**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:**

**Теоретичні відомості**

*Параметри командного рядка*

Одним із механізмом визначення параметрів необхідних для виконання програми – використання аргументів командного рядка. Для можливості використання аргументів командного необхідно підключити модуль argv

Розглянемо приклад:

|  |
| --- |
| from sys import argv  print(f"Script name: {argv[0]}")  print(f"Input parameter: {argv[1]}") |

В самому початку підключаються модуль argv, зо забезпечує можливість використання параметрів командного рядка. Все, що було вказано користувачем під час запуску програми зберігається в список доступ до елементів якого здійснюється використовуючи індекси.

Запустивши програму на виконання наступним чином:

|  |
| --- |
| python lab2.py lab2.csv |

Отримаємо результат:

|  |
| --- |
| Script name: lab2.py  Input parameter: lab2.csv |

Під нульовим індексом завжди зберігається ім’я програми, яка запускається. Починаючи з індексу один розміщуються параметри які буди вказані під час запуску програми на виконання.

*Робота з файлами*

Для роботи з файлами існують набір стандартних функцій, тож для відкриття, закриття, читання та запису інформації достатньо використовувати готовий функціонал.

Для того, щоб відкрити файл необхідно виконати функцію open() (<https://docs.python.org/3/library/functions.html#open>). Функція повертає посилання на файл, яке можна використовувати для наступних маніпуляцій. При закінченні використання файлу, його необхідно закрити.

|  |
| --- |
| file\_name = argv[1]  file = open(file\_name, "r")  file.close() |

Проте, використовуючи ключове слово with можна не використовувати close() в кінці використання файлу.

Після відкриття файлу існує можливість пройти всі рядки, що в ньому збережені. Розглянемо приклад:

|  |
| --- |
| file\_name = argv[1]  with open(file\_name, "r") as file:  for line in file:  print(f"line from file: {line}") |

Отримаємо результат

|  |
| --- |
| line from file: Name,Phone  line from file: Bob,1112233  line from file: Dilan,2223344  line from file: Zak,3334455 |

Для запису даних в файл використовується функція write()

|  |
| --- |
| with open(file\_name, "a") as file:  file.write("New student name,1231213") |

Для роботи з файлами формату CSV розроблено однойменний модуль, який одразу може сформувати словники з вхідних даних, що збережені в форматі CSV (<https://docs.python.org/3/library/csv.html>).

Функція DictReader() забезпечу читання файлу у вигляді словників

Розглянемо приклад:

|  |
| --- |
| file\_name = argv[1]  students = []  with open(file\_name) as file:  reader = csv.DictReader(file)  for row in reader:  students.append({"Name":row["Name"], "Phone":row["Phone"]})  print(students) |

Отримаємо результат

|  |
| --- |
| [{'Name': 'Bob', 'Phone': '1112233'}, {'Name': 'Dilan', 'Phone': '2223344'}, {'Name': 'Zak', 'Phone': '3334455'}] |

Для запису даних в CSV файл необхідно виконати дещо більше дій: відкрити файл для запису, створити обект класу DictWriter(), записати верхній рівень з назвою стовбців та безпосередньо записати данні в файл.

|  |
| --- |
| import csv  with open("lab2\_out.csv", "w", newline='') as csvfile:  fieldnames = ["Name", "Age"]  writer = csv.DictWriter(csvfile, fieldnames=fieldnames)  writer.writeheader()  writer.writerow({'Name': 'Ihor', 'Age': '37'}) |

*Юніт тести*

Процес виконання завдання може включати розбиття великої задачі на маленькі під задачі, при цьому результатом виконання маленької частина має бути певна кількість функцій, сумарний результат виконання яких задовольняє критеріям виділеної під задачі. Для того, щоб впевнитись, що написаний код виконує саме те, що вказано в завданні, розробник формує набір тестів яким піддається написаний в рамках виконання завдання код. Тестування коду – написання коду для тестування коду. При цьому поняття Юніт тест означає тестування окремого функціоналу, наприклад однієї функції (юніту).

