TRƯỜNG ĐẠI HỌC ĐẠI NAM

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**HỌC PHẦN: LẬP TRÌNH PYTHON**

**ĐỀ TÀI: QUẢN LÍ CÔNG VIỆC HÀNG NGÀY**

**Giảng viên: ThS. Lê Tuấn Anh**

**Sinh viên thực hiện:**

1. **Nguyễn Lê Gia Bảo – 1871020074**
2. **Phùng Quang Trung – 1871020590**

**Hà Nội, năm 2025**

TRƯỜNG ĐẠI HỌC ĐẠI NAM

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**HỌC PHẦN: LẬP TRÌNH PYTHON**

**ĐỀ TÀI: QUẢN LÍ CÔNG VIỆC HÀNG NGÀY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TT | Mã sv | Họ và Tên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 1 | ***1871020074*** | Nguyễn Lê Gia Bảo | 07/03/2006 |  |  |
| 2 | ***1871020590*** | Phùng Quang Trung | 15/02/2006 |  |  |

|  |  |
| --- | --- |
| **CÁN BỘ CHẤM THI 1**  **ThS. Lê Tuấn Anh** | **CÁN BỘ CHẤM THI 2** |

### 

**Hà Nội, năm 2025**

**LỜI NÓI ĐẦU**

Với nhịp sống bận rộn của sinh viên, từ việc chạy deadline bài tập, ôn thi, đến quản lý những công việc cá nhân, đôi khi mọi thứ trở nên rối rắm và khó theo dõi. Vì vậy, Nhóm 5 đã được giao đề tài "Quản lý công việc hàng ngày" cho bài tập lớn môn Lập trình Python, với mục tiêu xây dựng một ứng dụng đơn giản nhưng thiết thực, chạy trên giao diện dòng lệnh, giúp người dùng sắp xếp và theo dõi công việc hằng ngày một cách hiệu quả, như một cuốn sổ tay điện tử tiện lợi.

Qua đề tài này là cơ hội để nhóm áp dụng những kiến thức Python đã học, từ cách sử dụng list, dictionary, đến việc lưu và đọc dữ liệu từ file. Hơn thế nữa, đây còn là dịp để học cách làm việc chung, từ phân chia nhiệm vụ, thảo luận ý tưởng, đến cùng nhau sửa lỗi khi chương trình "trở chứng". Có những lúc nhóm ngồi hàng giờ để tìm cách tối ưu code, hoặc hào hứng khi thấy chương trình chạy đúng sau bao lần thử nghiệm thất bại. Những trải nghiệm này không chỉ giúp nhóm hiểu sâu hơn về lập trình Python mà còn rèn luyện kỹ năng giải quyết vấn đề, tư duy logic, và tinh thần đồng đội.

Báo cáo này sẽ trình bày chi tiết hành trình thực hiện dự án của Nhóm 5: từ việc phân tích yêu cầu, thiết kế các chức năng, viết mã nguồn, kiểm tra chương trình, đến kết quả đạt được và những bài học kinh nghiệm. Nhóm hy vọng ứng dụng này không chỉ đáp ứng yêu cầu của bài tập lớn mà còn trở thành một công cụ hữu ích, giúp bất kỳ ai cũng có thể quản lý công việc của mình một cách khoa học hơn.

Nhóm xin gửi lời cảm ơn chân thành đến thầy Lê Tuấn Anh đã tận tình hướng dẫn, giải đáp thắc mắc, và cung cấp những kiến thức nền tảng quý báu. Rất mong nhận được ý kiến đóng góp từ thầy để báo cáo và chương trình của nhóm được hoàn thiện hơn.

