**Звіт про виконання практичних завдань до лекцій з курсу Технології програмування на мові Python**

Звіт до 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.

Під час виконання Лабораторної роботи №2 було надано варіанти рішення до наступних задач:

Розширити функціональність телефонного довідника студентів групи task\_1.py (зробленого у Лабораторній роботі №1), наступними можливостями:

1) завантаження початкових даних для довідника з файлу lab2.csv. Ім’я файлу для завантаження даних повинно передаватись в якості параметрів командного рядку.

2) зберігання довідника студентів у форматі CSV перед виходом із програми.

3) покриття основного функціоналу програми Юніт тестами.

**Завантаження початкових даних для довідника з csv-файлу**

Для вирішення цього завдання я реалізував завантаження початкових даних для довідника з csv-файлу за допомогою функції load\_data\_from\_csv(file\_name). Параметр file\_name – це ім’я csv-файлу.

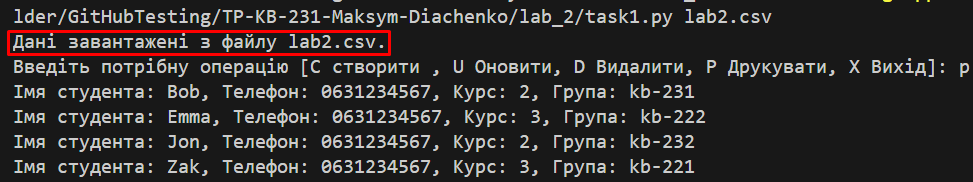
За допомогою конструкції Try-Except та команди With open() відбувається спроба відкриття csv-файлу.

Якщо файл відкрито успішно – то у змінну reader зчитуються строки csv-файлу командою csv.DictReader(file). Після цього на підставі reader створюється список студентів. Якщо відкриття файлу невдале – то видається повідомлення про помилку.

Для того щоб працювала функція csv.DictReader() потрібно на початку програми імпортувати бібліотеку роботи з csv-файлами, import csv.

Ім’я csv-файлу передається до програми і до функції load\_data\_from\_csv() за допомогою виклику цієї програми з командної строки: python task1.py lab2.csv.

Завантаження списку студентів із файлу lab2.csv робиться автоматично після запуску основної програми.



Текст функції load\_data\_from\_csv():

def load\_data\_from\_csv(file\_name):

    global students\_list

    try:

        with open(file\_name, mode='r', newline='', encoding='utf-8') as file:

            reader = csv.DictReader(file)

            students\_list = list(reader)

            print(f"Дані завантажені з файлу {file\_name}.")

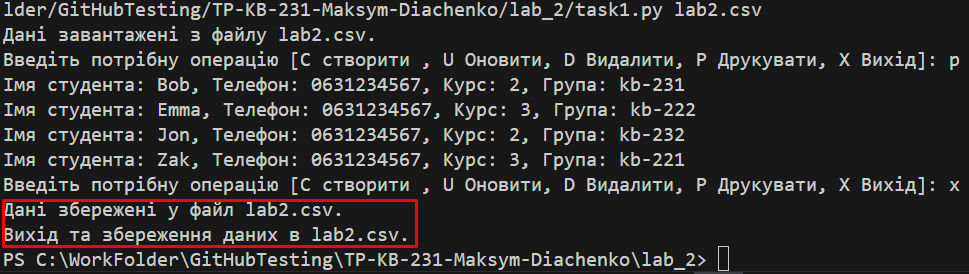
    except FileNotFoundError:

        print(f"Файл {file\_name} не знайдено.")

**Зберігання довідника студентів у форматі CSV перед виходом із програми.**

Для вирішення цього завдання я реалізував зберігання довідника студентів у форматі CSV перед виходом із програми за допомогою функції save\_data\_to\_csv(file\_name). Ця функція викликається з головної програми task1.py після введення користувачем букви «Х» - вихід з програми.

У даній функції першою командою виконується відкриття csv-файлу в режимі запису «W». Створюється список fieldnames із переліком назв полів списку студентів. За допомогою команди writer.writeheader() список полів виводиться першою строкою у csv-файл. Далі командою writer.writerows(students\_list) всі строки із списку студентів послідовно записуються у csv-файл.



Текст функції save\_data\_to\_csv():

def save\_data\_to\_csv(file\_name):

    global students\_list

    with open(file\_name, mode='w', newline='', encoding='utf-8') as file:

        fieldnames = ["name", "phone", "kurs", "group"]

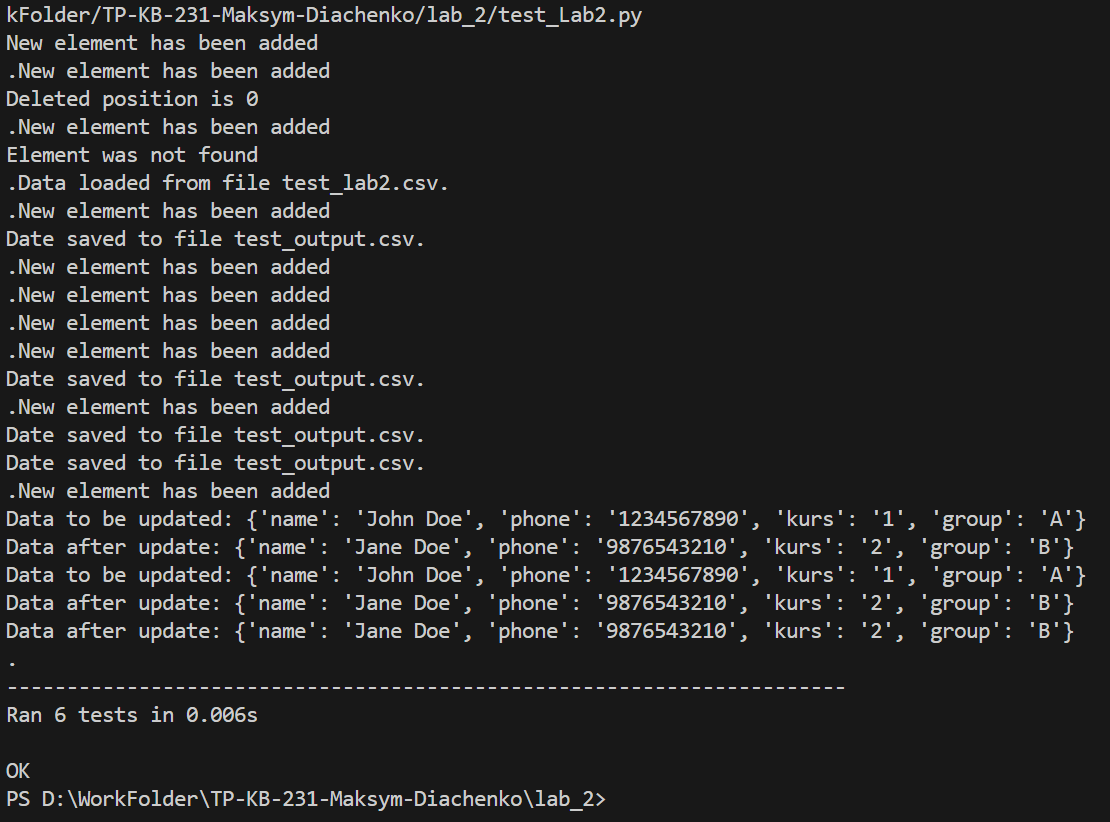
        writer = csv.DictWriter(file, fieldnames=fieldnames)

        writer.writeheader()

        writer.writerows(students\_list)

        print(f"Дані збережені у файл {file\_name}.")

**Покриття основного функціоналу програми Юніт тестами**



Текст програми для тестування Юніт-тестами:

import unittest

from Lab2 import loadDataFromCsv, saveDataToCsv, addNewElement, updateElement, deleteElement

global students

class TestLab2Functions(unittest.TestCase):

    def setUp(self):

        global students

        from Lab2 import students

        students.clear()

    def test\_addNewElement(self):

        addNewElement("John Doe", "1234567890", "1", "A")

        self.assertEqual(len(students), 1)

        self.assertEqual(students[0]["name"], "John Doe")

    def test\_deleteElement(self):

        addNewElement("John Doe", "1234567890", "1", "A")

        deleteElement("John Doe")

        self.assertEqual(len(students), 0)

    def test\_deleteElement\_not\_found(self):

        addNewElement("John Doe", "1234567890", "1", "A")

        deleteElement("Jane Smith")

        self.assertEqual(len(students), 1)

    def test\_updateElement(self):

        addNewElement("John Doe", "1234567890", "1", "A")

        updateElement("John Doe", "Jane Doe", "9876543210", "2", "B")

        self.assertEqual(students[0]["name"], "Jane Doe")

        self.assertEqual(students[0]["phone"], "9876543210")

    def test\_loadDataFromCsv(self):

        test\_csv = "test\_lab2.csv"

        with open(test\_csv, "w") as f:

            f.write("name,phone,kurs,group\n")

            f.write("JohnDoe,1234567890,1,A\n")

        loadDataFromCsv(test\_csv)

        self.assertEqual(len(students), 1)

        self.assertEqual(students[0]["name"], "JohnDoe")

    def test\_saveDataToCsv(self):

        test\_csv = "test\_output.csv"

        addNewElement("John Doe", "1234567890", "1", "A")

        saveDataToCsv(test\_csv)

        with open(test\_csv, "r") as f:

            lines = f.readlines()

        self.assertEqual(len(lines), 2)

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()

**Висновки.** Під час виконання цієї лабораторної роботи я навчився реалізовувати зчитування та зберігання даних csv-файлу використовуючи набір бібліотек csv. Також я навчився реалізовувати тестування коду програми за допомогою юніт-тестів.

Посилання на github:

<https://github.com/diachenkom/TP-KB-231-Maksym-Diachenko>

Знімок екрану з посилання на github:

