**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.

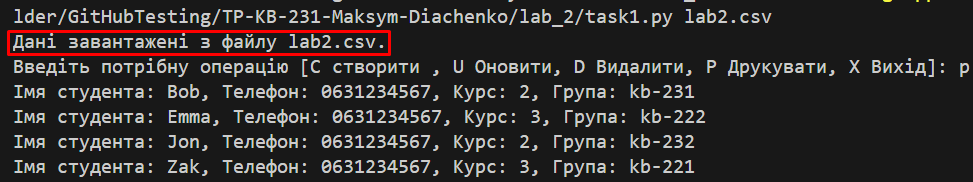
За допомогою конструкції Try-Except та команди With open() відбувається спроба відкриття csv-файлу.

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 loadDataFromCsv(file\_name):

    try:

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

            reader = csv.DictReader(file)

            for row in reader:

                students.append({"name":row["name"],"phone":row["phone"],"kurs":row["kurs"],"group":row["group"]})

            print(f"Data loaded from file {file\_name}.")

    except FileNotFoundError:

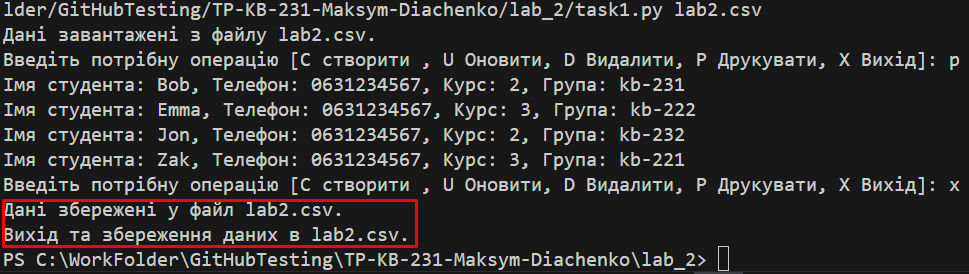
        print(f"File {file\_name} not found.")

    return

**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 saveDataToCsv(file\_name):

    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)

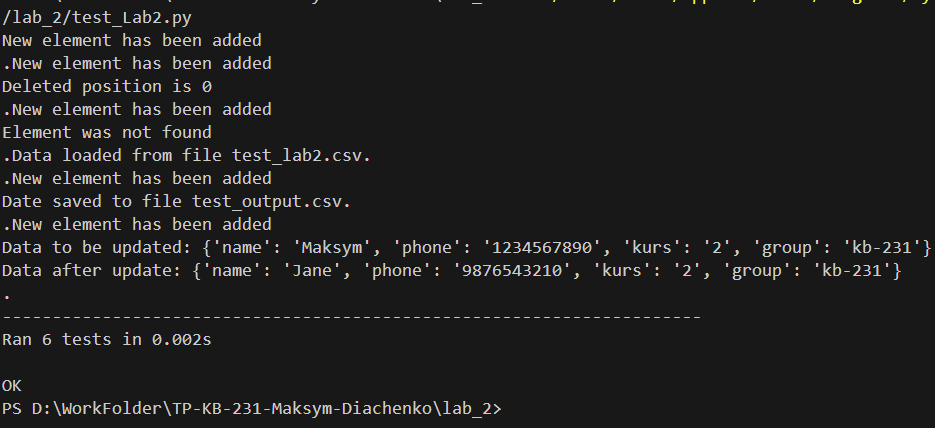
        print(f"Date saved to file {file\_name}.")

**Покриття основного функціоналу програми Юніт тестами**

Для вирішення цього завдання я розділив код програми на дві частини (файли main\_2.py та Lab2.py). У модулі Lab2.py я переніс всі функції по обробці списку студентів: додавання, видалення, редагування студента, завантаження та вивантаження списку студентів у файл csv. У модулі main\_2.py я залишив діалог програми з користувачем.

Для вирішення цього завдання я реалізував програму Юніт-тестів для модуля Lab2.py, вона тестує основні модулі обробки спиcку студентів, які знаходяться в Lab\_2.py: додавання, видалення, редагування студента в списку, завантаження та вивантаження списку студентів у файл csv.

Тести для головної програми main\_2.py я не створював, тому що в цій програмі прописаний діалог з користувачем та виклики потрібних функцій (модулів) з Lab\_2.py. Тестування такої програми як main\_2.py – це вже не модульне а інтеграційне тестування (тестування взаємодії різних модулів між собою). А в завданні було створення лише Юніт-тестів (модульне тестування).