**MỤC LỤC**

[CHƯƠNG 1. GIỚI THIỆU VỀ MỘT SỐ KHÁI NIỆM TRONG PYTHON 8](#_Toc202519942)

[1.1. Giới thiệu tổng quan về Python 8](#_Toc202519943)

[1.2. Biến và kiểu dữ liệu 8](#_Toc202519944)

[1.3. Câu lệnh điều kiện (if, elif, else) 9](#_Toc202519945)

[1.4. Vòng lặp (for, while) 10](#_Toc202519946)

[1.5. Hàm và cách sử dụng hàm 10](#_Toc202519947)

[1.6. Danh sách, từ điển và tuple 12](#_Toc202519948)

[1.7. Xử lí tệp (đọc/ghi tệp văn bản và JSON) 13](#_Toc202519949)

[1.8. Xử lí ngoại lệ (try-except) 13](#_Toc202519950)

[CHƯƠNG 2. GIỚI THIỆU, MÔ TẢ ĐỀ TÀI 15](#_Toc202519951)

[2.1. Lý do chọn đề tài 15](#_Toc202519952)

[2.2. Mục tiêu của ứng dụng 15](#_Toc202519953)

[2.3. Phạm vi áp dụng 16](#_Toc202519954)

[2.4. Các yêu cầu chức năng 16](#_Toc202519955)

[2.5. Các yêu cầu phi chức năng 17](#_Toc202519956)

[2.6. Công cụ và ngôn ngữ sử dụng 17](#_Toc202519957)

[2.7. Mô hình dữ liệu và lưu trữ 17](#_Toc202519958)

[CHƯƠNG 3. PHÁT TRIỂN VÀ TRIỂN KHAI ỨNG DỤNG 20](#_Toc202519959)

[3.1. Cấu trúc chương trình và cách tổ chức mã nguồn 20](#_Toc202519960)

[3.2. Khởi tạo và lưu trữ dữ liệu công việc 20](#_Toc202519961)

[3.3. Thêm công việc mới 21](#_Toc202519962)

[3.4 Hiển thị danh sách công việc 23](#_Toc202519963)

[3.5. Cập nhật thông tin công việc 23](#_Toc202519964)

[3.6. Xóa công việc 26](#_Toc202519965)

[3.7. Lọc và tìm kiếm công việc 27](#_Toc202519966)

[3.8. Sắp xếp và thống kê 29](#_Toc202519967)

[3.9. Hàm chính và Menu chương trình 30](#_Toc202519968)

[3.10. Kiểm thử chương trình 32](#_Toc202519969)

[3.10.1. Thêm công việc vào danh sách và hiển thị danh sách 32](#_Toc202519970)

[3.10.2. Xem công việc có trạng thái chưa bắt đầu 33](#_Toc202519971)

[3.10.3. Xem công việc theo trạng thái hoàn thành 33](#_Toc202519972)

[3.10.4. Tìm công việc theo tên công việc 34](#_Toc202519973)

[3.10.5. Tìm công việc theo mã công việc 34](#_Toc202519974)

[3.10.6. Tìm công việc theo tên người phụ trách 35](#_Toc202519975)

[3.10.7. Cập nhật tên công việc 35](#_Toc202519976)

[3.10.8. Cập nhật trạng thái công việc 36](#_Toc202519977)

**MỤC LỤC HÌNH ẢNH**

[Hình 3. 1 Kiểm thử chức năng 1 và 2 31](#_Toc202519914)

[Hình 3. 2 Kết quả kiểm thử chức năng 1 và 2 32](#_Toc202519915)

[Hình 3. 3 Kết quả kiểm thử chức năng 3 32](#_Toc202519916)

[Hình 3. 4 Kết quả kiểm thử chức năng 4 33](#_Toc202519917)

[Hình 3. 5 Kết quả kiểm thứ chức năng 5 33](#_Toc202519918)

[Hình 3. 6 Kết quả kiểm thử chức năng 6 34](#_Toc202519919)

[Hình 3. 7 Kết quả kiểm thử chức năng 7 34](#_Toc202519920)

[Hình 3. 8 Kiểm thứ chức năng 8 34](#_Toc202519921)

[Hình 3. 9 Kết quả kiểm thử chức năng 8 35](#_Toc202519922)

[Hình 3. 10 Kiểm thứ chức năng 9 35](#_Toc202519923)

