**Лабораторна робота №2**

**Тема**: Робота з файлами. Юніт тести.

**Мета**: 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.

**Завдання до лабораторної роботи**

1. Завантаження початкових данних для довідника з csv файлу. Приклад csv файлу (students.csv), та оригінальній файл формату xlsx розміщені поряд з вказівками. 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.

**Хід роботи**

Програма lab\_02.py дозволяє працювати зі списком студентів, використовуючи CSV-файл для збереження даних.

1. **Функції**:
   * Завантаження (load\_from\_csv) і збереження (save\_to\_csv) даних у CSV.
   * Перегляд списку (printAllList).
   * Додавання, видалення та оновлення студентів з дотриманням сортування за іменами.
2. **Робота**:
   * При запуску дані завантажуються з файлу (якщо вказано).
   * Користувач обирає дії через меню: додати, оновити, видалити студента, переглянути список або завершити роботу.
   * Зміни зберігаються у файл перед виходом.
3. **Тести (test\_lab\_02.py)**:
   * Перевіряють додавання, видалення та оновлення студентів.
   * Гарантують правильність функцій за допомогою бібліотеки unittest.

Текст програми:

|  |
| --- |
| # lab\_02.py  import csv  import sys  #Початковий список студентів, відсортований за іменами  students = []  #Функція для завантаження початкових даних із CSV файлу  def load\_from\_csv(data\_lab2):  try:  with open(data\_lab2, mode='r', encoding='utf-8') as file:  reader = csv.DictReader(file)  for row in reader:  students.append(row)  students.sort(key=lambda x: x['name']) #Сортуємо за іменами  print(f"Data loaded from {data\_lab2}")  except FileNotFoundError:  print(f"File {data\_lab2} not found. Starting with an empty phonebook.")  #Функція для збереження списку студентів у CSV файл  def save\_to\_csv(data\_lab2):  with open(data\_lab2, mode='w', encoding='utf-8', newline='') as file:  fieldnames = ["name", "phone", "email", "group"]  writer = csv.DictWriter(file, fieldnames=fieldnames)  writer.writeheader()  writer.writerows(students)  print(f"Data saved to {data\_lab2}")  #Функція для друку списку студентів  def printAllList():  for elem in students:  print(f"Name: {elem['name']}, Phone: {elem['phone']}, Email: {elem['email']}, Group: {elem['group']}")  #Функція для додавання нового студента  def addNewElement():  name = input("Enter student name: ")  phone = input("Enter student phone: ")  email = input("Enter student email: ")  group = input("Enter student group: ")  newItem = {"name": name, "phone": phone, "email": email, "group": group}  #Пошук позиції для вставки (щоб список залишався відсортованим)  insertPosition = 0  for item in students:  if name > item["name"]:  insertPosition += 1  else:  break  students.insert(insertPosition, newItem)  print("New student has been added.")  #Функція для видалення студента  def deleteElement():  name = input("Enter the name of the student to delete: ")  deletePosition = -1  for item in students:  if name == item["name"]:  deletePosition = students.index(item)  break  if deletePosition == -1:  print("Student not found.")  else:  del students[deletePosition]  print(f"Student '{name}' has been deleted.")  #Функція для зміни інформації про студента  def updateElement():  name = input("Enter the name of the student to update: ")  updatePosition = -1  for item in students:  if name == item["name"]:  updatePosition = students.index(item)  break  if updatePosition == -1:  print("Student not found.")  else:  print("Enter new details for the student (leave empty to keep current values):")  phone = input(f"Current phone ({students[updatePosition]['phone']}): ") or students[updatePosition]["phone"]  email = input(f"Current email ({students[updatePosition]['email']}): ") or students[updatePosition]["email"]  group = input(f"Current group ({students[updatePosition]['group']}): ") or students[updatePosition]["group"]  #Оновлення даних  students.pop(updatePosition)  newItem = {"name": name, "phone": phone, "email": email, "group": group}  #Вставка оновленого елемента у відповідне місце (зберігаємо сортування)  insertPosition = 0  for item in students:  if name > item["name"]:  insertPosition += 1  else:  break  students.insert(insertPosition, newItem)  print(f"Student '{name}' has been updated.")  #Основна програма  def main():  #Завантаження даних із файлу, якщо ім'я файлу передано у командному рядку  if len(sys.argv) > 1:  file\_name = sys.argv[1]  load\_from\_csv(file\_name)  else:  print("No input file specified. Starting with an empty phonebook.")  while True:  choice = input("Specify action [C create, U update, D delete, P print, X exit]: ")  match choice.lower():  case "c":  print("Adding a new student...")  addNewElement()  case "u":  print("Updating an existing student...")  updateElement()  case "d":  print("Deleting a student...")  deleteElement()  case "p":  print("Student list:")  printAllList()  case "x":  if len(sys.argv) > 1:  save\_to\_csv(sys.argv[1])  else:  save\_to\_csv("students.csv")  print("Exiting the program. Goodbye!")  break  case \_:  print("Invalid choice. Please try again.")  if \_\_name\_\_ == "\_\_main\_\_":  main()  # test\_lab\_02.py  import unittest  from lab\_02 import students, addNewElement, deleteElement, updateElement  class TestStudentDirectory(unittest.TestCase):  def setUp(self):  global students  students.clear()  students.extend([  {"name": "Bob", "phone": "0631234567", "email": "bob@example.com", "group": "CS231"},  {"name": "Emma", "phone": "0637654321", "email": "emma@example.com", "group": "CS231"},  ])  def test\_add\_new\_element(self):  input\_data = ["Alice", "0639999999", "alice@example.com", "CS232"]  with unittest.mock.patch('builtins.input', side\_effect=input\_data):  addNewElement()  self.assertEqual(len(students), 3)  self.assertEqual(students[0]['name'], "Alice")  def test\_delete\_element(self):  input\_data = ["Bob"]  with unittest.mock.patch('builtins.input', side\_effect=input\_data):  deleteElement()  self.assertEqual(len(students), 1)  self.assertEqual(students[0]['name'], "Emma")  def test\_update\_element(self):  input\_data = ["Emma", "", "emma\_new@example.com", ""]  with unittest.mock.patch('builtins.input', side\_effect=input\_data):  updateElement()  self.assertEqual(students[1]['email'], "emma\_new@example.com")  if \_\_name\_\_ == "\_\_main\_\_":  unittest.main() |

**Висновок:** Під час виконання роботи використовуючи теоретичне підґрунтя про роботу з файлами та тестування коду у мові Python розширено програму телефонного довідника студентів додавши вказаний функціонал.