Опишу коротко роботу програми Юніт-тестування test\_Lab2.py

Назва цієї програми сскладається з 2-х частин: “test\_” “Lab2.py” – так іменуються програми Юніт-тестів для коректної їх обробки бібліотекою тестування “unittest”

import unittest – спочатку імпортується бібліотека unittest

from Lab2 import loadDataFromCsv, saveDataToCsv, addNewElement, updateElement, deleteElement - ця команда імпортує з програми Lab2.py функції (модулі) по обробці списку студентів, які будуть тестуватись.

global students – оголошується глобальна змінна в якій буде зберігатись список студентів.

Class TestLab2Function(unittest.TestCase) – оголошується клас, в якому міститься набір тестів TestCase з бібліотеки unittest

def setUp(self):

global students

from Lab2 import students

students.clear()

Першим оголошується метод setup(), який автоматично викликається перед кожним юніт-тестом для очищення списку студентів перед початком тестування модуля. В цьому методі оголошується глобальна змінна students (список студентів), потім цей список імпортується з модуля Lab\_2.py і останньою командою список студентів очищається, щоб був порожнім перед тестуванням кожного модуля.

Метод

def test\_addNewElement(self):

addNewElement("Maksym", "1234567890", "2", "kb-231")

self.assertEqual(len(students), 1)

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

Наступний метод test\_addNewElement() виконує тестування додавання нового студента до списку students

Він так названий, бо до назви функції додавання студента addNewElement() добавлено префікс “test\_” – для обробки цієї функції бібліотекою unittest. Далі йде команда:

addNewElement(“Maksym”, “1234567890”, ”2”, “kb-231”)

Ця команда викликає функцію додавання нового студента до порожнього списку students. Після додавання студента запускаються 2 перевірки результату додавання:

self.assertEqual(len(students), 1) – довжина списку студентів має бути = 1

self.assertEqual(students[0]["name"], "Maksym") – ім’я “name” єдиного студента в списку з індексом [0] повинно бути =“Maksym”

Якщо ці 2 умови виконуються – тестування addNewElement() пройшло успішно, про що виводиться повідомлення в терміналі, чи командній стрічці.

Метод

def test\_deleteElement(self):

addNewElement("Maksym", "1234567890", "2", "kb-231")

deleteElement("Maksym")

self.assertEqual(len(students), 0)

Цей метод тестує функцію видалення студента який присутній у списку. Першою командою цього методу та всіх інших йде виклик функції addNewElement() – додавання 1-го студента до списку, т.я. список є пустим (його очищено методом setup())

Після додавання йде виклик функції видалення студента із заданим ім’ям deleteElement("Maksym")

Після видалення – перевірка кількості студентів у списку – має бути =0, якщо це так – тест видалення студента пройшов успішно, про що видається повідомлення в терміналі.

Метод

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

addNewElement("Maksym", "1234567890", "2", "kb-231")

deleteElement("Jane")

self.assertEqual(len(students), 1)

Цей метод тестує видалення студента, якого немає в списку з ім’ям "Jane".

Першою команду загружається 1 студент в список, наступною командою – виклик функції deleteElement(“Jane”) – спроба видалити студента, якого немає в списку.

Далі йде перевірка: довжина списку повинна залишитись незмінно =1. Якщо це так – ця перевірка успішна.

Метод

def test\_updateElement(self):

addNewElement("Maksym", "1234567890", "2", "kb-231")

updateElement("Maksym", "Jane", "9876543210", "1", "kb-242")

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

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

Цей метод тестує редагування даних студента, який є в списку.

Першою командою знову додається один і той же студент до пустого списку. Наступною командою – виклик функції updateElement(), яка змінює ldf атрибути студента який є в списку (ім’я та телефон).

Після цього виконуються 2 перевірки:

Перша – чи змінилось ім’я студента

Друга – чи змінився телефон студента

Якщо ці перевірки успішні – функція updateElement() переврена успішно, про що виводиться повідомлення в терміналі.

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("Maksym", "1234567890", "2", "kb-231")

        self.assertEqual(len(students), 1)

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

    def test\_deleteElement(self):

        addNewElement("Maksym", "1234567890", "2", "kb-231")

        deleteElement("Maksym")

        self.assertEqual(len(students), 0)

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

        addNewElement("Maksym", "1234567890", "2", "kb-231")

        deleteElement("Jane")

        self.assertEqual(len(students), 1)

    def test\_updateElement(self):

        addNewElement("Maksym", "1234567890", "2", "kb-231")

        updateElement("Maksym", "Jane", "9876543210", "2", "kb-231")

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

        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("Maksym,1234567890,2,kb-231\n")

        loadDataFromCsv(test\_csv)

        self.assertEqual(len(students), 1)

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

    def test\_saveDataToCsv(self):

        test\_csv = "test\_output.csv"

        addNewElement("Maksym", "1234567890", "2", "kb-231")

        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:

