Міністерство освіти і науки України

Національний університет «Полтавська політехніка імені Юрія Кондратюка»

Навчально-науковий інститут інформаційних технологій та робототехніки

Кафедра комп’ютерних та інформаційних технологій і систем

**Пояснювальна записка**

**до курсової роботи з дисципліни «Програмування»**

**на тему «Розробка додатку з використанням мови Python»**

Виконав студент гр. 203-ТК

Нонка Іван Андрійович

Керівник курсової роботи

к.ф-м.н., доцент

Двірна О.А.

Дата захисту:

« » травня 2024р.

Оцінка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Полтава 2024 р.

Міністерство освіти і науки України

Національний університет “Полтавська політехніка імені Юрія Кондратюка”

Навчально-науковий інститут інформаційних технологій та робототехніки

Кафедра комп’ютерних та інформаційних технологій і систем

**ЗАВДАННЯ**

**до курсової роботи з дисципліни «Програмування»**

**на тему «Розробка додатку з використанням мови Python»**

**Група** 203ТК

**Студент Нонка І.**

**Завдання:**

Розробити та реалізувати мовою Python програму - телефонний довідник, що містить такі відомості: ПІБ, адреса, електронна пошта, телефон (мобільний, домашній). Дані зберігаються в csv файлі.

Функції роботи довідника:

- додавання нових контактів;

- редагування існуючих контактів. При цьому на одного абонента може припадати кілька номерів телефону;

- перегляд існуючих контактів. Контакти повинні виводитися у відсортованому порядку;

- видалення контакту;

- пошук за параметрами – Прізвище, номер телефону.

Програма повинна містити довідку, де описуються основні можливості програми та інструкція для користувача.

Структура пояснювальної записки:

Титульна сторінка Завдання

Зміст

Вступ

Розділ 1. Теоретична частина.

Розділ 2. Практична частина.

Висновки

Список використаних джерел

Додатки

Завдання видала к.ф.-м.н., доц. Двірна О.А

**Зміст**

ВСТУП….………………………………………………………………………….4

РОЗДІЛ 1 ТЕОРЕТИЧНА ЧАСТИНА…………………………………………...6

* 1. Опис мови програмування та її переваг для розробки додатка………...6
  2. Опис бібліотеки для розробки додатків та елементів інтерфейсу……...7
  3. Опис використаних бібліотек…………………………………………...10
  4. Файлова структура додатку та імпорт модуля…………………………11

РОЗДІЛ 2. ПРАКТИЧНА ЧАСТИНА……………………………………………..12

ВИСНОВКИ………………………………………………………………….32

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……………………………………..33

Додаток А. Програмний код тестового додатку …………………………34

**ВСТУП**

У сучасному світі інформаційні технології та програмування відіграють важливу роль у багатьох різних аспектах життя, включаючи бізнес, освіту, медицину та інші сфери.

Одним із важливих завдань, які стоять перед суспільством, є ефективне управління та зберігання інформації.

У цьому контексті телефонна книга стає важливим інструментом, який дозволяє впорядковувати та швидко отримувати доступ до контактних даних.

Традиційні паперові телефонні довідники вже давно неактуальні через обмежену функціональність і незручність у використанні.

Натомість цифрові телефонні довідники пропонують користувачам більше можливостей, забезпечуючи зручність, швидкість і надійність доступу до важливої ​​інформації.

Вибір мови програмування Python для реалізації телефонної книги був не випадковим.

Python є однією з найпопулярніших мов програмування завдяки своїй простоті, гнучкості та багатим бібліотекам.

Python дозволяє швидко розробляти додатки різної складності і має широкий функціонал для роботи з даними.

Завдяки простому синтаксису Python є ідеальною мовою як для початківців, так і для професійних розробників, що робить його оптимальним вибором для створення ефективних і простих у використанні телефонних довідників.

Мета курсової роботи – розробити додаток “Телефонний довідник” засобами мови програмування Python, який буде містити такі функції:

- додавання нових контактів;

- редагування існуючих контактів. При цьому на одного абонента може припадати кілька номерів телефону;

-перегляд існуючих контактів. Контакти повинні виводитися у відсортованому порядку;

- видалення контакту;

- пошук за параметрами – Прізвище, номер телефону.

**Для досягнення поставленої мети** необхідно виконати такі завдання:

- вибір інструментів розробки;

- розробка дизайну/макету додатку;

- написання коду додатку відповідно до технічного завдання;

- тестування додатку.

**Об’єкт дослідження** процес розробки додатків засобами мови Python.

**Предмет дослідження** – розробка додатку “Телефонний довідник” з використанням мови Python.

Python — це мова програмування, яку все ще можна вважати новою з точки зору використання для розробки мобільних додатків, але вона довела свою ефективність і переваги.

Таким чином, написання програм на Python дуже підходить завдяки його перевагам простоти, гнучкості та ефективності, а також його великій спільноті та широкій спеціалізації в різних областях розробки програмного забезпечення.

**РОЗДІЛ 1 ТЕОРЕТИЧНА ЧАСТИНА**

Python є дуже популярною мовою програмування, яка використовується в багатьох популярних програмах і службах Instagram, одна з найбільших у світі соціальних мереж, використовує Python для розробки back-end.

Вони обрали Python через його простоту, ефективність і здатність швидко масштабуватися, використовуючи фреймворк Django для створення та підтримки своїх веб-сервісів.

Spotify - популярний сервіс потокової передачі музики, використовує Python для аналізу даних і розробки серверної частини, допомагаючи службі обробляти великі обсяги даних, аналізувати поведінку користувачів і створювати музичні рекомендації.

* 1. Опис мови програмування та її переваг для розробки додатка.

Опис мови програмування Python і її переваг для розробки додатків Python є однією з найпопулярніших мов програмування у світі, широко визнаною своєю простотою, гнучкістю та потужністю. Він був створений Гвідо ван Россумом і вперше випущений у 1991 році.

З того часу Python розвивався та вдосконалювався, що робить його одним із найкращих варіантів для розробників у різних сферах програмування.Однією з головних переваг Python є його простота.Синтаксис мови інтуїтивно зрозумілий і простий у вивченні, що дозволяє розробникам швидко створювати та читати код. Це робить Python ідеальною мовою для початківців, а також для професіоналів, які хочуть швидко створювати нові ідеї та проекти. Простота Python зменшує кількість помилок у коді та підвищує продуктивність розробника.

Ще однією важливою перевагою Python є гнучкість.

Ця мова підтримує багато парадигм програмування, включаючи об'єктно-орієнтоване програмування, процедурне програмування та функціональне програмування. Це дозволяє розробникам використовувати Python для різноманітних завдань, від веб-розробки та аналізу даних до автоматизації завдань і наукових обчислень. Python має велику кількість бібліотек і фреймворків, які значно спрощують розробку програм.

