

Параметры:

- text (str): Зашифрованная строка.
- shift (int): Количество позиций, на которое был выполнен сдвиг при шифровании.

Возвращаемое значение:

- str: Дешифрованная строка.

Описание работы:

Функция выполняет обратное шифрование, сдвигая каждую букву строки на заданное количество позиций в противоположную сторону. Символы, не являющиеся буквами, не изменяются.

reverse(text)

Описание функционала:

Отображает строку в обратном порядке.

Параметры:

- text (str): Исходная строка для обратного отображения.

Возвращаемое значение:

- str: Строка, отображённая в обратном порядке.

Описание работы:

Функция переворачивает строку, меняя порядок символов на обратный.

to_upper(text)

Описание функционала:

Преобразует все символы строки в верхний регистр.

Параметры:

- `text (str)`: Исходная строка.

Возвращаемое значение:

- `str`: Строка в верхнем регистре.

Описание работы:

Функция проходит по строке и преобразует каждую букву в верхний регистр. Символы, не являющиеся буквами, остаются неизменными.

to_lower(text)

Описание функционала:

Преобразует все символы строки в нижний регистр (с ошибкой: функция возвращает строку в верхнем регистре).

Параметры:

- `text (str)`: Исходная строка.

Возвращаемое значение:

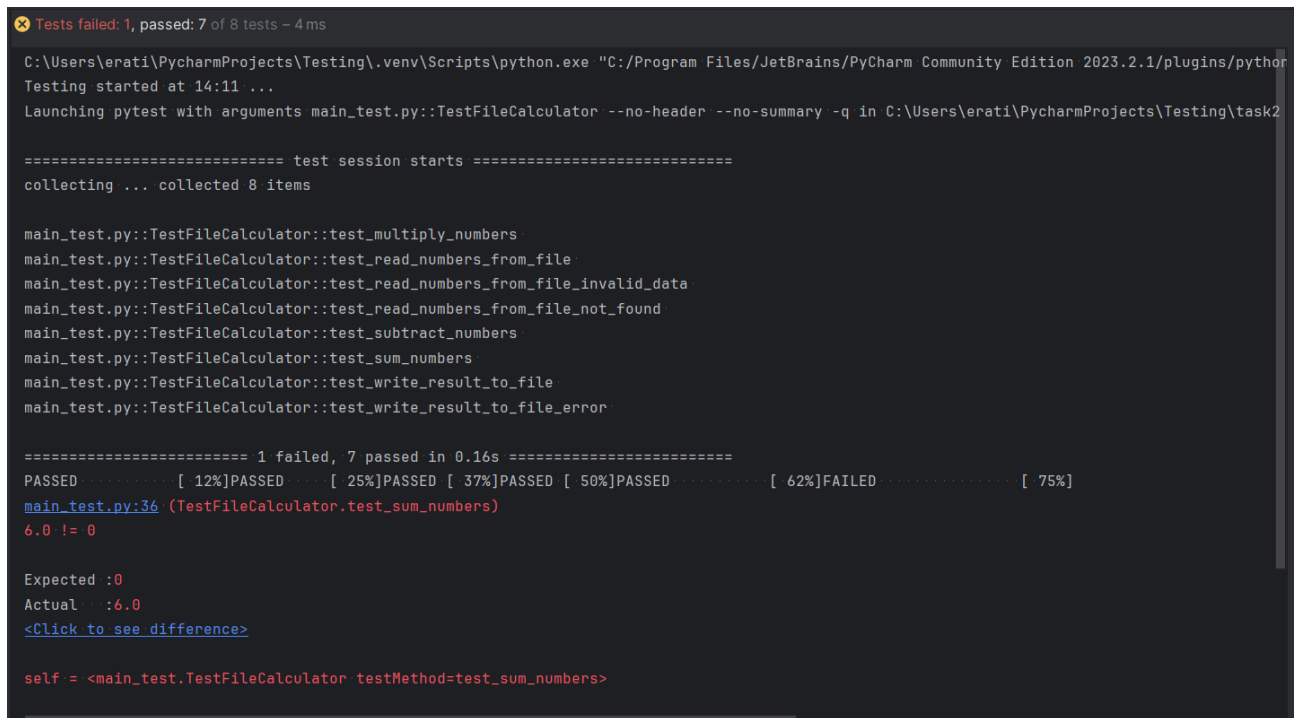
- `str`: Строка в нижнем регистре (ожидаемый результат).

Описание работы:

Функция должна преобразовать строку в нижний регистр, но из-за ошибки все символы преобразуются в верхний регистр вместо нижнего.

ТЕСТИРОВАНИЯ ПО

Были получены исходный код программы, которую необходимо протестировать, и документация. Для полученного ПО были написаны Unit-тесты для каждой из функций в программе. В итоге по одному из тестов была получена ошибка (Рис.1).



```

✖ Tests failed: 1, passed: 7 of 8 tests – 4 ms

C:\Users\erati\PycharmProjects\Testing\.venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2023.2.1/plugins/python
Testing started at 14:11 ...
Launching pytest with arguments main_test.py::TestFileCalculator --no-header --no-summary -q in C:\Users\erati\PycharmProjects\Testing\task2

===== test session starts =====
collecting ... collected 8 items

main_test.py::TestFileCalculator::test_multiply_numbers
main_test.py::TestFileCalculator::test_read_numbers_from_file
main_test.py::TestFileCalculator::test_read_numbers_from_file_invalid_data
main_test.py::TestFileCalculator::test_read_numbers_from_file_not_found
main_test.py::TestFileCalculator::test_subtract_numbers
main_test.py::TestFileCalculator::test_sum_numbers
main_test.py::TestFileCalculator::test_write_result_to_file
main_test.py::TestFileCalculator::test_write_result_to_file_error

===== 1 failed, 7 passed in 0.16s =====
PASSED [ 12%]PASSED [ 25%]PASSED [ 37%]PASSED [ 50%]PASSED [ 62%]FAILED [ 75%]
main_test.py:36 (TestFileCalculator.test_sum_numbers)
6.0 != 0

Expected :0
Actual   :6.0
<Click to see difference>

self = <main_test.TestFileCalculator testMethod=test_sum_numbers>
```

Рисунок 1 – Результат выполнения теста для функции test_sum_numbers

ИСПРАВЛЕНИЕ ОШИБКИ

Ошибка была задокументирована, документация была передана разработчику.

Краткое описание ошибки: «Неправильная операция суммирования».

Статус ошибки: открыта («Open»).

Категория ошибки: серьезная («Major»).

Тестовый случай: «Проверка алгоритма функционирования программы».

Описание ошибки:

1. Загрузить программу.
2. В качестве аргумента подать «[1.0, 2.0, 3.0]».
3. Нажать кнопку "Пуск".
4. Полученный результат: исключение «self = <main_test.TestFileCalculator testMethod=test_sum_numbers>».

