

TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP
KHOA ĐIỆN TỬ

Bộ môn: Công nghệ Thông tin



BÀI TẬP KẾT THÚC MÔN HỌC
MÔN HỌC
LẬP TRÌNH PYTHON

SINH VIÊN : ĐẬU VĂN KHÁNH
MSSV : K225480106099
LỚP : K58KTP
GIÁO VIÊN GIẢNG DẠY : TS. NGUYỄN VĂN HUY

Thái Nguyên - 2025

TRƯỜNG ĐHKTCN

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM

KHOA ĐIỆN TỬ

Độc lập - Tự do - Hạnh phúc

BÀI TẬP KẾT THÚC MÔN HỌC

MÔN HỌC: LẬP TRÌNH PYTHON

BỘ MÔN : CÔNG NGHỆ THÔNG TIN

Sinh viên: Đậu Văn Khánh

MSSV: K225480106099

Lớp: K58KTP

Ngành: Kỹ thuật máy tính

Giáo viên hướng dẫn: TS. Nguyễn Văn Huy

Ngày giao đề: 20/05/2025

Ngày hoàn thành: 09/06/2025

Tên đề tài : Xây ứng dụng quản lý danh bạ GUI đơn giản: thêm, sửa, xóa liên hệ với name, phone, email, lưu vào file JSON.

Yêu cầu :

1. Đầu vào – đầu ra:

- Đầu vào: Form nhập name, phone, email.*
- Đầu ra: Table hiển thị danh sách, lưu file contacts.json.*

2. Tính năng yêu cầu:

- Đọc/ghi JSON với module json.*
- Bắt lỗi format JSON, lưu khi thêm/sửa/xóa.*
- GUI: Entry, Buttons, Treeview.*
- Tìm kiếm theo tên.*

3. Kiểm tra & kết quả mẫu:

- Thêm “Nguyễn”, “0123” → xuất hiện trong Table.*
- Sửa số → file JSON cập nhật.*

GIÁO VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Thái Nguyên, ngày....tháng.....năm 2025

GIÁO VIÊN HƯỚNG DẪN

(Ký ghi rõ họ tên)

LỜI CẢM ƠN

Em xin chân thành cảm ơn thầy **Nguyễn Văn Huy** – người đã tận tình hướng dẫn, hỗ trợ và tạo điều kiện thuận lợi để em thực hiện đề tài “Xây ứng dụng quản lý danh bạ GUI đơn giản: thêm, sửa, xoá liên hệ với name, phone, email, lưu vào file JSON”.

Trong quá trình thực hiện đề tài, thầy đã truyền đạt cho em không chỉ những kiến thức chuyên môn bổ ích về lập trình giao diện người dùng (GUI), quản lý dữ liệu bằng JSON mà còn cả những kỹ năng tư duy logic, làm việc độc lập và giải quyết vấn đề thực tiễn. Sự chỉ dẫn tận tâm, nghiêm túc của Thầy là nguồn động lực lớn giúp em hoàn thiện sản phẩm đúng tiến độ và đạt được những kết quả tích cực.

Em cũng xin gửi lời cảm ơn đến các thầy, cô trong khoa Điện Tử đã tạo điều kiện và môi trường học tập tốt để em có thể ứng dụng kiến thức vào thực tiễn một cách hiệu quả.

Một lần nữa, em xin gửi lời tri ân sâu sắc đến thầy và kính chúc thầy luôn mạnh khỏe, thành công trong sự nghiệp giảng dạy và nghiên cứu.

Em xin chân thành cảm ơn!

MỤC LỤC

LỜI CẢM ƠN	3
DANH MỤC HÌNH ẢNH	5
LỜI NÓI ĐẦU	6
CHƯƠNG 1: GIỚI THIỆU	7
1.1. Giới thiệu ngôn ngữ lập trình Python	7
1.2. Giới thiệu lập trình GUI bằng Tkinter trong python	9
1.3. Giới thiệu đề tài	10
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	12
2.1. Ngôn ngữ lập trình sử dụng trong chương trình	12
2.2. Giao diện người dùng với Tkinter	12
2.3. Kiểu dữ liệu list (danh sách)	12
2.4. Kiểu dữ liệu Dictionary (từ điển)	13
2.5. Xử lý tệp JSON để lưu trữ dữ liệu	14
2.6. Các thao tác CRUD	14
2.7. Tìm kiếm liên hệ	14
2.8. Xử lý sự kiện và tương tác GUI	15
2.9. Lập trình hướng đối tượng (OOP)	15
CHƯƠNG 3: THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH	16
3.1. Sơ đồ khối hệ thống	16
3.2. Sơ đồ khối các thuật toán chính	20
3.3. Cấu trúc dữ liệu	24
3.4. Chương trình	24
CHƯƠNG 4: THỰC NGHIỆM VÀ KẾT LUẬN	28
4.1. Thực nghiệm	28
4.2. Kết luận	32
TÀI LIỆU THAM KHẢO	34

DANH MỤC HÌNH ẢNH

Hình 1.1: Ngôn ngữ lập trình Python

Hình 1.2: Một số đặc điểm của Python

Hình 1.3: Ứng dụng ngôn ngữ Python

Hình 2.1: Kiểu dữ liệu list được sử dụng trong chương trình

Hình 2.2: Hiển thị danh sách liên hệ

Hình 3.1: Biểu đồ phân cấp chức năng

Hình 3.2: Sơ đồ thuật toán thêm liên hệ

Hình 3.3: Sơ đồ thuật toán sửa liên hệ

Hình 3.4: Sơ đồ thuật toán xóa liên hệ

Hình 3.5: Sơ đồ thuật toán tìm kiếm liên hệ

Hình 4.1: Giao diện chính của chương trình

Hình 4.2: Giao diện khi thêm liên hệ

Hình 4.3: Giao diện khi sửa liên hệ

Hình 4.4: Giao diện khi tìm kiếm liên hệ

Hình 4.5: Giao diện khi xóa liên hệ

Hình 4.6: Hiển thị lỗi khi thêm liên hệ

Hình 4.7: Hiển thị lỗi khi sửa thông tin liên hệ

Hình 4.8: Hiển thị lỗi khi xóa liên hệ

Hình 4.9: Dữ liệu trong file contacts.json

LỜI NÓI ĐẦU

Trong thời đại công nghệ số hiện nay, việc lưu trữ và quản lý thông tin liên hệ một cách khoa học, thuận tiện và hiệu quả là nhu cầu thiết yếu của cá nhân cũng như tổ chức. Với sự phát triển mạnh mẽ của các ngôn ngữ lập trình hiện đại, đặc biệt là Python – một ngôn ngữ mạnh mẽ, linh hoạt và dễ tiếp cận, việc xây dựng các ứng dụng quản lý đơn giản nhưng hữu ích trở nên khả thi hơn bao giờ hết.

Đề tài “*Xây ứng dụng quản lý danh bạ GUI đơn giản: thêm, sửa, xóa liên hệ với name, phone, email, lưu vào file JSON*” được thực hiện nhằm giúp người học hiểu rõ cách xây dựng một chương trình có giao diện người dùng (GUI) đơn giản để quản lý danh bạ bao gồm các thao tác cơ bản như thêm, sửa, xóa và tìm kiếm liên hệ. Dữ liệu danh bạ được lưu trữ dưới dạng file JSON, giúp chương trình có thể dễ dàng lưu trữ và khôi phục thông tin khi cần thiết. Đồng thời, ứng dụng còn tích hợp các chức năng kiểm tra và xử lý lỗi định dạng dữ liệu để đảm bảo tính ổn định trong quá trình sử dụng.

Thông qua đề tài này, sinh viên có cơ hội tiếp cận và rèn luyện các kỹ năng lập trình hướng đối tượng, thao tác với file JSON, xử lý sự kiện trong giao diện đồ họa bằng thư viện Tkinter, cũng như áp dụng tư duy logic trong việc thiết kế và tổ chức mã nguồn theo mô-đun. Đây là một bước đệm quan trọng giúp sinh viên chuẩn bị tốt hơn cho các dự án lớn hơn trong lĩnh vực phát triển phần mềm.

Em xin chân thành cảm ơn sự hướng dẫn và hỗ trợ tận tình của giảng viên Nguyễn Văn Huy, cùng những ý kiến đóng góp quý báu của các bạn sinh viên trong quá trình hoàn thành đề tài này.

CHƯƠNG 1: GIỚI THIỆU

1.1. Giới thiệu ngôn ngữ lập trình Python

1.1.1. Khái niệm

Python là một ngôn ngữ lập trình thông dịch, dễ đọc và dễ hiểu. Nền tảng nổi tiếng với cú pháp đơn giản và được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau. Điển hình như phát triển các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học (ML).

Python có cú pháp linh hoạt và cấu trúc dữ liệu mạnh mẽ, công nghệ được hỗ trợ bởi một cộng đồng lớn. Điều này đã mang đến các thư viện và framework phong phú mà người dùng có thể sử dụng để xây dựng các ứng dụng phức tạp. Python cũng là một trong những ngôn ngữ phổ biến cho người mới học lập trình nhờ vào tính linh hoạt của nó.

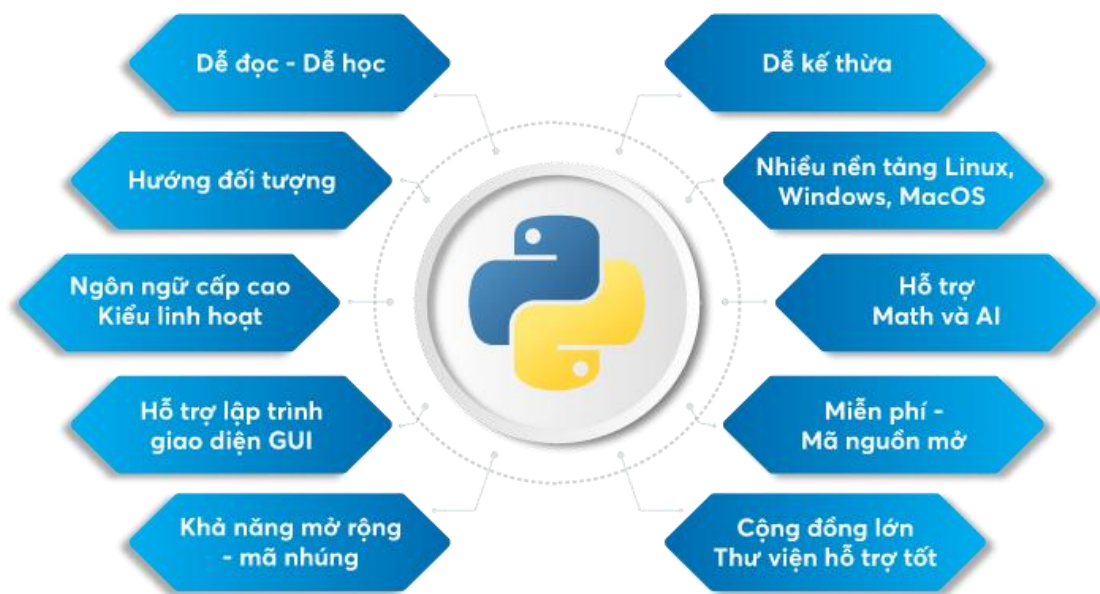


Hình 1.1: Ngôn ngữ lập trình Python

1.1.2. Đặc điểm của Python

- Python là một ngôn ngữ thông dịch: Điều này có nghĩa là ngôn ngữ trực tiếp chạy từng dòng mã. Nếu có lỗi mã trong chương trình nó sẽ ngừng chạy. Điều này giúp lập trình viên nhanh chóng tìm ra lỗi trong đoạn mã.
- Python là một ngôn ngữ dễ sử dụng: từ ngữ Python sử dụng giống trong tiếng Anh. Không như các ngôn ngữ lập trình khác, Python sử dụng thụt đầu dòng thay vì sử dụng dấu ngoặc ôm.

- Python là một ngôn ngữ linh hoạt: Đây là một đặc điểm giúp cho việc viết các chương trình Python nhanh chóng. Các lập trình viên không cần phải khai báo loại biến khi viết mã, vì Python sẽ xác định chúng vào thời điểm chạy.
- Python là một ngôn ngữ cấp cao: So với các ngôn ngữ lập trình khác thì Python gần gũi với ngôn ngữ con người hơn. Vì thế các lập trình viên không cần quá lo lắng về những chức năng cơ bản như kiến trúc và quản lý bộ nhớ.
- Python là ngôn ngữ lập trình hướng đối tượng: Điều này nghĩa là Python coi mọi thứ đều là đối tượng nhưng ngôn ngữ này cũng hỗ trợ các phương thức lập trình khác như lập trình cấu trúc và lập trình hàm.



Hình 1.2: Một số đặc điểm của Python

1.1.3. Một số ứng dụng của Python

❖ Lập trình ứng dụng web

Để tạo web app có khả năng mở rộng bằng cách sử dụng framework và hệ thống quản trị nội dung (CMS) được tích hợp trong Python. Một số nền tảng phổ biến tạo ra web app như: Flask, Plone, Pyramid, Instagram,... đều được viết bằng Python.

❖ Khoa học và tính toán

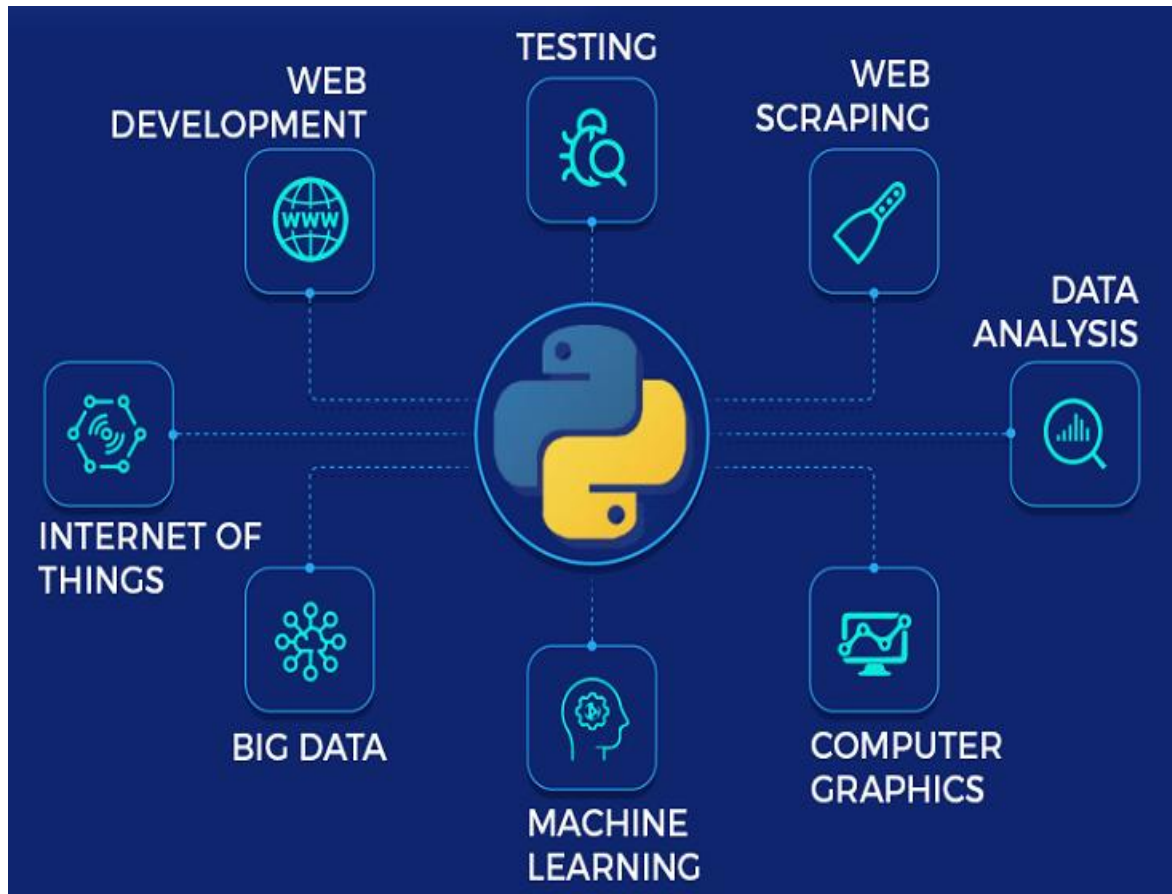
Nhiều thư viện trong Python cho khoa học và tính toán số liệu như SciPy và NumPy dùng cho mục đích tính toán. Bên cạnh đó còn có thư viện cho thiên văn học như: AstroPy, EarthPy,...

❖ Tạo nguyên mẫu phần mềm

Python là ngôn ngữ tuyệt vời để tạo những prototype. Ví dụ như có thể sử dụng Pygame để tạo nguyên mẫu game trước.

❖ Ngôn ngữ để dạy lập trình

Với những tính năng ứng dụng hữu ích và cú pháp đơn giản, dễ hiểu nên Python được nhiều trường học, trung tâm đào tạo lựa chọn làm ngôn ngữ để giảng dạy. Các ngành mũi nhọn của công nghệ thông tin như: trí tuệ nhân tạo (AI), dữ liệu lớn (big data), IoT... đang phát triển rất nhanh vì thế ngôn ngữ viết code Python là một lựa chọn phù hợp nhất để bắt đầu.



Hình 1.3: Ứng dụng ngôn ngữ Python

1.2. Giới thiệu lập trình GUI bằng Tkinter trong python

1.2.1. Giới thiệu GUI

GUI (Graphic user interface) hay còn gọi là giao diện người dùng đồ họa được sử dụng vô cùng phổ biến trong máy tính, các thiết bị đa phương tiện,... Sự hiện diện của GUI ngày càng gia tăng trong kỷ nguyên số, khi các thiết bị số trở thành một phần không thể thay thế trong đời sống con người.

1.2.2. Giới thiệu Python Tkinter

Tkinter là thư viện GUI tiêu chuẩn cho Python. Khi kết hợp với Tkinter, Python sẽ được cung cấp các công cụ một cách nhanh chóng và dễ dàng để tạo các ứng dụng GUI. Tkinter cung cấp giao diện hướng đối tượng mạnh mẽ đến các bộ công cụ Tk GUI.

1.2.3. Các Widgets của Tkinter Python

Tkinter cung cấp nhiều bảng điều khiển khác nhau được sử dụng trong một ứng dụng GUI như các nút, nhãn và hộp kiểm,... Những bảng điều khiển này thường được gọi là widget.

Một số kiểu widget trong Tkinter:

- Button: Tiện ích Button được sử dụng để hiển thị các nút trong ứng dụng
- Entry: được sử dụng để hiển thị trường văn bản một dòng để chấp nhận các giá trị từ người dùng.
- Canvas: Sử dụng để vẽ các hình dạng, chẳng hạn như đường thẳng, hình bầu dục, đa giác và hình chữ nhật, trong ứng dụng của bạn.
- Checkbutton: sử dụng để hiển thị một số tùy chọn dưới dạng hộp kiểm. Người dùng có thể chọn nhiều tùy chọn cùng một lúc.
- Frame: được sử dụng như một widget vùng chứa để sắp xếp các widget khác.
- Label: được sử dụng để cung cấp chú thích một dòng cho các tiện ích con khác. Nó cũng có thể chứa hình ảnh.
- Menu: được sử dụng để cung cấp các lệnh khác nhau cho người dùng. Các lệnh này được chứa bên trong Menubutton.
- Listbox: được sử dụng để cung cấp danh sách các tùy chọn cho người dùng.
- Text: được sử dụng để hiển thị văn bản trong nhiều dòng.

1.3. Giới thiệu đề tài

1.3.1. Tên đề tài

Xây ứng dụng quản lý danh bạ đơn giản: thêm, sửa, xóa liên hệ với name, phone, email, lưu vào file JSON.

1.3.2. Mô tả đề tài

Đề tài "Xây ứng dụng quản lý danh bạ đơn giản: thêm, sửa, xóa liên hệ với name, phone, email, lưu vào file JSON" nhằm phát triển một phần mềm đơn giản, thân thiện, cho phép người dùng quản lý thông tin liên hệ cá nhân. Ứng dụng hỗ trợ các chức năng cơ bản như: thêm, sửa, xóa và tìm kiếm liên hệ theo tên, với mỗi liên hệ gồm tên, số điện thoại và email.

Giao diện người dùng được xây dựng bằng Tkinter – thư viện GUI chuẩn trong Python, giúp thao tác dễ dàng. Dữ liệu được lưu dưới dạng JSON, thuận tiện cho việc lưu trữ và mở rộng.

Đề tài giúp sinh viên rèn luyện kỹ năng lập trình Python, thiết kế giao diện, xử lý sự kiện, đọc ghi dữ liệu JSON và tổ chức chương trình theo hướng rõ ràng, dễ bảo trì. Đây là bài tập ứng dụng kiến thức tổng hợp, phù hợp với mục tiêu môn học.

1.3.3. Các tính năng chính của chương trình

- Thêm liên hệ mới với các trường: họ tên, số điện thoại, email.
- Sửa thông tin liên hệ đã có.
- Xóa liên hệ không còn cần thiết.
- Tìm kiếm liên hệ theo tên.
- Hiển thị danh sách liên hệ bằng bảng Treeview.
- Tự động lưu dữ liệu vào file JSON sau mỗi thao tác thêm, sửa, xóa.
- Bắt lỗi định dạng JSON khi đọc dữ liệu từ file, đảm bảo ứng dụng không bị lỗi hoặc dừng đột ngột.