Наприклад, фреймворк Django широко використовується для створення веб-додатків завдяки своїй комплексності та потужним інструментам розробки серверної частини. Інші популярні бібліотеки включають Flask для легких веб-додатків, Pandas для аналізу даних, NumPy для наукових обчислень і TensorFlow для машинного навчання. Ці бібліотеки дозволяють розробникам значно скоротити час розробки та зосередитися на вирішенні конкретних проблем.

Python також відомий своєю гнучкістю та кросплатформеністю. Програми, написані на Python, можуть працювати в різних операційних системах, включаючи Windows, macOS і Linux, не вимагаючи значних змін коду. Це забезпечує легке розгортання та підтримку програми на різних платформах.

Ще однією важливою перевагою Python є його велика спільнота розробників. Спільнота Python активно підтримує та розвиває мову, про що свідчить велика кількість документації, навчальних ресурсів, форумів і відкритих проектів. Це полегшує розробникам пошук відповідей на свої запитання та швидке вирішення проблем, що виникають під час розробки. Завдяки численним перевагам Python є чудовим вибором для розробки різних програм. Його простота, універсальність, велика кількість бібліотек і фреймворків, гнучкість і велика спільнота роблять Python ідеальною мовою для створення як невеликих проектів, так і масштабних програмних рішенью

Використання Python для розробки програми телефонної книги забезпечить швидке й ефективне виконання завдань, а також дозволить легко розширювати та підтримувати програму в майбутньому.

**1.2** Опис бібліотеки для розробки додатків та елементів інтерфейсу

Різні бібліотеки та фреймворки можна використовувати для розробки додатків на мові програмування Python, включаючи додаток Phone Book, і для створення його інтерфейсу.

Однією з найпопулярніших бібліотек для розробки графічних інтерфейсів користувача (GUI) на Python є Tkinter. Це стандартна бібліотека для створення GUI на Python. Він заснований на Tcl/Tk, що робить його доступним на більшості платформ. Tkinter надає низку вбудованих віджетів, таких як кнопки, текстові поля, списки, перемикачі та багато інших, які можна використовувати для створення інтерфейсу користувача вашої програми. Він також має простий інтерфейс, що робить його популярним вибором для новачків.

Іншою популярною альтернативою для розробки GUI на Python є PyQt. PyQt — це зв’язка між Python і Qt, кросплатформною структурою GUI.Він має більш розширену функціональність, ніж Tkinter, і часто використовується для більш складних і професійних програм.PyGTK — ще один варіант розробки графічного інтерфейсу Python на основі GTK+, бібліотеки для створення графічного інтерфейсу користувача.

Він також підтримується на багатьох платформах і має багато віджетів і функцій.Ви можете використовувати бібліотеку Flask для створення інтерактивних веб-інтерфейсів.

Flask — це легкий мікрофреймворк для веб-розробки на Python, який дозволяє швидко створювати веб-додатки з невеликими зусиллями.

Він ідеально підходить для невеликих проектів, таких як телефонні довідники, і дозволяє легко створювати сторінки, керувати запитами користувачів та інтегрувати різноманітні функції.

Крім того, можна використовувати Django, повну веб-платформу для Python, для створення веб-інтерфейсів.

Він має вбудовані утиліти для роботи з базами даних, автентифікації користувачів, панелями адміністратора та іншими функціями, що робить його ідеальним вибором для великих і складних веб-проектів.

Залежно від потреб вашого проекту, ви можете вибрати одну з цих бібліотек або фреймворків для розробки інтерфейсу програми «Контакти».

Є переваги та недоліки, тому перед вибором тієї чи іншої технології важливо враховувати особливості проекту та власні вимоги.

Наприклад, якщо вам потрібен швидкий прототип або ви працюєте над невеликим проектом із простим інтерфейсом, Tkinter може стати чудовим вибором. З іншого боку, якщо ваш проект має складний і багатофункціональний інтерфейс, PyQt або PyGTK можуть бути більш актуальними варіантами. Для веб-додатків Flask може бути ідеальним для швидкого розгортання простих програм, тоді як Django може бути кращим вибором для складних веб-проектів із багатьма функціями та вимогами.

Яку бібліотеку чи фреймворк вибрати для розробки програми телефонного довідника, залежатиме від вашого досвіду, обсягу проекту, кінцевого терміну та інших факторів. Перш ніж прийняти рішення, важливо оцінити всі аспекти та вибрати технологію, яка найкраще відповідає вашим потребам і цілям проекту.

1.3 Опис використаних бібліотек

У проєкті, метою якого є розробка програми «Телефонний довідник» за допомогою мови програмування Python, використані бібліотеки та модулі матимуть наступне:

import csv: Ця бібліотека використовуватиметься для роботи з файлами CSV, які діятимуть як база даних для контактів. Кожен контакт буде збережено як рядок у файлі CSV. Під час додавання нового контакту або редагування наявного контакту дані зчитуються та записуються у форматі CSV.

import tkinter: Ця бібліотека використовуватиметься для створення графічного інтерфейсу користувача (GUI). За допомогою tkinter ви можете створювати вікна, кнопки, текстові поля, списки контактів тощо.

Графічний інтерфейс забезпечить зручну взаємодію користувача з програмою, дозволяючи додавати, редагувати, переглядати та видаляти контакти за допомогою графічних елементів.

Прості повідомлення та діалогові вікна: Ці модулі бібліотеки tkinter використовуватимуться для відображення повідомлень та отримання введених користувачем даних через діалогові вікна.

Наприклад, коли ви намагаєтеся видалити контакт або додати новий контакт, може з’явитися діалогове вікно з підтвердженням або запитом інформації від користувача.

import ttk: Цей модуль tkinter дозволяє створювати більш сучасний та елегантний інтерфейс, використовуючи теми та інноваційні віджети дизайну.

Наприклад, ttk можна використовувати для створення стильних кнопок, розкривних списків, перемикачів, які підсвічуються при виборі тощо. Це зробить інтерфейс програми більш привабливим і зручним.

* 1. Файлова структура додатку та імпорт модуля

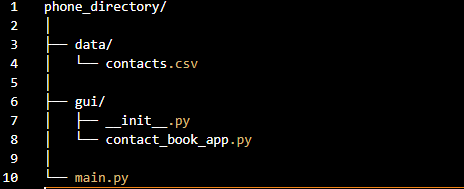


Рисунок 1.1

Опис файлів і папок:

* data/: Каталог для зберігання файлів контактів (contacts.csv).

Цей файл зберігає контактні дані у форматі CSV.

* gui/: Каталог, що містить файли, які використовуються для створення графічного інтерфейсу користувача.
* \_\_init\_\_.py: Порожній файл, який вказує, що gui/ є пакетом Python.
* contact\_book\_app.py: Файл, у якому реалізовано клас ContactBookApp, що містить головне вікно програми та її функції.
* main.py: Основний файл програми, у якому буде запущено програму. У цей файл буде імпортовано клас ContactBookApp модуля contact\_book\_app.py і запущено основний цикл програми.
* Що стосується імпорту модуля, файл main.py імпортує клас ContactBookApp з модуля contact\_book\_app.py:

**РОЗДІЛ 2. ПРАКТИЧНА ЧАСТИНА**

Перед початком виконання проєкту було обрано основні бібліотеки для створення графічного інтерфейсу та запису інформації, модулі для виведення повідомлень та отримання введених користувачем даних через діалогові вікна, створювати більш сучасний та стильний інтерфейс

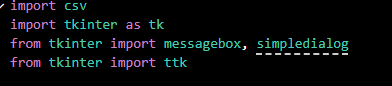


Рисунок 2.1 (список бібліотек та модулів)

Після вибору основних бібліотек та модулів, було розроблено головне вікно програми "Телефонний довідник, яке створює і відображає основні функції, які може виконати користувач.

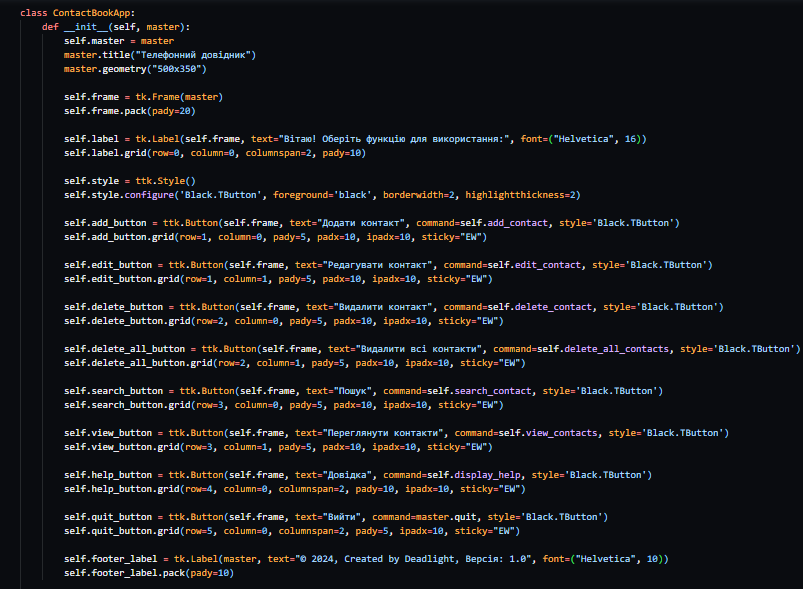


Рисунок 2.2(Уривок коду для створення головного вікна програми)

Детальний розбір всіх елементів у рядках функції:

1.self.master = master: Цей рядок встановлює атрибут master класу ContactBookApp, який буде посилатися на батьківське вікно Tkinter.

1. master.title("Телефонний довідник"): Встановлює заголовок головного вікна на "Телефонний довідник".
2. master.geometry("500x350"): Встановлює розміри головного вікна на 500 пікселів у ширину та 350 пікселів у висоту.
3. self.frame = tk.Frame(master): Створюється новий фрейм Tkinter, який буде вміщувати решту віджетів.
4. self.frame.pack(pady=20): Розміщує фрейм у вікні з вертикальним відступом 20 пікселів.
5. self.label = tk.Label(self.frame, text="Вітаю! Оберіть функцію для використання:", font=("Helvetica", 16)): Створює мітку (Label) з текстом "Вітаю! Оберіть функцію для використання:" та шрифтом Helvetica розміром 16. Вона розміщується в фреймі self.frame.
6. self.label.grid(row=0, column=0, columnspan=2, pady=10): Розміщує мітку у вікні за допомогою сітки з рядком 0 та стовпцем 0. columnspan=2 вказує на те, що мітка займає два стовпці. pady=10 встановлює вертикальний відступ у 10 пікселів.
7. self.style = ttk.Style(): Створюється об'єкт стилю для віджетів ttk.
8. self.style.configure('Black.TButton', foreground='black', borderwidth=2, highlightthickness=2): Налаштовує стиль кнопок ttk з ім'ям 'Black.TButton', встановлюючи чорний колір тексту, ширину рамки 2 пікселі та товщину підсвічування 2 пікселі.
9. self.add\_button = ttk.Button(self.frame, text="Додати контакт", command=self.add\_contact, style='Black.TButton'): Створює кнопку ttk з текстом "Додати контакт", яка буде викликати метод add\_contact при натисканні.
10. self.add\_button.grid(row=1, column=0, pady=5, padx=10, ipadx=10, sticky="EW"): Розміщує кнопку у вікні за допомогою сітки з рядком 1 та стовпцем 0. pady=5 встановлює вертикальний відступ у 5 пікселів, padx=10 - горизонтальний відступ, ipadx=10 - внутрішній горизонтальний відступ, а sticky="EW" вказує, що кнопка розтягується на всю ширину.

Аналогічно, кожен рядок коду ініціалізує та налаштовує віджети на головному вікні додатку.

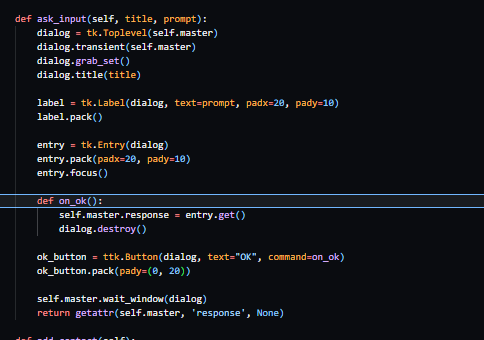


Рисунок 2.3 – Продовження коду графічного інтерфейсу

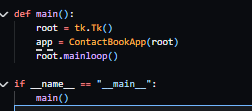


Рисунок 2.4 – Продовження коду графічного інтерфейсу

Детальний розбір всіх елементів у рядках функції:

1. dialog = tk.Toplevel(self.master): Створюється нове діалогове вікно, яке буде дочірнім до головного вікна master.
2. dialog.transient(self.master): Встановлює, що це діалогове вікно є дочірнім для головного вікна master, що означає, що воно пов'язане з головним вікном.
3. dialog.grab\_set(): Встановлює фокус на діалоговому вікні, що забороняє взаємодію з головним вікном, доки діалогове вікно не буде закрите.
4. dialog.title(title): Встановлює заголовок діалогового вікна на значення title.
5. label = tk.Label(dialog, text=prompt, padx=20, pady=10): Створюється мітка (Label) з текстом prompt та горизонтальним і вертикальним відступом 20 та 10 пікселів відповідно. Ця мітка розміщується в діалоговому вікні dialog.
6. label.pack(): Розміщує мітку в діалоговому вікні.
7. entry = tk.Entry(dialog): Створюється поле введення (Entry), яке буде використовуватися для введення даних користувачем. Це поле розміщується в діалоговому вікні dialog.
8. entry.pack(padx=20, pady=10): Розміщує поле введення в діалоговому вікні з горизонтальним і вертикальним відступом 20 та 10 пікселів відповідно.
9. entry.focus(): Встановлює фокус в поле введення, щоб користувач міг почати вводити текст безпосередньо.
10. Внутрішня функція on\_ok(): Ця функція викликається при натисканні на кнопку "OK". Вона зчитує текст з поля введення та зберігає його у властивості response головного вікна master, а потім закриває діалогове вікно.
11. ok\_button = ttk.Button(dialog, text="OK", command=on\_ok): Створюється кнопка "OK", яка при натисканні викликає функцію on\_ok.
12. ok\_button.pack(pady=(0, 20)): Розміщує кнопку "OK" у діалоговому вікні з вертикальним відступом відповідно 0 зверху і 20 знизу.
13. self.master.wait\_window(dialog): Зупиняє виконання програми до тих пір, поки діалогове вікно dialog не буде закрите.
14. return getattr(self.master, 'response', None): Повертає значення властивості response головного вікна master, або None, якщо такої властивості немає.
15. Функція main() є точкою входу в програму. Вона створює головне вікно програми та запускає головний цикл обробки подій Tkinter.
16. В першому рядку функції main(), root = tk.Tk() створює головне вікно програми Tkinter, яке є екземпляром класу Tk.
17. Наступний рядок app = ContactBookApp(root) створює екземпляр класу ContactBookApp, який, ймовірно, відображає головне вікно програми телефонного довідника. Зазвичай клас ContactBookApp визначається окремо в коді і містить логіку створення та керування віджетами GUI.
18. Коли програма закінчує роботу (наприклад, користувач закриває вікно програми), головний цикл Tkinter закінчується, і виконання повертається до функції main().
19. Умовний оператор if \_\_name\_\_ == "\_\_main\_\_": перевіряє, чи файл був запущений безпосередньо з командного рядка (а не імпортований у інший модуль). Якщо це так, викликається функція main(), і програма починає виконуватись.

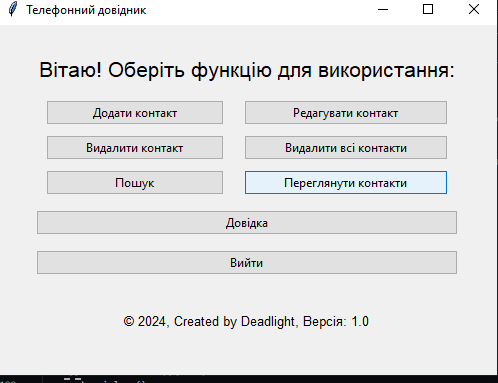
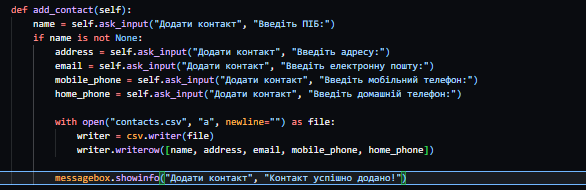


Рисунок 2.3 (Головне вікно програми)

При натисканні на кнопку “Додати контакт” викликається функція add\_contact(self) рис(2.4)

Рисунок 2.4 – Код функції add\_contact(self)

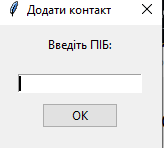
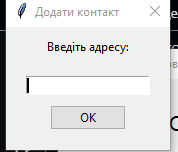


Рисунок 2.5 – Початкове повідомлення про введення ініціалів контакту

  
Рисунок 2.6 – Повідомлення про введення адреси проживання контакту

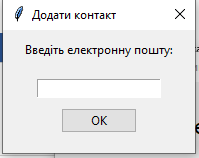


Рисунок 2.7 – Запит про введення електронної адреси контакту

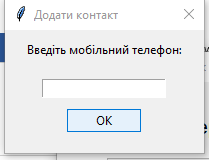
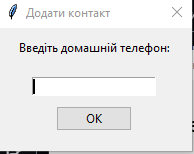


Рисунок 2.8 -Запит на введення мобільного телефону контакту  
  
Рисунок 2.9 – Запит на введення домашнього телефону контакту

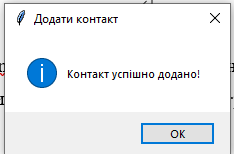


Рисунок 2.10 – Повідомлення про успішне додавання контакту до телефонного довідника

Детальний розбір всіх елементів у рядках функції:

Ця функція `add\_contact` призначена для додавання нового контакту в телефонний довідник. Давайте розберемо кожен рядок коду:

1. `name = self.ask\_input("Додати контакт", "Введіть ПІБ:")`: Викликається метод `ask\_input`, який створює діалогове вікно для введення ПІБ контакту. Введене значення зберігається в змінній `name`.

2. `if name is not None:`: Перевіряється, чи користувач ввів ПІБ. Якщо введено значення, програма продовжує виконання.

3. Аналогічно отримуються адреса (`address`), електронна пошта (`email`), мобільний телефон (`mobile\_phone`) і домашній телефон (`home\_phone`). Для кожного атрибута відбувається виклик методу `ask\_input`, де користувач вводить відповідні дані.

4. `with open("contacts.csv", "a", newline="") as file:`: Відкривається файл "contacts.csv" для додавання нового запису. Файл відкривається у режимі додавання ("a"), щоб зберегти існуючі дані та додати новий запис.

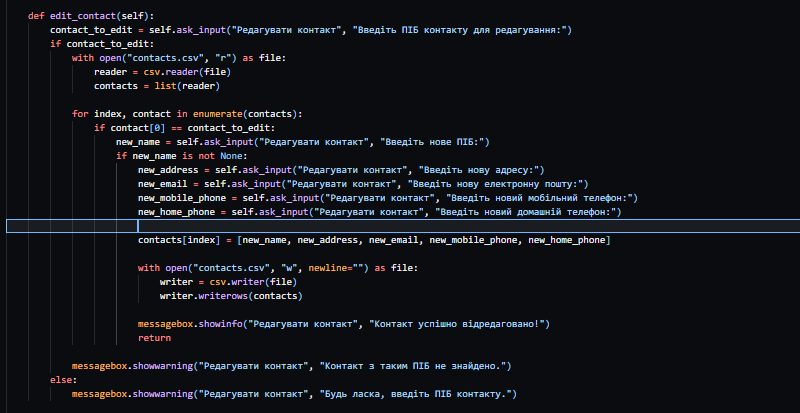
5. `writer = csv.writer(file)`: Створюється об'єкт `writer` для запису в файл `file` у форматі CSV.

