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 : Lương Ngọc Nam
Lớp : K58KTP.K01
Giáo viên GIẢNG DẠY : Nguyễn Văn Huy
Link GitHub: https://github.com/Luongngocnam/python

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 : Lương Ngọc Nam

Lớp :K58KTP.K01 *Ngành: Kỹ Thuật Máy Tính*

Giáo viên hướng dẫn :Nguyễn Văn Huy

Ngày giao đề:20/05/2025 *Ngày hoàn thành:10/06/2025*

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

Yêu cầu :

- Đầu vào: Form nhập name, phone, email.
- Đầu ra: Table hiển thị danh sách, lưu file contacts.json.
- Đọc/ghi JSON với module json.
- Bắt lỗi format JSON, lưu khi thêm/sửa/xoá.
- GUI: Entry, Buttons, Treeview.
- Tìm kiếm theo tên.
-

GIÁO VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Thái Nguyên, ngày…..tháng…..năm 20.....

GIÁO VIÊN HƯỚNG DẪN

(Ký ghi rõ họ tên)

Lời nói đầu

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ như hiện nay, các ứng dụng quản lý dữ liệu đóng vai trò thiết yếu trong việc tối ưu hóa công việc và nâng cao hiệu suất cá nhân cũng như tổ chức. Đặc biệt, việc quản lý danh bạ liên lạc một cách hiệu quả là một nhu cầu cơ bản và không thể thiếu đối với mọi cá nhân và doanh nghiệp.

Báo cáo này trình bày chi tiết quá trình xây dựng ứng dụng quản lý danh bạ GUI đơn giản, nhằm mục đích cung cấp một công cụ trực quan và dễ sử dụng để thêm, sửa, xóa và tìm kiếm thông tin liên hệ. Ứng dụng được phát triển dựa trên ngôn ngữ lập trình Python, kết hợp với các thư viện GUI để tạo ra một giao diện thân thiện với người dùng và sử dụng định dạng JSON để lưu trữ dữ liệu một cách linh hoạt.

Thông qua báo cáo này, chúng ta sẽ cùng tìm hiểu về các bước triển khai, các công nghệ được áp dụng, cũng như những kinh nghiệm thu được trong quá trình phát triển ứng dụng. Hy vọng rằng, sản phẩm này không chỉ đáp ứng được các yêu cầu đặt ra mà còn là nền tảng để phát triển những ứng dụng quản lý dữ liệu phức tạp hơn trong tương lai.

Mục Lục

Chương 1. Giới thiệu đề bài

1.1. Giới thiệu đề bài

Trong bài tập số 10, sinh viên được yêu cầu xây dựng một ứng dụng quản lý danh bạ đơn giản sử dụng ngôn ngữ lập trình Python kết hợp với thư viện tkinter để thiết kế giao diện đồ họa người dùng **(**GUI**).** Mục tiêu của bài tập là tạo ra một ứng dụng cho phép người dùng có thể thực hiện các chức năng cơ bản như thêm, sửa, xoá liên hệ và lưu trữ các thông tin này vào một tệp định dạng JSON**.**

Ứng dụng sẽ bao gồm các thành phần giao diện như: ô nhập liệu (`Entry`), nút bấm (`Button`) để thực hiện các thao tác, và bảng hiển thị danh sách liên hệ (`Treeview`) để người dùng dễ dàng quan sát và quản lý các mục đã lưu.

1.2. Các tính năng chính của chương trình

- Thêm liên hệ với các trường thông tin: Họ tên, Số điện thoại, Email.
- Chỉnh sửa và xoá liên hệ đã có trong danh sách.
- Tìm kiếm liên hệ theo tên.
- Hiển thị danh sách liên hệ bằng bảng (Treeview).
- Lưu trữ dữ liệu vào tệp contacts.json dưới định dạng JSON.
- Tự động đọc và hiển thị dữ liệu từ tệp khi khởi động chương trình.
- Xử lý lỗi định dạng JSON, đảm bảo chương trình không bị dừng khi đọc file lỗi.

1.3. Kiến thức được áp dụng

Để hoàn thành bài tập, sinh viên cần vận dụng kết hợp nhiều kiến thức và kỹ năng, bao gồm:

- Lập trình hướng đối tượng (OOP): Tổ chức mã nguồn thành các class như `ContactManager` để quản lý dữ liệu hiệu quả.
- Xử lý tệp tin JSON: Sử dụng module `json` để đọc, ghi và cập nhật dữ liệu.
- Thiết kế giao diện người dùng với `tkinter`: Tạo ra cửa sổ ứng dụng thân thiện, dễ sử dụng.
- Xử lý sự kiện giao diện: Gắn các thao tác (thêm, sửa, xoá, tìm kiếm) với các nút bấm.
- Tách biệt logic và giao diện: Tổ chức mã nguồn thành nhiều module (`contacts.py`, `main.py`) để dễ bảo trì.

1.4. Thách thức khi thực hiện

Trong quá trình phát triển ứng dụng, sinh viên có thể gặp một số khó khăn như:

- Đảm bảo tính đồng bộ giữa bảng giao diện và tệp JSON khi thực hiện thêm, sửa, xoá.
- Xác định và xử lý đúng liên hệ được chọn trong bảng Treeview.

- Bắt lỗi định dạng JSON nếu người dùng chỉnh sửa tệp ngoài chương trình.
- Thiết kế giao diện rõ ràng, dễ thao tác nhưng vẫn đảm bảo đầy đủ chức năng.

Chương 2. Cơ sở lý thuyết

2.1. Danh sách (List)

Trong Python, `list` là một cấu trúc dữ liệu cho phép lưu trữ nhiều phần tử cùng lúc, có thể thay đổi và được đánh chỉ số. Trong chương trình quản lý danh bạ, danh sách được sử dụng để lưu trữ tập hợp các liên hệ, mỗi liên hệ là một đối tượng kiểu `dict`. Thao tác thêm, sửa, xoá các liên hệ đều thực hiện trên danh sách này.

2.2. Từ điển (Dictionary)

Dict là kiểu dữ liệu ánh xạ trong Python cho phép lưu trữ các cặp khóa – giá trị. Mỗi liên hệ trong danh bạ được biểu diễn bằng một dict với các trường: "name", "phone" và "email". Kiểu dữ liệu này giúp dễ dàng truy xuất và cập nhật thông tin liên hệ.

2.3. Xử lý tệp JSON (module json)

Module json cho phép mã hóa và giải mã dữ liệu JSON. Trong bài này, dữ liệu danh bạ được lưu vào file contacts.json. Khi người dùng thao tác thêm, sửa, xoá, chương trình sẽ ghi lại toàn bộ danh sách vào tệp này.

- json.load(file) để đọc dữ liệu từ file JSON.
- json.dump(data, file) để ghi dữ liệu vào file JSON.

Việc xử lý JSON giúp đảm bảo dữ liệu được lưu trữ một cách có cấu trúc, dễ dàng tái sử dụng và tương thích với nhiều ứng dụng khác.

2.4. Giao diện đồ họa với Tkinter

Tkinter là thư viện chuẩn của Python dùng để xây dựng giao diện đồ họa. Trong chương trình này, tkinter được sử dụng để:

- Tạo các ô nhập liệu (Entry) cho người dùng nhập tên, số điện thoại, email.
- Tạo các nút bấm (Button) để người dùng thực hiện thao tác: Thêm, Sửa, Xoá, Tìm kiếm.
- Hiển thị danh sách liên hệ bằng bảng (Treeview) thuộc module ttk.
- Bắt sự kiện từ người dùng, ví dụ: chọn một dòng trong bảng, nhấn nút...

2.5. Lập trình hướng đối tượng (OOP)

Chương trình sử dụng lớp ContactManager để quản lý toàn bộ danh sách liên hệ và các thao tác liên quan (load, save, add, edit, delete, search). Việc sử dụng OOP giúp mã nguồn dễ bảo trì, phân tách rõ ràng giữa dữ liệu và giao diện.

2.6. Xử lý sự kiện (Event Handling)

Tkinter hỗ trợ cơ chế xử lý sự kiện để phản hồi các hành động từ người dùng. Ví dụ:

- Khi nhấn nút “Add”, chương trình sẽ lấy dữ liệu từ Entry, thêm vào danh sách và cập nhật Treeview.

- Khi chọn một dòng trên bảng Treeview, dữ liệu sẽ được đổ ngược lại vào form để sửa hoặc xoá.

2.7. Treeview và xử lý lựa chọn

- Treeview thuộc thư viện ttk giúp hiển thị dữ liệu theo dạng bảng. Khi người dùng nhấn vào một dòng, chương trình phải bắt được sự kiện và lấy ra giá trị của dòng được chọn để xử lý tiếp theo (sửa hoặc xoá).

2.8. Xử lý sự kiện và tương tác người dùng

- Tkinter sử dụng mô hình event-driven: các hành động (click chuột, nhập dữ liệu) sẽ kích hoạt các hàm tương ứng (`command`). Chương trình phải gán đúng sự kiện cho các nút để đảm bảo tính tương tác.

Chương 3. Thiết kế và xây dựng chương trình

3.1. Sơ đồ khối hệ thống

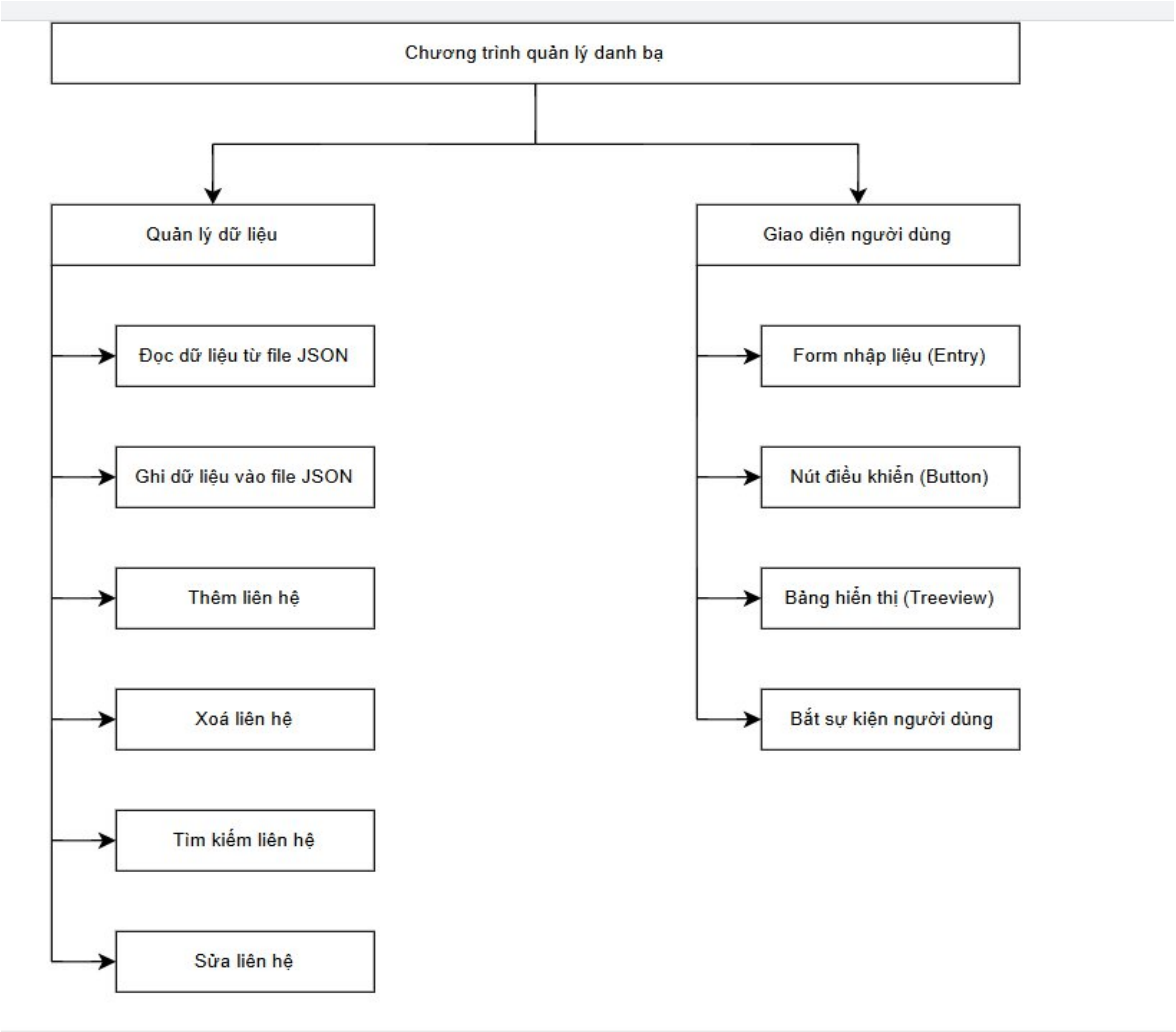
Chương trình được thiết kế chia thành hai phần chính, gồm:

- Phần xử lý dữ liệu (module `contacts.py`): Là nơi lưu trữ và thao tác dữ liệu danh bạ như thêm, sửa, xoá, tìm kiếm và lưu/đọc từ file `contacts.json`.
- Phần giao diện người dùng (GUI sử dụng Tkinter): Là nơi tiếp nhận thao tác của người dùng thông qua các ô nhập liệu và nút điều khiển, hiển thị danh sách danh bạ bằng bảng Treeview.

3.1.1. Mô tả các module chính:

Tên Module	Mô tả chức năng
`contacts.py`	Xử lý logic nghiệp vụ, đọc/ghi dữ liệu JSON, quản lý danh sách liên hệ.
`main.py` (hoặc GUI)	Hiển thị giao diện người dùng, xử lý sự kiện từ người dùng như thêm/sửa/xoá/tìm kiếm.

3.1.2. Biểu đồ phân cấp chức năng:



3.2. Sơ đồ khối các thuật toán chính

Các thuật toán chính được sử dụng trong chương trình đều xoay quanh thao tác với danh sách liên hệ. Dưới đây là mô tả đầu vào, xử lý, đầu ra và chức năng từng thuật toán:

3.2.1. Thuật toán thêm liên hệ

- Đầu vào: Tên, số điện thoại, email do người dùng nhập.
- Xử lý:
  - Tạo một từ điển từ dữ liệu nhập.
  - Thêm vào danh sách liên hệ.
  - Ghi lại toàn bộ danh sách vào file `contacts.json`.
  - Hiển thị lại danh sách trong bảng Treeview.
- Đầu ra: Giao diện cập nhật, file JSON lưu dữ liệu mới.

3.2.2. Thuật toán xoá liên hệ

- Đầu vào: Một dòng được chọn trong bảng Treeview.
- Xử lý:
  - Lấy chỉ số của dòng đang chọn.
  - Xoá phần tử tương ứng trong danh sách liên hệ.
  - Ghi lại file JSON.
  - Làm mới Treeview.
- Đầu ra: Liên hệ bị xoá khỏi giao diện và file.

3.2.3 Thuật toán sửa liên hệ

- Đầu vào: Liên hệ đang chọn và dữ liệu mới trong form.
- Xử lý:
  - Cập nhật phần tử trong danh sách liên hệ tại vị trí đang chọn.
  - Ghi lại file JSON.
  - Làm mới Treeview.
- Đầu ra: Dữ liệu hiển thị mới.

3.2.4 Thuật toán tìm kiếm liên hệ

- Đầu vào: Chuỗi từ khoá tên do người dùng nhập.
- Xử lý:
  - Duyệt danh sách, lọc những liên hệ chứa từ khoá trong trường `name`.
  - Hiển thị danh sách tìm được lên Treeview.
- Đầu ra: Treeview chỉ còn danh sách phù hợp.

3.3. Cấu trúc dữ liệu

Dữ liệu được lưu dưới dạng danh sách các từ điển, mỗi từ điển đại diện cho một liên hệ:

```python
contacts = [
    {"name": "Nguyen Van A", "phone": "0123456789", "email": "a@gmail.com"},
    {"name": "Tran Thi B", "phone": "0987654321", "email": "b@gmail.com"}
]
```

3.3.1. Các trường thông tin:

Trường	Kiểu dữ liệu	Mô tả
`name`	`str`	Họ tên liên hệ
`phone`	`str`	Số điện thoại
`email`	`str`	Địa chỉ email

3.3.2. File dữ liệu `contacts.json`

File được ghi ở định dạng JSON như sau:

```json
[
  {
    "name": "Nguyen Van A",
    "phone": "0123456789",
    "email": "a@gmail.com"
  },
  {
    "name": "Tran Thi B",
    "phone": "0987654321",
    "email": "b@gmail.com"
  }
]
```

3.4. Các hàm trong chương trình

Chương trình được thiết kế theo hướng chia module rõ ràng giữa xử lý dữ liệu và giao diện. Dưới đây là mô tả các hàm chính:

Trong **contacts.py**

Tên hàm	Chức năng
`load_contacts()`	Đọc dữ liệu từ file JSON, trả về danh sách liên hệ
`save_contacts(data)`	Ghi danh sách liên hệ vào file JSON
`add_contact(contact)`	Thêm một liên hệ mới vào danh sách
`edit_contact(index, contact)`	Cập nhật liên hệ tại vị trí `index`
`delete_contact(index)`	Xoá liên hệ tại vị trí `index`
`search_contacts(keyword)`	Tìm kiếm các liên hệ có tên chứa `keyword`

Trong giao diện Tkinter

Tên hàm	Chức năng
`on_add()`	Lấy dữ liệu từ Entry, thêm vào danh sách và cập nhật Treeview
`on_edit()`	Cập nhật lại thông tin liên hệ đang chọn
`on_delete()`	Xoá dòng được chọn khỏi danh sách
`on_search()`	Lọc và hiển thị danh sách theo từ khoá tên
`load_to_treeview()`	Hiển thị lại toàn bộ danh sách lên Treeview
`clear_entries()`	Xoá trắng các ô nhập liệu

Chương 4. Thực nghiệm và kết luận

4.1. Giới thiệu về Python

Python là một ngôn ngữ lập trình bậc cao, thông dịch, được phát triển bởi Guido van Rossum và phát hành lần đầu tiên vào năm 1991. Python nổi bật nhờ cú pháp rõ ràng, dễ đọc, dễ viết và có cộng đồng phát triển rất lớn. Đây là một trong những ngôn ngữ phổ biến nhất hiện nay trong nhiều lĩnh vực như: phát triển phần mềm, trí tuệ nhân tạo, xử lý dữ liệu, phát triển web, tự động hoá và cả lập trình ứng dụng giao diện người dùng (GUI).

