Технології програмування

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.

**Завдання до лабораторної роботи:**

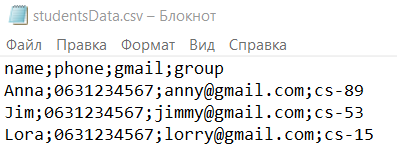
Розширити функціональність телефонного довідника студентів групи, що був розроблений у Лабораторній роботі №1 наступними можливостями:

1. Завантаження початкових даних для довідника з csv файлу. Ім’я файлу для завантаження даних повинно передаватись в якості параметрів командного рядку.
2. Зберігання довідника студентів у форматі cvs перед виходом із програми.
3. Покриття основного функціоналу програми Юніт тестами.

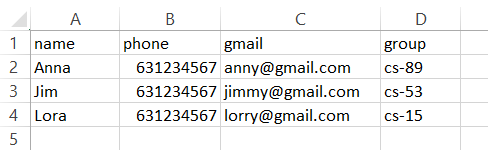
**Хід роботи:**

*Завдання 1. Завантаження початкових даних для довідника з csv файлу.*

1. В директорії lab\_02 створила файл studentsData.csv. Відкрила файл у Блокноті та записала перший рядок який містить назви полів у нашому довіднику студентів. Також файл заповнила новими даними про трьох студентів. Кожне слово відокремлюється символом “;”.



1. Зберегли файл. Клацнувши на файл двічі, файл відкривається у Excel.



1. Підключили дві бібліотеки csv і sys. Модуль csv дозволяє зчитувати та записувати дані в CSV файли. Модуль sys використовується для роботи з параметрами командного рядка, вихідними та вхідними потоками.
2. До основного коду телефонного довідника студентів з Лабораторної роботи №1 додали функцію loadDataCSV, що зчитує данні з studentsData.csv файлу і вставляє данні з файлу в лист студентів.

Код:

|  |
| --- |
| def loadDataCSV(filename):  try:  with open(filename, 'r', newline='') as csvfile:  reader = csv.DictReader(csvfile, delimiter=';')  for row in reader:  print(row)  newItem = {  "name": row["name"],  "phone": row["phone"],  "gmail": row["gmail"],  "group": row["group"]  }  studentList.append(newItem)  studentList.sort(key=lambda x: x['name'])  print(f"The data from the {filename} was loaded successfully.")  except Exception as e:  print(f"Error loading data to {filename}: {e}") |

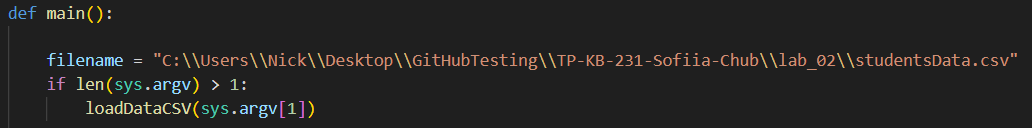
1. Принцип роботи:

Функція відкриває csv файл за допомогою конструкції with open(filename, 'r', newline='').

* filename: ім'я файлу, який передається функції як аргумент.
* 'r': режим відкриття файлу для читання.
* newline='': параметр, що визначає, як обробляються нові рядки в файлі (це для коректної обробки переносу рядків у CSV).

Для читання файлу використовується csv.DictReader, що дозволяє читати кожен рядок csv файлу як словник (dictionary). delimiter=';': вказує, що стовпці в csv розділені символом “;”. Функція перебирає кожен рядок у CSV файлі за допомогою циклу for row in reader. Кожен рядок є словником, де ключі – це назви стовпців, а значення – відповідні значення з рядка. Для кожного рядка створюється новий словник newItem, який містить ключі: name, phone, gmail, group (ці значення беруться з відповідних стовпців у csv файлі). Словник newItem додається до списку studentList. Після обробки всіх рядків csv файлу, список studentList сортується за ім’ям студента в алфавітному порядку за допомогою методу sort. Після успішного запису даних виводиться повідомлення про успіх: " The data from the {filename} was loaded successfully.". Якщо під час роботи виникає помилка, функція виводить повідомлення з деталями помилки через блок except.

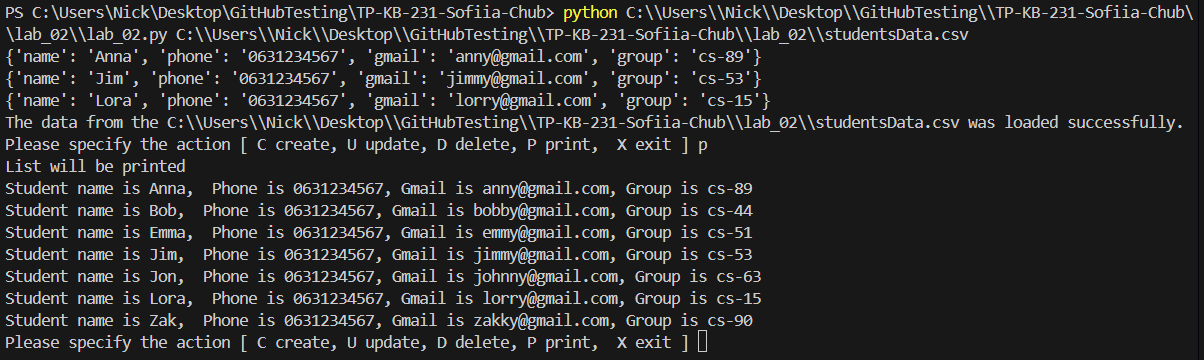
Щоб записати данні в функцію main() додали таку умову:



Визначається шлях до cvs файлу в змінній filename. Потім вона перевіряє, чи передано додаткові аргументи через командний рядок, використовуючи sys.argv. Якщо аргумент командного рядка надано, то викликається функція loadDataCSV(). Якщо ж аргумент не передано, функція не викликається і завантаження даних з файлу csv не відбувається.

1. Результат роботи:

З командного рядка запускаємо файл даючи аргумент у вигляді імені csv файлу.



*Завдання 2. Зберігання довідника студентів у форматі cvs.*

1. До основного коду додали функцію saveDataCSV, що зберігає данні з листа студентів і вставляє данні в файл studentsData.csv.

Код:

|  |
| --- |
| def saveDataCSV(filename):  try:  with open(filename, 'w', newline='', encoding='utf-8') as csvfile:  fieldnames = ['name', 'phone', 'gmail', 'group']  writer = csv.DictWriter(csvfile, fieldnames=fieldnames, delimiter=';')  writer.writeheader()  writer.writerows(studentList)  print(f"Data has been successfully saved to {filename}.")  except Exception as e:  print(f"Error saving data to {filename}: {e}") |

1. Принцип роботи:

Функція відкриває CSV файл для запису за допомогою конструкції with open(filename, 'w', newline='', encoding='utf-8').

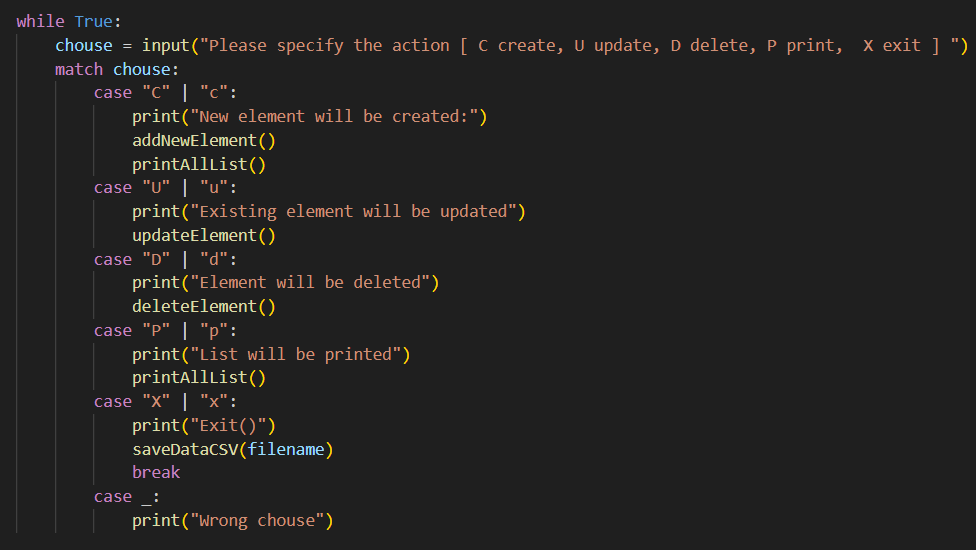
filename: ім'я файлу, в який будуть збережені дані, передається функції як аргумент.

