**Лабораторна робота №2**

Завдання: 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.**

Виконання завдання:

1. Спочатку потрібно відмовитися від списку словарів, оскільки тепер ми будемо формувати цей список словарів через читання даних csv-файлу із заданим набором даних. Ця функція буде називатися load\_csv() вона буде приймати параметр file\_name(ім’я файлу). До речі, ім’я файлу будемо отримувати з масиву arg елемент з індексом 1(Рис. 3.) цей елемент буде передано в якості аргумента функції load\_csv(). Далі в функції ми створюємо масив, відразу будемо блоки try … except… щоб перехопити помилки, якщо вони будуть. У try відкриваємо файл і зчитуємо спеціальною функцією DictReader(функція з модуля csv) дані і по рядку додаємо в масив. Коли все зчитування завершиться, повернемо результат функції через return. Результатом функції буде список list\_students. Далі цей list\_students передаємо у всі функції(printAllList, addNewElement, deleteElement, updateElement – у цих вказаних функціях робимо так, що вони приймають аргумент list\_students). На цьому створення цієї функції завершено. Не забудемо поставити виклик цієї функції в main().
2. Тепер створимо зберігання даних в csv-файл. Функція матиме назву save\_csv(), прийматиме два аргументи назву файлу та масив, які потрібно записати. Відкриваємо файл у режимі перезапису. Робимо список полів fieldnames. Далі за допомогою спеціальної функції DictWriter(з модуля csv) записуємо спочатку поля в файл lab2.csv(його передадуть у виклику). А далі робимо цикл, який перебирає масив list\_students(який передадуть при виклику функції) і записує в файл дані по рядку. На цьому створення функції завершено, тепер викличемо цю функцію, коли користувач захоче вийти, передаємо цій функції параметри назву файлу з якого ми зчитували дані(lab2.csv) і масив, який потрібно записати в цей файл(list\_students).
3. Тепер напишемо юніт-тести для функцій addNewElement(), deleteElement(), updateElement(), (save\_csv(), load\_csv() – ці дві функції будуть у test\_save\_csv()). Імпортуємо модулі для тестів та їх необхідні функції, далі імпортуємо всі функції з нашого основного файлу lab\_02.py. І ще імпортуємо модуль os для перевірки зберігання файлу. Створимо клас TestLab2, аргумент класу unittest.TestCase. Далі в ньому пропишемо функцію setUp() з аргументом self, яка буде надавати початкові дані для тестів. Початковими даними для тесту буде масив словарів list\_students. Наповнимо його. Не забуваємо, що потрібно його оголошувати через self, тобто self.list\_students. self як аргумент додаємо в кожну тест-функцію. І приступаємо реалізовувати наші тести. Тестуємо функцію addNewElement() у тест-функції test\_addNewElement(). Надалі будь-яку тест функцію називаємо так test\_«назва функції». Пишемо конструкцію with patch передаємо їй параметри builtins.input(це вказуємо, що будемо імітувати роботу функції input) і side\_effect, якому присвоюємо масив значень, які мають бути отримані внаслідок свого роду «вводу користувача». Тобто, що типу користувач як ім’я увів Denys, як номер 0673838838, як електронну пошту [denys@gmail.com](mailto:denys@gmail.com) і як групу CS-221. Далі викликаємо функцію(яка тестується) addNewElement і передає їй початкові дані self.list\_students. Коли вона відпрацює, то ми за допомогою функції тесту і передачею параметрів, що порівнюються:

self.assertEqual(self.list\_students[1]["name"], "Denys"). І для точності ще додамо такий самий тест тільки вже для Emma, щоб точно переконатися в тому, що Denys став після додавання в алфавітному порядку. Далі перевіряємо функцію оновлення даних(updateElement()). Тут все аналогічно, як з попереднім тестом для addNewElement(). Дві відмінності лише в тому, що у side\_effect вже п’ять параметрів: ім’я, нове ім’я, телефон, електронна пошта, група, а друга відмінність, що ми перевіряємо, чи залишалася Емма в списку за допомогою такої конструкції: self.assertNotIn({"name": "Emma", "phone": "0637457545", "email": "emma@gmail.com", "group": "CS-221"}, self.list\_students). Першим аргументом є словар з даними Емми, а другий аргумент, у якому списку потрібно його перевірити, що його там немає. Далі напишемо тест для функції deleteElement(). Тут аналогічна дія, як у попередньому тесті, коли ми перевіряли, що Емми вже немає в списку, а тепер перевіряємо, що немає Michael у списку. Тепер напишемо тест для функції збереження(save\_csv()), в ній протестуємо не тільки збереження файлу, а й функцію зчитування(load\_csv()) даних з файлу. Помістимо код у конструкцію try… finally… finally виконує завжди якусь дію після того, як конструкція try…(буває try… except…) відпрацює. У нашому випадку ми залишимо її блок пустим за допомогою pass. У блоці try викликаємо функцію save\_csv() і передаємо їй параметри назву файлу(test\_lab2.csv) та список self.list\_students, який необхідно зберегти у файл. Потім перевіряємо за допомогою модуля os та такої конструкції:

self.assertTrue(os.path.isfile('test\_lab2.csv')), чи справді файл було збережено і він там дійсно існує. Далі викликаємо функцію load\_csv(), передаємо їй параметр test\_lab2.csv(щойно збережений файл), цей виклик присвоїмо в змінну data. І далі за допомогою нам вже знайомої конструкції self.assertEqual(data, self.list\_students) ми порівнюємо завантажені дані з даними, які ми очікували.

Написання тестів завершено, час їх запустити.

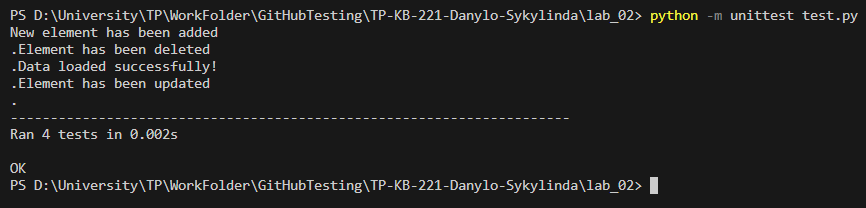


Рис. 1. Показано, всі тести виконуються успішно.

Усі тести проходять, отже ми все зробили правильно.

Код завдання:

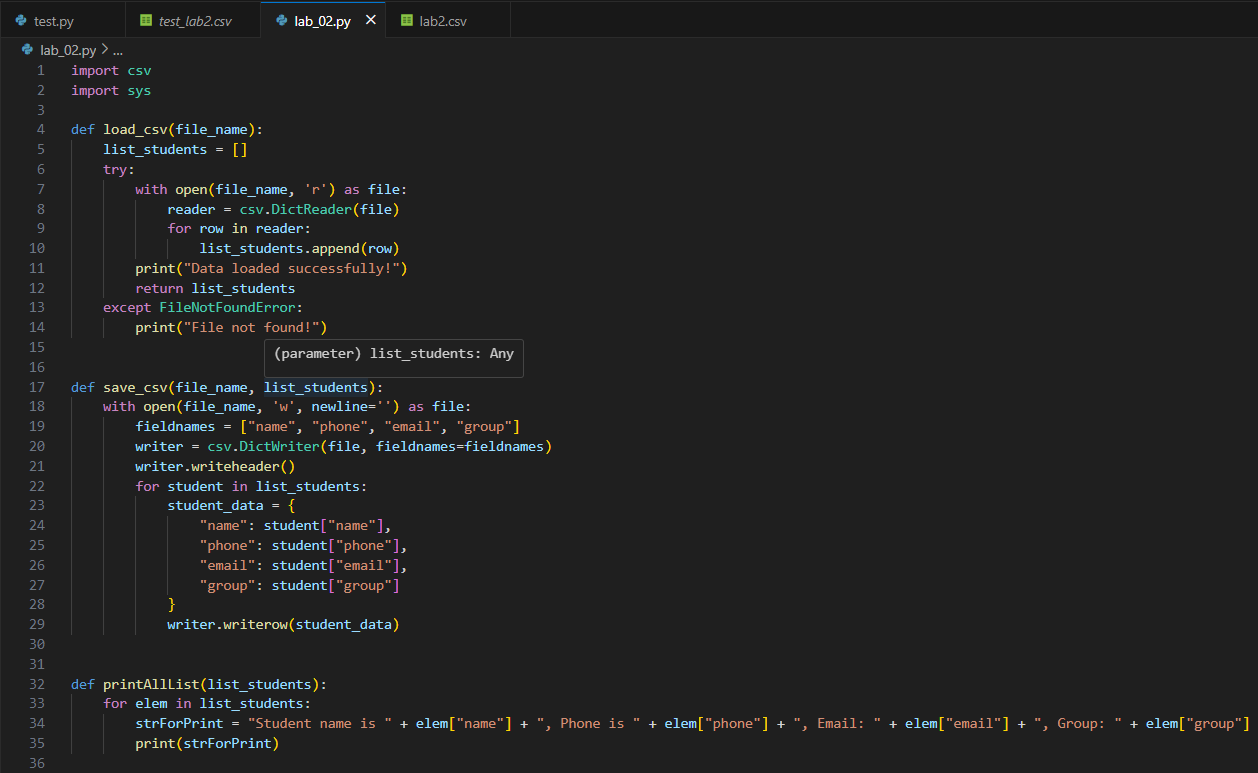


Рис. 2. Код файлу lab\_02.py.

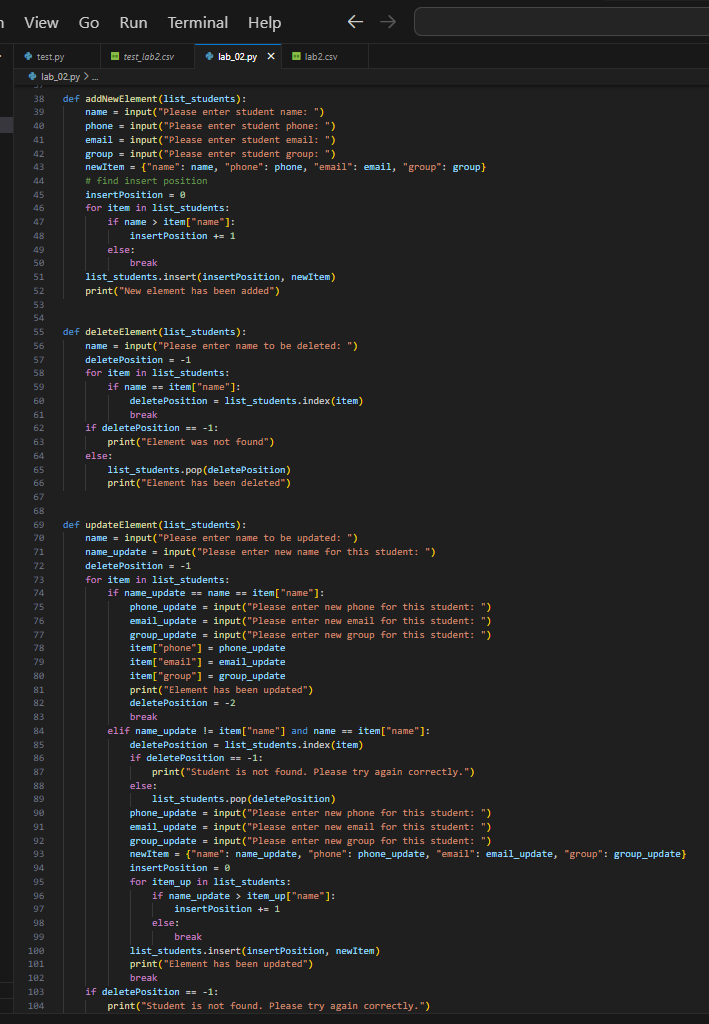


Рис. 3. Продовження коду файлу lab\_02.py.

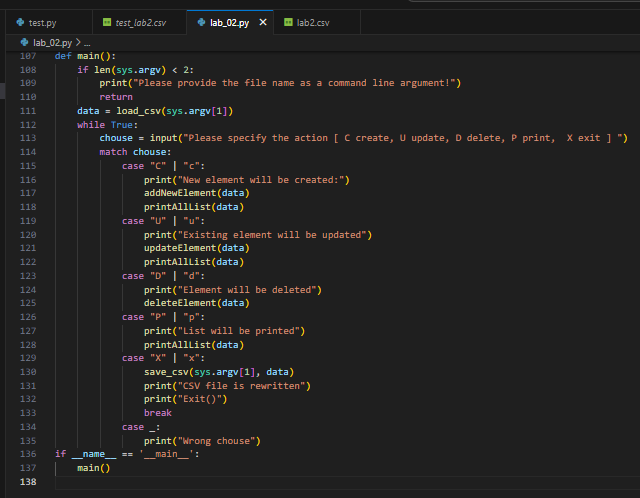


Рис. 4. Продовження коду файлу lab\_02.py.

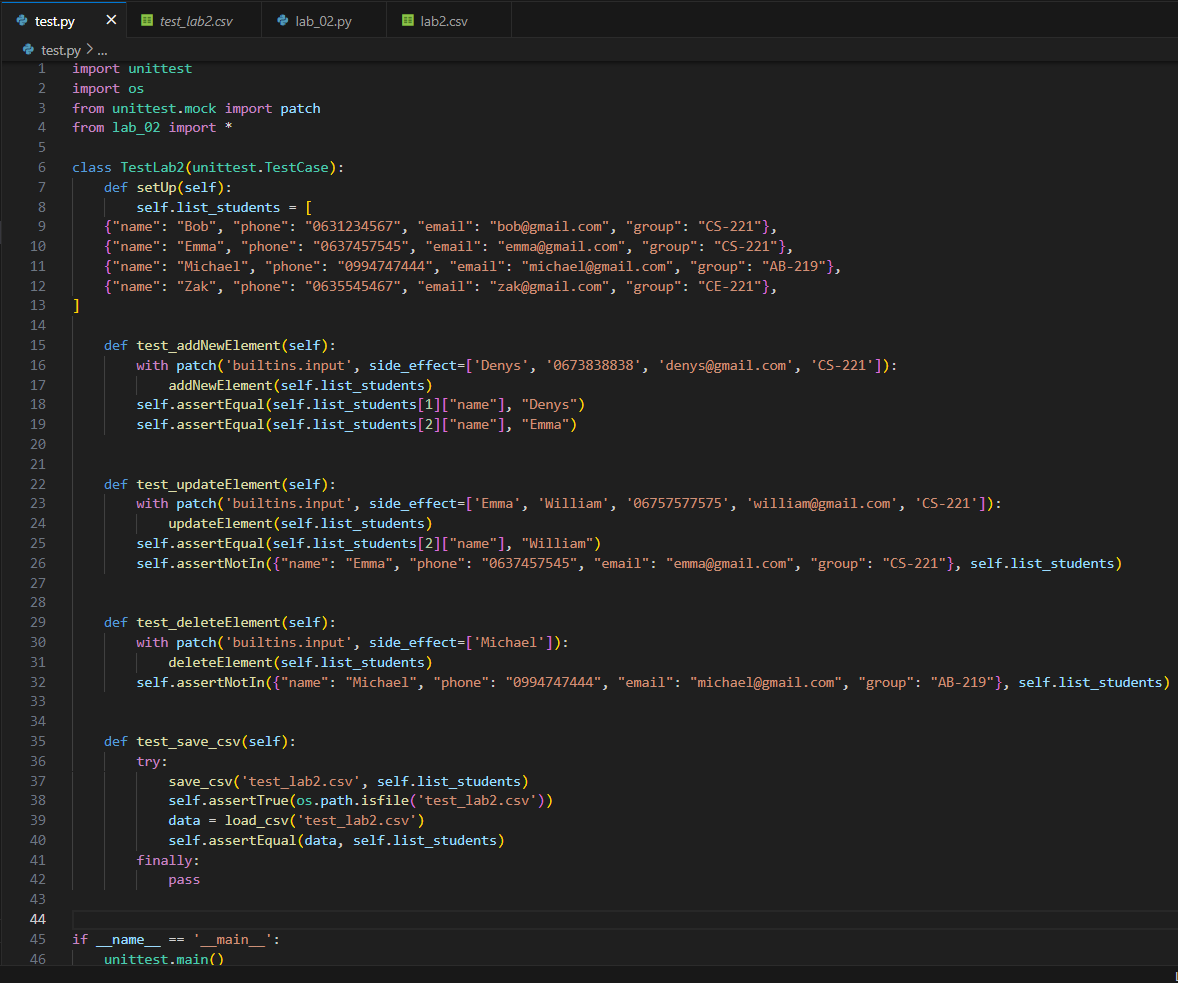


Рис. 5. Код файлу test.py.

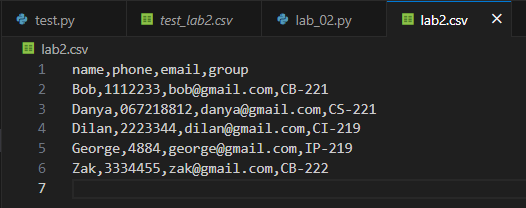


Рис. 6. Дані з файлу lab2.csv для роботи коду файлу lab\_02.py.

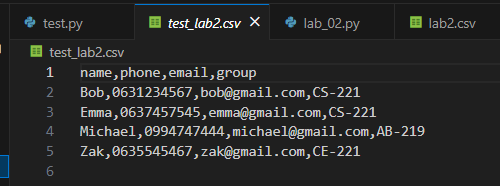


