Технології програмування

Лабораторна робота №2

**Робота з файлами. Юніт тести**

**Новик Максим Вікторович, КБ-231**

**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. Розширити функціональність телефонного довідника студентів групи, що був розроблений у Лабораторній роботі №1 наступними можливостями:

1. **завантаження** початкових данні для довідника з csv файлу. Приклад csv файлу (lab2.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**.

Перед виконанням роботи слід ознайомитись з прикладами.

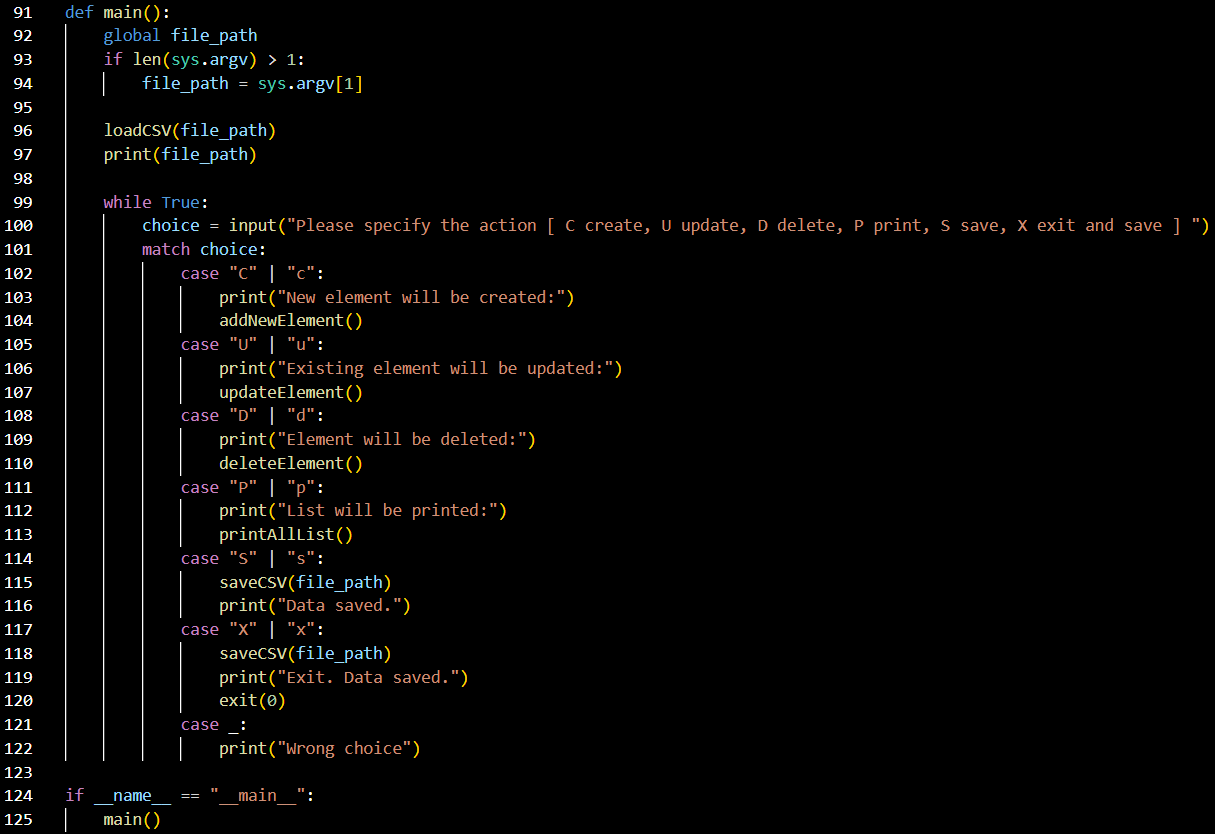
Текст програми разом зі звітом розмістити в директорії lab\_02. Директорію lab\_02 розмістити в директорії, що використовується для виконання практичних завдань по кожній лекції та має назву **TP-KB-23[1 or 2]-Name-Surname**.

**Хід виконання завдання:**

Перш за все було додано функції loadCSV та saveCSV, які забезпечують завантаження даних із CSV-файлу та їхнє збереження у нього. Дані автоматично завантажуються при запуску програми, а також зберігаються при виході з неї або за запитом користувача. Дані зберігаються у змінній data\_list, яка синхронізується з CSV-файлом.



Подивляючись на lab\_01, можна відмітити ще розширення функціональності управління даними в вигляді доданої команду S для ручного збереження даних у CSV-файл і зміненої команди X, для зберігання дані у файл перед виходом із програми.



