Кондратенко Катерина КБ-232

Лабораторна робота №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**

**Завдання до лабораторної роботи:** Розширити функціональність телефонного довідника студентів групи, що був розроблений у Лабораторній роботі №1 наступними можливостями:

1. **завантаження** початкових данні для довідника з csv файлу. 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**.

Текст програми:

**studentscsv.py**

import csv

data = [

    {"name": "Bob", "phone": "0631234567", "email": "bob@example.com", "address": "895 Street"},

    {"name": "Emma", "phone": "0631234567", "email": "emma@example.com", "address": "123 Street"},

    {"name": "Jon", "phone": "0631234567", "email": "jon@example.com", "address": "27 Street"},

    {"name": "Zak", "phone": "0631234567", "email": "zak@example.com", "address": "9 Street"}

]

file\_name = "students.csv"

# Створення CSV файлу

with open(file\_name, mode='w', newline='') as file:

    writer = csv.DictWriter(file, fieldnames=["name", "phone", "email", "address"])

    writer.writeheader()

    writer.writerows(data)

print(f"{file\_name} has been created.")

**phone\_directory.py**

import csv

import sys

directory = []

def loadFromCsv(file\_name):

    global directory

    try:

        with open(file\_name, mode='r') as file:

            reader = csv.DictReader(file)

            directory = [row for row in reader]

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

    except FileNotFoundError:

        print(f"File {file\_name} not found. Starting with an empty directory.")

    return

def saveToCsv(file\_name):

    global directory

    with open(file\_name, mode='w', newline='') as file:

        fieldnames = ["name", "phone", "email", "address"]

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

        writer.writeheader()

        writer.writerows(directory)

    print(f"Data successfully saved to {file\_name}")

    return

def printAllList():

    if not directory:

        print("Directory is empty.")

    for elem in directory:

        strForPrint = f"Student name: {elem['name']}, Phone: {elem['phone']}, Email: {elem['email']}, Address: {elem['address']}"

        print(strForPrint)

    return

def addNewElement(name=None, phone=None, email=None, address=None):

    if name is None:

        name = input("Please enter student name: ")

    if phone is None:

        phone = input("Please enter student phone: ")

    if email is None:

        email = input("Please enter student email: ")

    if address is None:

        address = input("Please enter student address: ")

    newItem = {"name": name, "phone": phone, "email": email, "address": address}

    insertPosition = 0

    for item in directory:

        if name.lower() > item["name"].lower():

            insertPosition += 1

        else:

            break

    directory.insert(insertPosition, newItem)

    print("New element has been added.")

    return

def deleteElementByName(name):

    deletePosition = -1

    for item in directory:

        if name.lower() == item["name"].lower():

            deletePosition = directory.index(item)

            break

    if deletePosition != -1:

        del directory[deletePosition]

        print(f"Element with name {name} has been deleted.")

    else:

        print(f"No element with name {name} found.")

    return

def updateElement():

    name = input("Please enter the name of the student to be updated: ")

    student = next((item for item in directory if item["name"] == name), None)

    if not student:

        print("Element was not found.")

        return

    print("Leave the field blank if you don't want to update it.")

    new\_name = input(f"Enter new name (current: {student['name']}): ") or student['name']

    phone = input(f"Enter new phone (current: {student['phone']}): ") or student['phone']

    email = input(f"Enter new email (current: {student['email']}): ") or student['email']

    address = input(f"Enter new address (current: {student['address']}): ") or student['address']

    deleteElementByName(name)

    addNewElement(new\_name, phone, email, address)

    print(f"Information about {name} has been updated.")

    return

def main():

    if len(sys.argv) > 1:

        loadFromCsv(sys.argv[1])

    while True:

        choice = input("Please specify the action [ C create, U update, D delete, P print, X exit ]: ")

        match choice:

            case "C" | "c":

                print("New element will be created:")

                addNewElement()

                printAllList()

            case "U" | "u":

                print("Existing element will be updated:")

                updateElement()

                printAllList()

            case "D" | "d":

                print("Element will be deleted:")

                name\_to\_delete = input("Please enter name to be deleted: ")

                deleteElementByName(name\_to\_delete)

                printAllList()

            case "P" | "p":

                print("List will be printed:")

                printAllList()

            case "X" | "x":

                if len(sys.argv) > 1:

                    saveToCsv(sys.argv[1])

                print("Exit()")

                break

            case \_:

                print("Wrong choice")

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

    main()

**test\_phone\_directory.py**

import unittest

from unittest.mock import patch

from phone\_directory import addNewElement, deleteElementByName, updateElement, directory

class TestPhoneBook(unittest.TestCase):

    def setUp(self):

        global directory

        directory = [

        {"name": "Bob", "phone": "0631234567", "email": "bob@example.com", "address": "895 Street"},

        {"name": "Emma", "phone": "0631234567", "email": "emma@example.com", "address": "123 Street"}