6. `writer.writerow([name, address, email, mobile\_phone, home\_phone])`: Записує рядок з даними контакту у файл CSV. Кожен атрибут контакту розділений комами.

7. `messagebox.showinfo("Додати контакт", "Контакт успішно додано!")`: Показує діалогове вікно з повідомленням про успішне додавання контакту.

Після створення функції add\_contact(self), яка створює основний функціонал кнопки “Додати контакт”, ми переходимо до написання функціоналу кнопки “Редагувати контакт”:

При натисканні на кнопку “ Редагувати контакт ” викликається функція edit\_contact(self) рис(2.11)

  
Рисунок 2.11 – Код функції edit\_contact(self)

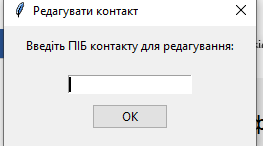


Рисунок 2.12 – Запит на існуючі ініціали контакту для знаходження його в телефонному довіднику

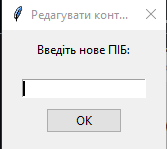


Рисунок 2.13 – Запит на редагування ініціалів контакту

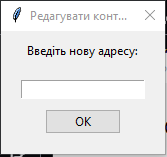


Рисунок 2.14 – Запит на редагування адреси контакту

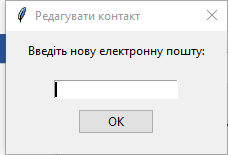
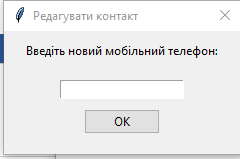
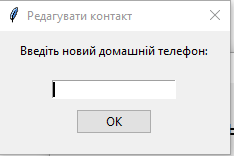
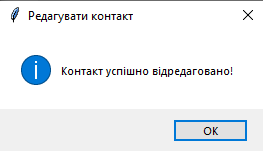


Рисунок 2.15 -Запит на редагування електронної пошти

  
Рисунок 2.16.- Запит на редагування номеру мобільного телефону контакту  
  
Рисунок 2.17 – Запит на редагування номеру домашнього телефону

  
Рисунок 2.18 – Повідомлення про успішне редагування контакту

Розглянемо застереження програми:

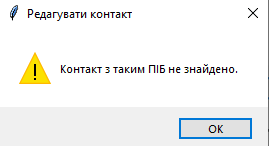
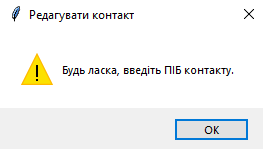


Рисунок 2.19 – Повідомлення про помилку, так як контакту з такими ініціалами не знайдено у базі даних  
  
Рисунок 2.20 – Повідомлення про те, що користувач не ввів у відповідне поле ініціали контакту

Детальний розбір всіх елементів у рядках функції:

Функція edit\_contact(self) призначена для редагування ініціалів, номеру телефону, електронної пошти, тощо. Давайте розберемо її вміст:  
Виводить повідомлення з запитом на введення ПІБ контакту, який вони хочуть відредагувати.

1. Перевіряє, чи користувач ввів дані. Якщо так, продовжується виконання коду.
2. Відкриває файл contacts.csv для читання.
3. Зчитує вміст файлу CSV у список contacts, де кожен елемент цього списку - це рядок контакту.
4. Проходиться по кожному контакту у списку contacts і порівнює ПІБ контакту з тим, який користувач ввів.
5. Якщо знаходить контакт з таким ПІБ, запитує користувача нові дані (нове ПІБ, адреса, електронна пошта, мобільний телефон, домашній телефон).
6. Оновлює дані цього контакту у списку contacts.
7. Записує оновлений список контактів у файл contacts.csv.
8. Виводить повідомлення про успішне відредагування контакту за допомогою messagebox.showinfo.
9. Після успішного відредагування контакту повертається з методу.
10. Якщо контакт із введеним ПІБ не знайдено, показує повідомлення про це користувачеві за допомогою messagebox.showwarning.
11. Якщо користувач не ввів ПІБ контакту, також показує відповідне повідомлення.

Тепер розробимо функцію “Видалити контакт”

При натисканні на кнопку “Видалити контакт” викликається функція delete\_contact(self) рис(2.21)

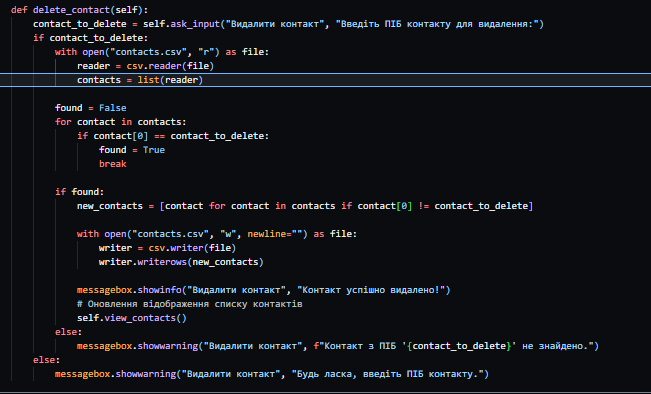


Рисунок 2.21. – Код функції delete\_contact(self)

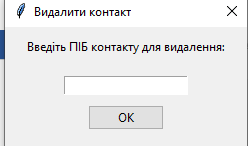


Рисунок 2.22 – Запит на ініціали контакту для видалення

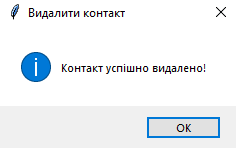


Рисунок 2.23 – Повідомлення про успішне видалення контакту

Детальний розбір всіх елементів у рядках функції:

1. Користувачу пропонується ввести ПІБ контакту, який вони бажають видалити.

2. Якщо користувач ввів дані, код продовжується.

3. Відкриває файл `contacts.csv` для читання.

4. Зчитує вміст CSV-файлу у список `contacts`.

5. Перевіряє, чи є контакт із введеним ПІБ у списку `contacts`.

6. Якщо контакт знайдено, він видаляється зі списку `contacts`.

7. Створюється новий список `new\_contacts`, в якому відсутній видалений контакт.

8. Оновлений список `new\_contacts` записується у файл `contacts.csv`, перезаписуючи попередні дані.

