**Лабораторна робота №2**

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

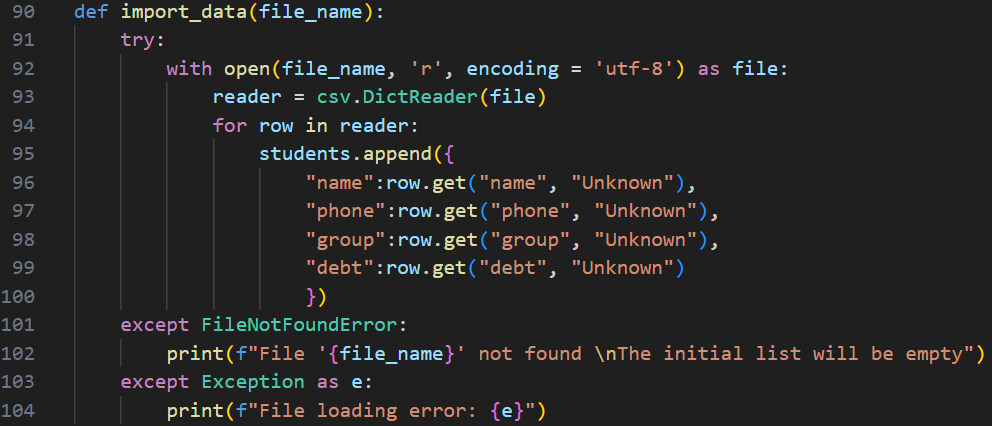
**Мета**: Використовуючи теоретичне підґрунтя про роботу з файлами та тестування коду у мові Python розширити програму телефонного довідника студентів додавши функціонал, що буде вказано в завданні до лабораторної роботи.

**Завдання до лабораторної роботи**

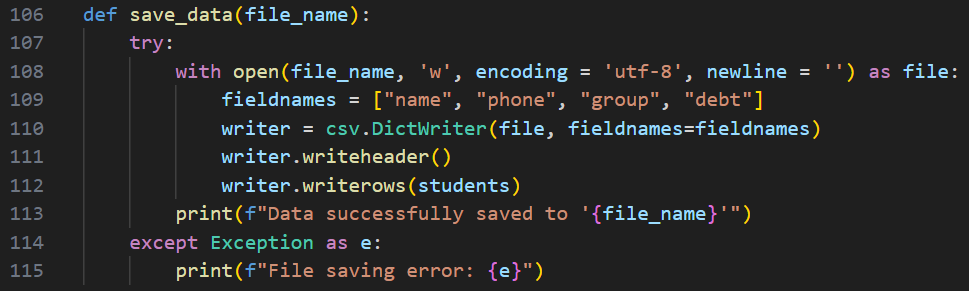
1. **завантаження** початкових данних для довідника з csv файлу. Приклад csv файлу (lab2.csv), та оригінальній файл формату xlsx розміщені поряд з вказівками. Ім’я файлу для завантаження даних повинно передаватись в якості параметрів командного рядку.
2. **зберігання** довідника студентів у форматі CSV перед виходом із програми.
3. покриття основного функціоналу програми **Юніт тестами**.

**Хід роботи**

Для початку я створив функцію завантаження потакових даних, яка приймає назву файла, після чого відкриває його, зчитує та записує дані з рядків у список студентів, якщо ж поля нема, то записується Unknown. У випадку відсутності файла з заданою назвою, список залишиться пустим:

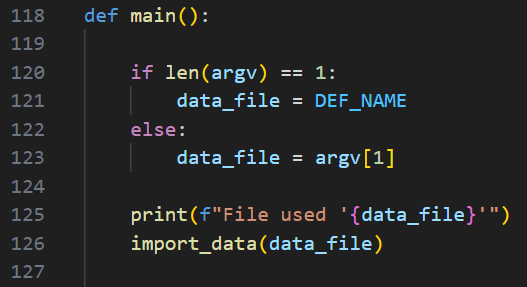


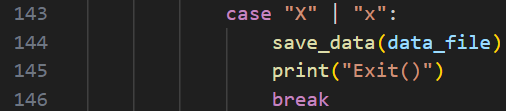
Далі створив функцію зберігання даних, вона так само приймає назву файла, відкриває його, вказує назви полів, це буде 1 рядок, його значення таке сайме, як і у ключів в словниках, після чого записує весь список:



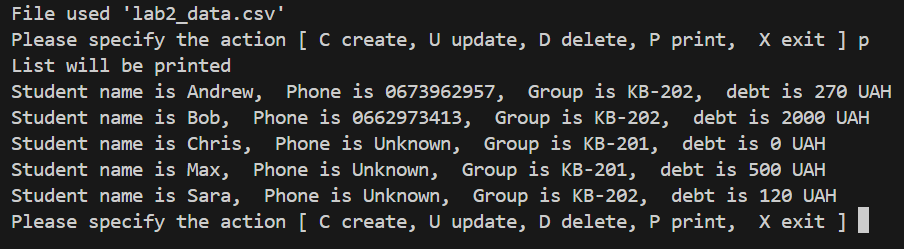
Далі треба ці функції використати, для цього на початку коду прописую константу зі стандартою назвою файлу, після чого до циклу в функції main() зчитую параметри командного рядка, якщо довжина 1, то є лише назва самого файлу, отже для файлу даних використовується стандартна назва, а якщо ні, то було введено другий параметр, що означає, що було введено назву файла даних, в такому випадку використовується введена назва. Після вибору виходу в циклі спочатку зберігаються дані, а потім завершується цикл:



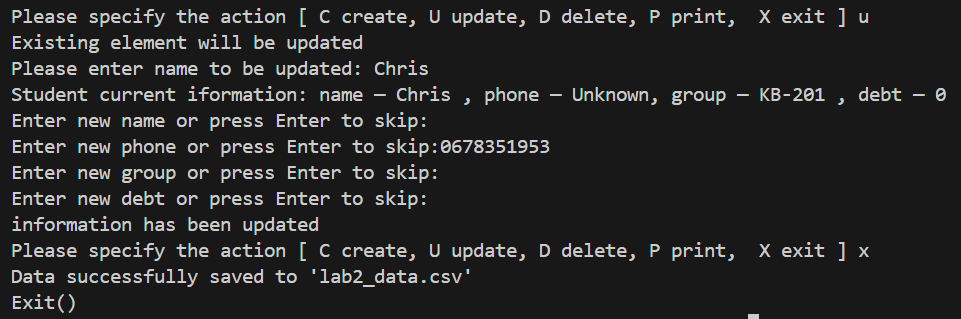


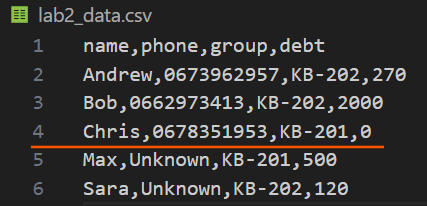


Результат запуску програми:



Результат завершення програми зі зміною 1 елемента:

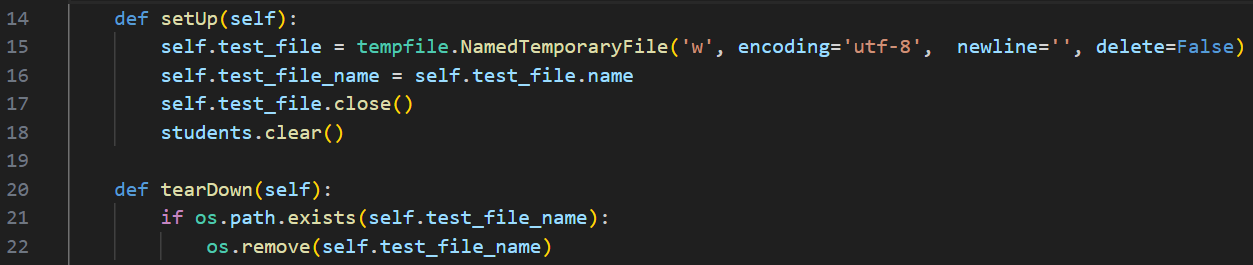




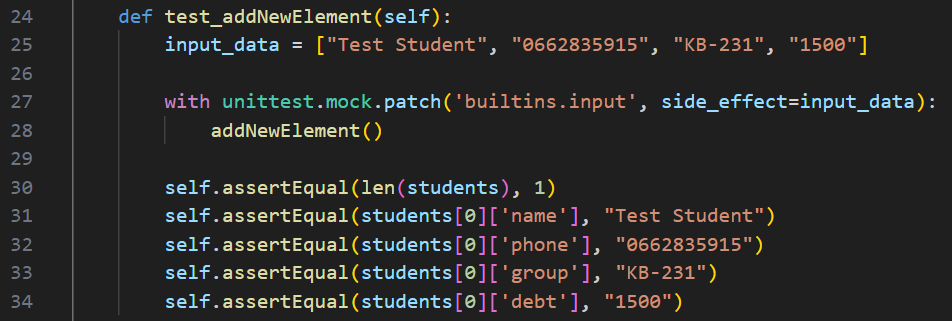
Текст програми:

|  |
| --- |
| import csv  from sys import argv  DEF\_NAME = "lab2\_data.csv"  students = []  def printAllList():      if not students:          print("The list of students is empty")          return      for elem in students:          strForPrint = "Student name is " + elem["name"] + ",  Phone is " + elem["phone"] + ",  Group is " + elem["group"] + ",  debt is " + elem["debt"] + " UAH"          print(strForPrint)      return  def addNewElement():      name = input("Pease enter student name: ") or "Unknown"      phone = input("Please enter student phone: ") or "Unknown"      group = input("Pease enter student group: ") or "Unknown"      debt = input("Please enter student debt (in UAH): ") or "Unknown"      newItem = {"name": name, "phone": phone, "group": group, "debt": debt}      insertPosition = 0      for item in students:          if name > item["name"]:              insertPosition += 1          else:              break      students.insert(insertPosition, newItem)      print("New element has been added")      return  def deleteElement():      name = input("Please enter name to be delated: ")      deletePosition = -1      for item in students:          if name == item["name"]:              deletePosition = students.index(item)              break      if deletePosition == -1:          print("Element was not found")      else:          print("Delete position " + str(deletePosition))          del students[deletePosition]      return  def updateElement():      name = input("Please enter name to be updated: ")      updatePosition = -1      for item in students:          if name == item["name"]:              updatePosition = students.index(item)              break      if updatePosition == -1:          print("Student not found")      else:          name1 = students[updatePosition]["name"]          phone1 = students[updatePosition]["phone"]          group1 = students[updatePosition]["group"]          debt1 = students[updatePosition]["debt"]          curinf = "Student current iformation: name — " + name1 + " , phone — " + phone1 + ", group — " + group1 + " , debt — " + debt1          print(curinf)          name = input("Enter new name or press Enter to skip:") or name1          phone = input("Enter new phone or press Enter to skip:") or phone1          group = input("Enter new group or press Enter to skip:") or group1          debt = input("Enter new debt or press Enter to skip:") or debt1            if name == name1 and phone == phone1 and group == group1 and debt == debt1:              print("you haven`t updated student information")          elif name == name1:              students[updatePosition]["phone"] = phone              students[updatePosition]["group"] = group              students[updatePosition]["debt"] = debt          else:              updatedItem = {"name": name, "phone": phone, "group": group, "debt": debt}              del students[updatePosition]              insertPosition = 0              for item in students:                  if name > item["name"]:                      insertPosition += 1              students.insert(insertPosition, updatedItem)          print("information has been updated")      return  def import\_data(file\_name):      try:          with open(file\_name, 'r', encoding = 'utf-8') as file:              reader = csv.DictReader(file)              for row in reader:                  students.append({                      "name":row.get("name", "Unknown"),                      "phone":row.get("phone", "Unknown"),                      "group":row.get("group", "Unknown"),                      "debt":row.get("debt", "Unknown")                      })      except FileNotFoundError:          print(f"File '{file\_name}' not found \nThe initial list will be empty")      except Exception as e:          print(f"File loading error: {e}")  def save\_data(file\_name):      try:          with open(file\_name, 'w', encoding = 'utf-8', newline = '') as file:              fieldnames = ["name", "phone", "group", "debt"]              writer = csv.DictWriter(file, fieldnames=fieldnames)              writer.writeheader()              writer.writerows(students)          print(f"Data successfully saved to '{file\_name}'")      except Exception as e:          print(f"File saving error: {e}")  def main():      if len(argv) == 1:          data\_file = DEF\_NAME      else:          data\_file = argv[1]        print(f"File used '{data\_file}'")      import\_data(data\_file)      while True:          chouse = input("Please specify the action [ C create, U update, D delete, P print,  X exit ] ")          match chouse:              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 "X" | "x":                  save\_data(data\_file)                  print("Exit()")                  break              case \_:                  print("Wrong chouse")  if \_\_name\_\_ == "\_\_main\_\_":      main() |