1.3.4. Những thách thức trong quá trình xây dựng

- Xử lý các lỗi định dạng dữ liệu JSON, đảm bảo tính ổn định và liên tục của ứng dụng.
- Thiết kế giao diện đơn giản, dễ sử dụng nhưng vẫn đảm bảo đầy đủ chức năng.
- Đồng bộ dữ liệu giữa bảng hiển thị (Treeview) và dữ liệu lưu trữ trong file JSON.

1.3.5. Kiến thức vận dụng để thực hiện đề tài

- Lập trình hướng đối tượng (OOP) trong Python để tổ chức mã nguồn gọn gàng, dễ bảo trì.
- Xử lý sự kiện và ràng buộc dữ liệu, liên kết giữa các thành phần GUI và logic xử lý phía sau.
- Xây dựng giao diện người dùng với thư viện Tkinter, sử dụng các thành phần như Label, Entry, Button, Treeview.
- Thao tác file JSON với thư viện json để lưu trữ và xử lý dữ liệu.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Ngôn ngữ lập trình sử dụng trong chương trình

Chương trình sử dụng ngôn ngữ Python, là một ngôn ngữ lập trình thông dịch, đa mục đích, được thiết kế với cú pháp rõ ràng, dễ đọc và dễ học. Python hỗ trợ lập trình hướng đối tượng, thủ tục và hàm. Trong bài này, Python được sử dụng để xử lý dữ liệu danh bạ và xây dựng giao diện người dùng với thư viện Tkinter.

2.2. Giao diện người dùng với Tkinter

Tkinter là thư viện GUI tiêu chuẩn đi kèm với Python, cho phép tạo các cửa sổ, nút, nhãn, hộp nhập liệu và các thành phần đồ họa khác.

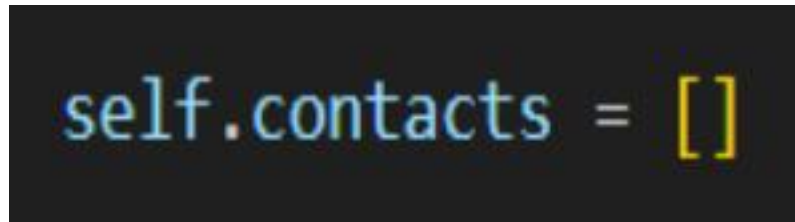
Một số thành phần chính sử dụng trong ứng dụng:

- Tk(): Tạo cửa sổ chính của ứng dụng.
- Label, Entry: Dùng để hiển thị nhãn và nhận dữ liệu nhập từ người dùng (họ tên, số điện thoại, email).
- Button: Các nút bấm để thực hiện các chức năng như Thêm, Sửa, Xóa, Tìm kiếm.
- Treeview (từ ttk): Là bảng dữ liệu giúp hiển thị các liên hệ theo dạng hàng, cột.
- messagebox: Dùng để hiển thị các hộp thoại cảnh báo, thông báo, hoặc xác nhận.

2.3. Kiểu dữ liệu list (danh sách)

- ❖ List là một kiểu dữ liệu thu thập (collection data) trong Python. List là một dạng danh sách có thể chứa nhiều phần tử các kiểu dữ liệu khác nhau, được phân tách bởi dấu phẩy “,” và được đặt trong cặp dấu ngoặc vuông []. Ví dụ: ['Apple', 'Mango', 'Cherry']. Phần tử của list không giới hạn có thể là bất kỳ kiểu dữ liệu từ cơ bản đến phức tạp. Để gán tên cho list, ta sử dụng toán tử “=”. Ví dụ: fruits = ['Apple', 'Mango', 'Cherry'].
- ❖ Một số đặc điểm nổi bật của List trong Python:
 - List có thể lưu trữ bất kỳ loại phần tử nào, bao gồm các kiểu dữ liệu cơ bản như số, chuỗi,...và các kiểu dữ liệu phức tạp hơn như list khác, tuple, dictionary, set,...
 - List được sắp xếp nhất định, các phần tử sẽ có một thứ tự xác định và sẽ không thay đổi. Nếu thêm phần tử mới vào list thì phần tử đó sẽ xuất hiện ở cuối danh sách.

- List có thể thay đổi được, có nghĩa là các phần tử trong list có thể được thêm, sửa đổi và xóa sau khi đã tạo ra.
- List cho phép các phần tử trùng lặp, có nghĩa là một giá trị có thể xuất hiện nhiều lần trong list.
- Các phần tử trong list được đánh chỉ mục (indexed), mỗi phần tử trong list được gán một chỉ số, bắt đầu từ 0. Bằng cách sử dụng chỉ số, bạn có thể truy cập vào các phần tử trong list.



Hình 2.1: Kiểu dữ liệu list được sử dụng trong chương trình

- Câu lệnh `self.contacts = []` để tạo 1 danh sách rỗng. Có nghĩa là khi bắt đầu chương trình, nếu chưa có bất kỳ dữ liệu nào thì khởi tạo danh sách rỗng.

2.4. Kiểu dữ liệu Dictionary (từ điển)

- ❖ Kiểu dữ liệu dictionary trong Python là một kiểu dữ liệu lưu trữ các giá trị chứa key và value, nhìn một cách tổng quát thì nó giống với **Json**. Và đối với kiểu dữ liệu này thì các giá trị bên trong nó không được sắp xếp theo một trật tự nào cả.
- ❖ Để khai báo một dictionary chúng ta sử dụng cặp dấu `{}` theo cú pháp sau:

`{key1: value1, key2: value2,..., keyN: valueN}`

Trong đó, `key1: value1, key2: value2,..., keyN: valueN` là các key và giá trị của kiểu dữ liệu dictionary. Và tên của key thì các bạn phải tuân thủ theo một số quy tắc sau:

- Các phần tử đều phải có key.
- Và Key chỉ có thể là số hoặc chuỗi.
- Key phải là duy nhất, nếu không nó sẽ nhận giá trị của phần tử có key được xuất hiện cuối cùng.
- Key khi đã được khai báo thì không thể đổi được tên.
- Key có phân biệt hoa thường.
- ❖ Dùng dictionary để biểu diễn từng liên hệ với các trường thông tin như họ tên, số điện thoại, email.

2.5. Xử lý tệp JSON để lưu trữ dữ liệu

2.5.1. Định dạng JSON

JSON (JavaScript Object Notation) là định dạng nhẹ để lưu trữ và truyền dữ liệu. Trong bài này, JSON được sử dụng để lưu và đọc danh sách liên hệ từ tệp `contacts.json`.

2.5.2. Các hàm xử lý JSON

- `json.load(file)`: Đọc dữ liệu từ tệp JSON và chuyển thành danh sách Python.
- `json.dump(data, file, indent=4, ensure_ascii=False)`: Ghi danh sách liên hệ ra tệp JSON.

2.6. Các thao tác CRUD

CRUD là viết tắt của Create, Read, Update, Delete – các thao tác cơ bản trong quản lý dữ liệu. Chương trình thực hiện đầy đủ cả 4 thao tác này.

2.6.1. Thêm liên hệ (Create)

- Dữ liệu từ các ô nhập được thu thập bằng hàm `get_form_data()`.
- Sau đó thêm vào danh sách và lưu vào tệp: `self.manager.add_contact(contact)`.

2.6.2. Hiển thị danh sách liên hệ (Read)

Danh bạ được tải và hiển thị trên Treeview bằng hàm `load_contacts()`:

```
for contact in self.manager.contacts:
    self.tree.insert('', tk.END, values=(contact["name"], contact["phone"], contact["email"]))
```

Hình 2.2: Hiển thị danh sách liên hệ

2.6.3. Cập nhật liên hệ (Update)

- Khi người dùng chọn một liên hệ, thông tin sẽ được đưa vào form.
- Sau khi chỉnh sửa, nhấn nút "Edit" để lưu lại thay đổi:

```
self.manager.update_contact(index, contact)
```

2.6.4. Xóa liên hệ (Delete)

- Chọn liên hệ và nhấn nút "Delete", sau đó liên hệ sẽ bị xóa khỏi danh sách:

```
self.manager.delete_contact(index)
```

2.7. Tìm kiếm liên hệ

- Dữ liệu người dùng nhập vào ô tìm kiếm sẽ được dùng để lọc danh bạ:

```
if name in contact["name"].lower():
    self.tree.insert('', tk.END, values=(contact["name"], contact["phone"], contact["email"]))
```

- Các kết quả phù hợp sẽ được hiển thị lại trên Treeview.

2.8. Xử lý sự kiện và tương tác GUI

- Ứng dụng có các sự kiện giúp tăng tính tương tác:
`self.tree.bind("<<TreeviewSelect>>", self.on_select)`
- Khi người dùng chọn một liên hệ trong bảng, thông tin của liên hệ đó sẽ được hiển thị trong các ô nhập liệu.
- Hàm `select_first()`: Dùng để chọn liên hệ đầu tiên khi cần, giúp người dùng thao tác nhanh hơn.

2.9. Lập trình hướng đối tượng (OOP)

- ❖ Chương trình được xây dựng theo mô hình hướng đối tượng:
 - `ContactManager`: quản lý dữ liệu liên hệ và thao tác với file JSON.
 - `ContactApp`: tạo giao diện và xử lý tương tác người dùng.
- ❖ Cách tổ chức này giúp chia tách rõ ràng giữa logic xử lý dữ liệu và giao diện hiển thị.