9. Виводиться повідомлення про успішне видалення контакту за допомогою `messagebox.showinfo`.

10. Якщо контакт не знайдено, показується повідомлення про це за допомогою `messagebox.showwarning`.

11. Якщо користувач не ввів ПІБ контакту, також показується відповідне повідомлення.

Цей метод також викликає метод `self.view\_contacts()`, який відповідає за оновлення відображення списку контактів після видалення одного з них.

На черзі розробка функція “Пошук контактів”:

При натисканні на кнопку “ Пошук контактів ” викликається функція search\_contact(self) рис(2.24)

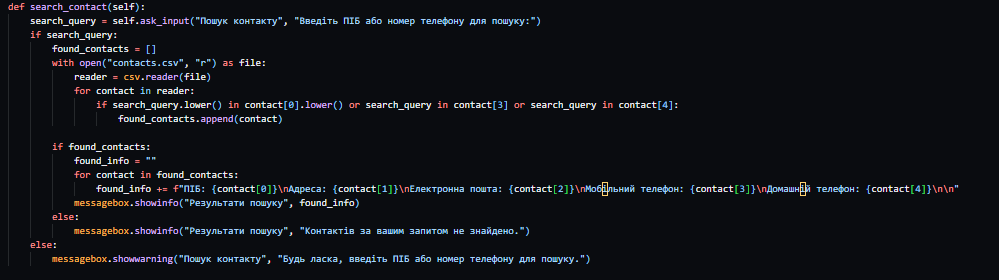


Рисунок 2.24 – Код функції search\_contact(self)

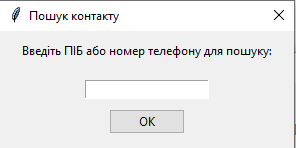


Рисунок 2.25 -Запит на пошут контакту по ініціалам або номеру телефону

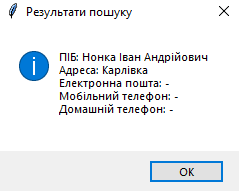


Рисунок 2.26 -Повідомлення про успіх пошуку контакту

Розглянемо застереження програми функції:

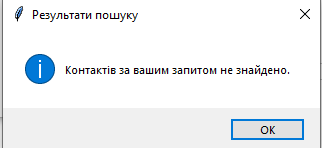
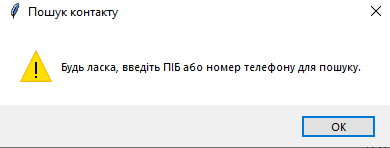


Рисунок 2.27 – Повідомлення про невдалий пошук, через те, що в базі даних не знайдено контакту за даними ініціалами

  
Рисунок 2.27 – Повідомлення про невдалий пошук, так як не було введено ініціали або номер телефону контакту

Детальний розбір всіх рядків функції:

1. Користувачу пропонується ввести запит для пошуку (ПІБ або номер телефону).
2. Якщо користувач ввів запит, код продовжується.
3. Створюється порожній список found\_contacts, в якому будуть зберігатися знайдені контакти.
4. Відкривається файл contacts.csv для читання.
5. Перебираються всі рядки у файлі CSV.
6. Перевіряється, чи міститься введений запит у першому елементі (ПІБ) або третьому чи четвертому елементі (мобільний або домашній телефон) кожного контакту.
7. Якщо контакт відповідає критеріям пошуку, він додається до списку found\_contacts.
8. Якщо знайдено контакти, їхні дані форматуються у рядок found\_info.
9. Відображається вікно з результатами пошуку, в якому відображається інформація про знайдені контакти.
10. Якщо контакти за введеним запитом не знайдено, відображається повідомлення про це.
11. Якщо користувач не ввів запит для пошуку, показується відповідне повідомлення.

Розглянемо розробку функції “Перегляд контактів”

При натисканні на кнопку “ Перегляд контактів ” викликається функція view\_contacts(self) рис(2.28)

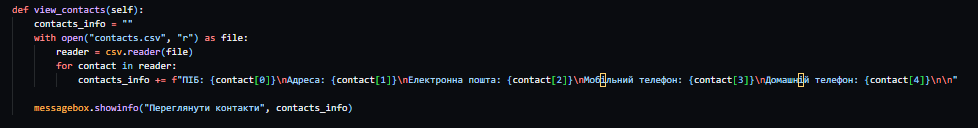


Рисунок 2.28 – Код функції view\_contacts(self)

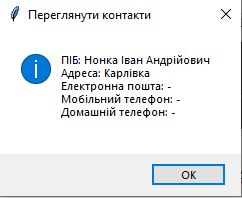


Рисунок 2.29 – При натисненні на кнопку “Переглянути контакти”, виводиться повідомлення про контакти, які містяться у базі даних

Розглянемо останню функцію у завданні:  
При натисканні на кнопку “Довідка ” викликається функція display\_help(self) рис(2.30)

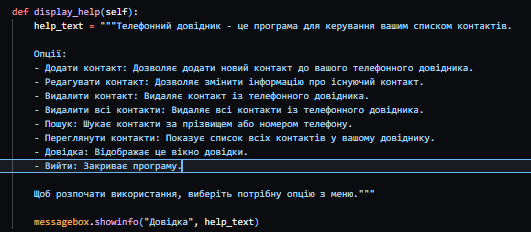


Рисунок 2.30 – Код функції display\_help(self)

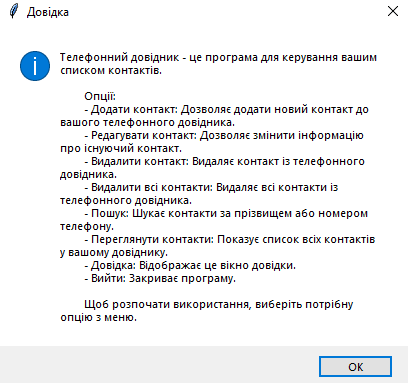


Рисунок 2.31 -Результат взаємодії з кнопкою “Довідка”

Детальний розбір всіх рядків функції:

Цей код реалізує метод `view\_contacts`, який відповідає за відображення інформації про всі контакти з файлу `contacts.csv`. Ось як він працює:1. 1.Створюється порожній рядок `contacts\_info`, який буде містити інформацію про всі контакти.

2. Відкривається файл `contacts.csv` для читання.

3. Створюється об'єкт читача `reader`, який дозволяє почергово читати рядки з CSV-файлу.

4. Перебираються всі рядки у файлі CSV.

5. Для кожного рядка створюється рядок інформації про контакт, який додається до змінної `contacts\_info`. У цьому рядку вказується ім'я, адреса, електронна пошта, мобільний телефон та домашній телефон контакту, розділені новими рядками для кращої читабельності.