* 'w': режим відкриття файлу для запису. Якщо файл не існує, він буде створений; якщо існує — перезаписаний.
* newline='': параметр, що визначає, як обробляються нові рядки при записі в файл, щоб уникнути додаткових порожніх рядків.
* encoding='utf-8': параметр, який визначає кодування файлу для коректного збереження тексту, зокрема для підтримки символів з різних мов.

Далі визначаються заголовки стовпців через список fieldnames = ['name', 'phone', 'gmail', 'group'], що відповідає ключам словників, які будуть записані у файл. Для запису даних використовується об'єкт csv.DictWriter, що дозволяє записувати словники в csv файл, де ключі словників відповідають назвам стовпців. Параметр delimiter=';' вказує на використання крапки з комою як роздільника стовпців. Спочатку записуються заголовки стовпців за допомогою методу writeheader(). Потім за допомогою методу writerows(studentList) записуються всі елементи зі списку studentList, де кожен елемент це словник, що містить дані для одного студента.

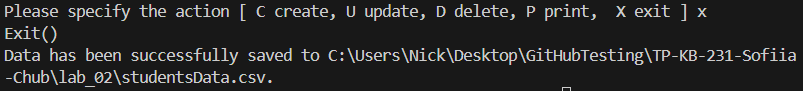
Після успішного запису даних виводиться повідомлення про успіх: "Data has been successfully saved to {filename}.". Якщо під час роботи виникає помилка, функція виводить повідомлення з деталями помилки через блок except.

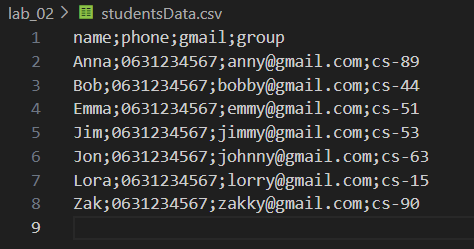
Щоб зберегти данні в кейс “E | e” функції main() додали виклик функції saveDateCSV():



1. Результат роботи:

Вибираємо опцію Exit.





*Завдання 3. Юніт тести.*

1. Розробили тести для трьох функцій: додавання, зміни та видалення даних студента. У юніт тесті перевіряється коректність роботи програми.

Код:

|  |
| --- |
| import pytest  from lab\_02 import studentList, addNewElement, deleteElement, updateElement    def test\_add\_new\_element():  initial\_len = len(studentList)  addNewElement()  assert len(studentList) == initial\_len + 1  assert any(student["name"] == "Sofi" for student in studentList)    def test\_update\_element():  updateElement()  updated\_student = next(student for student in studentList if student["name"] == "Sofi")  assert updated\_student["group"] == "cs-45"    def test\_delete\_element():  initial\_len = len(studentList)  deleteElement()  assert len(studentList) == initial\_len - 1  assert not any(student["name"] == "Sofi" for student in studentList) |

1. Принцип роботи:

Спочатку імпортуються необхідні модулі:

* import pytest: імпортується бібліотека pytest, яка використовується для тестування коду.
* from lab\_02 import studentList, addNewElement, deleteElement, updateElement: імпортуються функції та змінні з модуля lab\_02.

Тест на додавання нового елемента (test\_add\_new\_element):

Спочатку зберігається початкова довжина списку студентів в змінній initial\_len. Викликається функція addNewElement(), яка має додати нового студента. Перевіряється, що довжина списку studentList збільшилася на 1 за допомогою асертації assert len(studentList) == initial\_len + 1. Також перевіряється, чи містить список студента з іменем "Sofi" через асертацію assert any(student["name"] == "Sofi" for student in studentList).

