**Бобер Євгеній Антонович**

**Звіт про виконання Лабораторної роботи №2**

**Робота з файлами. Юніт тести**

**Мета роботи**: Використовуючи теоретичне підґрунтя про роботу з файлами та тестування коду у мові Python розширити програму телефонного довідника студентів додавши функціонал, що буде вказано в завданні до лабораторної роботи.

**Хід роботи**

1. Завантаження даних з CSV-файлу

Першим завданням було навчити програму зчитувати початковий список студентів з файлу, ім'я якого передається через командний рядок.

* Імпорт необхідних модулів: На початку скрипту було додано імпорти модулів sys для доступу до аргументів командного рядка та csv для зручної роботи з CSV-форматом.
* Створення функції load\_data(filename): Була розроблена функція, яка:
* Приймає ім'я файлу як аргумент.
* Використовує конструкцію try...except FileNotFoundError для коректної обробки ситуації, коли файл ще не існує. Це дозволяє програмі запуститися з порожнім списком, якщо дані відсутні.
* Відкриває файл за допомогою with open(...), що гарантує його автоматичне закриття.
* Використовує csv.DictReader для читання рядків файлу одразу у форматі словників, що ідеально відповідає структурі даних програми.
* Після завантаження даних, список студентів сортується за іменем для підтримки логіки програми.
* Інтеграція в main(): У головній функції main() було додано логіку, яка:
* Перевіряє, чи було передано ім'я файлу як аргумент командного рядка (len(sys.argv) < 2).
* Якщо так, викликає функцію load\_data(), передаючи їй ім'я файлу (sys.argv[1]).

1. Збереження даних у CSV-файл

Наступним кроком була реалізація збереження всіх змін (додавання, оновлення, видалення) назад у той самий файл перед закриттям програми.

* Створення функції save\_data(filename): Була створена функція, яка:
* Відкриває файл на запис (mode='w').
* Визначає назви стовпців (fieldnames), динамічно отримуючи ключі з першого запису у списку студентів.
* Створює об'єкт csv.DictWriter, який дозволяє записувати словники у CSV-файл.
* Записує рядок із заголовками (writer.writeheader()).
* Записує весь список студентів у файл (writer.writerows(students)).
* Інтеграція в main(): Виклик функції save\_data() було додано в головний цикл програми під опцією виходу (case "x"), щоб збереження відбувалося безпосередньо перед завершенням роботи.

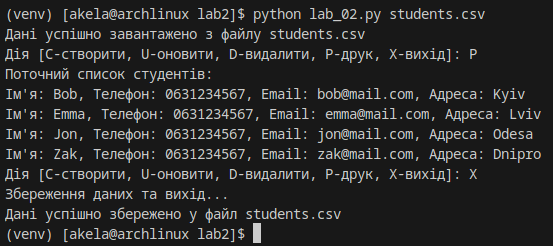
1. Покриття функціоналу Юніт-тестами

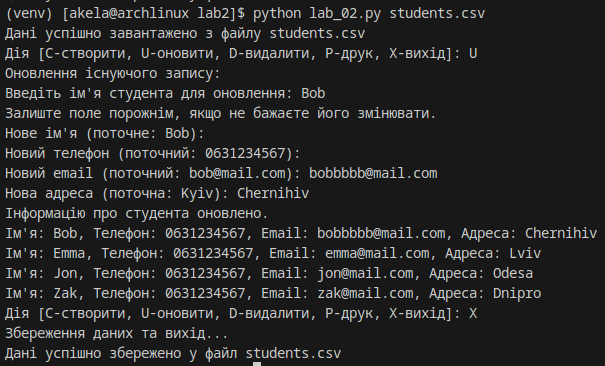
Для перевірки коректності роботи ключових функцій (addNewElement, deleteElement, updateElement) було створено набір автоматизованих тестів за допомогою бібліотеки Pytest.