CHƯƠNG 3: THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH

3.1. Sơ đồ khối hệ thống

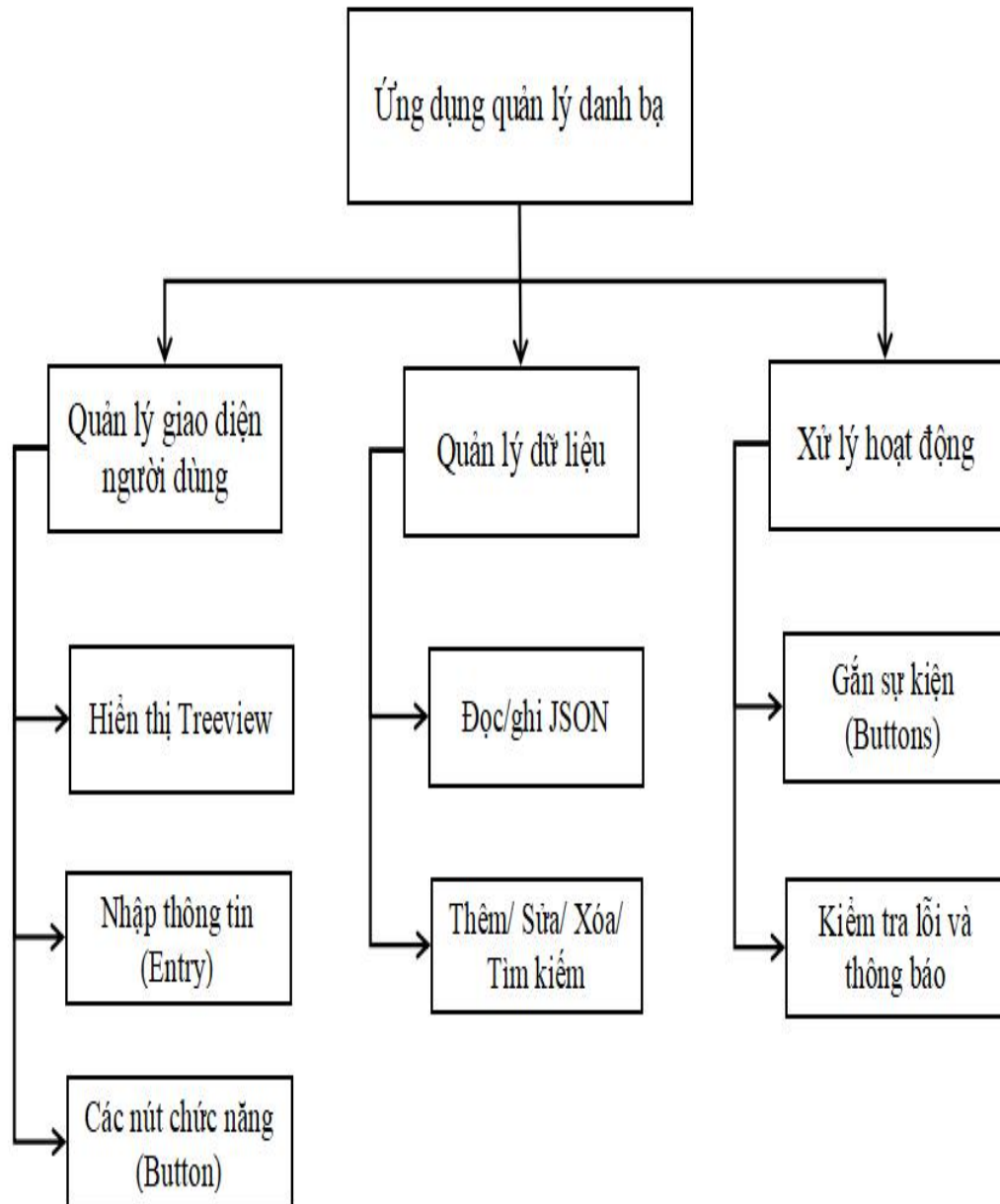
3.1.1. Mô tả các module chính

Ứng dụng "Quản lý danh bạ" được tổ chức thành ba module chính, nhằm tách biệt rõ ràng giữa các phần xử lý dữ liệu, giao diện người dùng và hoạt động xử lý sự kiện. Việc chia module như vậy giúp chương trình dễ mở rộng, dễ bảo trì và nâng cao tính linh hoạt trong phát triển sau này. Cụ thể:

- ❖ Module giao diện người dùng (GUI):
 - ✓ Đây là module chịu trách nhiệm tạo và quản lý toàn bộ giao diện đồ họa (Graphical User Interface) của ứng dụng. Giao diện được xây dựng bằng thư viện Tkinter – một thư viện tiêu chuẩn trong Python cho phát triển ứng dụng GUI đơn giản.
 - ✓ Chức năng chính:
 - Tạo cửa sổ chính cho ứng dụng.
 - Hiển thị các thành phần nhập liệu và điều khiển:
 - + Entry: Nhập tên, số điện thoại và email của người dùng.
 - + Button: Gồm các nút chức năng như Thêm, Sửa, Xóa, Tìm kiếm.
 - + Treeview: Một bảng hiển thị tất cả liên hệ đã lưu, gồm ba cột: Tên, Số điện thoại, Email.
 - Khi người dùng thực hiện một thao tác như thêm hoặc sửa liên hệ, giao diện sẽ:
 - + Gửi dữ liệu người dùng nhập vào tới module xử lý dữ liệu.
 - + Hiển thị kết quả tương ứng, ví dụ: thêm dòng mới vào bảng, cập nhật dòng hiện có,...
- ❖ Module quản lý dữ liệu (contacts.py):
 - ✓ Đây là trái tim của ứng dụng, nơi lưu trữ, cập nhật và thao tác với dữ liệu danh bạ. Toàn bộ dữ liệu được lưu trong một file có tên là contacts.json, dưới dạng danh sách các đối tượng JSON (mỗi liên hệ là một từ điển chứa tên, điện thoại và email).
 - ✓ Chức năng chính:
 - Đọc dữ liệu (load_contacts): Mỗi khi ứng dụng khởi động, module này sẽ đọc file JSON và nạp dữ liệu vào danh sách.
 - Ghi dữ liệu (save_contacts): Sau mỗi thay đổi như thêm, sửa hoặc xóa, module này sẽ ghi lại toàn bộ danh sách liên hệ vào file JSON để đảm bảo tính bền vững.

- Thêm liên hệ (add_contact): Tạo một liên hệ mới từ dữ liệu đầu vào và thêm vào danh sách.
- Sửa liên hệ (edit_contact): Cập nhật thông tin của một liên hệ đã tồn tại trong danh sách.
- Xóa liên hệ (delete_contact): Xóa một liên hệ theo chỉ số xác định.
- Tìm kiếm liên hệ (search_contacts): Lọc các liên hệ có tên chứa từ khóa tìm kiếm, không phân biệt chữ hoa hay chữ thường.
- ✓ Xử lý lỗi: Nếu file JSON bị lỗi định dạng hoặc không tồn tại, module sẽ tự động khởi tạo danh sách rỗng để tránh làm sập chương trình.
- ❖ Module xử lý hoạt động (logic điều khiển sự kiện):
- ✓ Đây là lớp trung gian giữa giao diện và dữ liệu. Module này đảm nhiệm việc phản ứng với các hành động của người dùng và đảm bảo chương trình thực hiện đúng chức năng.
- ✓ Chức năng chính:
 - Gắn chức năng cho nút bấm: Mỗi khi người dùng nhấn một nút (ví dụ: "Thêm"), module này sẽ lấy dữ liệu từ các ô nhập (Entry), gọi hàm tương ứng từ module dữ liệu, rồi cập nhật lại bảng hiển thị (Treeview).
 - Chọn dòng dữ liệu trong bảng để sửa: Khi người dùng click vào một dòng trong Treeview, thông tin sẽ được đổ ngược lên các ô nhập để tiện cho việc sửa đổi.
 - Kiểm tra hợp lệ dữ liệu:
 - + Kiểm tra các trường không được để trống.
 - + Kiểm tra xem có chọn dòng trong bảng trước khi sửa hoặc xóa hay không.
 - + Xử lý lỗi người dùng: Nếu người dùng thao tác sai (ví dụ: nhấn "Sửa" khi chưa chọn dòng nào), chương trình sẽ hiển thị thông báo lỗi thay vì bị lỗi chương trình.

3.1.2. Biểu đồ phân cấp chức năng



Hình 3.1: Biểu đồ phân cấp chức năng

❖ Quản lý giao diện người dùng

- Đây là phần giao diện (GUI), dùng để hiển thị và cho phép người dùng thao tác với ứng dụng.

- Hiển thị Treeview

+ Treeview là bảng hiển thị danh sách các liên hệ.

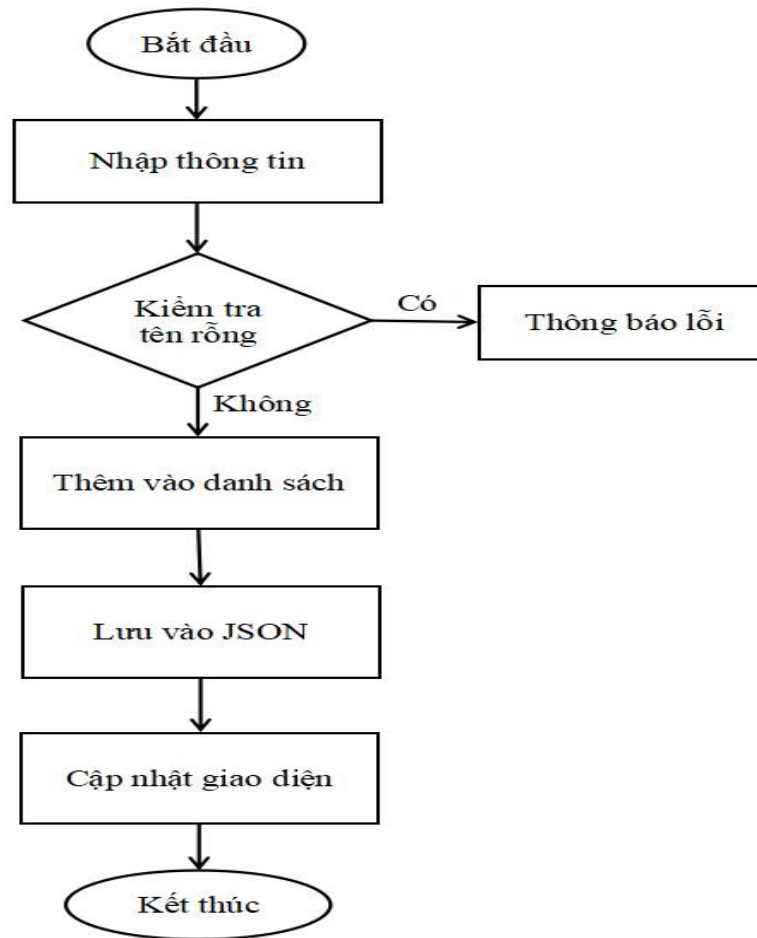
+ Khi chương trình khởi động hoặc sau khi thêm/sửa/xóa, dữ liệu sẽ được cập nhật và hiển thị ở đây.

+ Mỗi dòng trong Treeview tương ứng với một liên hệ (gồm tên, số điện thoại, email).

- Nhập thông tin (Entry)
 - + Gồm các ô nhập liệu cho người dùng gõ vào: name (tên), phone (số điện thoại), email (địa chỉ email).
 - + Dữ liệu từ các ô này sẽ được sử dụng khi người dùng nhấn các nút chức năng (thêm, sửa...).
- Các nút chức năng (Button):
 - + Gồm các nút thao tác như: Thêm (Add), Sửa (Edit), Xóa (Delete), Tìm kiếm (Search).
 - + Mỗi nút sẽ gọi hàm xử lý tương ứng (ở module xử lý hoạt động), truyền dữ liệu từ Entry.
- ❖ Quản lý dữ liệu
 - Chức năng xử lý phía sau, quản lý danh sách liên hệ trong bộ nhớ và lưu/đọc từ file JSON.
 - Đọc/ghi JSON
 - + Khi khởi động ứng dụng: Đọc dữ liệu từ file contacts.json (nếu tồn tại).
 - + Sau mỗi thao tác thêm, sửa, xóa: Tự động ghi lại dữ liệu mới vào file JSON.
 - Thêm / Sửa / Xóa / Tìm kiếm
 - + Đây là các phương thức thao tác dữ liệu chính:
 - + Thêm liên hệ mới.
 - + Sửa thông tin liên hệ đã có.
 - + Xóa liên hệ khỏi danh sách.
 - + Tìm kiếm tên liên hệ (theo chuỗi nhập vào).
- ❖ Xử lý hoạt động
 - Phần này xử lý logic khi người dùng thao tác trên giao diện.
 - Gắn sự kiện (Buttons)
 - + Mỗi nút (Add, Edit...) sẽ gắn với một sự kiện/hàm xử lý như: add_contact(), edit_contact(), delete_contact()
 - + Khi người dùng nhấn nút, chương trình lấy dữ liệu từ Entry → gọi hàm tương ứng.
 - Kiểm tra lỗi và thông báo
 - + Trước khi xử lý, chương trình sẽ kiểm tra: Có ô nào bị bỏ trống không?, Email có đúng định dạng không?, Có chọn dòng trong Treeview trước khi sửa/xóa không?
 - + Nếu có lỗi, chương trình sẽ hiển thị thông báo (messagebox) hướng dẫn người dùng sửa.

3.2. Sơ đồ khối các thuật toán chính

3.2.1. Sơ đồ thuật toán thêm liên hệ



Hình 3.2: Sơ đồ thuật toán thêm liên hệ

- ❖ Mục đích: Thêm một liên hệ mới (gồm tên, số điện thoại, email) vào danh sách, lưu vào file và hiển thị trên giao diện.
- ❖ Các bước xử lý:
 - Thu thập dữ liệu: Khi người dùng nhập thông tin và nhấn nút "Thêm", chương trình đọc dữ liệu từ các ô nhập (Entry).
 - Kiểm tra tính hợp lệ:
 - + Kiểm tra xem có trường nào bị bỏ trống không.
 - + Nếu có thì gửi thông báo lỗi bằng messagebox.
 - Tạo đối tượng liên hệ: Tạo một từ điển (hoặc đối tượng) chứa thông tin họ tên, số điện thoại, email.
 - Lưu dữ liệu:
 - + Thêm liên hệ mới vào danh sách đang quản lý.
 - + Ghi danh sách này vào file contacts.json bằng module json.
 - Cập nhật giao diện: Cập nhật liên hệ mới vào bảng Treeview.

❖ Kết quả:

- Liên hệ mới hiển thị trên bảng.
- File contacts.json được cập nhật.

3.2.2. Sơ đồ thuật toán sửa liên hệ



Hình 3.3: Sơ đồ thuật toán sửa liên hệ

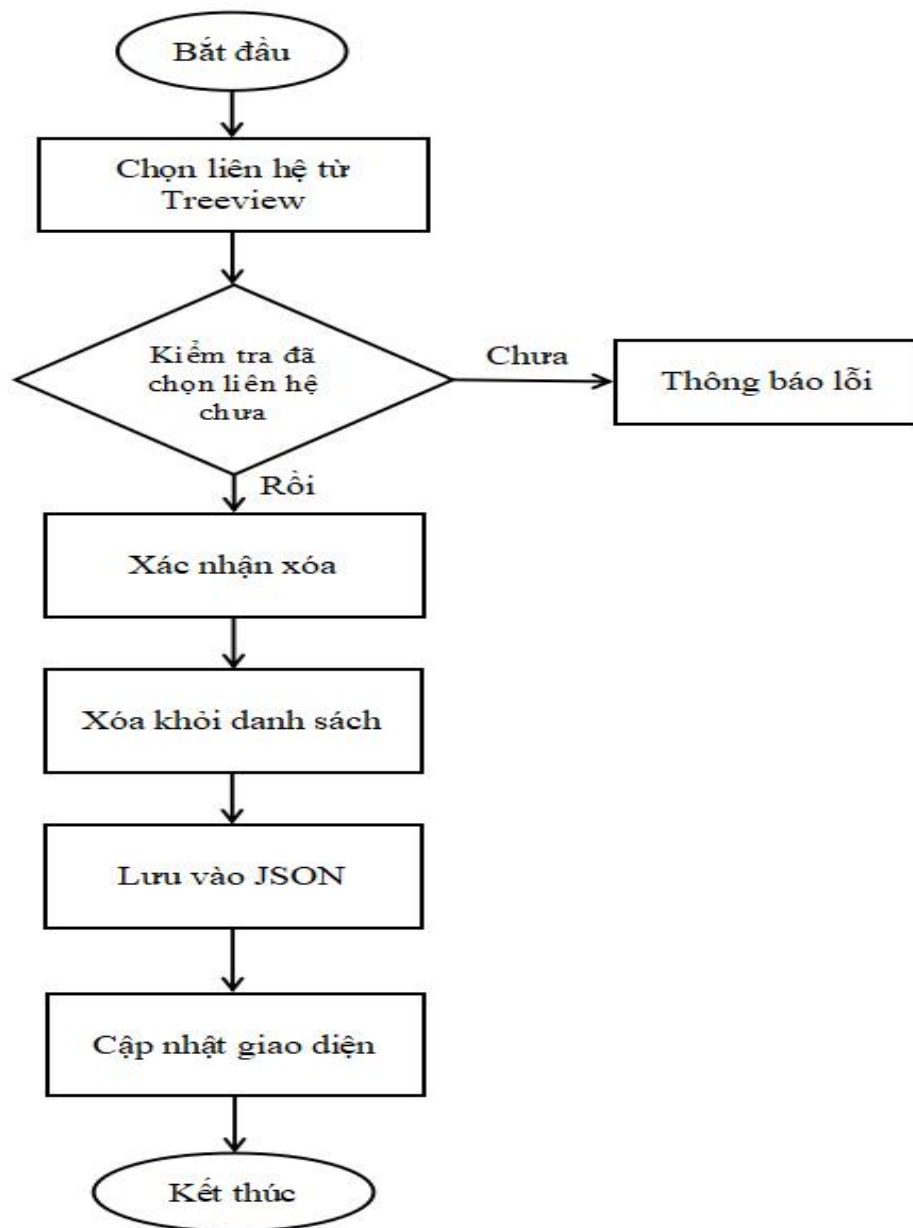
❖ Mục đích: Cho phép người dùng cập nhật thông tin liên hệ đã có.

❖ Các bước xử lý:

- Lấy lựa chọn:
 - + Khi nhấn nút "Sửa", kiểm tra xem người dùng đã chọn dòng nào trên Treeview chưa.
 - + Nếu chưa chọn hiển thị thông báo lỗi.
- Cập nhật thông tin mới.
- Lấy dữ liệu mới từ Entry: Đọc lại tên, số điện thoại, email mà người dùng đã chỉnh sửa.

- Cập nhật danh sách:
 - + Xác định chỉ số của dòng trong danh sách.
 - + Cập nhật dữ liệu tương ứng trong danh sách.
 - Ghi lại file JSON: Cập nhật dữ liệu đã sửa vào contacts.json.
 - Làm mới Treeview: Cập nhật dữ liệu mới từ danh sách.
- ❖ Kết quả:
- Treeview cập nhật liên hệ đã sửa.
 - Dữ liệu trong file JSON phản ánh thông tin mới.

3.2.3. Sơ đồ thuật toán xóa liên hệ



Hình 3.4: Sơ đồ thuật toán xóa liên hệ

- ❖ Mục đích: Xóa một liên hệ ra khỏi danh sách và file dữ liệu.

❖ Các bước xử lý:

- Kiểm tra lựa chọn:

- + Khi người dùng nhấn nút "Xóa", kiểm tra xem có chọn dòng nào không.
- + Nếu không thì hiển thị thông báo lỗi.

- Xóa dữ liệu:

- + Xác định vị trí (index) liên hệ trong danh sách.
- + Xóa phần tử đó khỏi danh sách liên hệ.

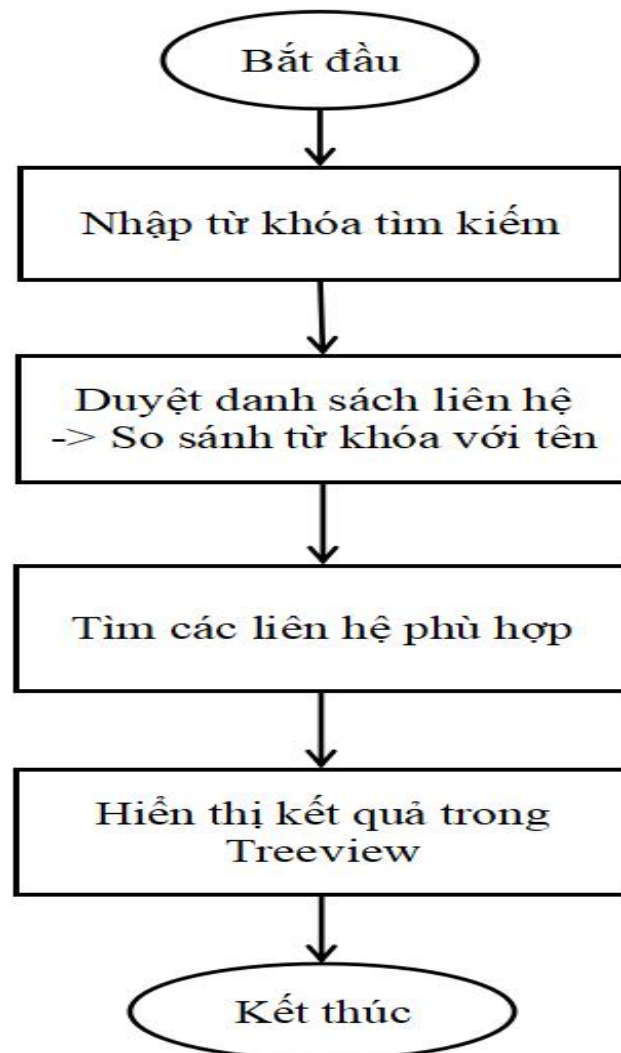
- Ghi lại file JSON: Ghi danh sách đã cập nhật vào contacts.json.

- Cập nhật Treeview: Xóa dòng bị xóa khỏi bảng.

❖ Kết quả:

- Liên hệ bị loại bỏ khỏi danh sách và giao diện.
- Dữ liệu trong contacts.json được cập nhật.

3.2.4. Sơ đồ thuật toán tìm kiếm liên hệ



Hình 3.5: Sơ đồ thuật toán tìm kiếm liên hệ

- ❖ Mục đích: Cho phép người dùng tìm nhanh một liên hệ theo tên.
- ❖ Các bước xử lý:
 - Lấy từ khóa tìm kiếm: Người dùng nhập vào ô tìm kiếm và nhấn nút "Tìm kiếm".
 - Lọc dữ liệu:
 - + Duyệt từng liên hệ trong danh sách.
 - + So sánh từ khóa với tên liên hệ (dùng `.lower()` để không phân biệt chữ hoa/thường).
 - Hiển thị kết quả:
 - + Treeview bị xóa tạm thời.
 - + Chỉ hiển thị các liên hệ phù hợp với từ khóa.
 - Kết quả: Treeview chỉ còn các dòng chứa tên phù hợp với từ khóa.