Для мові Python існує стороння бібліотека pytest, яка реалізує механізм написання тестів. Детальний опис бібліотеки доступний за посиланням <https://docs.pytest.org/en/stable/>

Для початку використання бібліотеки необхідно виконати інсталяцію

|  |
| --- |
| pip install pytest |

Розглянемо в якості прикладу завдання на написання програми калькулятор, що запитує у користувача два параметри над якими необхідно виконати дії. В даному прикладі операції додавання та множення винесені в окремі функції. Саме ці функціями будуть виступати окремими юнітами для тестування.

|  |
| --- |
| def add(a, b):  return a + b  def mul(a, b):  return a \* b  def main():  a = int(input("What is a: "))  b = int(input("What is b: "))  print(add(a, b))  print(mul(a, b))  if \_\_name\_\_ == "\_\_main\_\_":  main() |

Перед початком написання тестів необхідно відмітити існування ключового слова assert, що надає можливість перевірити на правдивість вказану умову. Також необхідно вказати, згідно конвенції назва юніт тесту має починатися зі слова test\_

Наведемо приклад файлу з тестами:

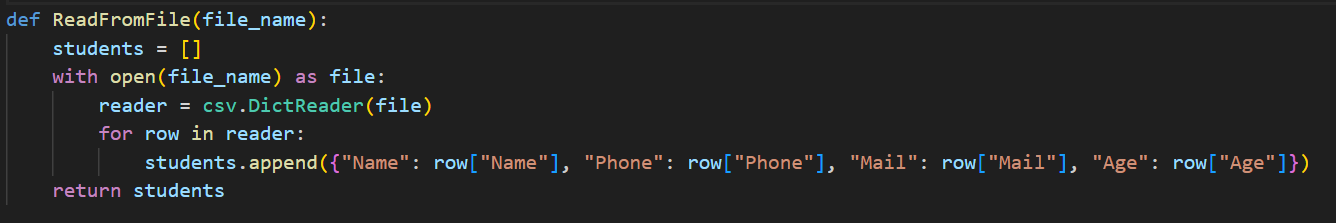
|  |
| --- |
| from lab2\_sample\_calc import add  from lab2\_sample\_calc import mul  def test\_add():  assert add(2, 2) == 4  assert add(3, 2) == 5  assert add(4, 2) == 6  def test\_mul\_positive\_both():  assert mul(2, 2) == 4  assert mul(3, 2) == 6  assert mul(4, 2) == 8  def test\_mul\_positive\_and\_negative():  assert mul(2, -2) == -4  assert mul(-3, 2) == -6  assert mul(4, -2) == -8  def test\_mul\_negative\_both():  assert mul(-2, -2) == 4  assert mul(-3, -2) == 6  assert mul(-4, -2) == 8 |

З самого початку необхідно підключити файл і функції які будуть піддаватись тестуванню. Далі відбувається тестування юніту додавання та юніту множення. Зверніть увагу, що для юніту множення розроблено три незалежних тести.

**Хід роботи:**

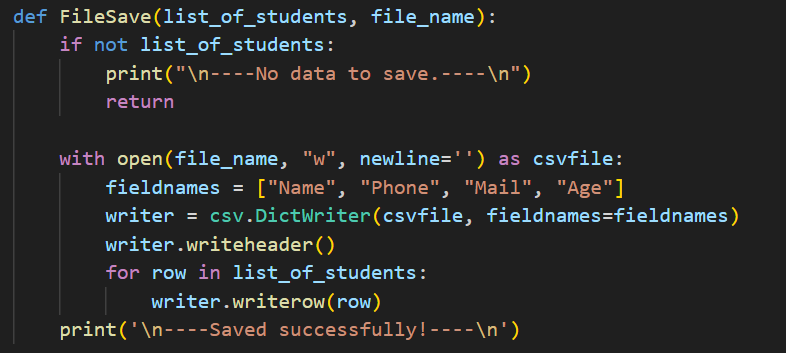
Формат збереження даних:

У коді використовується формат CSV для збереження та читання даних студентів. Це робить обмін даними між програмою та файлом більш стандартизованим і зручним для роботи з іншими інструментами.



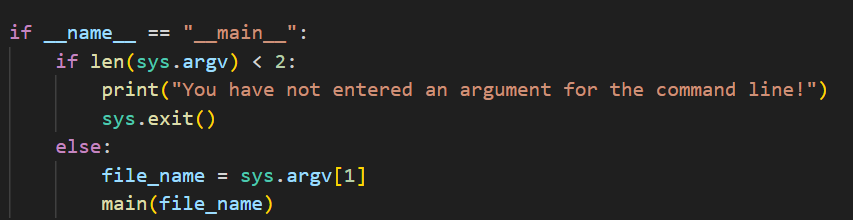
Робота із збереженням:

Додана функція FileSave, яка зберігає зміни у файлі після кожної операції, що забезпечує постійність та доліксність даних навіть після виходу з програми.