    ]

    def test\_addNewElement(self):

        addNewElement("Zak", "0631234567", "zak@example.com", "9 Street")

        self.assertEqual(len(directory), 3)

        self.assertEqual(directory[2]["name"], "Zak")

    def test\_deleteElementByName(self):

        deleteElementByName("Emma")

        self.assertEqual(len(directory), 1)

        self.assertEqual(directory[0]["name"], "Bob")

    def test\_updateElement(self):

        addNewElement("Jon", "0631234567", "jon@example.com", "27 Street")

        with patch("builtins.input", side\_effect=["Jon", "John", "0998765432", "", "New Address"]):

            updateElement()

        updatedStudent = next((s for s in directory if s["name"] == "John"), None)

        self.assertIsNotNone(updatedStudent)

        self.assertEqual(updatedStudent["phone"], "0998765432")

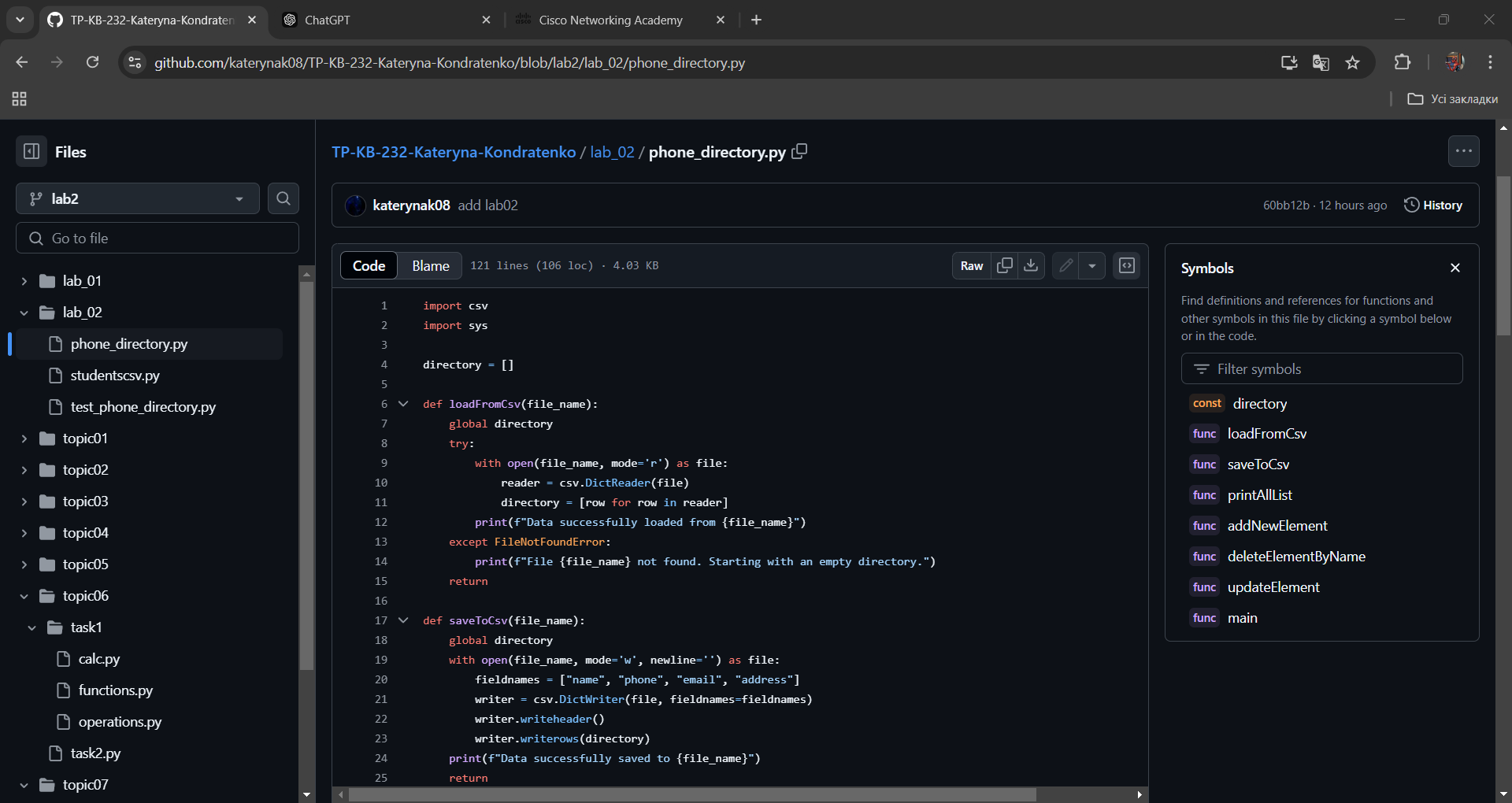
if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()

Посилання на github:

https://github.com/katerynak08/TP-KB-232-Kateryna-Kondratenko/tree/lab2/lab\_02

Знімок екрану з посилання на github:



**Висновок:** у цій лабораторній роботі я розширила функціонал телефонного довідника студентів, додавши можливість роботи з файлами. Також ознайомилася з принципами юніт тестування та написала тести для перевірки основних функцій програми. Завдяки цьому вдалося забезпечити надійну роботу коду та збереження даних. Отримані знання стануть в нагоді при подальшій розробці програмного забезпечення.