Một số đặc điểm chính của Python:

- Dễ học và dễ đọc: Cú pháp đơn giản, gần giống ngôn ngữ tự nhiên giúp người học nhanh chóng tiếp cận.
- Hỗ trợ lập trình hướng đối tượng: Cho phép tổ chức mã nguồn theo hướng module, class và tái sử dụng mã.
- Thư viện phong phú: Có sẵn nhiều thư viện như `json`, `tkinter`, `os`, `re`, `pandas`... giúp lập trình viên phát triển nhanh các ứng dụng.
- Tương thích đa nền tảng: Chạy được trên Windows, Linux, macOS mà không cần chỉnh sửa mã nguồn.

Lý do chọn Python cho bài toán quản lý danh bạ:

- Có sẵn thư viện GUI là `tkinter` rất phù hợp để thiết kế ứng dụng đơn giản với các thành phần như: Entry, Button, Treeview...
- Hỗ trợ tốt cho xử lý dữ liệu JSON: Python có module `json` rất tiện lợi để lưu trữ và đọc dữ liệu danh bạ.
- Cú pháp gọn gàng, rõ ràng giúp tập trung vào giải quyết bài toán hơn là xử lý lỗi cú pháp phức tạp.
- Tốc độ phát triển nhanh: Phù hợp với yêu cầu làm dự án cá nhân, học tập trong thời gian ngắn.

Công cụ sử dụng để lập trình

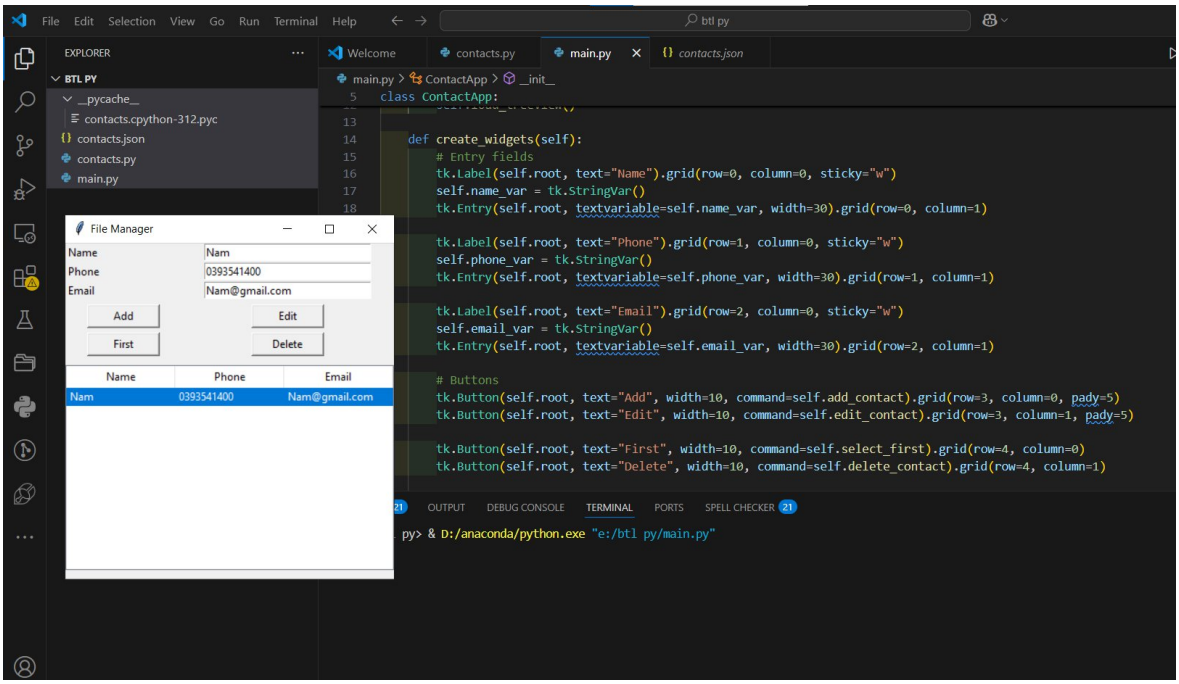
- Python phiên bản 3.x (khuyến nghị từ 3.9 trở lên).
- Môi trường phát triển: VS Code hoặc IDLE.
- Thư viện sử dụng:
    - `tkinter`: để thiết kế giao diện người dùng.
    - `json`: để lưu và đọc file dữ liệu.

4.2. Thực nghiệm

Sau khi hoàn thiện chương trình, nhóm đã tiến hành chạy thử và kiểm tra lần lượt các chức năng chính của ứng dụng quản lý danh bạ. Các kết quả thực nghiệm được ghi lại như sau:

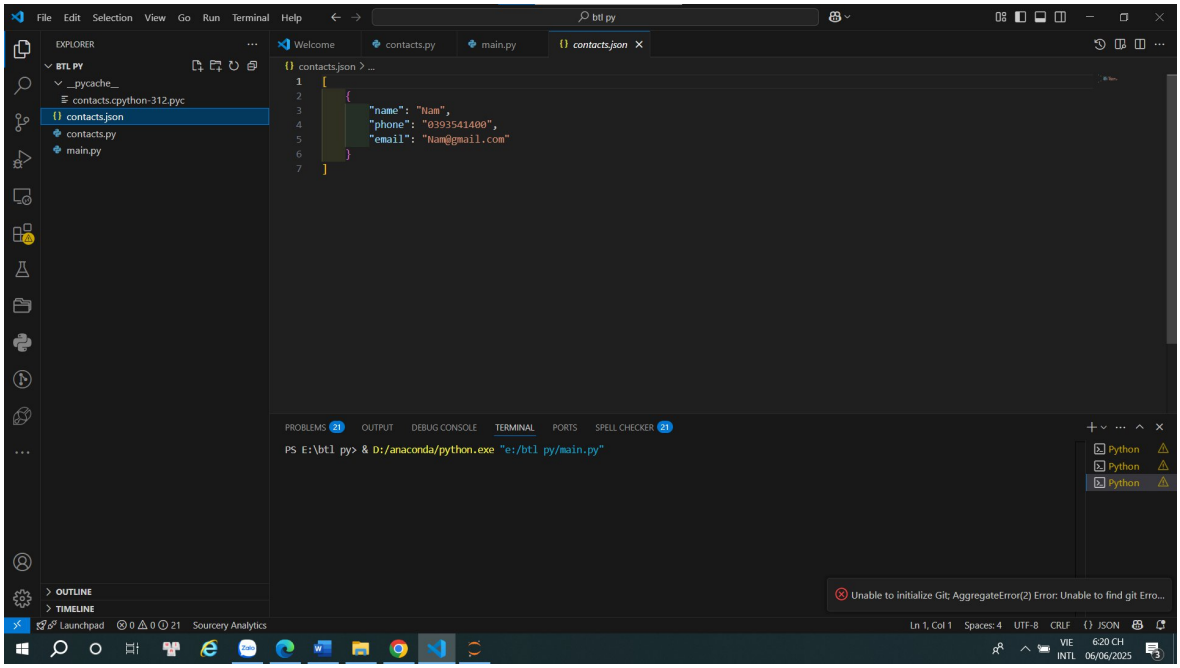
4.2.1. Chức năng: Thêm liên hệ

Chức năng nhập họ tên, số điện thoại, email và nhấn nút “Add”.



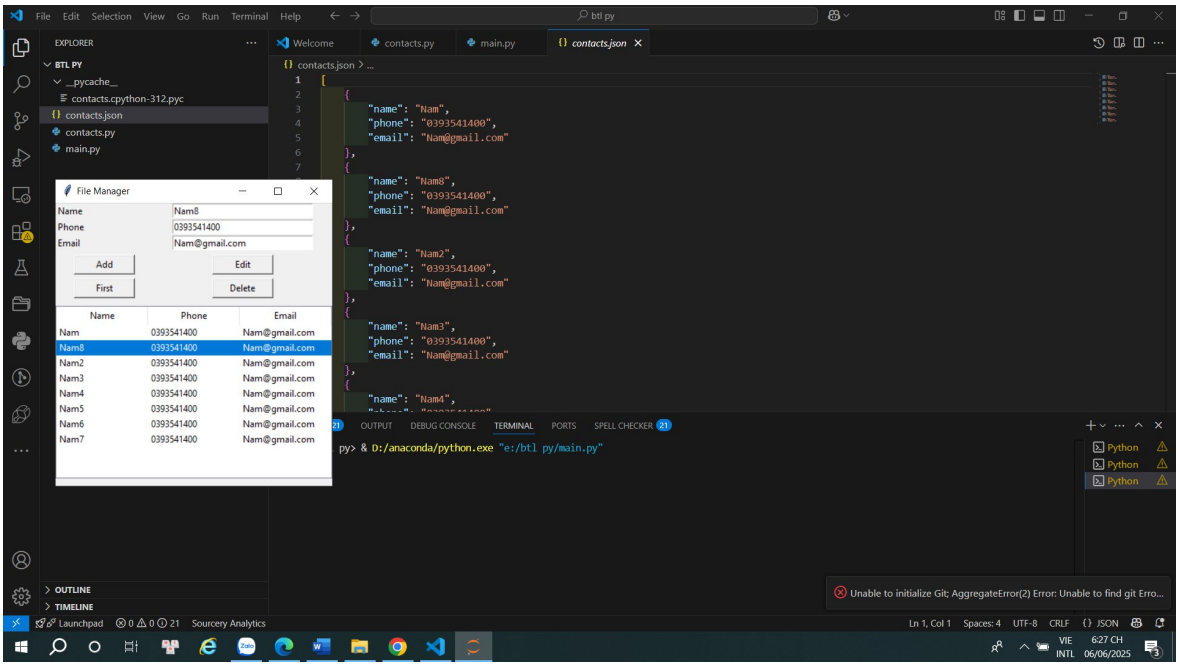
Kết quả mong đợi: Dữ liệu mới xuất hiện trong bảng và được lưu trong file contacts.json.

Kết quả thực tế: Đúng như mong đợi, liên hệ mới được hiển thị ngay lập tức và file JSON được cập nhật.

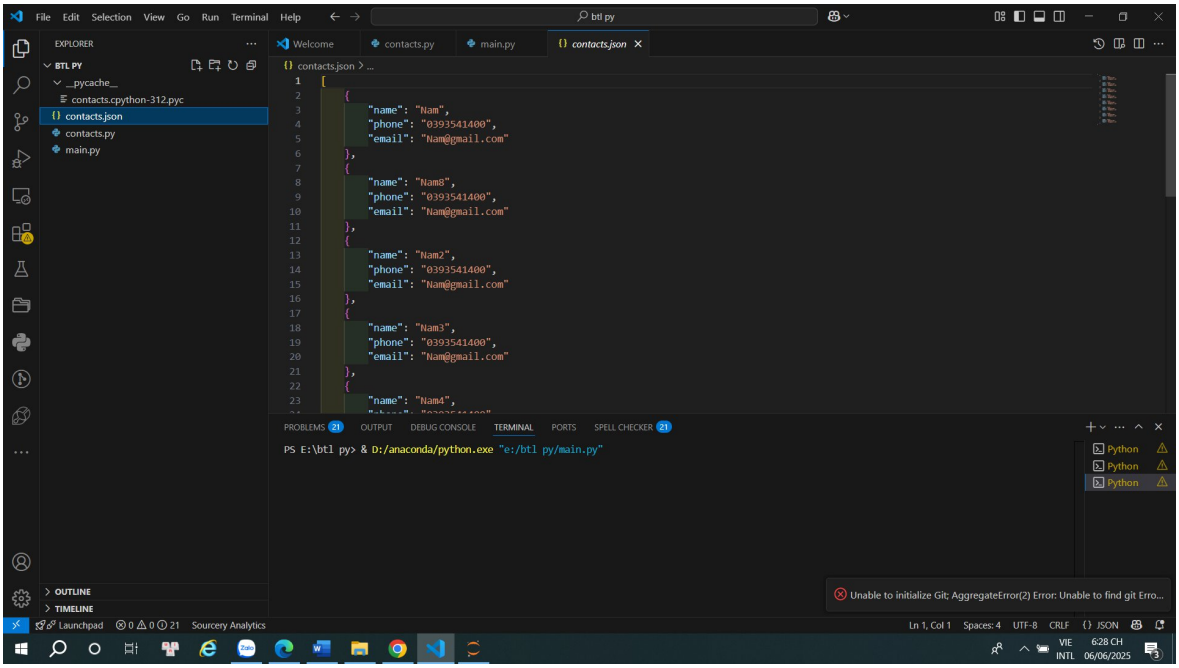


4.2.2. Chức năng: Sửa liên hệ

- Mô tả: Sửa “Nam1” thành “Nam8” trong danh sách, thay đổi thông tin rồi nhấn “Edit”.

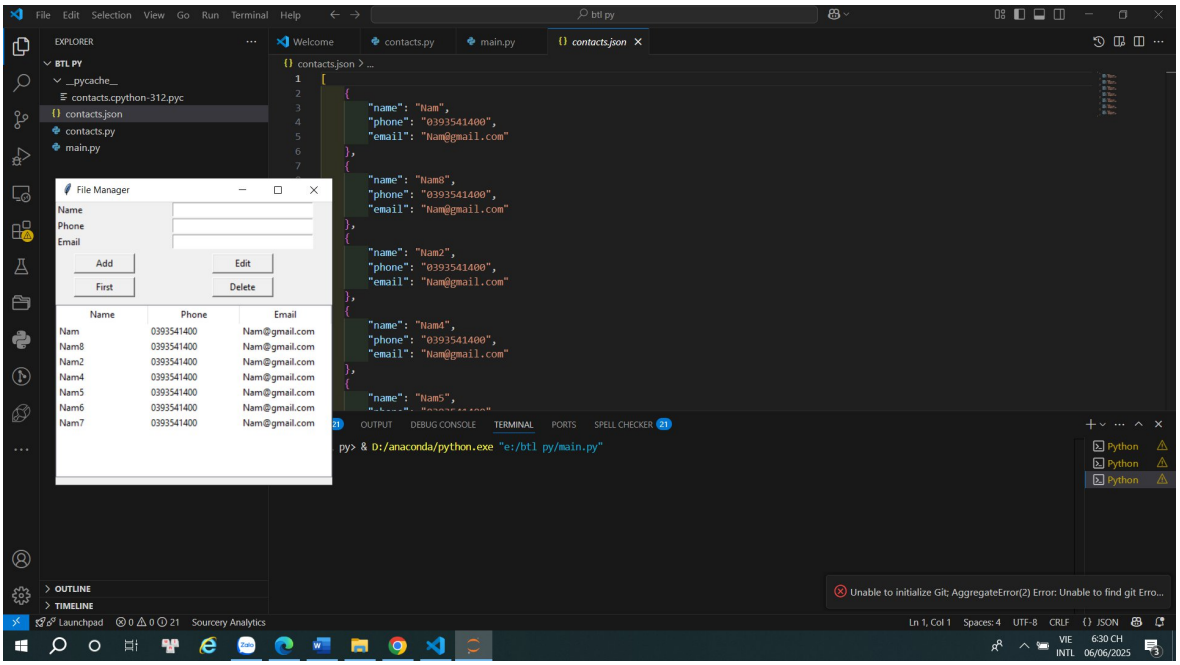


- Kết quả mong đợi: Dữ liệu được cập nhật cả trong giao diện và file contacts.json.
- Kết quả thực tế: Hệ thống cập nhật chính xác và không bị lỗi.

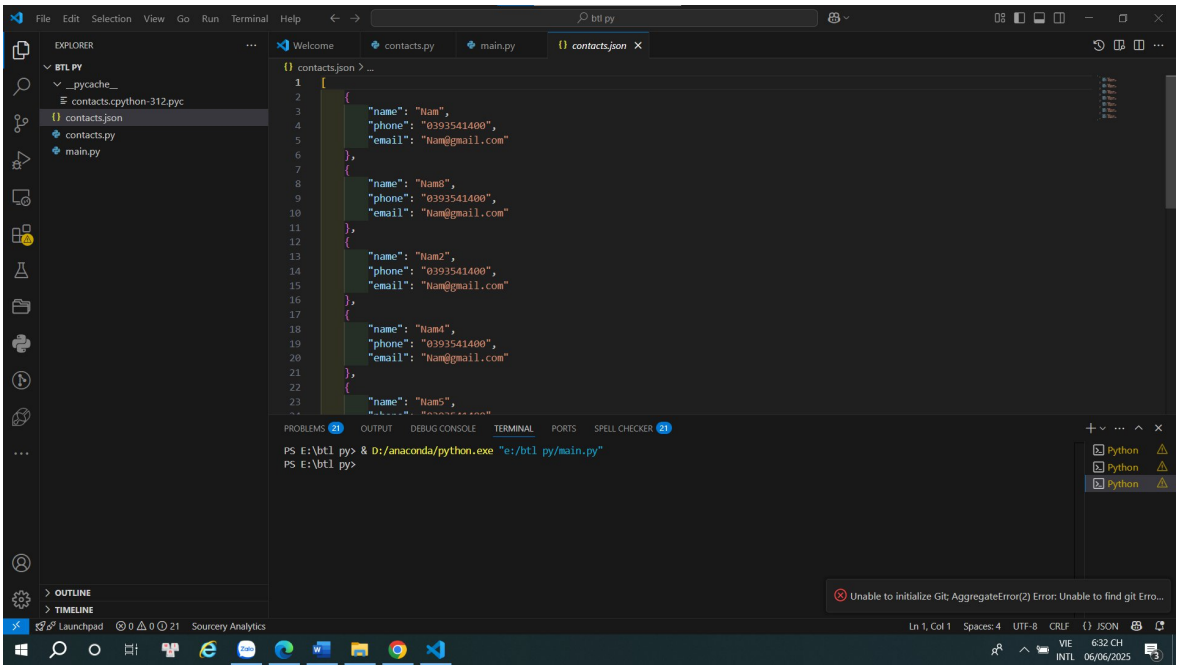


4.2.3. Chức năng Xoá liên hệ

- Mô tả: Chọn liên hệ “Nam3” và nhấn nút “Delete”.