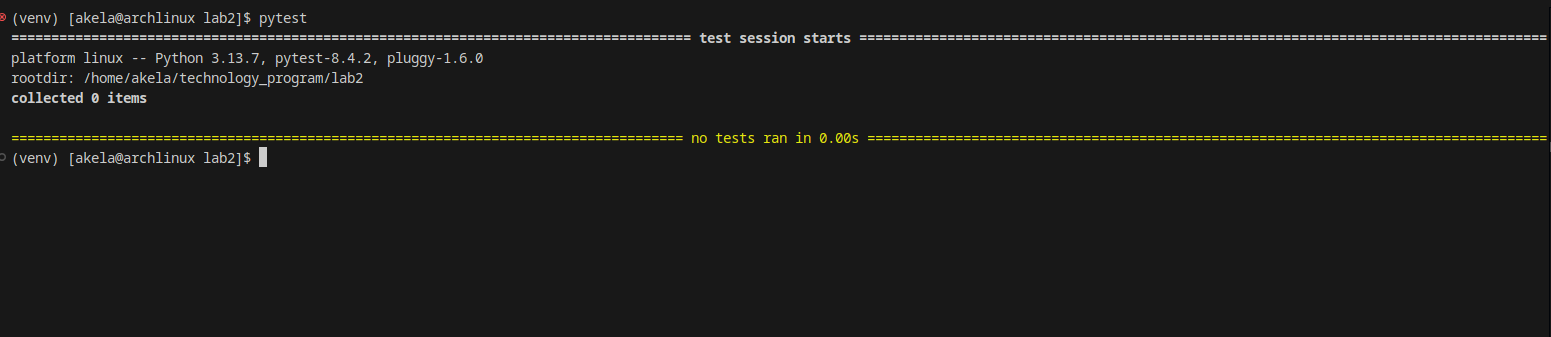
* Створення тестового файлу: Був створений файл test\_main.py.
* Використання фікстури: За допомогою @pytest.fixture була створена функція setup\_students, яка перед кожним тестом ініціалізує список students однаковими тестовими даними. Це забезпечує ізольованість та передбачуваність тестів.
* Імітація вводу користувача: Оскільки тестовані функції очікують вводу даних через input(), був використаний інструмент monkeypatch з pytest. Він дозволяє тимчасово "підмінити" стандартну функцію input() на власну, яка повертає заздалегідь визначені значення.
* Написання тестів
* test\_add\_new\_element: Перевіряє, чи правильно додається новий студент і чи залишається список відсортованим.
* test\_delete\_element\_exists: Перевіряє коректне видалення існуючого запису.
* test\_delete\_element\_not\_exists: Перевіряє, що програма не ламається і список не змінюється при спробі видалити неіснуючий запис.
* test\_update\_element: Перевіряє, що дані студента оновлюються правильно, і список пересортовується, якщо ім'я було змінено.
* Перевірка результатів: У кожному тесті використовуються інструкції assert для порівняння фактичного результату роботи функції (наприклад, довжина списку або імена в ньому) з очікуваним результатом.

**Посилання на Github:** <https://github.com/akelaqe/TP-KB-241-Bober>

**Скріншоти з фрагмента роботи програми:**

****

****

****

**Висновок**

В ході виконання цієї лабораторної роботи було успішно модернізовано програму-довідник, що значно розширило її функціональні можливості та надійність. Ключовим досягненням стала реалізація персистентності даних: програма навчилася зчитувати початкові дані з CSV-файлу при запуску та зберігати всі внесені зміни назад у файл перед завершенням роботи, що перетворило її на повноцінний інструмент для керування даними. Гнучкість програми було підвищено завдяки використанню модуля sys для обробки аргументів командного рядка, що дозволяє динамічно вказувати робочий файл. Крім того, надійність коду була значно посилена шляхом впровадження автоматизованого тестування за допомогою фреймворку pytest; створений набір юніт-тестів тепер перевіряє коректність основних операцій додавання, оновлення та видалення записів, захищаючи від можливих помилок у майбутньому. В процесі роботи я також здобув практичні навички використання віртуальних середовищ (venv) для ізоляції залежностей проєкту, що є стандартом у сучасній розробці. Таким чином, усі поставлені завдання були виконані, а я отримав важливі навички роботи з файловою системою, інструментами тестування та параметрами запуску, що є невід'ємною частиною розробки якісного програмного забезпечення.