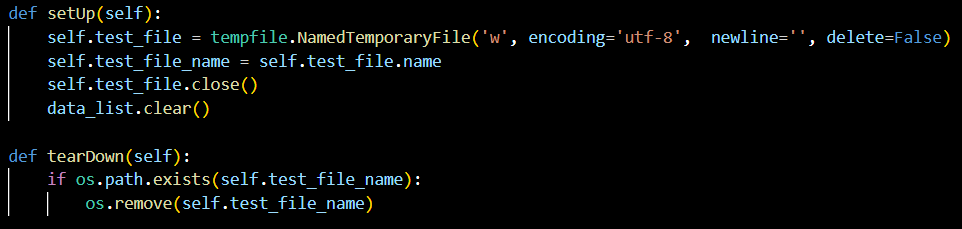
Текст програми:

|  |
| --- |
| import csv  import sys  import os  file\_path = os.path.join(os.path.dirname(\_\_file\_\_), "log.csv")  data\_list = []  def loadCSV(file\_path):      global data\_list      try:          with open(file\_path, "r") as file:              reader = csv.DictReader(file)              for row in reader:                  data\_list.append({                      "name":row.get("name", "Unknown"),                      "phone":row.get("phone", "Unknown"),                      "age":row.get("age", "Unknown"),                      "email":row.get("email", "Unknown")                      })          print(f"Data loaded.")      except FileNotFoundError:          print(f"File not found. Starting with an empty list.")  def saveCSV(file\_path):      with open(file\_path, "w", newline="") as file:          fieldnames = ["name", "phone", "age", "email"]          writer = csv.DictWriter(file, fieldnames=fieldnames)          writer.writeheader()          writer.writerows(data\_list)  def printAllList():      for elem in data\_list:          print("Student name is " + elem["name"] + ",  Phone is " + elem["phone"] + ", Age is " + elem["age"] + ", Email is " + elem["email"])  def addNewElement():      name = input("Please enter student name: ")      phone = input("Please enter student phone: ")      age = input("Please enter student age: ")      email = input("Please enter student email: ")      newItem = {"name": name, "phone": phone, "age": age, "email": email}      insertPosition = 0      for item in data\_list:          if name > item["name"]:              insertPosition += 1          else:              break      data\_list.insert(insertPosition, newItem)      print("New element has been added")  def deleteElement():      name = input("Please enter name to be deleted: ")      deletePosition = -1      for item in data\_list:          if name == item["name"]:              deletePosition = data\_list.index(item)              break      if deletePosition == -1:          print("Element was not found")      else:          del data\_list[deletePosition]          print(f"Element with name {name} has been deleted.")  def updateElement():      name = input("Please enter name to be updated: ")      insertPosition = -1      for item in data\_list:          if name == item["name"]:              insertPosition = data\_list.index(item)              break      if insertPosition == -1:          print("Name not found")      else:          old\_element = data\_list[insertPosition]          print(f"Student name is {old\_element['name']}, Phone is {old\_element['phone']}, Age is {old\_element['age']}, Email is {old\_element['email']}")          newname = input("Please enter student new name or skip: ") or old\_element['name']          newphone = input("Please enter student new phone or skip: ") or old\_element['phone']          newage = input("Please enter student new age or skip: ") or old\_element['age']          newemail = input("Please enter student new email or skip: ") or old\_element['email']          updateElement = {"name": newname, "phone": newphone, "age": newage, "email": newemail}          del data\_list[insertPosition]          updatePosition = 0          for item in data\_list:              if newname > item["name"]:                  updatePosition += 1          data\_list.insert(updatePosition, updateElement)          print("Element has been updated.")  def main():      global file\_path      if len(sys.argv) > 1:          file\_path = sys.argv[1]      loadCSV(file\_path)      print(file\_path)      while True:          choice = input("Please specify the action [ C create, U update, D delete, P print, S save, X exit and save ] ")          match choice:              case "C" | "c":                  print("New element will be created:")                  addNewElement()              case "U" | "u":                  print("Existing element will be updated:")                  updateElement()              case "D" | "d":                  print("Element will be deleted:")                  deleteElement()              case "P" | "p":                  print("List will be printed:")                  printAllList()              case "S" | "s":                  saveCSV(file\_path)                  print("Data saved.")              case "X" | "x":                  saveCSV(file\_path)                  print("Exit. Data saved.")                  exit(0)              case \_:                  print("Wrong choice")  if \_\_name\_\_ == "\_\_main\_\_":      main() |