6. Після обробки всіх контактів, відображається вікно з інформацією про всі контакти за допомогою `messagebox.showinfo`. Заголовок вікна - "Переглянути контакти", а текст - інформація про всі контакти, яка зберігається у змінній `contacts\_info`.

Цей метод дозволяє користувачу переглянути всі контакти у зручному форматі.

Тепер розглянемо додаткову функцію під назвою “Видалення всіх контактів”:

При натисканні на кнопку “Видалення всіх контактів” викликається функція delete\_all\_contacts(self) рис(2.32)

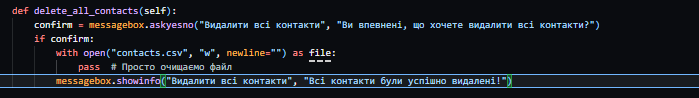


Рисунок 2.32 – Код функції delete\_all\_contacts(self)

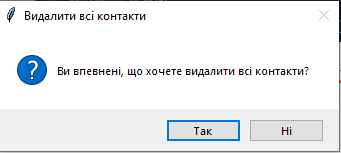
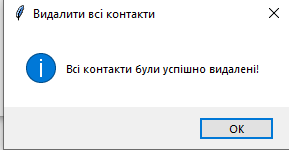


Рисунок 2.33 – Підтвердження про видалення усіх контактів



2.34 – Повідомлення про видалення всіх контактів

Детальний розбір всіх рядків функції:

1. Показується вікно підтвердження за допомогою messagebox.askyesno, в якому користувачу ставиться питання про підтвердження видалення всіх контактів.
2. Якщо користувач підтверджує видалення (натискає "Так"), змінна confirm буде містити True, і код продовжується.
3. Відкривається файл contacts.csv для запису з параметром w, що дозволяє очистити файл.
4. Фактично, в цьому випадку, у тілі with-блоку немає жодного коду, оскільки за допомогою pass просто пропускається будь-який код. Очищення файлу відбувається саме через відкриття файлу з режимом запису (який очищає файл) і одразу його закриття.
5. Після видалення всіх контактів користувачу показується повідомлення про успішне видалення за допомогою messagebox.showinfo.

Цей метод забезпечує можливість користувачеві безпечно видалити всі контакти, якщо вони більше не потрібні.

**ВИСНОВКИ**

Пояснювальна записка для цього завдання містить пояснення особливостей програми телефонної книги, методів, описаних у коді, її призначення та принцип роботи.

Було розглянуто архітектуру програми, яка використовує об’єктно-орієнтований підхід до керування контактними даними за допомогою класів і методів.

Завдяки цій курсовій роботі були вивчені важливі аспекти операцій з файлами, наприклад читання та запис даних із файлів CSV.

Також важливо підкреслити інтерфейс взаємодії з користувачем, особливо вікна підтвердження та інформаційні вікна, які забезпечують зручність і зрозумілість взаємодії з програмою.

Для більш детального та закріплюючого ефекту вивчення дисципліни можна вдосконалити програму, шляхом зміни бібліотек графічного інтерфейсу на складніші та додати нових функцій

У результаті виконання курсової роботи було успішно розроблено програма телефонного довідника, яка може бути корисною для керування контактами. Робота над проектом дозволила отримати досвід роботи з мовою програмування Python та поглибити знання з розробки програмного забезпечення.

**Список використаних джерел**

1.https://dystosvita.org.ua/mod/page/view.php?id= **Графічний інтерфейс користувача (tkinter)**

2. <https://docs.python.org/uk/3/library/csv.html>  **— CSV File Reading and Writing**

3.<https://programer.in.ua/index.php/uroky/uroky-python> Відеоуроки по Python

4. <https://replace.org.ua/forum/21/> україномовний форум з програмування, веб-дизайну, SEO та всього пов'язаного з інтернетом та комп'ютерами.

5. <https://replit.com/~> Середовище розробки програмного забезпечення (застосовувалось як компілятор для тестування проєкту)

Додаток А. Програмний код тестового додатку

import csv

import tkinter as tk

from tkinter import messagebox, simpledialog

from tkinter import ttk

class ContactBookApp:

    def \_\_init\_\_(self, master):

        self.master = master

        master.title("Телефонний довідник")

        master.geometry("500x350")

        self.frame = tk.Frame(master)

        self.frame.pack(pady=20)

        self.label = tk.Label(self.frame, text="Вітаю! Оберіть функцію для використання:", font=("Helvetica", 16))

        self.label.grid(row=0, column=0, columnspan=2, pady=10)

        self.style = ttk.Style()

        self.style.configure('Black.TButton', foreground='black', borderwidth=2, highlightthickness=2)

        self.add\_button = ttk.Button(self.frame, text="Додати контакт", command=self.add\_contact, style='Black.TButton')

        self.add\_button.grid(row=1, column=0, pady=5, padx=10, ipadx=10, sticky="EW")

        self.edit\_button = ttk.Button(self.frame, text="Редагувати контакт", command=self.edit\_contact, style='Black.TButton')

        self.edit\_button.grid(row=1, column=1, pady=5, padx=10, ipadx=10, sticky="EW")

        self.delete\_button = ttk.Button(self.frame, text="Видалити контакт", command=self.delete\_contact, style='Black.TButton')

        self.delete\_button.grid(row=2, column=0, pady=5, padx=10, ipadx=10, sticky="EW")

        self.delete\_all\_button = ttk.Button(self.frame, text="Видалити всі контакти", command=self.delete\_all\_contacts, style='Black.TButton')

        self.delete\_all\_button.grid(row=2, column=1, pady=5, padx=10, ipadx=10, sticky="EW")

        self.search\_button = ttk.Button(self.frame, text="Пошук", command=self.search\_contact, style='Black.TButton')

        self.search\_button.grid(row=3, column=0, pady=5, padx=10, ipadx=10, sticky="EW")

        self.view\_button = ttk.Button(self.frame, text="Переглянути контакти", command=self.view\_contacts, style='Black.TButton')

        self.view\_button.grid(row=3, column=1, pady=5, padx=10, ipadx=10, sticky="EW")

        self.help\_button = ttk.Button(self.frame, text="Довідка", command=self.display\_help, style='Black.TButton')

        self.help\_button.grid(row=4, column=0, columnspan=2, pady=10, ipadx=10, sticky="EW")

        self.quit\_button = ttk.Button(self.frame, text="Вийти", command=master.quit, style='Black.TButton')

        self.quit\_button.grid(row=5, column=0, columnspan=2, pady=5, ipadx=10, sticky="EW")

        self.footer\_label = tk.Label(master, text="© 2024, Created by Deadlight, Версія: 1.0", font=("Helvetica", 10))