Рис. 7. Дані з файлу test\_lab2.csv для роботи тесту test.py коду файлу lab\_02.py.

GitHub:

Посилання на GitHub на цю папку з файлами лабораторної роботи: <https://github.com/danylosykylinda/TP-KB-221-Danylo-Sykylinda/tree/main/lab_02>.

Скріншот коду з GitHub:

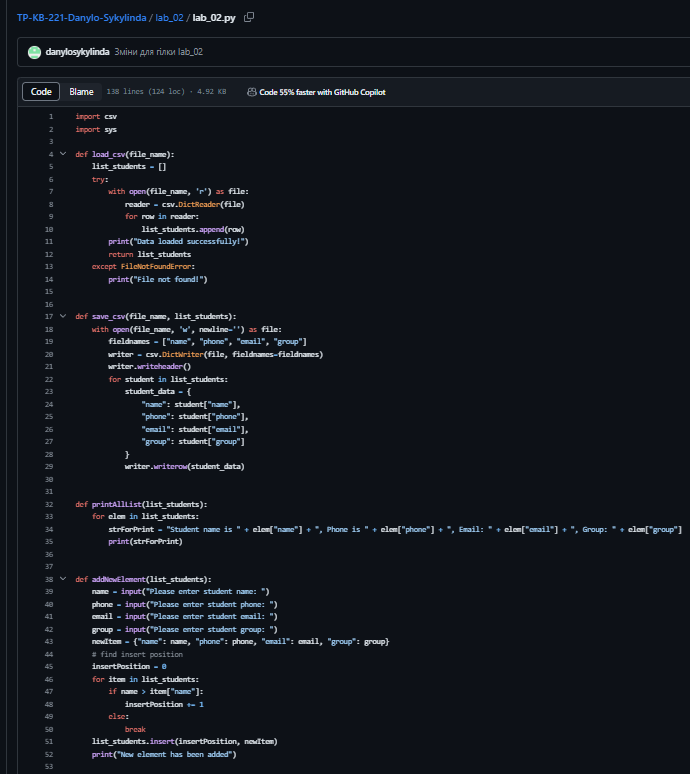


Рис. 8. Код файлу lab\_02.py.