**UNIT-тести**

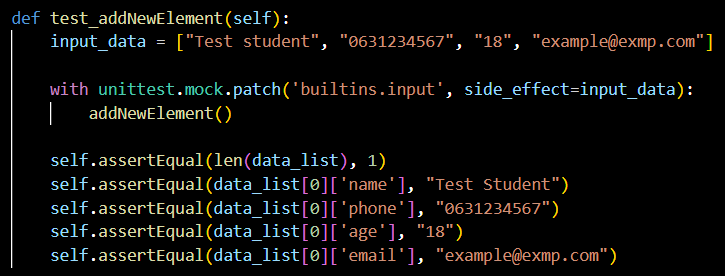
Ініціалізація тестів (setUp і tearDown):

* Створюється тимчасовий файл за допомогою tempfile.NamedTemporaryFile, в якому будуть проводитися тести для функцій імпорту/експорту даних.
* Очищається список data\_list перед кожним тестом, що гарантує чистий стан перед кожним виконанням тестів.
* Після виконання тестів перевіряється наявність тимчасового файлу, і якщо він існує, файл видаляється, щоб не залишити зайвих файлів після завершення тестів.



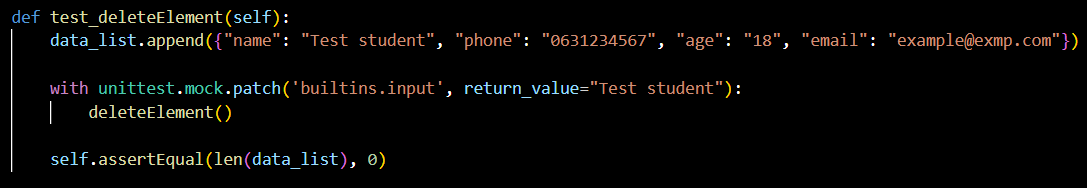
Тестування функції addNewElement():

* Використовується мок для введення даних через input() за допомогою unittest.mock.patch, що дозволяє тестувати без реального вводу користувача.
* Перевіряється додавання одного елемента в data\_list і коректність значень полів елемента (ім'я, телефон, вік, email).



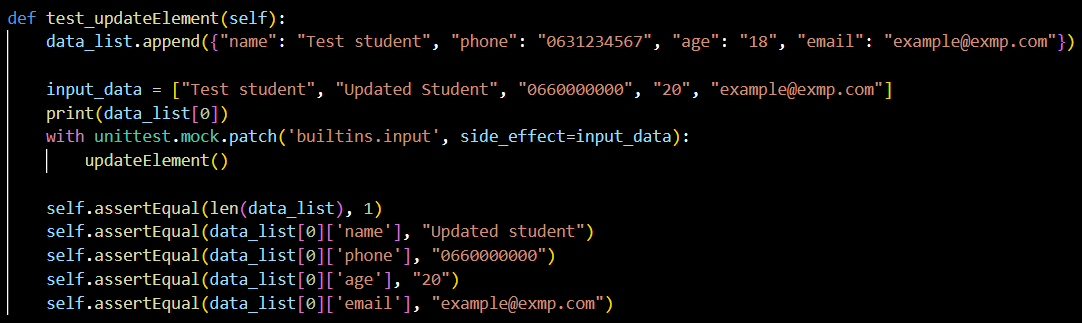
Тестування функції deleteElement():

* Перед тестом до списку додається один елемент.
* Мок для input() дозволяє вказати ім'я студента, якого потрібно видалити.
* Перевіряється список після видалення елемента стає порожнім.



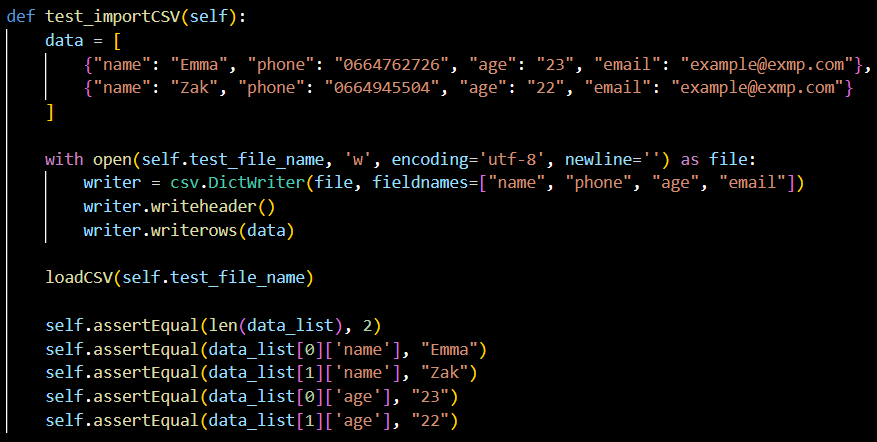
Тестування функції updateElement():

* Додається елемент до списку, після чого через мок змінюються дані цього елемента.
* Перевіряється оновлені значення в списку після виконання функції.