        self.footer\_label.pack(pady=10)

    def ask\_input(self, title, prompt):

        dialog = tk.Toplevel(self.master)

        dialog.transient(self.master)

        dialog.grab\_set()

        dialog.title(title)

        label = tk.Label(dialog, text=prompt, padx=20, pady=10)

        label.pack()

        entry = tk.Entry(dialog)

        entry.pack(padx=20, pady=10)

        entry.focus()

        def on\_ok():

            self.master.response = entry.get()

            dialog.destroy()

        ok\_button = ttk.Button(dialog, text="OK", command=on\_ok)

        ok\_button.pack(pady=(0, 20))

        self.master.wait\_window(dialog)

        return getattr(self.master, 'response', None)

    def add\_contact(self):

        name = self.ask\_input("Додати контакт", "Введіть ПІБ:")

        if name is not None:

            address = self.ask\_input("Додати контакт", "Введіть адресу:")

            email = self.ask\_input("Додати контакт", "Введіть електронну пошту:")

            mobile\_phone = self.ask\_input("Додати контакт", "Введіть мобільний телефон:")

            home\_phone = self.ask\_input("Додати контакт", "Введіть домашній телефон:")

            with open("contacts.csv", "a", newline="") as file:

                writer = csv.writer(file)

                writer.writerow([name, address, email, mobile\_phone, home\_phone])

            messagebox.showinfo("Додати контакт", "Контакт успішно додано!")

    def edit\_contact(self):

        contact\_to\_edit = self.ask\_input("Редагувати контакт", "Введіть ПІБ контакту для редагування:")

        if contact\_to\_edit:

            with open("contacts.csv", "r") as file:

                reader = csv.reader(file)

                contacts = list(reader)

            for index, contact in enumerate(contacts):

                if contact[0] == contact\_to\_edit:

                    new\_name = self.ask\_input("Редагувати контакт", "Введіть нове ПІБ:")

                    if new\_name is not None:

                        new\_address = self.ask\_input("Редагувати контакт", "Введіть нову адресу:")

                        new\_email = self.ask\_input("Редагувати контакт", "Введіть нову електронну пошту:")

                        new\_mobile\_phone = self.ask\_input("Редагувати контакт", "Введіть новий мобільний телефон:")

                        new\_home\_phone = self.ask\_input("Редагувати контакт", "Введіть новий домашній телефон:")

                        contacts[index] = [new\_name, new\_address, new\_email, new\_mobile\_phone, new\_home\_phone]

                        with open("contacts.csv", "w", newline="") as file:

                            writer = csv.writer(file)

                            writer.writerows(contacts)

                        messagebox.showinfo("Редагувати контакт", "Контакт успішно відредаговано!")

                        return

            messagebox.showwarning("Редагувати контакт", "Контакт з таким ПІБ не знайдено.")

        else:

            messagebox.showwarning("Редагувати контакт", "Будь ласка, введіть ПІБ контакту.")

    def delete\_contact(self):

        contact\_to\_delete = self.ask\_input("Видалити контакт", "Введіть ПІБ контакту для видалення:")

        if contact\_to\_delete:

            with open("contacts.csv", "r") as file:

                reader = csv.reader(file)

                contacts = list(reader)

            found = False

            for contact in contacts:

                if contact[0] == contact\_to\_delete:

                    found = True

                    break

            if found:

                new\_contacts = [contact for contact in contacts if contact[0] != contact\_to\_delete]

                with open("contacts.csv", "w", newline="") as file:

                    writer = csv.writer(file)

                    writer.writerows(new\_contacts)

                messagebox.showinfo("Видалити контакт", "Контакт успішно видалено!")

                # Оновлення відображення списку контактів

                self.view\_contacts()

            else:

                messagebox.showwarning("Видалити контакт", f"Контакт з ПІБ '{contact\_to\_delete}' не знайдено.")

        else:

            messagebox.showwarning("Видалити контакт", "Будь ласка, введіть ПІБ контакту.")

    def delete\_all\_contacts(self):

        confirm = messagebox.askyesno("Видалити всі контакти", "Ви впевнені, що хочете видалити всі контакти?")

        if confirm:

            with open("contacts.csv", "w", newline="") as file:

                pass  # Просто очищаємо файл

            messagebox.showinfo("Видалити всі контакти", "Всі контакти були успішно видалені!")

    def search\_contact(self):

        search\_query = self.ask\_input("Пошук контакту", "Введіть ПІБ або номер телефону для пошуку:")

        if search\_query:

            found\_contacts = []

            with open("contacts.csv", "r") as file:

                reader = csv.reader(file)

                for contact in reader:

                    if search\_query.lower() in contact[0].lower() or search\_query in contact[3] or search\_query in contact[4]:

                        found\_contacts.append(contact)

            if found\_contacts:

                found\_info = ""

                for contact in found\_contacts:

                    found\_info += f"ПІБ: {contact[0]}\nАдреса: {contact[1]}\nЕлектронна пошта: {contact[2]}\nМобільний телефон: {contact[3]}\nДомашній телефон: {contact[4]}\n\n"

                messagebox.showinfo("Результати пошуку", found\_info)

            else:

                messagebox.showinfo("Результати пошуку", "Контактів за вашим запитом не знайдено.")

        else:

            messagebox.showwarning("Пошук контакту", "Будь ласка, введіть ПІБ або номер телефону для пошуку.")

    def view\_contacts(self):

        contacts\_info = ""

        with open("contacts.csv", "r") as file:

            reader = csv.reader(file)

            for contact in reader:

                contacts\_info += f"ПІБ: {contact[0]}\nАдреса: {contact[1]}\nЕлектронна пошта: {contact[2]}\nМобільний телефон: {contact[3]}\nДомашній телефон: {contact[4]}\n\n"

        messagebox.showinfo("Переглянути контакти", contacts\_info)

    def display\_help(self):

        help\_text = """Телефонний довідник - це програма для керування вашим списком контактів.

        Опції:

        - Додати контакт: Дозволяє додати новий контакт до вашого телефонного довідника.

        - Редагувати контакт: Дозволяє змінити інформацію про існуючий контакт.

        - Видалити контакт: Видаляє контакт із телефонного довідника.

        - Видалити всі контакти: Видаляє всі контакти із телефонного довідника.

        - Пошук: Шукає контакти за прізвищем або номером телефону.

        - Переглянути контакти: Показує список всіх контактів у вашому довіднику.

        - Довідка: Відображає це вікно довідки.

        - Вийти: Закриває програму.

        Щоб розпочати використання, виберіть потрібну опцію з меню."""

        messagebox.showinfo("Довідка", help\_text)

def main():

    root = tk.Tk()

    app = ContactBookApp(root)

    root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

    main()