[Hình 3. 11 Kết quả kiểm thử chức năng 9 36](#_Toc202519924)

**MỤC LỤC BẢNG**

[Bảng 1. 1 Các kiểu dữ liệu trong Python 8](#_Toc202517117)

# CHƯƠNG 1. GIỚI THIỆU VỀ MỘT SỐ KHÁI NIỆM TRONG PYTHON

## 1.1. Giới thiệu tổng quan về Python

Python là một ngôn ngữ lập trình bậc cao, thông dịch, có cú pháp đơn giản, dễ học và dễ đọc. Python được phát triển bởi Guido van Rossum và phát hành lần đầu vào năm 1991. Với triết lý “rõ ràng hơn phức tạp”, Python rất phù hợp cho người mới bắt đầu học lập trình.

Một số đặc điểm nổi bật của Python:

* Cú pháp rõ ràng, dễ hiểu và gần với ngôn ngữ tự nhiên.
* Quản lý bộ nhớ tự động, không cần khai báo kiểu biến.
* Hỗ trợ nhiều mô hình lập trình: thủ tục, hướng đối tượng, hàm.
* Thư viện chuẩn phong phú cùng với khả năng mở rộng bằng thư viện bên ngoài.
* Ứng dụng rộng rãi trong các lĩnh vực như: phát triển web, trí tuệ nhân tạo, tự động hóa, phân tích dữ liệu,...

Trong đề tài này, Python được sử dụng để xây dựng một ứng dụng dòng lệnh đơn giản giúp người dùng quản lý các công việc hằng ngày, với chức năng như thêm, xóa, chỉnh sửa, lưu trữ công việc,...

## 1.2. Biến và kiểu dữ liệu

Biến là tên đại diện cho một vùng nhớ trong máy tính, dùng để lưu trữ dữ liệu. Python không yêu cầu khai báo kiểu dữ liệu khi khởi tạo biến — kiểu sẽ được suy luận tự động dựa trên giá trị gán.

ten\_cong\_viec = "Viết báo cáo"

do\_uu\_tien = "Cao"

so\_luong\_cong\_viec = 5

|  |  |  |
| --- | --- | --- |
| Kiểu dữ liệu | Mô tả | Ví dụ |
| Int | Số nguyên | 3, 2025 |
| float | Số thực | 3.14, 100.5 |
| Str | Chuỗi ký tự | "Công việc A" |
| bool | Giá trị logic | True, False |

*Bảng 1. 1 Các kiểu dữ liệu trong Python*

## 1.3. Câu lệnh điều kiện (if, elif, else)

Câu lệnh điều kiện dùng để kiểm tra một điều kiện logic. Nếu điều kiện đúng (True), chương trình sẽ thực hiện một khối lệnh tương ứng.

Cú pháp:

if điều\_kiện:

    # hành động khi điều\_kiện đúng

elif điều\_kiện\_khác:

    # hành động khi điều\_kiện\_khác đúng

else:

    # hành động nếu không điều kiện nào đúng

Ví dụ trong quản lí công việc:

if cong\_viec["trang\_thai"] == "Hoàn thành":

    print("Công việc đã xong.")

else:

    print("Công việc chưa hoàn thành.")

Cách hoạt động: Python sẽ lần lượt kiểm tra từng điều kiện từ trên xuống. Khi gặp điều kiện đúng đầu tiên, nó thực hiện khối lệnh tương ứng và bỏ qua các điều kiện còn lại.

## 1.4. Vòng lặp (for, while)

Vòng lặp dùng để lặp lại một khối lệnh nhiều lần.

Vòng lặp for: dùng để lặp qua một dãy (list, dict, string,...)

for cong\_viec in danh\_sach\_cong\_viec:

    print(cong\_viec)

Vòng lặp while: lặp lại khi điều kiện còn đúng

Cách hoạt động:

* for sẽ lấy từng phần tử từ danh sách và thực hiện khối lệnh.
* while sẽ kiểm tra điều kiện trước mỗi vòng, nếu True thì lặp tiếp.

i = 0

while i < 5:

    print(i)

    i += 1

## 1.5. Hàm và cách sử dụng hàm

Hàm là một khối lệnh được đặt tên để thực hiện một tác vụ cụ thể, có thể gọi lại nhiều lần giúp tránh lặp mã.

