**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.

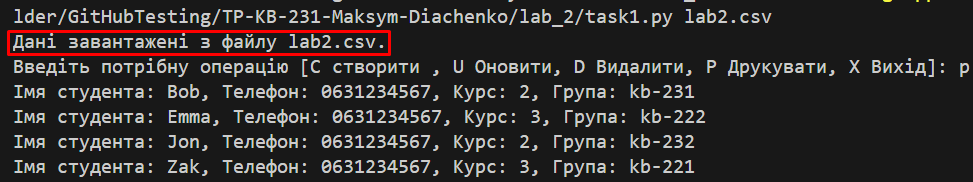
За допомогою конструкції Try-Except та команди With open() відбувається спроба відкриття csv-файлу.

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}.")

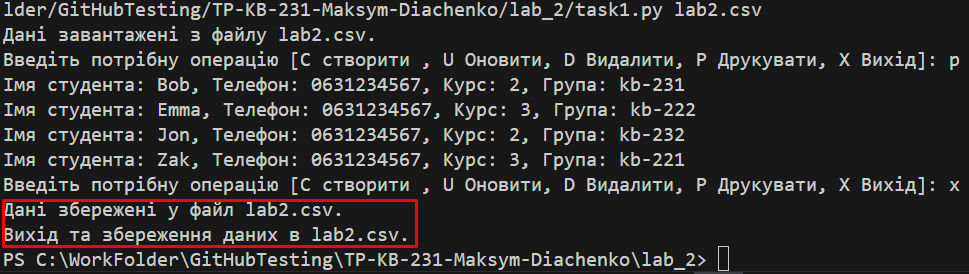
    except FileNotFoundError:

        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**

Для вирішення Завдання 3 (Юніт-тести) довелось робити копію task1\_t.py програмного файлу до першого та другого завдань task1.py, та вносити в нього певні зміни для того, щоб перевірити його Юніт-тестами. Проблема була в тому, що в кожній функції обробки довідника студентів (додавання, видалення та редагування запису) дані вводились через діалог з користувачем. А Юніт-тести мають перевіряти правильність роботи коду функцій автоматично, без втручання користувача. Тому мені довелось адаптувати 3 функції редагування довідника студентів, видаляючи з них команди діалогу з користувачем:

а) в заголовку функції add\_new\_element(new\_item) в дужках я додав параметр new\_item – запис нового студента в довідник. Потім я закоментував усі команди введення даних нового студента і команду присвоєння запису для нового студента в змінну new\_item, так як цей запис передається в функцію під час її виклику з процедури Юніт-тесту test\_add\_new\_element().

Цей Юніт-тест test\_add\_new\_element() додає новий запис в довідник, після чого перевіряє: чи збільшилась на 1 кількість записів у довіднику (спочатку було 4, після додавання 1-го – стало 5), це робить команда:

self.assertEqual(len(students\_list), 5)

б) в заголовку функції delete\_element(name) в дужках я додав параметр name – ім’я студента, запис якого треба видалити. Потім я закоментував команди введення імені студента для видалення, так як це ім’я передається в функцію delete\_element(name) як параметр під час її виклику з Юніт-тесту test\_delete\_element()

Цей Юніт-тест спочатку знаходить та видаляє запис студента із заданим ім’ям «Zak». Після цього виконується перевірка зменшення на 1 кількості записів в довіднику (до видалення було 5, після видалення стало 4) командою:

self.assertEqual(len(students\_list), 4)

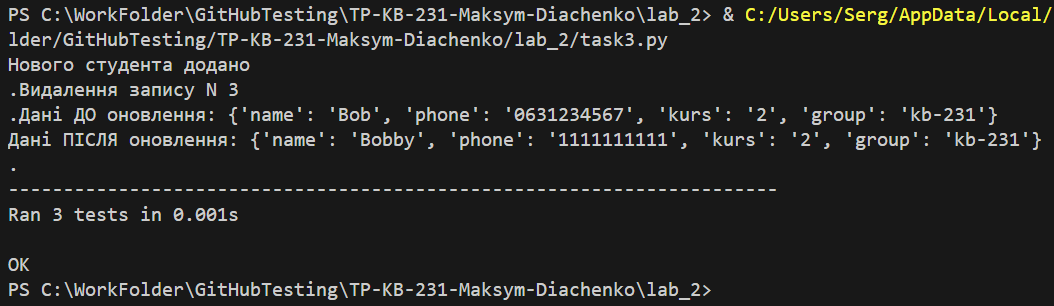
Після цього перевіряється відсутність запису видаленого студента в довіднику:

self.assertNotIn({"name": "Zak", "phone": "0631234567", "kurs": "3", "group": "kb-221"}, students\_list)

в) в заголовку функції update\_element(name, item\_upd) в дужках я додав 2 параметри, які будуть передані до неї з Юніт-тесту test\_update\_element(). Це такі параметри: name – ім’я студента запис якого треба знайти і змінити, item\_upd – повністю змінений запис студента, який буде збережено в довіднику функцією update\_element(name, item\_upd). В коді цієї функції я закоментував команду вводу імені студента, запис якого треба редагувати, та 4 команди вводу оновлених даних цього студента. Адже всі ці дані передаються в функцію update\_element(name, item\_upd) під час її виклику.