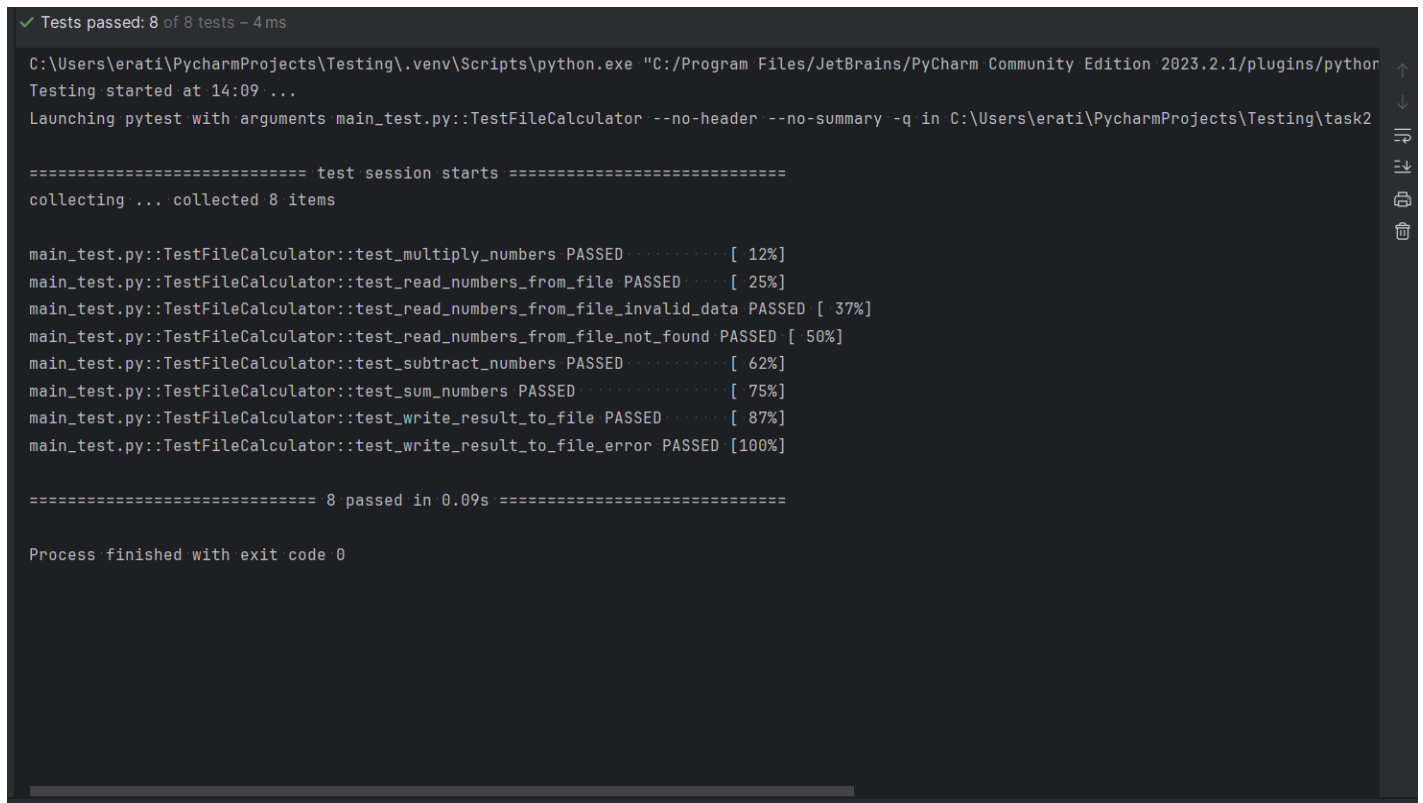
Ожидаемый результат: «

Expected :0

Actual :6.0».

ИТОГОВОЕ ТЕСТИРОВАНИЕ

После исправления ошибки, ПО было повторно протестировано. В результате все тесты проходят успешно (Рис. 2).



```
✓ Tests passed: 8 of 8 tests – 4 ms

C:\Users\erati\PycharmProjects\Testing\.venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2023.2.1/plugins/python
Testing started at 14:09 ...
Launching pytest with arguments main_test.py::TestFileCalculator --no-header --no-summary -q in C:\Users\erati\PycharmProjects\Testing\task2

===== test session starts =====
collecting ... collected 8 items

main_test.py::TestFileCalculator::test_multiply_numbers PASSED [ 12%]
main_test.py::TestFileCalculator::test_read_numbers_from_file PASSED [ 25%]
main_test.py::TestFileCalculator::test_read_numbers_from_file_invalid_data PASSED [ 37%]
main_test.py::TestFileCalculator::test_read_numbers_from_file_not_found PASSED [ 50%]
main_test.py::TestFileCalculator::test_subtract_numbers PASSED [ 62%]
main_test.py::TestFileCalculator::test_sum_numbers PASSED [ 75%]
main_test.py::TestFileCalculator::test_write_result_to_file PASSED [ 87%]
main_test.py::TestFileCalculator::test_write_result_to_file_error PASSED [100%]

===== 8 passed in 0.09s =====

Process finished with exit code 0
```

Рисунок 2 – Повторное тестирование

ЗАКЛЮЧЕНИЕ

В ходе практической работы были получены навыки проектирования и реализации модульных тестов для отдельных компонентов программного обеспечения.