Cách định nghĩa:

def ten\_ham(tham\_so):

    # khối lệnh

    return ket\_qua

Ví dụ:

def in\_cong\_viec(cong\_viec):

    print(f"Tên: {cong\_viec['ten']}")

    print(f"Trạng thái: {cong\_viec['trang\_thai']}")

Cách hoạt động: Khi hàm được gọi, Python sẽ chuyển đến khối lệnh trong hàm, truyền các tham số và thực hiện các lệnh, sau đó quay lại vị trí gọi.

Hàm main() – Hàm điều phối chính của chương trình

Trong các chương trình lớn hoặc có nhiều chức năng, người ta thường định nghĩa một hàm main() làm điểm bắt đầu thực thi chính của chương trình. Đây là nơi gọi các chức năng khác như thêm công việc, hiển thị, tìm kiếm,…

Ví dụ:

def main():

    danh\_sach\_cong\_viec = []

    while True:

        print("1. Thêm công việc")

        print("2. Thoát")

        lua\_chon = input("Chọn chức năng: ")

        if lua\_chon == '1':

            ten = input("Nhập tên công việc: ")

            trang\_thai = input("Nhập trạng thái: ")

            cong\_viec = {"ten": ten, "trang\_thai": trang\_thai}

            danh\_sach\_cong\_viec.append(cong\_viec)

            print("Đã thêm công việc.")

        elif lua\_chon == '2':

            print("Thoát chương trình.")

            break

        else:

            print("Lựa chọn không hợp lệ.")

Cách gọi hàm main():

Để đảm bảo hàm main() chỉ được gọi khi file đang chạy trực tiếp (không phải được nhập dưới dạng module), Python sử dụng điều kiện sau:

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

    main()

Cách hoạt động:

* Khi chạy chương trình, Python kiểm tra xem file hiện tại có phải là file chính không.
* Nếu có, main() sẽ được gọi, và toàn bộ xử lý chương trình bắt đầu từ đó.
* Việc sử dụng main() giúp tách rõ giữa phần định nghĩa hàm và phần thực thi, làm cho chương trình dễ bảo trì hơn.

## 1.6. Danh sách, từ điển và tuple

Danh sách (list): lưu nhiều phần tử, có thể thay đổi, có thứ tự

cong\_viec = ["Làm bài", "Dọn nhà", "Học Python"]

Từ điển (dict): lưu theo cặp key: value

cong\_viec = {

    "ten": "Học Python",

    "uu\_tien": "Cao",

    "trang\_thai": "Chưa bắt đầu"

}

Tuple: giống list nhưng không thay đổi được

thoi\_gian = ("05/06/2025", "07/06/2025")

Cách hoạt động: Python lưu các phần tử trong vùng nhớ liền kề. Với dict, tra cứu theo khóa (key) rất nhanh, phù hợp để quản lý công việc có nhiều thông tin như tên, trạng thái, hạn, v.v.

## 1.7. Xử lí tệp (đọc/ghi tệp văn bản và JSON)

Để lưu công việc giữa các lần chạy chương trình, cần ghi và đọc từ tệp.

Ghi vào file JSON:

import json

with open("data.json", "w", encoding="utf-8") as f:

    json.dump(danh\_sach\_cong\_viec, f, ensure\_ascii=False, indent=4)

Đọc từ file JSON:

with open("data.json", "r", encoding="utf-8") as f:

    danh\_sach\_cong\_viec = json.load(f)

Cách hoạt động:

* Khi ghi, json.dump sẽ chuyển từ dict Python sang chuỗi JSON và ghi vào file.
* Khi đọc, json.load sẽ chuyển từ JSON trong file về dict Python.