3.3. Cấu trúc dữ liệu

- ❖ Dữ liệu được lưu trong tệp `contacts.json` theo định dạng JSON. Mỗi liên hệ là một đối tượng (dictionary) trong danh sách các liên hệ.
- ❖ Cấu trúc của một liên hệ:

Trường dữ liệu	Kiểu dữ liệu	Mô tả
Họ tên	String	Họ tên của người liên hệ
Số điện thoại	String	Số điện thoại
Email	String	Địa chỉ email

- ❖ Đặc điểm:
 - Danh sách liên hệ là một mảng các đối tượng JSON.
 - Tệp `contacts.json` được đọc khi khởi động chương trình và ghi lại sau mỗi thao tác thêm/sửa/xóa.
 - Nếu tệp không tồn tại hoặc lỗi định dạng, chương trình sẽ khởi tạo danh sách trống hoặc báo lỗi (bắt lỗi định dạng JSON).

3.4. Chương trình

Chương trình gồm hai thành phần chính:

- Lớp `ContactManager`: Quản lý dữ liệu liên hệ (đọc/ghi file JSON).
- Lớp `ContactApp`: Quản lý giao diện người dùng và xử lý tương tác.

3.4.1. Lớp `ContactManager`

Lớp này đóng vai trò quản lý dữ liệu liên hệ, thực hiện thao tác đọc, ghi, tìm kiếm trong file JSON `contacts.json`.

- ❖ Hàm khởi tạo `__init__(self, filename='contacts.json')`
 - Chức năng: Hàm khởi tạo, gán tên file lưu trữ, khởi tạo danh sách liên hệ và gọi hàm `load_contacts()`.
 - Tham số: `filename` – tên file JSON chứa dữ liệu liên hệ.
 - Gán:


```
self.filename = filename
self.contacts = []
```
- ❖ Hàm `load_contacts(self)`
 - Chức năng: Đọc dữ liệu từ file `contacts.json` và lưu vào `self.contacts`.
 - Xử lý lỗi:
 - + Nếu file không tồn tại hoặc lỗi định dạng JSON → khởi tạo `self.contacts = []`.
- ❖ Hàm `save_contacts(self)`
 - Chức năng: Ghi dữ liệu `self.contacts` vào file JSON.
 - Định dạng lưu: Dùng `json.dump()` với `indent=4` và `ensure_ascii=False` để đảm bảo dễ đọc và hỗ trợ Unicode (tiếng Việt).
- ❖ Hàm `add_contact(self, contact)`
 - Chức năng: Thêm một liên hệ vào danh sách `self.contacts`, sau đó ghi lại file JSON.
 - Đầu vào: `contact` là một dictionary với các khóa: "họ tên", "số điện thoại", "email".
- ❖ Hàm `update_contact(self, index, new_contact)`
 - Chức năng: Cập nhật thông tin liên hệ tại vị trí `index` bằng dữ liệu mới.
 - Kiểm tra: Nếu `index` nằm trong phạm vi danh sách, thực hiện cập nhật.
- ❖ Hàm `delete_contact(self, index)`
 - Chức năng: Xóa liên hệ tại vị trí `index` trong danh sách, sau đó ghi lại file.
 - Kiểm tra: Đảm bảo `index` hợp lệ.
- ❖ Hàm `search_contacts(self, keyword)`
 - Chức năng: Tìm kiếm liên hệ theo từ khóa tên.
 - Cách hoạt động:
 - + Duyệt toàn bộ danh sách liên hệ.
 - + Từ khóa và tên liên hệ đều được chuyển về chữ thường. Nếu từ khóa nằm trong tên liên hệ thì liên hệ đó sẽ được đưa vào danh sách kết quả.

3.4.2. Lớp **ContactApp**

- ❖ Hàm khởi tạo `__init__(self, root)`: Khởi tạo chương trình
 - Chức năng: Khởi tạo cửa sổ chính, gọi hàm tạo giao diện và tải danh sách liên hệ.

- Hoạt động:
 - + Khởi tạo trình quản lý liên hệ `self.manager = ContactManager()`.
 - + Thiết lập tiêu đề, màu nền cửa sổ.
 - + Gọi `create_widgets()` để tạo các thành phần giao diện.
 - + Gọi `load_contacts()` để hiển thị dữ liệu.
- ❖ Hàm `create_widgets(self)`: Tạo giao diện
 - Chức năng: Tạo các Entry, Button, Treeview cho người dùng.
 - Giao diện gồm:
 - + Entry nhập: Name, Phone, Email.
 - + Button: Thêm, Sửa, Xóa.
 - + Ô tìm kiếm và các nút Tìm kiếm, Đặt lại.
 - + Bảng Treeview hiển thị danh sách liên hệ.
- ❖ Hàm `load_contacts(self)`: Hiển thị danh sách liên hệ
 - Chức năng: Nạp dữ liệu từ bộ nhớ (đã đọc từ file JSON).
 - Hoạt động:
 - + Xóa toàn bộ dòng trong Treeview.
 - + Duyệt qua danh sách `self.manager.contacts`, hiển thị từng dòng.
- ❖ Hàm `add_contact(self)`: Thêm liên hệ mới
 - Chức năng: Thêm một liên hệ mới từ Entry.
 - Quy trình:
 - + Lấy dữ liệu bằng `get_form_data()`.
 - + Kiểm tra trường Name có rỗng → nếu có: cảnh báo.
 - + Gọi `self.manager.add_contact(contact)` để lưu.
 - + Gọi `load_contacts()` để cập nhật giao diện.
 - + Gọi `clear_fields()` để làm trống ô nhập.
- ❖ Hàm `edit_contact(self)`: Sửa thông tin liên hệ
 - Chức năng: Cập nhật liên hệ đang được chọn.
 - Quy trình:
 - + Kiểm tra đã chọn dòng chưa → nếu chưa: cảnh báo.
 - + Lấy chỉ số dòng được chọn từ Treeview.
 - + Lấy thông tin mới từ Entry bằng `get_form_data()`.
 - + Gọi `self.manager.update_contact(index, contact)` để cập nhật.
 - + Gọi `load_contacts()` để làm mới bảng.
- ❖ Hàm `delete_contact(self)`: Xóa liên hệ
 - Chức năng: Xóa liên hệ được chọn trong bảng.
 - Quy trình:

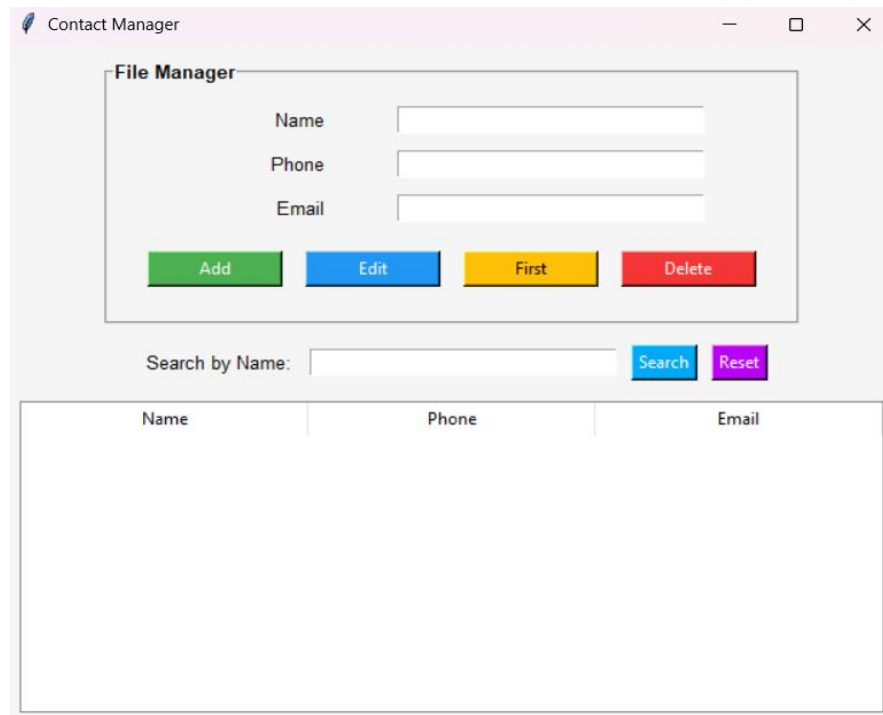
- + Kiểm tra đã chọn dòng chưa.
- + Lấy chỉ số dòng từ Treeview.
- + Gọi `self.manager.delete_contact(index)` để xóa khỏi danh sách.
- + Gọi `load_contacts()` để làm mới bảng.
- + Gọi `clear_fields()` để xóa các Entry.
- ❖ Hàm `search_contact(self)`: Tìm kiếm liên hệ
 - Chức năng: Tìm kiếm theo tên người dùng nhập.
 - Quy trình:
 - + Lấy từ khóa từ ô nhập.
 - + Nếu từ khóa rỗng → cảnh báo.
 - + Xóa bảng tạm thời.
 - + Duyệt danh sách, lọc các liên hệ chứa từ khóa trong trường name.
 - + Hiện thị kết quả phù hợp trong Treeview.
- ❖ Hàm `on_select(self, event)`: Hiện thị dữ liệu dòng được chọn
 - Chức năng: Khi người dùng chọn dòng trong bảng, các Entry sẽ tự động điền thông tin tương ứng.
 - Hoạt động: Lấy dòng được chọn → đọc các trường name, phone, Email → điền vào Entry.
- ❖ Hàm `get_form_data(self)`: Lấy dữ liệu từ ô nhập
 - Chức năng: Đóng gói thông tin từ Entry thành một dictionary.
 - Kết quả trả về:


```
{
    "name": ...,
    "phone": ...,
    "email": ...
}
```
- ❖ Hàm `clear_fields(self)`: Xóa trắng ô nhập
 - Chức năng: Làm trống toàn bộ các Entry.