Тест на оновлення елемента (test\_update\_element):

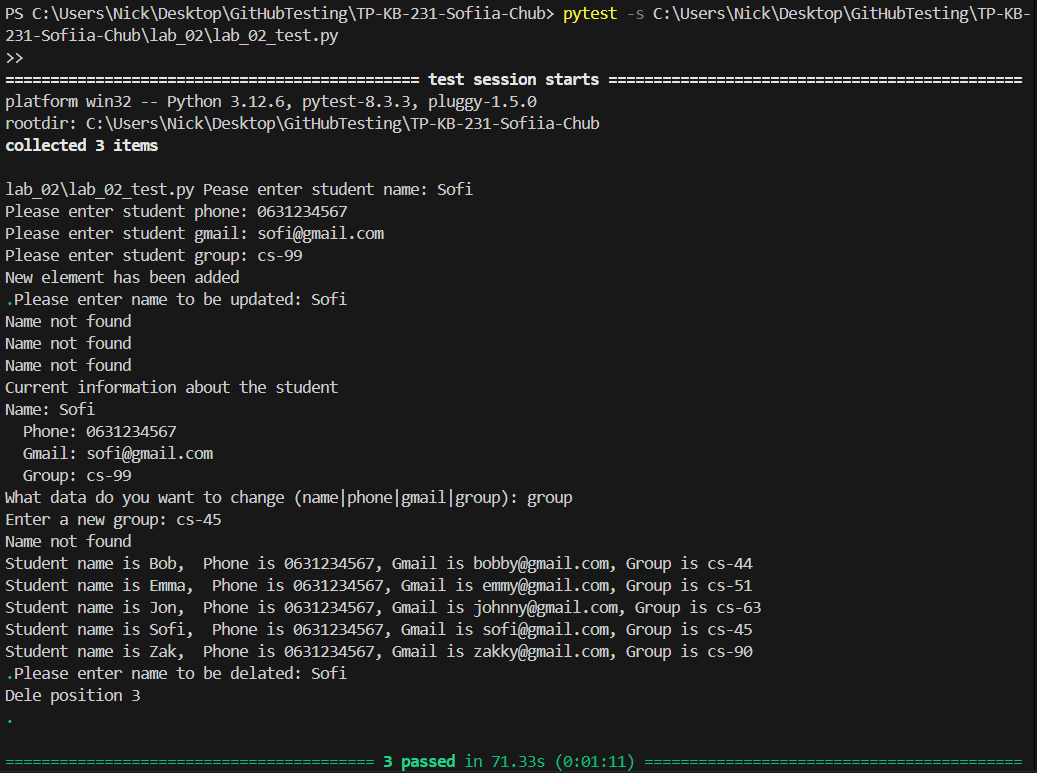
Викликається функція updateElement(), яка має оновити інформацію про студента. Знаходиться оновлений студент за допомогою генератора, який шукає студента з ім'ям "Sofi". Перевіряється, чи був оновлений його параметр "group" на значення "cs-45" через асертацію assert updated\_student["group"] == "cs-45".

Тест на видалення елемента (test\_delete\_element):

Спочатку зберігається початкова довжина списку студентів в змінній initial\_len. Викликається функція deleteElement(), яка має видалити студента зі списку. Перевіряється, що довжина списку зменшилася на 1 за допомогою асертації assert len(studentList) == initial\_len - 1. Також перевіряється, чи не містить список студента з ім'ям "Sofi" після видалення через асертацію assert not any(student["name"] == "Sofi" for student in studentList).

1. Результат роботи:

Програма запускається через командний рядок з ключем -s для вводу тестових даних. Ввели параметри згідно нашого коду.



**Повний код програми**

Код телефонного довідника студентів (lab\_02.py):