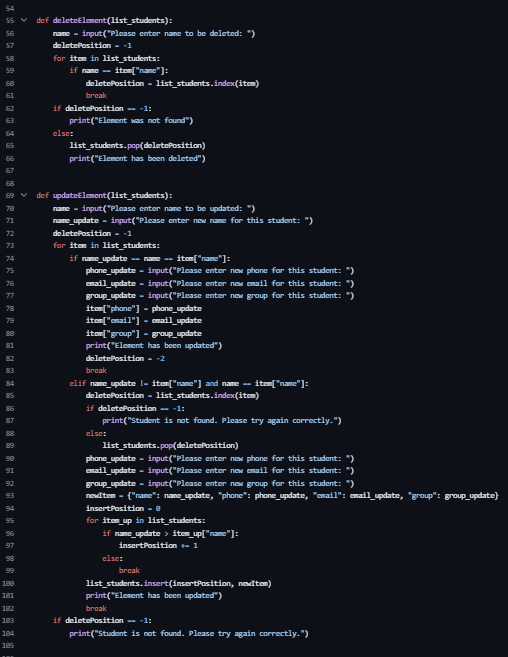


Рис. 9. Продовження коду файлу lab\_02.py.

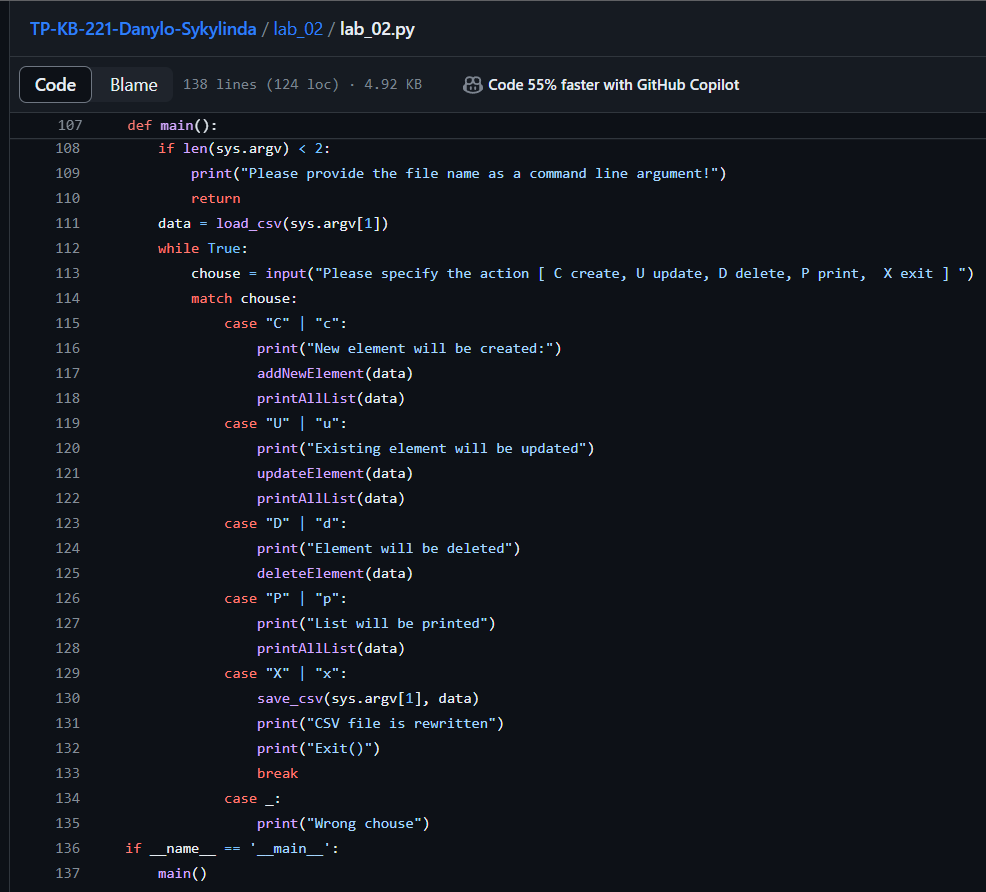


Рис. 10. Продовження коду файлу lab\_02.py.