- Kết quả mong đợi: Liên hệ bị xoá khỏi danh sách và file JSON.
- Kết quả thực tế: Hoạt động đúng, không còn dữ liệu đã xoá trong file.



4.2.4. Code file contacs.py

```
import json
import os

class ContactManager:
    def __init__(self, filename="contacts.json"):
        self.filename = filename
        self.contacts = []
        self.load_contacts()
```

```
•
•       def load_contacts(self):
•           if os.path.exists(self.filename):
•               try:
•                   with open(self.filename, "r") as f:
•                       self.contacts = json.load(f)
•               except json.JSONDecodeError:
•                   print("Lỗi định dạng JSON.")
•                   self.contacts = []
•
•       def save_contacts(self):
•           with open(self.filename, "w") as f:
•               json.dump(self.contacts, f, indent=4)
•
•       def add_contact(self, contact):
•           self.contacts.append(contact)
•           self.save_contacts()
•
•       def update_contact(self, index, contact):
•           self.contacts[index] = contact
•           self.save_contacts()
•
•       def delete_contact(self, index):
•           if 0 <= index < len(self.contacts):
•               del self.contacts[index]
•               self.save_contacts()
•
```

4.2.5. Code file main.py

```
import tkinter as tk
from tkinter import ttk, messagebox
from contacts import ContactManager

class ContactApp:
    def __init__(self, root):
        self.root = root
        self.root.title("File Manager")
        self.manager = ContactManager()

        self.create_widgets()
        self.load_treeview()

    def create_widgets(self):
        # Entry fields
        tk.Label(self.root, text="Name").grid(row=0, column=0, sticky="w")
        self.name_var = tk.StringVar()
        tk.Entry(self.root, textvariable=self.name_var, width=30).grid(row=0, column=1)
```

```
        tk.Label(self.root, text="Phone").grid(row=1, column=0, sticky="w")
        self.phone_var = tk.StringVar()
        tk.Entry(self.root, textvariable=self.phone_var, width=30).grid(row=1, column=1)

        tk.Label(self.root, text="Email").grid(row=2, column=0, sticky="w")
        self.email_var = tk.StringVar()
        tk.Entry(self.root, textvariable=self.email_var, width=30).grid(row=2, column=1)

        # Buttons
        tk.Button(self.root, text="Add", width=10, command=self.add_contact).grid(row=3,
column=0, pady=5)
        tk.Button(self.root, text="Edit", width=10, command=self.edit_contact).grid(row=3,
column=1, pady=5)

        tk.Button(self.root, text="First", width=10, command=self.select_first).grid(row=4,
column=0)
        tk.Button(self.root, text="Delete", width=10,
command=self.delete_contact).grid(row=4, column=1)

        # Treeview
        self.tree = ttk.Treeview(self.root, columns=("Name", "Phone", "Email"),
show="headings")
        for col in ("Name", "Phone", "Email"):
            self.tree.heading(col, text=col)
            self.tree.column(col, width=120)
        self.tree.grid(row=5, column=0, columnspan=2, pady=10)
        self.tree.bind("<<TreeviewSelect>>", self.load_selected)

    def get_form_data(self):
        return {
            "name": self.name_var.get(),
            "phone": self.phone_var.get(),
            "email": self.email_var.get()
        }

    def clear_form(self):
        self.name_var.set("")
        self.phone_var.set("")
        self.email_var.set("")

    def load_treeview(self):
        for row in self.tree.get_children():
            self.tree.delete(row)
        for idx, contact in enumerate(self.manager.contacts):
            self.tree.insert("", "end", iid=idx, values=(contact["name"], contact["phone"],
contact["email"]))
```

```
    def add_contact(self):
        contact = self.get_form_data()
        if not contact["name"] or not contact["phone"]:
            messagebox.showwarning("Thiếu dữ liệu", "Vui lòng nhập tên và số điện thoại.")
            return
        self.manager.add_contact(contact)
        self.load_treeview()
        self.clear_form()

    def edit_contact(self):
        selected = self.tree.selection()
        if not selected:
            messagebox.showwarning("Chọn liên hệ", "Chọn liên hệ để sửa.")
            return
        index = int(selected[0])
        contact = self.get_form_data()
        self.manager.update_contact(index, contact)
        self.load_treeview()
        self.clear_form()

    def delete_contact(self):
        selected = self.tree.selection()
        if not selected:
            messagebox.showwarning("Chọn liên hệ", "Chọn liên hệ để xoá.")
            return
        index = int(selected[0])
        self.manager.delete_contact(index)
        self.load_treeview()
        self.clear_form()

    def load_selected(self, event):
        selected = self.tree.selection()
        if selected:
            index = int(selected[0])
            contact = self.manager.contacts[index]
            self.name_var.set(contact["name"])
            self.phone_var.set(contact["phone"])
            self.email_var.set(contact["email"])

    def select_first(self):
        items = self.tree.get_children()
        if items:
            self.tree.selection_set(items[0])
            self.tree.focus(items[0])
            self.load_selected(None)

if __name__ == "__main__":
    root = tk.Tk()
```

```
app = ContactApp(root)
root.mainloop()
```

4.3. Kết luận

4.3.1. Những gì sản phẩm làm được.

Với việc hoàn thành ứng dụng quản lý danh bạ GUI, chúng ta đã thành công xây dựng một công cụ thiết thực, đáp ứng đầy đủ các yêu cầu đã đặt ra. Ứng dụng không chỉ cho phép người dùng thêm, sửa, xóa liên hệ một cách dễ dàng thông qua giao diện trực quan với các thành phần như Entry, Buttons và Treeview, mà còn đảm bảo dữ liệu được lưu trữ ổn định và an toàn trong tệp JSON.

Khả năng đọc/ghi dữ liệu JSON sử dụng module `json` đã được triển khai hiệu quả, cùng với cơ chế bắt lỗi định dạng JSON giúp tăng cường tính bền vững của ứng dụng. Chức năng tìm kiếm theo tên cũng là một điểm cộng lớn, nâng cao trải nghiệm người dùng bằng cách giúp họ nhanh chóng truy cập thông tin mong muốn. Các bước triển khai rõ ràng từ việc tạo module `contacts.py` quản lý dữ liệu đến việc thiết kế giao diện người dùng và ánh xạ các chức năng đã chứng minh một quy trình phát triển có hệ thống. Ứng dụng tự động tải dữ liệu khi khởi động, đảm bảo tính liên tục và tiện lợi.

4.3.2. Những gì đã học được.

Trong quá trình phát triển ứng dụng này, chúng ta đã tiếp thu được nhiều kiến thức và kỹ năng quan trọng:

- Lập trình giao diện người dùng (GUI) với Tkinter: Nắm vững cách sử dụng các widget cơ bản như Entry, Button, Label, và đặc biệt là Treeview để hiển thị dữ liệu dạng bảng.
- Xử lý dữ liệu JSON: Hiểu rõ cách đọc và ghi dữ liệu từ/vào tệp JSON sử dụng thư viện json của Python, bao gồm cả việc xử lý các trường hợp lỗi định dạng.
- Quản lý dữ liệu hiệu quả: Xây dựng cấu trúc lớp để quản lý danh sách liên hệ, đảm bảo tính đóng gói và dễ bảo trì.
- Tổ chức mã nguồn: Áp dụng nguyên tắc module hóa (chia thành contacts.py và phần GUI riêng biệt) giúp mã nguồn gọn gàng, dễ đọc và tái sử dụng.
- Xử lý sự kiện: Hiểu cách liên kết các sự kiện từ giao diện người dùng (nhấn nút) với các hàm xử lý logic nghiệp vụ.
- Kiểm tra và gỡ lỗi: Thực hiện kiểm tra các tính năng như thêm, sửa, xóa và tìm kiếm để đảm bảo ứng dụng hoạt động chính xác.

4.3.3. Các cải tiến trong tương lai

Mặc dù ứng dụng đã hoàn thành các yêu cầu đặt ra, vẫn có nhiều hướng để cải thiện và mở rộng tính năng, nâng cao trải nghiệm người dùng và tính mạnh mẽ của sản phẩm:

- Tính năng xác thực dữ liệu: Thêm kiểm tra đầu vào cho các trường như số điện thoại (chỉ cho phép số, định dạng chuẩn) và email (kiểm tra định dạng email hợp lệ) để tránh nhập liệu sai.
- Phân trang hoặc cuộn vô hạn: Đối với danh bạ có số lượng liên hệ lớn, việc triển khai phân trang hoặc cuộn vô hạn sẽ giúp quản lý và hiển thị dữ liệu hiệu quả hơn, tránh tình trạng giao diện bị chậm.
- Tính năng sắp xếp: Cho phép người dùng sắp xếp danh bạ theo tên, số điện thoại hoặc email để dễ dàng tìm kiếm và quản lý.
- Xuất/nhập dữ liệu: Bổ sung chức năng xuất dữ liệu sang các định dạng khác như CSV, Excel hoặc nhập dữ liệu từ các nguồn khác.

- Giao diện người dùng nâng cao: Cải thiện thẩm mỹ giao diện bằng cách sử dụng các thư viện GUI khác mạnh mẽ hơn như PyQt hoặc Kivy, hoặc thêm các hiệu ứng chuyển động, biểu tượng trực quan.
- Chức năng sao lưu/khôi phục: Tích hợp tính năng tự động sao lưu dữ liệu hoặc cho phép người dùng sao lưu/khôi phục thủ công để phòng tránh mất mát dữ liệu.
- Tối ưu hóa hiệu suất tìm kiếm: Đối với danh bạ rất lớn, có thể xem xét các thuật toán tìm kiếm hiệu quả hơn hoặc sử dụng cơ sở dữ liệu thay vì file JSON để tăng tốc độ.
- Hỗ trợ đa ngôn ngữ: Cho phép người dùng chuyển đổi ngôn ngữ giao diện.

Việc tiếp tục phát triển theo những hướng này sẽ biến ứng dụng quản lý danh bạ thành một công cụ toàn diện và thân thiện hơn với người dùng.

Lời cảm ơn

Chúng tôi xin gửi lời cảm ơn chân thành đến tất cả những cá nhân và tổ chức đã đóng góp và hỗ trợ trong quá trình hoàn thành báo cáo và phát triển ứng dụng quản lý danh bạ GUI này.

Đầu tiên, chúng tôi xin bày tỏ lòng biết ơn sâu sắc đến Thầy Đỗ Duy Cốp đã tận tình giảng dạy, truyền đạt kiến thức và kinh nghiệm quý báu. Những hướng dẫn, góp ý và sự động viên của Thầy/Cô chính là kim chỉ nam giúp chúng tôi định hướng và hoàn thành tốt nhiệm vụ này.

Chúng tôi cũng xin cảm ơn Trường Đại Học Công Nghiệp Thái Nguyên đã tạo mọi điều kiện thuận lợi về cơ sở vật chất, tài liệu và môi trường học tập, nghiên cứu, giúp chúng tôi có thể tập trung phát triển sản phẩm một cách hiệu quả nhất.

Một lần nữa, chúng tôi xin chân thành cảm ơn sự giúp đỡ, hỗ trợ và đồng hành của tất cả mọi người. Đây là nguồn động lực to lớn giúp chúng tôi không ngừng học hỏi và phát triển.

Trân trọng,

Tài Liệu Tham Khảo

[1]. Python Official Documentation: https://docs.python.org/

[2]. Tkinter GUI Programming Guide: https://tkdocs.com/

[3]. Stack Overflow – Cộng đồng hỗ trợ lập trình viên: https://stackoverflow.com/

[4]. W3Schools Python Tutorial: https://www.w3schools.com/python/

[5]. GeeksforGeeks – Python JSON & GUI Examples: https://www.geeksforgeeks.org/