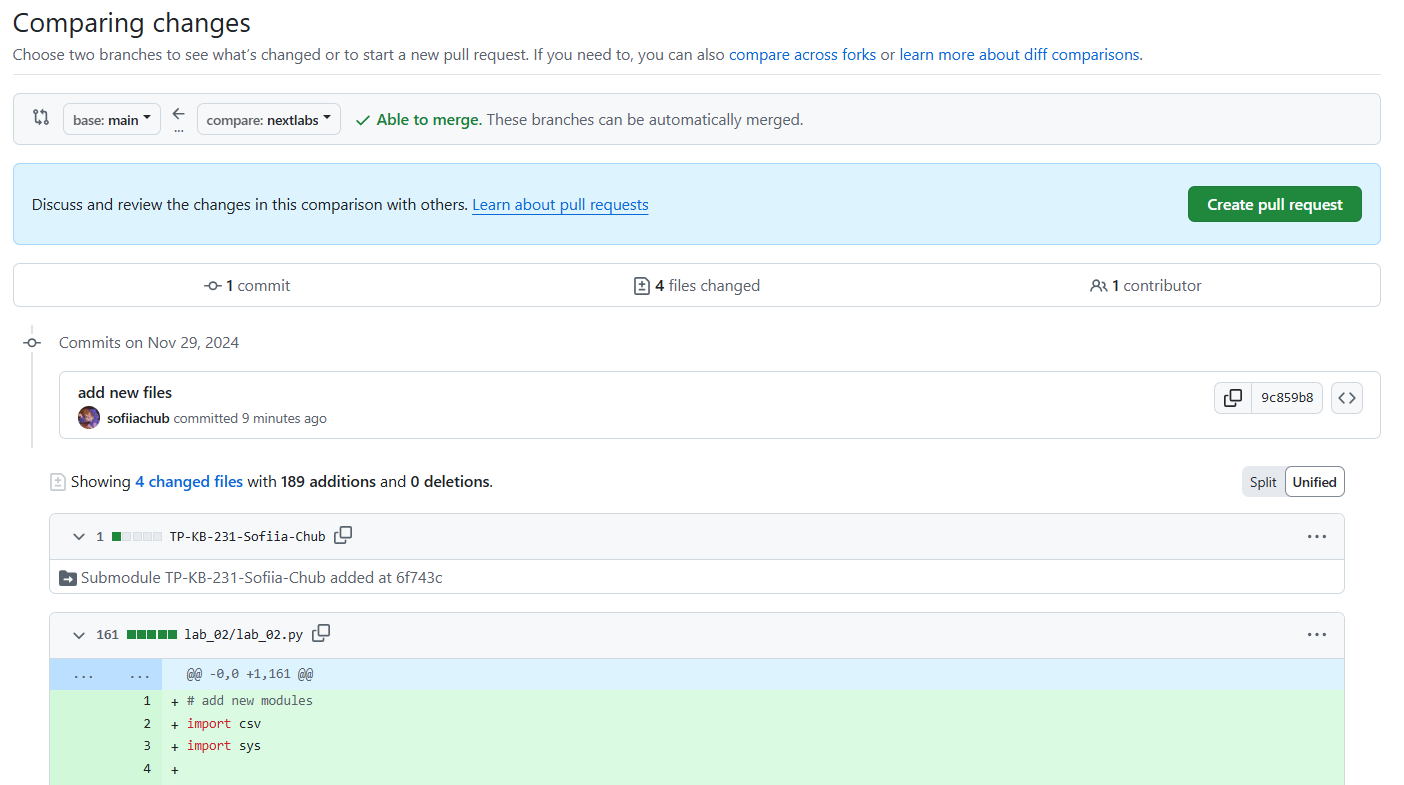
|  |
| --- |
| # add new modules  import csv  import sys    studentList = [  # added parameters "gmail" and "group" in list  {"name":"Bob", "phone":"0631234567", "gmail":"bobby@gmail.com", "group":"cs-44"},  {"name":"Emma", "phone":"0631234567", "gmail":"emmy@gmail.com", "group":"cs-51"},  {"name":"Jon", "phone":"0631234567", "gmail":"johnny@gmail.com", "group":"cs-63"},  {"name":"Zak", "phone":"0631234567", "gmail":"zakky@gmail.com", "group":"cs-90"}  ]    def loadDataCSV(filename):  try:  with open(filename, 'r', newline='') as csvfile:  reader = csv.DictReader(csvfile, delimiter=';')  for row in reader:  print(row)  newItem = {  "name": row["name"],  "phone": row["phone"],  "gmail": row["gmail"],  "group": row["group"]  }  studentList.append(newItem)  studentList.sort(key=lambda x: x['name'])  print(f"The data from the {filename} was loaded successfully.")  except Exception as e:  print(f"Error loading data to {filename}: {e}")    # saving date to a CSV file  def saveDataCSV(filename):  try:  with open(filename, 'w', newline='', encoding='utf-8') as csvfile:  fieldnames = ['name', 'phone', 'gmail', 'group']  writer = csv.DictWriter(csvfile, fieldnames=fieldnames, delimiter=';')  writer.writeheader()  writer.writerows(studentList)  print(f"Data has been successfully saved to {filename}.")  except Exception as e:  print(f"Error saving data to {filename}: {e}")    def printAllList():  for elem in studentList:  # added parameters "gmail" and "group" for print all list  strForPrint = "Student name is " + elem["name"] + ", Phone is " + elem["phone"] + ", Gmail is " + elem["gmail"] + ", Group is " + elem["group"]  print(strForPrint)  return    def addNewElement():  # added parameters "gmail" and "group" for add new element  name = input("Pease enter student name: ")  phone = input("Please enter student phone: ")  gmail = input("Please enter student gmail: ")  group = input("Please enter student group: ")  newItem = {"name": name, "phone": phone, "gmail": gmail, "group": group}    insertPosition = 0  for item in studentList:  if name > item["name"]:  insertPosition += 1  else:  break  studentList.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 studentList:  if name == item["name"]:  deletePosition = studentList.index(item)  break    if deletePosition == -1:  print("Element was not found")  else:  print("Dele position " + str(deletePosition))  del studentList[deletePosition]  return    def updateElement():  # the date change function is implemented  name = input("Please enter name to be updated: ")    studentNames = [elem["name"] for elem in studentList]    if name in studentNames:  for elem in studentList:  if elem["name"] == name:  