CHƯƠNG 4: THỰC NGHIỆM VÀ KẾT LUẬN

4.1. Thực nghiệm

4.1.1. Giao diện chính của chương trình

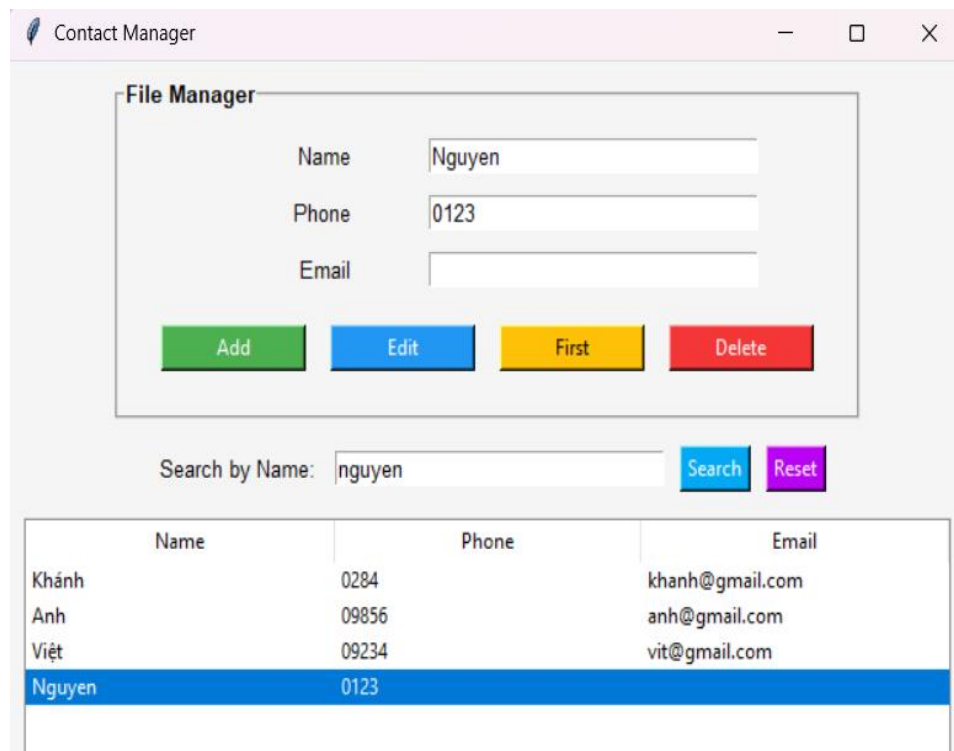


The screenshot shows the 'Contact Manager' application window. It features a 'File Manager' section with three input fields: 'Name', 'Phone', and 'Email'. Below these fields are four buttons: 'Add' (green), 'Edit' (blue), 'First' (yellow), and 'Delete' (red). Below the buttons is a search bar labeled 'Search by Name:' with a 'Search' button (cyan) and a 'Reset' button (purple). At the bottom is a table with three columns: 'Name', 'Phone', and 'Email'. The table is currently empty.

Hình 4.1: Giao diện chính của chương trình

4.1.2. Chức năng thêm liên hệ (Add)

- Thêm “Nguyễn”, “0123” → xuất hiện trong Table.



The screenshot shows the 'Contact Manager' application window after adding a new contact. The 'Name' field is filled with 'Nguyen' and the 'Phone' field is filled with '0123'. The 'Search by Name:' bar now contains 'nguyen'. The table at the bottom now displays four rows of data:

Name	Phone	Email
Khánh	0284	khanh@gmail.com
Anh	09856	anh@gmail.com
Việt	09234	vit@gmail.com
Nguyen	0123	

The 'Nguyen' row is highlighted in blue.

Hình 4.2: Giao diện khi thêm liên hệ

4.1.3. Chức năng sửa liên hệ (Edit)

- Trước khi sửa thông tin cần phải chọn liên hệ muốn sửa trong Treeview.
- Sửa số → file JSON cập nhật.

File Manager

Name:

Phone:

Email:

Search by Name:

Name	Phone	Email
Khánh	0284	khanh@gmail.com
Anh	09856	anh@gmail.com
Việt	09234	vit@gmail.com
Nguyen	0123456789	

Hình 4.3: Giao diện khi sửa liên hệ

4.1.4. Chức năng tìm kiếm liên hệ (Search)

File Manager

Name:

Phone:

Email:

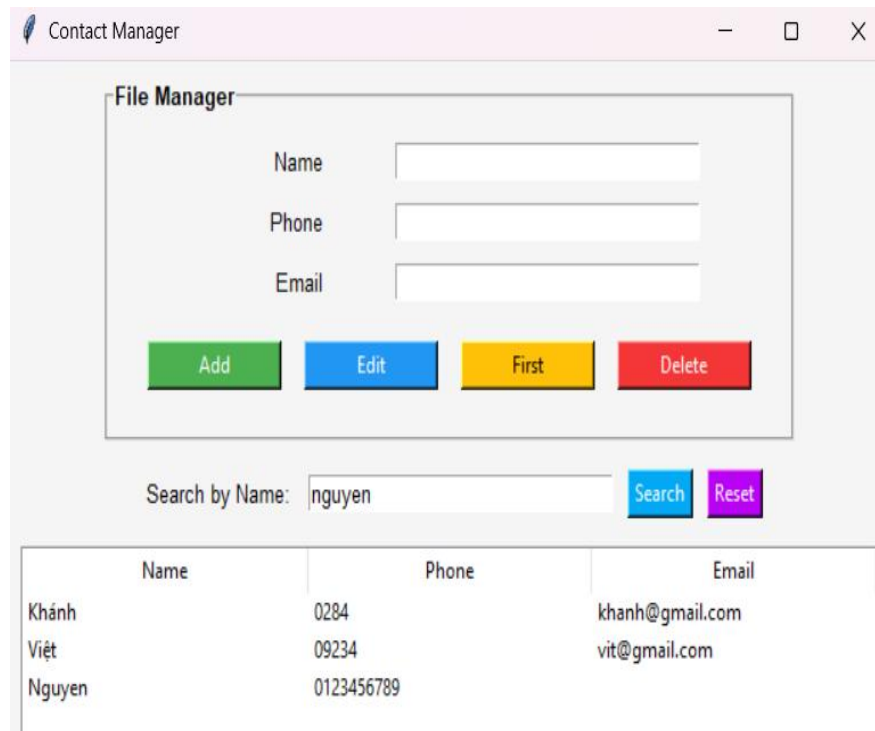
Search by Name:

Name	Phone	Email
Nguyen	0123456789	

Hình 4.4: Giao diện khi tìm kiếm liên hệ

4.1.5. Chức năng xóa liên hệ (Delete)

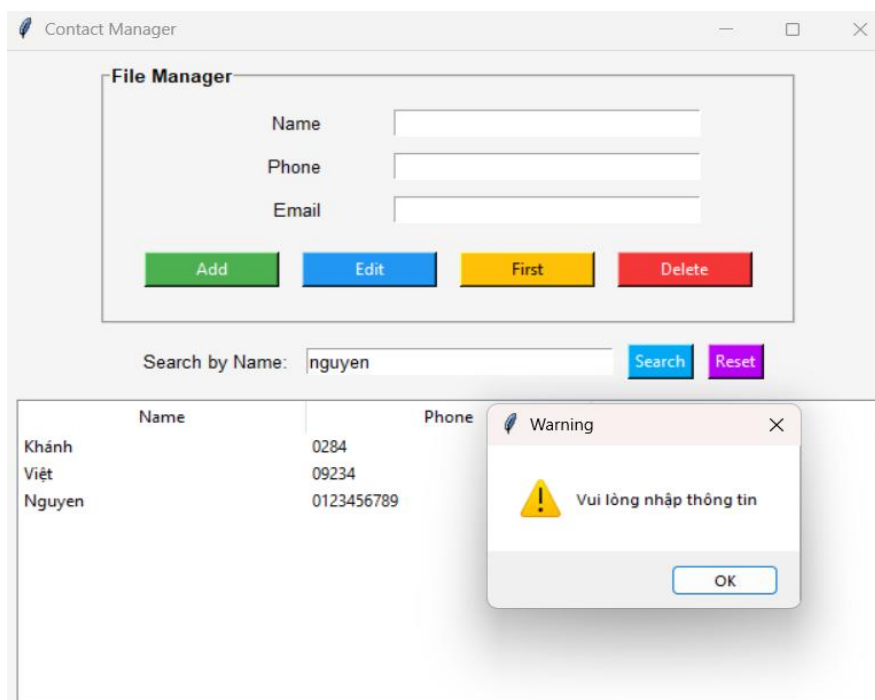
- Trước khi xóa cần nhấn chọn liên hệ muốn xóa trong Treeview.
- Xóa liên hệ “Anh”, “09856”.



Hình 4.5: Giao diện khi xóa liên hệ

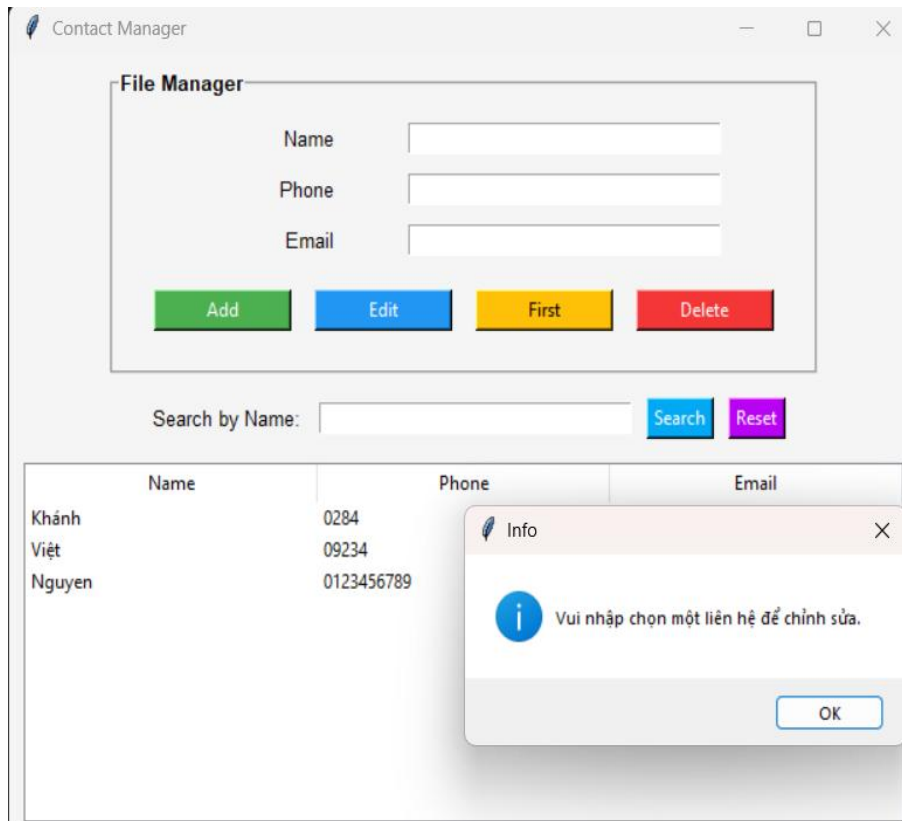
4.1.6. Bắt lỗi format JSON, khi thêm/sửa/xóa

- Khi muốn thêm liên hệ thì cần phải nhập thông tin tên liên hệ (Không nhất thiết phải nhập số điện thoại và email). Nếu không nhập thông tin sẽ hiển thị lỗi.

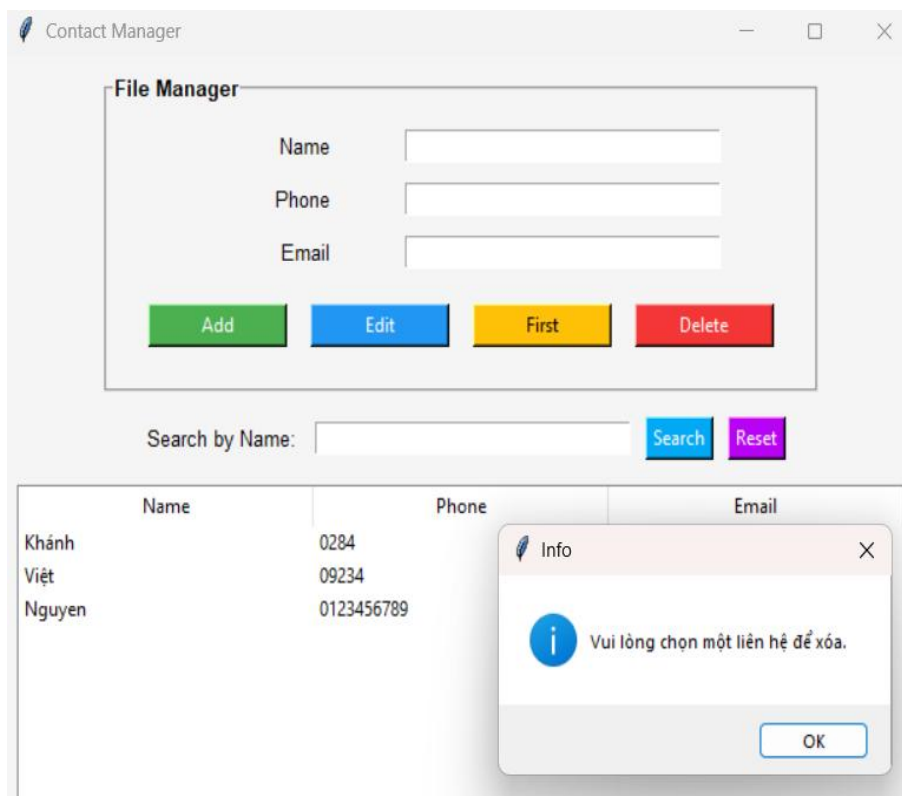


Hình 4.6: Hiển thị lỗi khi thêm liên hệ

- Khi muốn sửa hoặc xóa liên hệ thì cần phải chọn liên hệ muốn sửa trong Treeview. Nếu không chọn sẽ hiển thị lỗi.

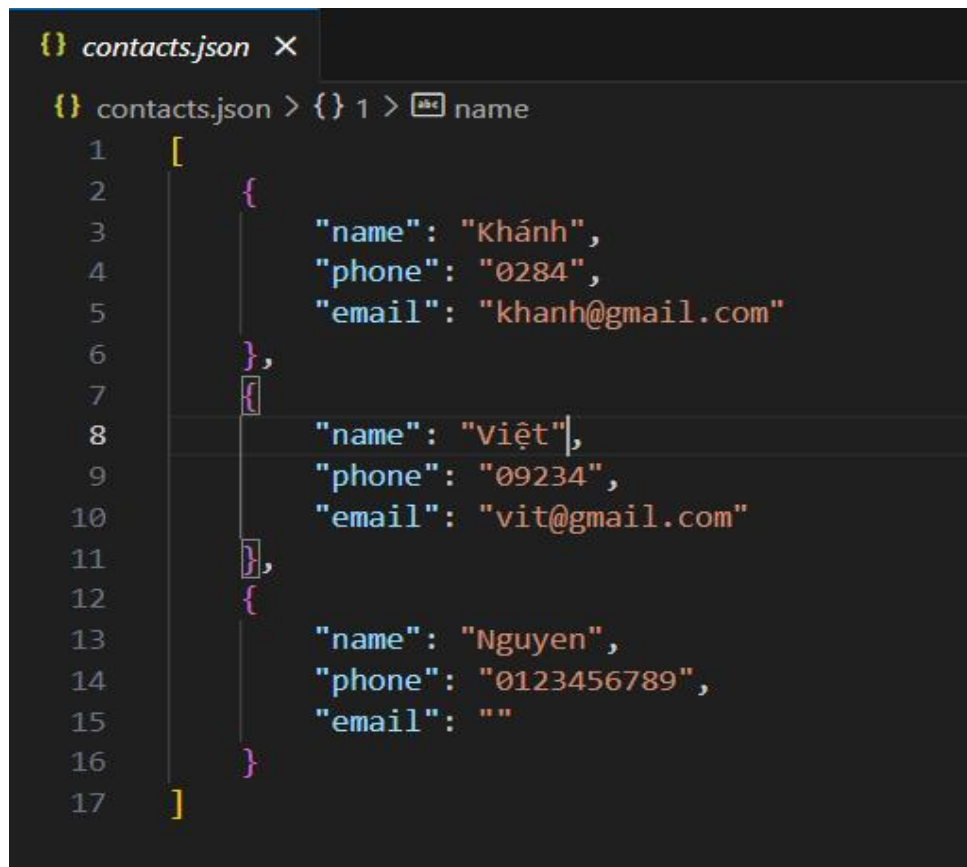


Hình 4.7: Hiển thị lỗi khi sửa thông tin liên hệ



Hình 4.8: Hiển thị lỗi khi xóa liên hệ

4.1.7. Dữ liệu JSON trong contacts.json



```

{} contacts.json ×
{} contacts.json > {} 1 > name
1  [
2      {
3          "name": "Khánh",
4          "phone": "0284",
5          "email": "khanh@gmail.com"
6      },
7      {
8          "name": "Việt",
9          "phone": "09234",
10         "email": "vit@gmail.com"
11     },
12     {
13         "name": "Nguyễn",
14         "phone": "0123456789",
15         "email": ""
16     }
17 ]
    
```

Hình 4.9: Dữ liệu trong file contacts.json

4.2. Kết luận

4.2.1. Sản phẩm đã làm đã làm được những gì?

- ❖ Trong quá trình thực hiện đề tài, nhóm đã xây dựng thành công một ứng dụng quản lý danh bạ với giao diện trực quan bằng ngôn ngữ lập trình Python kết hợp thư viện Tkinter. Cụ thể:
 - Thiết kế giao diện người dùng (GUI) đơn giản, dễ sử dụng, thân thiện với người dùng cuối.
 - Lưu trữ dữ liệu danh bạ bằng định dạng JSON (contacts.json), giúp đảm bảo tính linh hoạt và dễ dàng cập nhật, quản lý dữ liệu mà không cần sử dụng đến cơ sở dữ liệu phức tạp.
 - Thực hiện đầy đủ các chức năng chính như:
 - + Thêm mới liên hệ.
 - + Sửa thông tin liên hệ.
 - + Xóa liên hệ khỏi danh sách.
 - + Tìm kiếm liên hệ theo tên.
 - + Hiện thị danh sách liên hệ dưới dạng bảng (Treeview).

- Xử lý một số lỗi thường gặp như:
 - + Không cho phép thêm liên hệ khi chưa nhập tên.
 - + Cảnh báo khi chưa chọn liên hệ muốn sửa hoặc xóa.
 - + Kiểm tra và phản hồi cho người dùng thông qua messagebox.
- ❖ Những kết quả này chứng minh rằng ứng dụng đã hoạt động ổn định, đáp ứng tốt yêu cầu bài toán đề ra, đồng thời tạo nền tảng thuận lợi để phát triển thêm trong tương lai.

4.2.2. Những điều học được trong quá trình làm đề tài

- ❖ Về ngôn ngữ Python:
 - Củng cố kiến thức lập trình hướng đối tượng, danh sách (list), xử lý chuỗi và thao tác với file.
 - Hiểu rõ cách sử dụng module json để lưu trữ và truy xuất dữ liệu hiệu quả.
- ❖ Về xây dựng giao diện người dùng (Tkinter):
 - Nắm bắt được quy trình xây dựng GUI với các thành phần như Entry, Button, Treeview.
 - Xử lý sự kiện (event binding), cập nhật dữ liệu theo thao tác người dùng.
- ❖ Về phân tích và tổ chức mã nguồn:
 - Biết cách phân chia chương trình thành các lớp (Class) độc lập để dễ quản lý: tách lớp xử lý dữ liệu (ContactManager) và lớp giao diện (ContactApp).
 - Thực hành tốt kỹ năng xử lý lỗi và kiểm thử tính năng (testing thủ công).
- ❖ Về kỹ năng mềm:
 - Nâng cao khả năng tự học qua tài liệu, diễn đàn, ví dụ thực tiễn.
 - Rèn luyện tư duy logic và kiên nhẫn trong quá trình debug, tối ưu giao diện.

4.2.3. Hướng cải tiến trong tương lai

- ❖ Xác thực dữ liệu nâng cao: Kiểm tra số điện thoại hợp lệ và email đúng định dạng, hiển thị cảnh báo khi nhập sai.
- ❖ Nhập/xuất dữ liệu: Hỗ trợ lưu và đọc danh bạ từ Excel, CSV để sao lưu hoặc chia sẻ dễ dàng.
- ❖ Giao diện: Cải thiện màu sắc, biểu tượng, font chữ và bổ sung đa ngôn ngữ để thân thiện hơn với người dùng.
- ❖ Đóng gói phần mềm: Sử dụng PyInstaller tạo file .exe, giúp người dùng không cần cài Python vẫn chạy được chương trình.
- ❖ Tìm kiếm và phân loại: Hỗ trợ lọc, sắp xếp và phân nhóm liên hệ theo nhu cầu sử dụng.

TÀI LIỆU THAM KHẢO

1. Cuốn sách “Automate the Boring Stuff with Python: Practical Programming for Total Beginners” của tác giả Al Sweigart.
2. Cuốn sách “Python Programming for the Absolute Beginner- 3rd Edition” của tác giả Michael Dawson.
3. Mark Lutz (2013). Learning Python (5th Edition). O’Reilly Media.
4. TkDocs – Tkinter Documentation and Tutorials: <https://tkdocs.com>
5. W3Schools Python Tutorial: <https://www.w3schools.com/python/>
6. <https://t3h.edu.vn/tin-tuc/python-tkinter-lap-trinh-gui-bang-tkinter-trong-python>
7. <https://www.youtube.com/watch?v=mop6g-c5HEY>
8. <https://www.youtube.com/watch?v=kvd6GQZASno>
9. https://www.tutorialspoint.com/python/python_gui_programming.htm
10. <https://blog.tooljet.ai/python-gui-framework/>

MÃ QR GITHUB



MÃ QR YOUTUBE