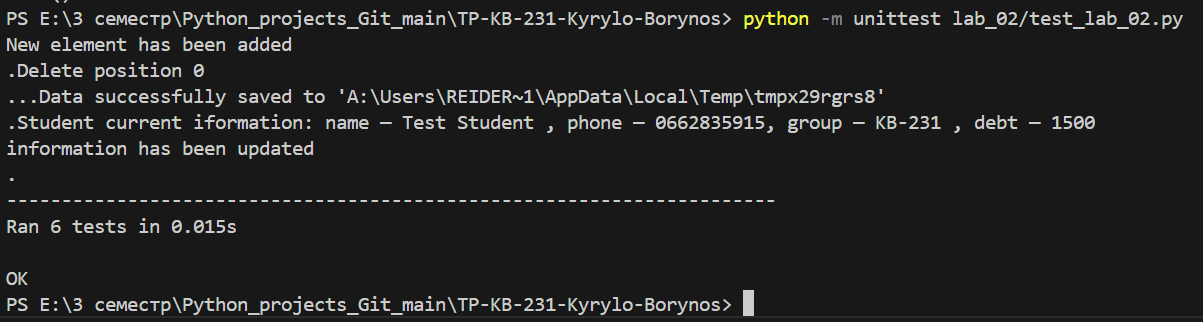
Останнім було покриття функціоналу юніт-тестами, функції setUp() та tearDown() виконуються перед та після кожного тесту відповідно, перша створює тимчасовий файл, записує його назву та очищує список студентів, а друга видаляє цей файл, якщо він є:



На прикладі тесту додавання елемента поясню, як це працює: спочатку створюється список з даними які треба вводити користувачу, а потім імітується ввденення цих даних при виклику потрібної функції, після цього перевіряється довжина списку та значення першого елементу:



Результат виконання тестів:



Текст програми:

|  |
| --- |
| import unittest  import unittest.mock  from io import StringIO  import sys  import os  import tempfile  import csv  from .lab\_02 import students, addNewElement, deleteElement, updateElement, import\_data, save\_data, printAllList  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()          students.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", "0662835915", "KB-231", "1500"]          with unittest.mock.patch('builtins.input', side\_effect=input\_data):              addNewElement()          self.assertEqual(len(students), 1)          self.assertEqual(students[0]['name'], "Test Student")          self.assertEqual(students[0]['phone'], "0662835915")          self.assertEqual(students[0]['group'], "KB-231")          self.assertEqual(students[0]['debt'], "1500")      def test\_deleteElement(self):          students.append({"name": "Test Student", "phone": "0662835915", "group": "KB-231", "debt": "1500"})          with unittest.mock.patch('builtins.input', return\_value="Test Student"):              deleteElement()          self.assertEqual(len(students), 0)      def test\_updateElement(self):          students.append({"name": "Test Student", "phone": "0662835915", "group": "KB-231", "debt": "1500"})          input\_data = ["Test Student", "Updated Student", "0669999999", "KB-232", "800"]          with unittest.mock.patch('builtins.input', side\_effect=input\_data):              updateElement()          self.assertEqual(len(students), 1)          self.assertEqual(students[0]['name'], "Updated Student")          self.assertEqual(students[0]['phone'], "0669999999")          self.assertEqual(students[0]['group'], "KB-232")          self.assertEqual(students[0]['debt'], "800")      def test\_import\_data(self):          data = [              {"name": "Alice", "phone": "0664682546", "group": "KB-231", "debt": "300"},              {"name": "Bob", "phone": "0666483595", "group": "KB-232", "debt": "700"}          ]          with open(self.test\_file\_name, 'w', encoding='utf-8', newline='') as file:              writer = csv.DictWriter(file, fieldnames=["name", "phone", "group", "debt"])              writer.writeheader()              writer.writerows(data)          import\_data(self.test\_file\_name)            self.assertEqual(len(students), 2)          self.assertEqual(students[0]['name'], "Alice")          self.assertEqual(students[1]['name'], "Bob")          self.assertEqual(students[0]['group'], "KB-231")          self.assertEqual(students[1]['group'], "KB-232")      def test\_save\_data(self):          students.append({"name": "Chris", "phone": "0667777777", "group": "KB-232", "debt": "500"})          save\_data(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'], "0667777777")              self.assertEqual(rows[0]['group'], "KB-232")              self.assertEqual(rows[0]['debt'], "500")      def test\_printAllList(self):          students.append({"name": "Test Student", "phone": "0667777777", "group": "KB-232", "debt": "100"})          captured\_output = StringIO()          sys.stdout = captured\_output          printAllList()          sys.stdout = sys.\_\_stdout\_\_          self.assertIn("Test Student", captured\_output.getvalue())  if \_\_name\_\_ == '\_\_main\_\_':      unittest.main() |

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