print("Current information about the student")  strForPrint = "Name: " + elem["name"] + "\n Phone: " + elem["phone"] + "\n Gmail: " + elem["gmail"] + "\n Group: " + elem["group"]  print(strForPrint)    choice = input("What data do you want to change (name|phone|gmail|group): ")    match choice:  case "name":  newName = input("Enter a new name: ")    studentList.remove(elem)  newItem = {"name": newName, "phone": elem["phone"], "gmail": elem["gmail"], "group": elem["group"]}    insertPosition = 0  for item in studentList:  if newName > item["name"]:  insertPosition += 1  else:  break  studentList.insert(insertPosition, newItem)    case "phone":  newPhone = input("Enter a new phone: ")  elem["phone"] = newPhone  case "gmail":  newGmail = input("Enter a new gmail: ")  elem["gmail"] = newGmail  case "group":  newGroup = input("Enter a new group: ")  elem["group"] = newGroup  case \_:  print("You didn't choose from the proposed options")  break  else:  print("Name not found")    printAllList()  return    def main():    filename = "C:\\Users\\Nick\\Desktop\\GitHubTesting\\TP-KB-231-Sofiia-Chub\\lab\_02\\studentsData.csv"  if len(sys.argv) > 1:  loadDataCSV(sys.argv[1])    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()  printAllList()  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":  print("Exit()")  saveDataCSV(filename)  break  case \_:  print("Wrong chouse")    if \_\_name\_\_ == '\_\_main\_\_':  main() |

Код юніт тестів (lab\_02\_test.py):

|  |
| --- |
| import pytest  from lab\_02 import studentList, addNewElement, deleteElement, updateElement    def test\_add\_new\_element():  initial\_len = len(studentList)  addNewElement()  assert len(studentList) == initial\_len + 1  assert any(student["name"] == "Sofi" for student in studentList)    def test\_update\_element():  updateElement()  updated\_student = next(student for student in studentList if student["name"] == "Sofi")  assert updated\_student["group"] == "cs-45"    def test\_delete\_element():  initial\_len = len(studentList)  deleteElement()  assert len(studentList) == initial\_len - 1  assert not any(student["name"] == "Sofi" for student in studentList) |

Посилання на github: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub/compare/main...nextlabs?diff=unified&w=>

Знімок екрану з посилання на github:



**Висновки:** Використовуючи теоретичне підґрунтя про csv файли, способи зчитування та записування інформації, і розглянула як працюють параметри командного рядку. Додала дві нові функції завантаження та збереження даних з використанням csv файлу в програму. Також створила юніт тести і перевірила, що функції працюють правильно. Засвоїла вивчений матеріал та використала його на практиці.