Тестування функції loadCSV():

* Створюється тестовий файл, який містить кілька записів студентів.
* Завантаження даних здійснюється через функцію loadCSV(), і перевіряється кількість елементів у списку після завантаження й коректність даних.



Тестування функції saveCSV():

* До списку додається новий елемент, після чого цей список зберігається за допомогою функції saveCSV().
* Перевіряється коректність записаних даних у файл і вірність значень полів.



Текст програми:

|  |
| --- |
| import unittest  import unittest.mock  import os  import tempfile  import csv  from .lab\_02 import data\_list, loadCSV, saveCSV, addNewElement, deleteElement, updateElement  class TestStudenList(unittest.TestCase):      def setUp(self):          self.test\_file = tempfile.NamedTemporaryFile('w', encoding='utf-8',  newline='', delete=False)          self.test\_file\_name = self.test\_file.name          self.test\_file.close()          data\_list.clear()      def tearDown(self):          if os.path.exists(self.test\_file\_name):              os.remove(self.test\_file\_name)      def test\_addNewElement(self):          input\_data = ["Test student", "0631234567", "18", "example@exmp.com"]          with unittest.mock.patch('builtins.input', side\_effect=input\_data):              addNewElement()          self.assertEqual(len(data\_list), 1)          self.assertEqual(data\_list[0]['name'], "Test Student")          self.assertEqual(data\_list[0]['phone'], "0631234567")          self.assertEqual(data\_list[0]['age'], "18")          self.assertEqual(data\_list[0]['email'], "example@exmp.com")      def test\_deleteElement(self):          data\_list.append({"name": "Test student", "phone": "0631234567", "age": "18", "email": "example@exmp.com"})          with unittest.mock.patch('builtins.input', return\_value="Test student"):              deleteElement()          self.assertEqual(len(data\_list), 0)      def test\_updateElement(self):          data\_list.append({"name": "Test student", "phone": "0631234567", "age": "18", "email": "example@exmp.com"})          input\_data = ["Test student", "Updated Student", "0660000000", "20", "example@exmp.com"]          print(data\_list[0])          with unittest.mock.patch('builtins.input', side\_effect=input\_data):              updateElement()            self.assertEqual(len(data\_list), 1)          self.assertEqual(data\_list[0]['name'], "Updated student")          self.assertEqual(data\_list[0]['phone'], "0660000000")          self.assertEqual(data\_list[0]['age'], "20")          self.assertEqual(data\_list[0]['email'], "example@exmp.com")      def test\_importCSV(self):          data = [              {"name": "Emma", "phone": "0664762726", "age": "23", "email": "example@exmp.com"},              {"name": "Zak", "phone": "0664945504", "age": "22", "email": "example@exmp.com"}          ]          with open(self.test\_file\_name, 'w', encoding='utf-8', newline='') as file:              writer = csv.DictWriter(file, fieldnames=["name", "phone", "age", "email"])              writer.writeheader()              writer.writerows(data)          loadCSV(self.test\_file\_name)            self.assertEqual(len(data\_list), 2)          self.assertEqual(data\_list[0]['name'], "Emma")          self.assertEqual(data\_list[1]['name'], "Zak")          self.assertEqual(data\_list[0]['age'], "23")          self.assertEqual(data\_list[1]['age'], "22")      def test\_saveCSV(self):          data\_list.append({"name": "Jon", "phone": "0667283939", "age": "26", "email": "example@exmp.com"})          saveCSV(self.test\_file\_name)          with open(self.test\_file\_name, 'r', encoding='utf-8') as file:              reader = csv.DictReader(file)              rows = list(reader)              self.assertEqual(len(rows), 1)              self.assertEqual(rows[0]["name"], "Chris")              self.assertEqual(rows[0]['phone'], "0667283939")              self.assertEqual(rows[0]['age'], "26")              self.assertEqual(rows[0]['email'], "example@exmp.com")    if \_\_name\_\_ == '\_\_main\_\_':      unittest.main() |

**Висновок:**

Під час виконання лабораторної роботи я використовував теоретичне підґрунтя про роботу з файлами та тестування коду у мові Python розширивши програму телефонного довідника студентів додавши функціонал, що був вказаний в завданні до лабораторної роботи.

**Посилання на github:**

https://github.com/GAGGAGX/TP-KB-231-Novyk-Maksym

**Знімок екрану з посилання на github:**

