Санкт-Петербургский Национальный Исследовательский
Университет Информационных Технологий, Механики и Оптики

Факультет «Инфокоммуникационных Технологий»
Направление подготовки «Программирование в
инфокоммуникационных системах»

Лабораторная работа №1

Выполнил:
Крылов Дан Станиславович
Группа №3322
Проверил:
Кочубеев Николай Сергеевич

Санкт-Петербург
2024

Цель работы.

протестировать работу открытого кода из GitHub с помощью unit-тестов.

Задачи.

1. Выбор репозитория с GitHub
2. Анализ тестируемых функциональностей
3. Написание тестов

Ход работы.

1. **Выбор репозитория.**

   Был выбран репозиторий - https://github.com/ndleah/python-mini-project/tree/main/HangMan

   Он содержит код для игры «Виселица».

2. **Анализ тестируемых функциональностей**

   Функциональные элементы:

   - is_valid_guess (guess) - проверяет, является ли символ действительной буквой в нижнем регистре
   - display_word (secret_word, guessed) - отображает текущее состояние слова, замещая не предсказанные буквы подчеркиваниями
   - get_random_word_from_file(filename)- генерирует случайное слово из файла слов.
   - Main () - обрабатывает логику игры, запрашивая у пользователя буквы для отгадывания, управляя попытками и выводя результат.

   Критические части системы:

   - Проверка ввода пользователя (is_valid_guess() и проверка на повторные буквы)
   - Отображение результата текущего состояния слова (display_word())
   - Логика main()

•Получение слов из файла (get_random_word_from_file())

Важные случаи использования (Use Cases):

- Успешное угадывание слова
- Неудачное угадывание (исчерпание попыток)
- Ввод некорректного символа
- Повторное использование одной и той же буквы
- Повторная игра

3. **Написание тестов.**

На рисунке 1 приведены тесты.

```
import unittest
from unittest.mock import patch, mock_open
from hangman import is_valid_guess, display_word, get_random_word_from_file

class TestHangmanFunctions(unittest.TestCase):

    def test_is_valid_guess_valid(self):
        guess = 'a'
        result = is_valid_guess(guess)
        self.assertTrue(result)

    def test_is_valid_guess_invalid_non_alpha(self):
        guess = '1'
        result = is_valid_guess(guess)
        self.assertFalse(result)

    def test_is_valid_guess_invalid_uppercase(self):
        guess = 'A'
        result = is_valid_guess(guess)
        self.assertFalse(result)

    def test_display_word_correct_output(self):
        secret_word = "hangman"
        guessed = {'h', 'a'}
        expected_output = "h a _ _ _ a _"
        with patch('sys.stdout') as mock_stdout:
            display_word(secret_word, guessed)
            self.assertEqual(mock_stdout.getvalue(), expected_output)

    def test_get_random_word_from_file(self):
        mock_words = "example\nhangman\ntest\n"
        with patch("builtins.open", mock_open(read_data=mock_words)):
            result = get_random_word_from_file("fake_file.txt")
            self.assertIn(result, ['example', 'hangman', 'test'])

if name == "main"
```

Рисунок 1 – unit тесты

1. test_is_valid_guess_valid: проверяет, что функция правильно определяет допустимый ввод (одна строчная буква).

2. test_is_valid_guess_invalid_non_alpha: проверяет, что функция отклоняет недопустимый ввод (не буквенное значение).

3. test_is_valid_guess_invalid_uppercase: проверяет, что функция отклоняет недопустимый ввод (заглавная буква).

4. test_display_word_correct_output: проверяет правильный вывод функции display_word с помощью захвата вывода.

5. test_get_random_word_from_file: проверяет, что функция корректно выбирает случайное слово из заданного содержания файла.

Тесты соответствуют принципам AAA и FIRST. На примере функции применении AAA:

```
def test_is_valid_guess_valid(self):
# Arrange
guess = 'a'
# Act
result = is_valid_guess(guess)
# Assert
self.assertTrue(result)
```

FIRST реализовано, так как тесты быстрые, не зависят друг от друга, автоматически проверяются, своевременны и при повторе не меняют итогов. Результат запуска тестов приведен на рисунке 2.

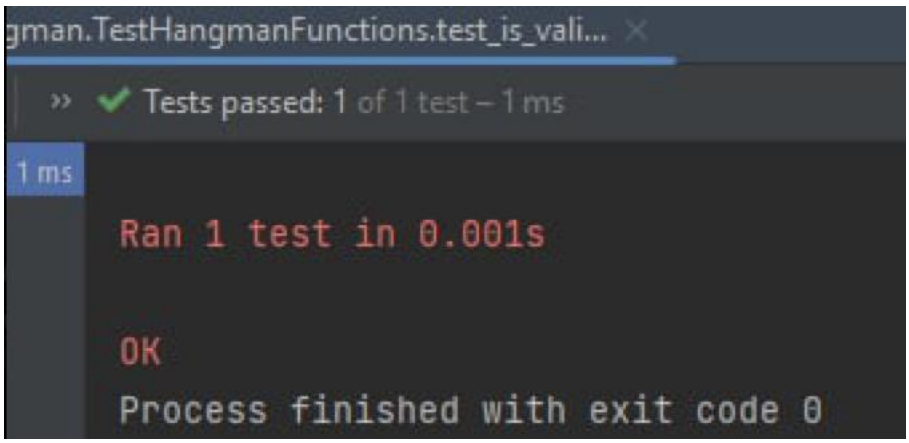

Рисунок 2 – Результат.

Вывод:

Достигнута цель работы - протестирована работа кода «Виселицы» из GitHub с помощью unit-тестов.