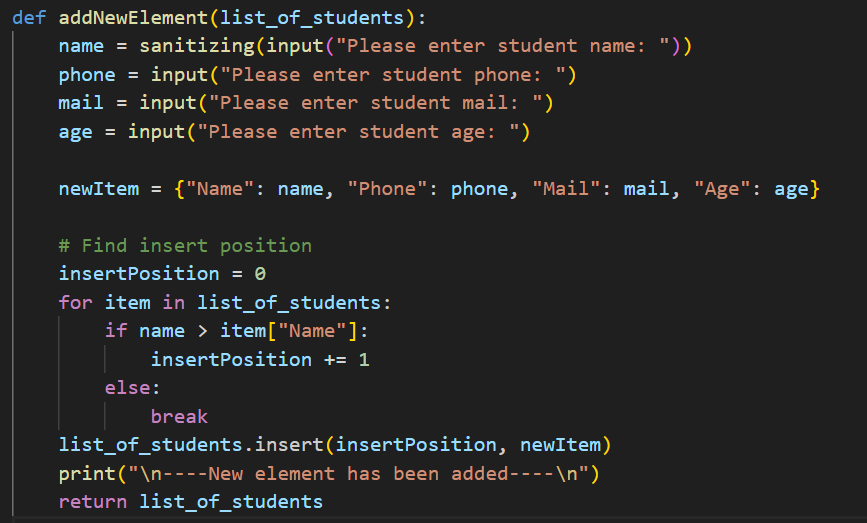
Командний рядок:

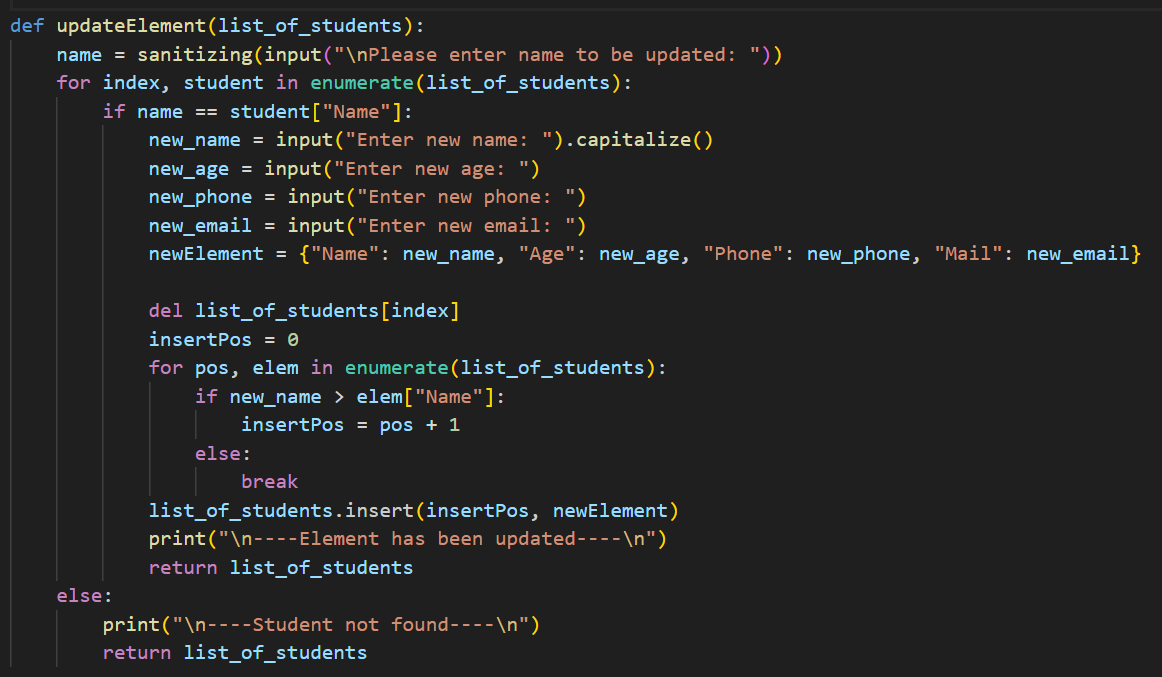
В коді додано можливість передачі імені файлу як аргумента командного рядка. Це робить програму більш гнучкою, дозволяючи працювати з різними файлами даних.

