|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ |
| УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ |
| «СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» |
|  |
| Институт информационных технологий и управления в технических системах |
| (полное название института) |
|  |
| кафедра «Информационные системы» |
| (полное название кафедры) |

**ОТЧЁТ**

по лабораторной работе №5

на тему«Исследование способов модульного тестирования программного обеспечения в среде NUnit»

по дисциплине **«**Тестирование программного обеспечения»

Вариант 22

Выполнил

студент группы ПИ/б-18-1-о

Маринин Игорь

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | | | | | | | | |
|  |  | (должность, учёная степень преподавателя) | | | | | | | | |
|  |  |  | | | | | | | | |
|  |  | (ФИО преподавателя) | | | | | | | | |
|  |  | « |  | » |  |  |  | 20 | 20 | г. |
|  |  |  | | | | | | | | |
|  |  | (оценка) | | | | | | | | |
|  |  |  | | | | | | | | |
|  |  |  | | | | | | | | |

Севастополь 2020

* 1. **Цель работы**

Исследовать эффективность использования методологии TDD (Test-driven development) при разработке программного обеспечения. Получить практические навыки использования фреймворка NUnit для модульного тестирования программного обеспечения.

* 1. **Постановка задачи**

Применить методологию TDD при разработке класса Word с использованием фреймворка NUnit.из этого файла.

* 1. **Описание обязанностей тестируемого класса**

Тестируемый класс – Lab5.

Данный класс содержит метод ReplaceAllAfterDiez, принимающий строку. Возвращает преобразованную строку.

Для начала необходимо определить области эквивалентности входных данных. В рамках данной задачи они делятся на следующие категории:

1. По размеру строки

1. Строка из одного символа;
2. Строка, состоящая из более чем одного символа;
3. Пустая строка.

2. По расположению символов:

1. Все символы «#»;
2. Без символов «#»;
3. Символ «#» в начале строки;
4. Символ «#» в конце строки;
5. Символ «#» в середине сроки
   1. **Текст программы**

**main.py (пустой)**

**class** **Lab5(object):**

**def** ReplaceAllAfterDiez**(**s**):**

**pass**

**tests.py**

**import** os

**from** main **import** Lab5

**import** unittest

**class** **testStringReplacer(**unittest**.**TestCase**):**

**def** testOneSymbolString**(**self**):**

s **=** 'A'

expected **=** 'A'

actual **=** Lab5**.**ReplaceAllAfterDiez**(**s**);**

self**.**assertEqual**(**expected**,** actual**,**

f'Expected: {expected}\nGot: {actual}'**)**

**def** testEmptyString**(**self**):**

s **=** ''

expected **=** ''

actual **=** Lab5**.**ReplaceAllAfterDiez**(**s**);**

self**.**assertEqual**(**expected**,** actual**,**

f'Expected: {expected}\nGot: {actual}'**)**

**def** testAllDiezString**(**self**):**

s **=** '##########'

expected **=** '#@@@@@@@@@'

actual **=** Lab5**.**ReplaceAllAfterDiez**(**s**);**

self**.**assertEqual**(**expected**,** actual**,**

f'Expected: {expected}\nGot: {actual}'**)**

**def** testStringWithoutDiez**(**self**):**

s **=** 'Hello, world!'

expected **=** 'Hello, world!'

actual **=** Lab5**.**ReplaceAllAfterDiez**(**s**);**

self**.**assertEqual**(**expected**,** actual**,**

f'Expected: {expected}\nGot: {actual}'**)**

**def** testStringWithDiez**(**self**):**

s **=** 'Hello,#world!'

expected **=** 'Hello,#@@@@@@'

actual **=** Lab5**.**ReplaceAllAfterDiez**(**s**);**

self**.**assertEqual**(**expected**,** actual**,**

f'Expected: {expected}\nGot: {actual}'**)**

**def** testStringWithManyDiez**(**self**):**

s **=** 'Hel#o,#world!'

expected **=** 'Hel#@@@@@@@@@'

actual **=** Lab5**.**ReplaceAllAfterDiez**(**s**);**

self**.**assertEqual**(**expected**,** actual**,**

f'Expected: {expected}\nGot: {actual}'**)**

**def** testStringFirstDiez**(**self**):**

s **=** '#ello, world!'

expected **=** '#@@@@@@@@@@@@'

actual **=** Lab5**.**ReplaceAllAfterDiez**(**s**);**

self**.**assertEqual**(**expected**,** actual**,**

f'Expected: {expected}\nGot: {actual}'**)**

**def** testStringLastDiez**(**self**):**

s **=** 'Hello, world#'

expected **=** 'Hello, world#'

actual **=** Lab5**.**ReplaceAllAfterDiez**(**s**);**

self**.**assertEqual**(**expected**,** actual**,**

f'Expected: {expected}\nGot: {actual}'**)**

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

logFile **=** 'log\_file.txt'

**with** **open(**logFile**,** 'w'**)** **as** f**:**

runner **=** unittest**.**TextTestRunner**(**f**)**

unittest**.**main**(**testRunner **=** runner**,** verbosity**=**2**)**

**main.py (рабочий)**

**class** **Lab5(object):**

**def** ReplaceAllAfterDiez**(**s**):**

**try:**

first\_in **=** s**.**index**(**'#'**)** **+** 1

length **=** **len(**s**)**

**if** first\_in **>** length**:**

**return** s

**else:**

**return** s**[**0**:**first\_in**]** **+** '@' **\*** **(**length **-** first\_in**)**

**except** **ValueError:**

**return** s

* 1. **Тестирование**
     1. Проведём тестирование на ещё не реализованном методе (рисунок 5.1).

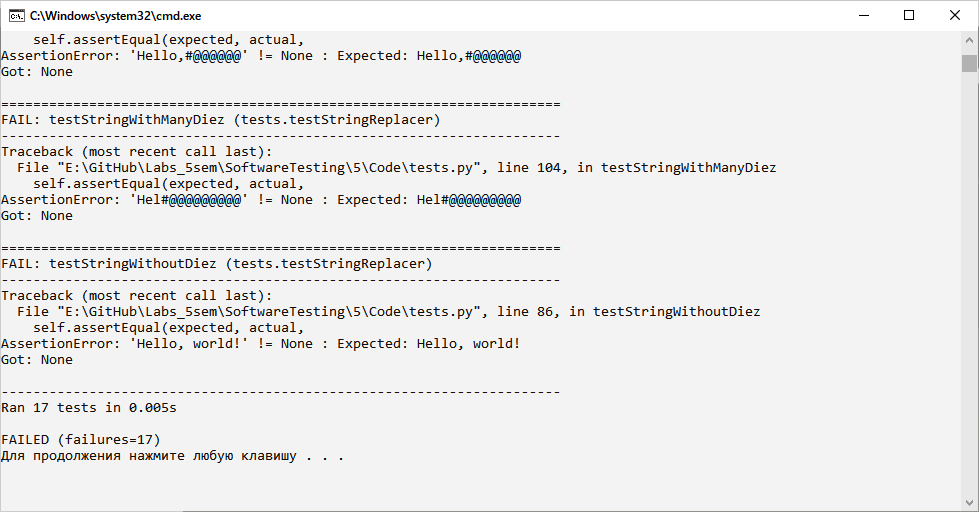


Рисунок 5.1 – Результат тестирования нереализованного метода

Как можно заметить, все тесты были успешно провалены.

* + 1. Реализуем метод ReplaceAllAfterDiez и посмотрим на результаты повторного теста (рисунок 5.2).

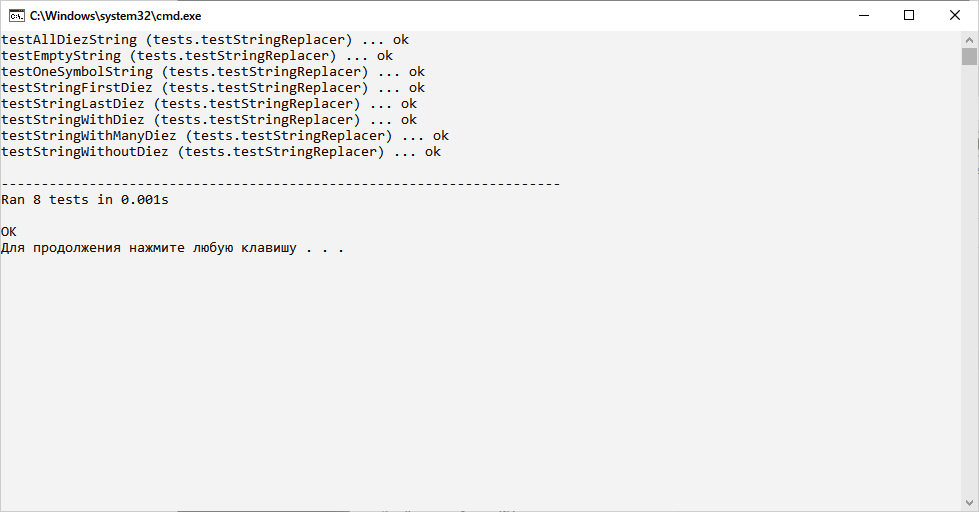


Рисунок 5.2 – Результат тестирования реализованного метода

**Вывод**

В ходе выполнения данной лабораторной работы были изучены основные принципы тестирования с помощью Unit-тестов. Для этого использовалась библиотека unittest.

В ходе выполнения лабораторной работы было выяснено, что метод ReplaceAllAfterDiez, находящийся в классе Lab5, работает верно.