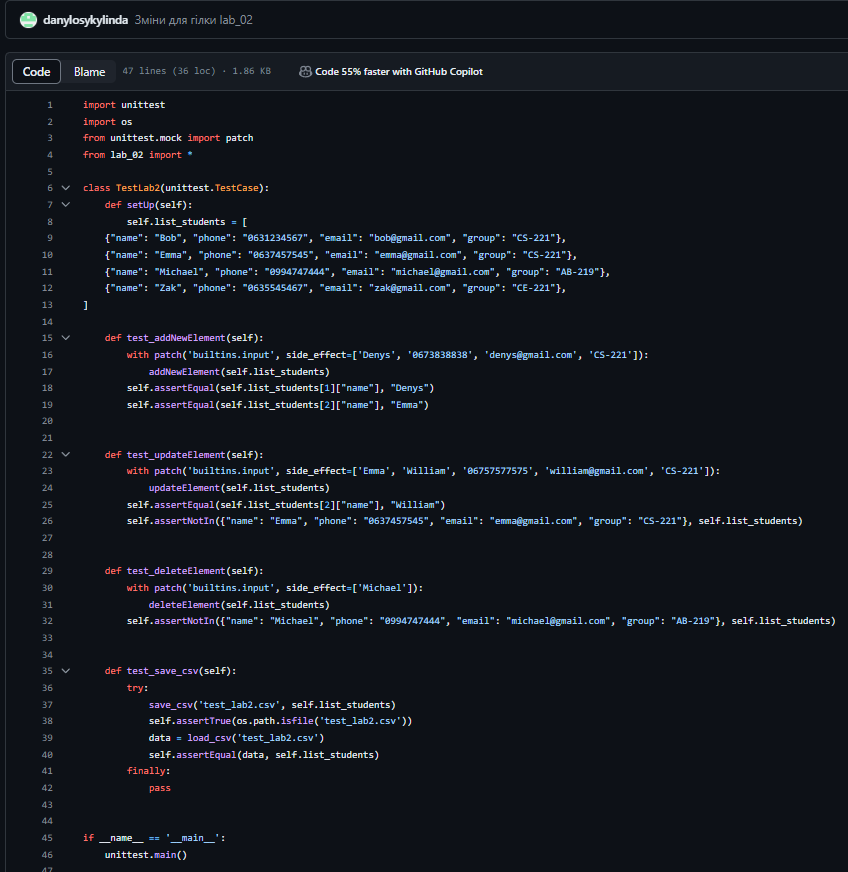


Рис. 11. Код файлу test.py для юніт-тестів для коду файлу lab\_02.py.

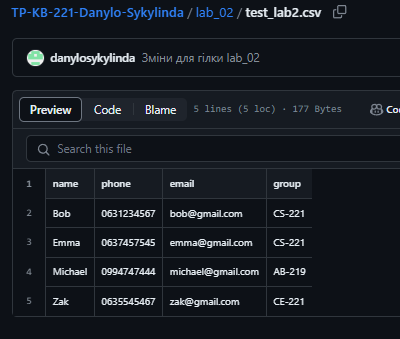


Рис. 12. Файл test\_lab2.csv з початковими даними для юніт-тестів.

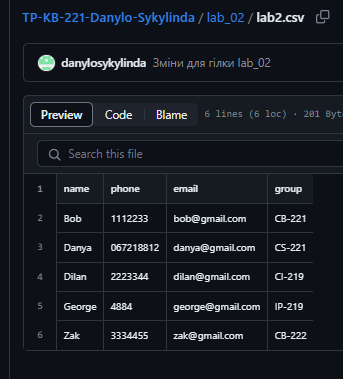


Рис. 13. Файл lab2.csv з початковими даними для роботи коду lab\_02.py.