

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»

Выполнил:
Данилецкий Дмитрий Витальевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. технических
наук, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

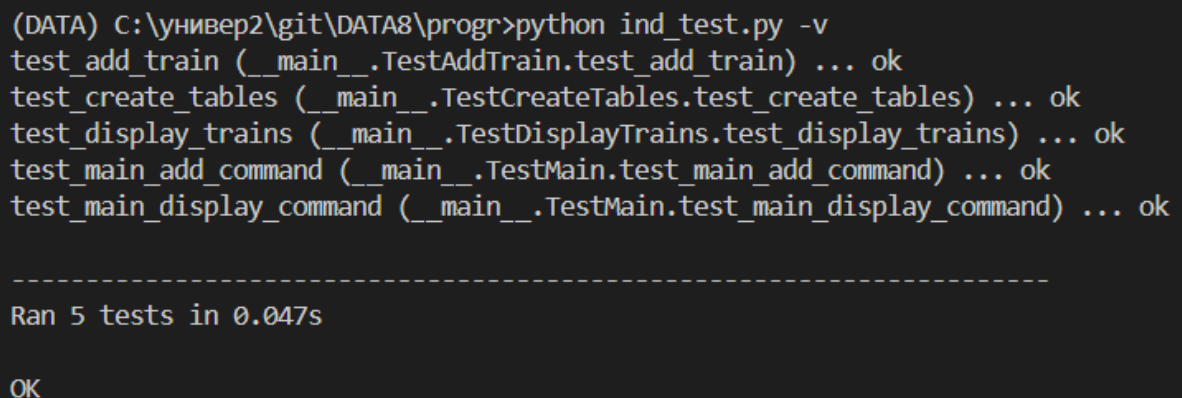
Тема: Тестирование в Python [unittest]

Цель работы: приобретение навыков написания автоматизированных тестов на языке программирования Python версии 3.x.

Ход работы

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python. Выполнил клонирование созданного репозитория.
2. Дополнил файл .gitignore необходимыми правилами.
3. Организовал созданный репозиторий в соответствии с необходимыми требованиями.
4. Проработал примеры лабораторной работы.
5. Выполнил индивидуальные задания, согласно варианту 8. Привёл в отчете скриншоты работы программ.

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



```
(DATA) C:\универ2\git\DATA8\progr>python ind_test.py -v
test_add_train (__main__.TestAddTrain.test_add_train) ... ok
test_create_tables (__main__.TestCreateTables.test_create_tables) ... ok
test_display_trains (__main__.TestDisplayTrains.test_display_trains) ... ok
test_main_add_command (__main__.TestMain.test_main_add_command) ... ok
test_main_display_command (__main__.TestMain.test_main_display_command) ... ok

-----
Ran 5 tests in 0.047s

OK
```

Рисунок 1. .работа программы индивидуального задания

Контрольные вопросы

1. Для чего используется автономное тестирование?

Автономное тестирование используется для автоматической проверки корректности работы программного обеспечения без вмешательства

человека. Это позволяет обнаруживать ошибки, повышать качество кода и ускорять процесс разработки.

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

Фреймворки Python для автономного тестирования: наибольшее распространение получили unittest (входит в стандартную библиотеку Python), pytest и nose (несмотря на то что разработка nose остановлена, его форк nose2 продолжает развиваться).

3. Какие существуют основные структурные единицы модуля unittest?

Основные структурные единицы модуля unittest:

TestCase: класс, представляющий отдельный тестовый случай.

TestSuite: коллекция тестовых случаев или тестовых наборов.

TestLoader: для загрузки тестов из TestCase и TestSuite.

TextTestRunner: класс для выполнения тестов и вывода результатов в текстовой форме.

TestResult: хранит результаты тестов.

4. Какие существуют способы запуска тестов unittest?

Способы запуска тестов unittest:

Использование интерфейса командной строки для запуска тестов (например, `python -m unittest discover`).

Использование `unittest.main()` внутри скрипта для запуска тестов.

Создание и использование объекта TestSuite для более сложных сценариев запуска.

5. Каково назначение класса TestCase?

Назначение класса TestCase: предоставляет рамки для создания тестовых случаев, включая методы для подготовки перед тестами (например, `setUp`) и очистки после тестов (например, `tearDown`), а также методы для самого тестирования.

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

Методы класса `TestCase`, выполняющиеся при запуске и завершении работы тестов:

`setUp()`: вызывается перед каждым тестовым методом.

`tearDown()`: вызывается после каждого тестового метода.

`setUpClass()`: вызывается перед запуском первого тестового метода в классе.

`tearDownClass()`: вызывается после завершения всех тестов в классе.

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

Методы класса `TestCase` для проверки условий и генерации ошибок включают `assertEqual()`, `assertTrue()`, `assertFalse()`, `assertRaises()` и многие другие.

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

Методы класса `TestCase` для сбора информации о тесте не столь явно выражены, как методы для проверки, но можно использовать `id()`, `shortDescription()` для получения информации о тестовом случае.

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

Назначение класса `TestSuite` - группировка и последовательный запуск тестов. Загрузка тестов осуществляется через `TestLoader` или путем добавления тестовых случаев и наборов в `TestSuite` вручную.

10. Каково назначение класса `TestResult`?

Назначение класса `TestResult` - хранение и представление результатов тестов. Он используется в `TestRunner` для сбора информации о прохождении тестов, включая количество успешных, неудачных и пропущенных тестов.

11. Для чего может понадобиться пропуск отдельных тестов?

Пропуск отдельных тестов может понадобиться, если тест временно неприменим, требует еще разработки или зависит от условий, которые в данный момент не выполнены.

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

Безусловный пропуск: декоратор `@unittest.skip("причина")`. Условный пропуск: декораторы вроде `@unittest.skipIf(condition, "причина")`. Пропуск класса тестов: использование тех же декораторов на уровне класса.

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

Алгоритм проведения тестирования в PyCharm:

Создайте тестовый файл в своем проекте.

Используйте структуры unittest, например, классы `TestCase`, для написания тестов.

В PyCharm, щелкните правой кнопкой мыши на тестовом файле или тестовом методе и выберите "Run 'Unit tests in <имя файла>" для запуска тестов.

Посмотрите результаты во вкладке Run, где PyCharm покажет успешные тесты и тесты с ошибками, предоставив детальную информацию по каждому случаю.

Вывод: в результате выполнения лабораторной работы были получены навыки по написанию автоматизированных тестов на языке программирования Python версии 3.x