Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №8
дисциплины «Анализ данных»

Выполнила:
Кубанова Ксения Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р. А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: тестирование

Цель: приобрести навыки тестирования

Порядок выполнения работы

Индивидуальное задание.

Для индивидуального задания лабораторной работы 2.21 добавьте тесты с использованием модуля unittest, проверяющие операции по работе с базой данных.

```
> def added_train(name_bd, nomer, punkts, time): ···


class indTest(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        """Set up for class"""
        print("Проверка работы операций с базами данных")
        print("==========")

    @classmethod
    def tearDownClass(cls):
        """Tear down for class"""
        print("==========")
        print("Конец")

    def test_select_all_last(self):
        self.assertEqual(len(ind.select_all("test_bd")[-1]), 3)

    def test_select_all_first(self):
        self.assertEqual(len(ind.select_all("test_bd")[0]), 3)
```

Рисунок 1 –тестируемые функции

В коде представлены функции для тестирования валидности столбцов.

Данный код реализован в файле testind.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import unittest
import testind


if __name__ == "__main__":
    ind_test_suite = unittest.TestSuite()
    ind_test_suite.addTest(unittest.makeSuite(testind.indTest))
    runner = unittest.TextTestRunner(verbosity=2)
    runner.run(ind_test_suite)

```

Рисунок 2 – код для запуска файла тестирования

Код на рис. 2 необходим для того, чтобы корректно запустить файл с тестами. Его код реализован в testrun.py.

```
C:\Users\Сергей\OneDrive\Рабочий стол\ДЗ\2 курс\анализ данных\Lab8_Ad\prog>python testrun.py
C:\Users\Сергей\OneDrive\Рабочий стол\ДЗ\2 курс\анализ данных\Lab8_Ad\prog\testrun.py:10: DeprecationWarning: unittest.makeSuite() is deprecated and
 will be removed in Python 3.13. Please use unittest.TestLoader.loadTestsFromTestCase() instead.
  ind_test_suite.addTest(unittest.makeSuite(testind.indTest))
Проверка работы операций с базами данных
==========
test_select_all_first (testind.indTest.test_select_all_first) ... ok
test_select_all_last (testind.indTest.test_select_all_last) ... ok
==========
Конец
```

Рисунок 3 – результат тестирования

Тестируемый код находится в ind.py, а его база данных – test_bd.

Контрольные вопросы

1 Для чего используется автономное тестирование?

Автономное тестирование используется для автоматизации процесса тестирования программного обеспечения. Оно позволяет проверить работоспособность отдельных компонентов (функций, классов, модулей) или всей программы без необходимости ручного вмешательства.

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

unittest: встроенный модуль в стандартную библиотеку Python.

pytest: популярный сторонний фреймворк с богатым набором функциональности.

nose: еще один сторонний фреймворк, который расширяет функциональность unittest и pytest.

3 Какие существуют основные структурные единицы модуля unittest?

TestCase: класс, представляющий тестовый случай или набор тестов.

TestSuite: класс, представляющий собой набор тестовых случаев для выполнения.

TestLoader: класс, используемый для загрузки тестов из модулей и создания наборов тестов.

TestResult: класс, представляющий результаты выполнения тестов.

TestRunner: класс, используемый для запуска тестов и отображения результатов.

4 Какие существуют способы запуска тестов unittest?

Через командную строку с помощью утилиты unittest.

Используя среды разработки, такие как PyCharm, VSCode и другие, которые предоставляют встроенные инструменты для запуска тестов.

Через автоматизированные средства сборки и непрерывной интеграции, такие как Jenkins, Travis CI, GitHub Actions и т. д.

5 Каково назначение класса TestCase?

Класс TestCase предназначен для создания тестовых случаев. Он содержит методы для проверки различных аспектов функциональности программы.

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

setUp(): выполняется перед запуском каждого теста.

tearDown(): выполняется после завершения каждого теста.

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

assertEqual(): проверяет, что два объекта равны.

assertTrue(): проверяет, что условие истинно.

assertFalse(): проверяет, что условие ложно.

и другие методы, такие как assertRaises(), assertIn(), assertIsInstance() и т. д.

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

setUpClass(): выполняется один раз перед запуском всех тестов в классе.

tearDownClass(): выполняется один раз после завершения всех тестов в классе.

setUp() и tearDown(): как уже упоминалось, выполняются перед и после каждого теста соответственно.

Методы, начинающиеся с test_: представляют отдельные тесты и выполняются по одному.

9 Каково назначение класса TestSuite? Как осуществляется загрузка тестов?

Класс TestSuite используется для организации набора тестов. Тесты могут быть добавлены в TestSuite с помощью метода addTest() или addTests(). Загрузка тестов осуществляется путем создания экземпляра TestSuite и добавления в него соответствующих тестовых случаев.

Вывод: в ходе выполнения лабораторной работы были приобретены навыки тестирования