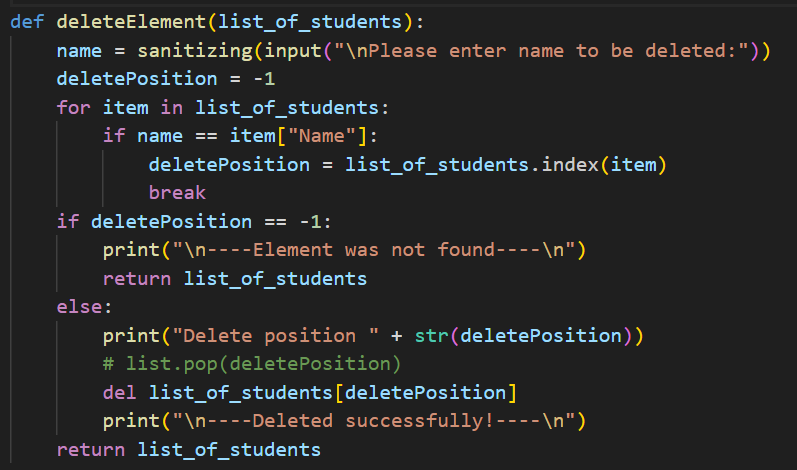


Оновлена логіка операцій:

Оновлені функції addNewElement, updateElement, та deleteElement для роботи із словниками, що використовують ключі "Name", "Phone", "Mail", "Age", що є більш читабельним та адаптованим до формату CSV.

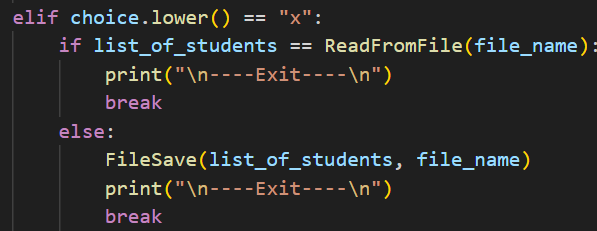
****

****

****

Умови виходу:

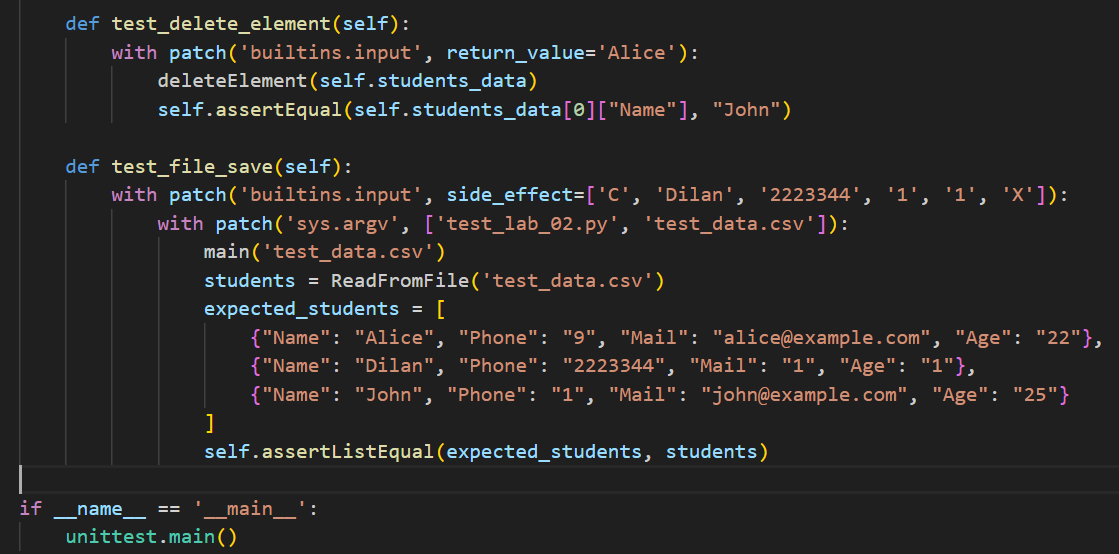
Додана умова, яка перевіряє, чи внесено зміни до даних перед виходом з програми. Якщо так, то користувачеві пропонується зберегти їх у файл.



Юніт тести:

Створені юніт тести що покривають основні функції програми.





**Висновок:** виконавши лабораторну роботу, я, використовуючи теоретичне підґрунтя про роботу з файлами та тестування коду у мові Python, розширив програму телефонного довідника студентів додавши функціонал, що було вказано в завданні до лабораторної роботи.