ПРИЛОЖЕНИЯ

Приложение А – Программный код своего ПО.

Приложение Б – Программный код Unit-тестов тестируемого ПО.

Приложение А

Программный код своего ПО.

Листинг А.1 – Код программы

```
def encrypt(text, shift):
    encrypted_text = ""
    for char in text:
        if char.isalpha():
            shift_base = ord('a') if char.islower() else ord('A')
            encrypted_text += chr((ord(char) - shift_base + shift) % 26 +
shift_base)
        else:
            encrypted_text += char
    return encrypted_text

def decrypt(text, shift):
    decrypted_text = ""
    for char in text:
        if char.isalpha():
            shift_base = ord('a') if char.islower() else ord('A')
            decrypted_text += chr((ord(char) - shift_base - shift) % 26 +
shift_base)
        else:
            decrypted_text += char
    return decrypted_text

def reverse(text):
    return text[::-1]

def to_upper(text):
    return text.upper()

def to_lower(text):
    return text.upper() # используется upper вместо lower
```

Приложение Б

Программный код Unit-тестов тестируемого ПО.

Листинг Б.1 – Код тестов

```
import unittest
from unittest.mock import patch, mock_open
import main # Импортируем основной модуль с функциями

class TestFileCalculator(unittest.TestCase):

    # Тест для функции read_numbers_from_file
    @patch("builtins.open", new_callable=mock_open, read_data="1\n2\n3\n")
    def test_read_numbers_from_file(self, mock_file):
        result = main.read_numbers_from_file("input.txt")
        self.assertEqual(result, [1.0, 2.0, 3.0])

    @patch("builtins.open", new_callable=mock_open,
read_data="invalid\n2\n3\n")
    def test_read_numbers_from_file_invalid_data(self, mock_file):
        result = main.read_numbers_from_file("input.txt")
        self.assertEqual(result, "Ошибка: Некорректные данные в файле.")

    @patch("builtins.open", side_effect=FileNotFoundError)
    def test_read_numbers_from_file_not_found(self, mock_file):
        result = main.read_numbers_from_file("non_existent.txt")
        self.assertEqual(result, "Ошибка: Файл не найден.")

    # Тест для функции write_result_to_file
    @patch("builtins.open", new_callable=mock_open)
    def test_write_result_to_file(self, mock_file):
        result = main.write_result_to_file("output.txt", 30.0)
        self.assertIsNone(result)
        mock_file().write.assert_called_with("Результат: 30.0\n")

    @patch("builtins.open", side_effect=OSError("Ошибка ввода/вывода."))
    def test_write_result_to_file_error(self, mock_file):
        result = main.write_result_to_file("output.txt", 30.0)
        self.assertIn("Ошибка записи в файл:", result)

    # Тест для функции sum_numbers
    def test_sum_numbers(self):
        result = main.sum_numbers([1.0, 2.0, 3.0])
        self.assertEqual(result, 6.0)

    # Тест для функции subtract_numbers
    def test_subtract_numbers(self):
        result = main.subtract_numbers([10.0, 2.0, 3.0])
        self.assertEqual(result, 5.0)

    # Тест для функции multiply_numbers
    def test_multiply_numbers(self):
        result = main.multiply_numbers([2.0, 3.0, 4.0])
        self.assertEqual(result, 24.0)

if __name__ == '__main__':
    unittest.main()
```