Цей Юніт-тест спочатку знаходить в довіднику запис студента «Bob» та змінює його ім’я на «Bobby» та номер телефону на «1111111111». Потім виконується перевірка, чи ім’я першого студента в довіднику «Bob» змінено на «Bobby»

Після внесення таких змін Юніт-тести почали працювати без помилок:



Текст програми для тестування Юніт-тестами:

import unittest

from task1\_t import students\_list, add\_new\_element, delete\_element, update\_element

class TestStudentDirectory(unittest.TestCase):

    def setUp(self):

        students\_list = [

    {"name": "Bob", "phone": "0631234567", "kurs": "2", "group": "kb-231"},

    {"name": "Emma", "phone": "0631234567", "kurs": "3", "group": "kb-222"},

    {"name": "Jon", "phone": "0631234567", "kurs": "2", "group": "kb-232"},

    {"name": "Zak", "phone": "0631234567", "kurs": "3", "group": "kb-221"}

]

    def test\_add\_new\_element(self):

        add\_new\_element({"name": "Charlie", "phone": "5555555555", "kurs": "3", "group": "kb-303"})

        self.assertEqual(len(students\_list), 5)

    def test\_delete\_element(self):

        delete\_element("Zak")

        self.assertEqual(len(students\_list), 4)

        self.assertNotIn({"name": "Zak", "phone": "0631234567", "kurs": "3", "group": "kb-221"}, students\_list)

    def test\_update\_element(self):

        update\_element("Bob", {"name": "Bobby", "phone": "1111111111", "kurs": "2", "group": "kb-231"})

        self.assertEqual(students\_list[0]["name"], "Bobby")

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

    unittest.main()

Текст програми task1\_t.py, що являє собою відредаговану програму із першого завдання для можливості тестування Юніт-тестами:

import csv

import sys

# Початковий список студентів

students\_list = [

    {"name": "Bob", "phone": "0631234567", "kurs": "2", "group": "kb-231"},

    {"name": "Emma", "phone": "0631234567", "kurs": "3", "group": "kb-222"},

    {"name": "Jon", "phone": "0631234567", "kurs": "2", "group": "kb-232"},

    {"name": "Zak", "phone": "0631234567", "kurs": "3", "group": "kb-221"}

]

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.")

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}.")

def print\_all\_list():

    for elem in students\_list:

        str\_for\_print = f"Імя студента: {elem['name']}, Телефон: {elem['phone']}, Курс: {elem['kurs']}, Група: {elem['group']}"

        print(str\_for\_print)

def add\_new\_element(new\_item):

#    name = input("Уведіть ім'я студента: ")

#    phone = input("Уведіть номер телефону студента: ")

#    kurs = input("Уведіть курс: ")

#    group = input("Уведіть групу: ")

#    new\_item = {"name": name, "phone": phone, "kurs": kurs, "group": group}

    students\_list.append(new\_item)

    print("Нового студента додано")

def delete\_element(name):

#   name = input("Уведіть ім'я студента якого бажаєте видалити: ")

    deletePosition = -1

    for item in students\_list:

        if name == item["name"]:

            deletePosition = students\_list.index(item)

        if deletePosition != -1:

            print("Видалення запису № " + str(deletePosition))

            del students\_list[deletePosition]

            return

def update\_element(name, item\_upd):

#    name = input("Уведіть ім'я студента дані якого бажаєте редагувати: ")

    for item in students\_list:

        if item["name"] == name:

            print("Дані ДО оновлення:", item)

#            item["name"] = input("Уведіть нове ім'я: ")

            item["name"] = item\_upd["name"]

#            item["phone"] = input("Уведіть новий телефон: ")

            item["phone"] = item\_upd["phone"]

#            item["kurs"] = input("Уведіть новий курс: ")

            item["kurs"] = item\_upd["kurs"]

#            item["group"] = input("Уведіть нову групу: ")

            item["group"] = item\_upd["group"]

            print("Дані ПІСЛЯ оновлення:", item)

            return

    print("Елемент не знайдено.")

def main():

    if len(sys.argv) > 1:

        load\_data\_from\_csv(sys.argv[1])

    while True:

        choice = input("Введіть потрібну операцію [C -створити , U -Оновити, D -Видалити, P Друкувати, X Вихід]: ").lower()

        if choice == "c":

            add\_new\_element()

        elif choice == "u":

            update\_element()

        elif choice == "d":

            delete\_element()

        elif choice == "p":

            print\_all\_list()

        elif choice == "x":

            save\_data\_to\_csv("lab2.csv")

            print("Вихід та збереження даних в lab2.csv.")

            break

        else:

            print("Невірний код операції, спробуйте ще раз")

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

    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:

