Северо-Кавказский федеральный университет

Институт математики и информационных технологий

ОТЧЕТ
о выполнении лабораторной работы №9
по дисциплине
«Основы Программной Инженерии»

Выполнил:

Ботвинкин Никита Сергеевич

студент 2 курса, ПИЖ-б-о-21-1 группы
бакалавриата «Программная инженерия»

очной формы обучения

Ставрополь, 2023

СКРИНШОТЫ ВЫПОЛНЕНИЯ ИНДИВИДУАЛЬНОГО

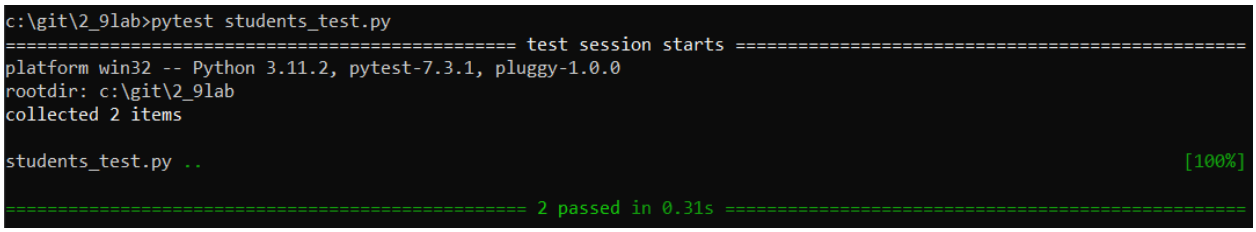

Рисунок 9.1 – Выполнение теста

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего используется автономное тестирование?

- Для тестирования функций, классов, методов и т.д. с целю выявления ошибок в работе в этих отдельных единицах общей программы.

2. Какие фреймворки Python получили наибольшее распространение для решения задач автономного тестирования?

- unittest, nose, pytest

3. Какие существуют основные структурные единицы модуля unittest?

- Test fixture, Test case, Test suite, Test runner

4. Какие существуют способы запуска тестов unittest?

- Запуск тестов можно сделать как из командной строки, так и с помощью графического интерфейса пользователя (GUI).

5. Каково назначение класса TestCase?

- Он представляет собой класс, который должен являться базовым для всех остальных классов, методы которых будут тестировать те или иные автономные единицы исходной программы.

6. Какие методы класса TestCase выполняются при запуске и завершении работы тестов?

- setUp(), teardown()

7. Какие методы класса TestCase используются для проверки условий и генерации ошибок?

`assertEqual(a, b)`	a == b
`assertNotEqual(a, b)`	a != b
`assertTrue(x)`	bool(x) is True
`assertFalse(x)`	bool(x) is False
`assertIs(a, b)`	a is b
`assertIsNot(a, b)`	a is not b
`assertIsNone(x)`	x is None
`assertIsNotNone(x)`	x is not None
`assertIn(a, b)`	a in b
`assertNotIn(a, b)`	a not in b
`assertIsInstance(a, b)`	isinstance(a, b)
`assertNotIsInstance(a, b)`	not isinstance(a, b)

Рисунок 9.2 – Методы проверки

`assertRaises(exc, fun, *args, **kwds)`	Функция `fun(*args, **kwds)` вызывает исключение exc
`assertRaisesRegex(exc, r, fun, *args, **kwds)`	Функция `fun(*args, **kwds)` вызывает исключение exc, сообщение которого совпадает с регулярным выражением r
`assertWarns(warn, fun, *args, **kwds)`	Функция `fun(*args, **kwds)` выдает сообщение warn
`assertWarnsRegex(warn, r, fun, *args, **kwds)`	Функция `fun(*args, **kwds)` выдает сообщение warn и оно совпадает с регулярным выражением r

Рисунок 9.3 – Исключения

`assertAlmostEqual(a, b)`	round(a-b, 7) == 0
`assertNotAlmostEqual(a, b)`	round(a-b, 7) != 0
`assertGreater(a, b)`	a > b
`assertGreaterEqual(a, b)`	a >= b
`assertLess(a, b)`	a < b
`assertLessEqual(a, b)`	a <= b
`assertRegex(s, r)`	r.search(s)
`assertNotRegex(s, r)`	not r.search(s)
`assertCountEqual(a, b)`	a и b имеют одинаковые элементы (порядок неважен)

Рисунок 9.4 – Проверка различных ситуаций

8. Какие методы класса TestCase позволяют собирать информацию о самом тесте?

- countTestCases(), id(), shortDescription()

9. Каково назначение класса TestSuite? Как осуществляется загрузка тестов?

- Класс TestSuite используется для объединения тестов в группы, которые могут включать в себя как отдельные тесты, так и заранее созданные группы.

10. Каково назначение класса TestResult?

- Класс TestResult используется для сбора информации о результатах прохождения тестов.

11. Для чего может понадобиться пропуск отдельных тестов?

- Во избежание ошибок тестирования, так как некоторые тесты могут давать заведомо неправильный результат в зависимости от какого-либо условия.

12. Как выполняется безусловный и условных пропуск тестов? Как выполнить пропуск класса тестов?

- @unittest.skip(reason) – Безусловный пропуска

- @unittest.skipIf(condition, reason) – Условный

- @unittest.skipUnless(condition, reason) - Условный

13. Самостоятельно изучить средства по поддержке тестов unittest в PyCharm. Приведите обобщенный алгоритм проведения тестирования с помощью PyCharm.

- В PyCharm есть встроенная поддержка unit тестов, которая позволяет создавать шаблон класса для тестирования и его дальнейшей настройки.

1) Необходимо создать класс для тестирования.

2) Написание кода тестов в классе для тестирования частей программы.

3) Запуск тестов

4) Debug тестов при необходимости.

5) Автоматизация тестов. PyCharm поддерживает автоматизацию тестов – установив её, вы можете сфокусироваться в написании кода самой программы, а IDE будет в автоматическом режиме проводить тестирование по мере изменения кода.