## 1.8. Xử lí ngoại lệ (try-except)

Dùng để ngăn chương trình bị dừng đột ngột do lỗi (ví dụ: nhập sai kiểu dữ liệu, file không tồn tại,...)

Ví dụ:

try:

    ngay = int(input("Nhập ngày (số): "))

except ValueError:

    print("Lỗi: bạn phải nhập số.")

Cách hoạt động:

Python thử thực hiện khối lệnh trong try. Nếu có lỗi xảy ra, nó chuyển sang except và thực hiện lệnh xử lý lỗi ở đó.

# CHƯƠNG 2. GIỚI THIỆU, MÔ TẢ ĐỀ TÀI

## 2.1. Lý do chọn đề tài

Trong cuộc sống hiện đại, mỗi người đều có rất nhiều công việc cần hoàn thành mỗi ngày, từ các công việc học tập, làm việc đến việc cá nhân, gia đình. Việc ghi nhớ và quản lý chúng một cách thủ công (bằng sổ tay, ghi chú giấy hoặc trí nhớ) dễ dẫn đến việc quên, chồng chéo hoặc trì hoãn công việc, gây ra áp lực không đáng có.

Trong khi đó, với sự phát triển mạnh mẽ của công nghệ thông tin, đặc biệt là các ngôn ngữ lập trình đơn giản và hiệu quả như Python, việc xây dựng một ứng dụng quản lý công việc hằng ngày trở nên dễ tiếp cận hơn bao giờ hết.

Lý do chọn đề tài:

* Giải quyết nhu cầu thực tế về quản lý công việc hằng ngày.
* Rèn luyện kỹ năng lập trình thực tế bằng Python.
* Nâng cao hiểu biết về cách tổ chức dữ liệu, xử lý logic, lưu trữ, và giao tiếp với người dùng qua dòng lệnh.
* Mở rộng nền tảng để phát triển ứng dụng nâng cao hơn trong tương lai (có giao diện đồ họa, tích hợp đám mây,...)

## 2.2. Mục tiêu của ứng dụng

Mục tiêu tổng quát của đề tài là xây dựng một chương trình dòng lệnh bằng Python cho phép người dùng quản lý các công việc hằng ngày một cách dễ dàng và hiệu quả. Các mục tiêu cụ thể bao gồm:

* Cho phép người dùng thêm, sửa, xóa, xem công việc một cách trực quan.
* Hỗ trợ tìm kiếm, lọc công việc theo trạng thái, mức độ ưu tiên, hạn hoàn thành,...
* Lưu trữ dữ liệu công việc bằng tệp JSON để sử dụng lâu dài.
* Hiển thị thông tin dưới dạng dễ đọc, rõ ràng.
* Cấu trúc chương trình rõ ràng, dễ bảo trì và phát triển.

## 2.3. Phạm vi áp dụng

Ứng dụng được xây dựng hướng đến người dùng cá nhân với nhu cầu theo dõi, sắp xếp, quản lý công việc hằng ngày trên máy tính cá nhân.

Phạm vi cụ thể:

* Giao diện dòng lệnh (CLI), không có giao diện đồ họa.
* Chạy trên môi trường Python 3, không yêu cầu cài đặt nâng cao.
* Dữ liệu được lưu cục bộ dưới dạng tệp JSON.
* Không yêu cầu tài khoản đăng nhập, bảo mật nâng cao.
* Phù hợp cho học sinh, sinh viên, nhân viên văn phòng muốn quản lý lịch làm việc cá nhân.

## 2.4. Các yêu cầu chức năng

Đây là các chức năng chính mà hệ thống cần có để đáp ứng mục tiêu đề ra:

1. Thêm công việc mới: Nhập tên, mô tả, trạng thái, mức ưu tiên, hạn hoàn thành, nhóm công việc,...
2. Xem danh sách công việc: Hiển thị toàn bộ công việc hiện có.
3. Xóa công việc: Xóa một công việc cụ thể theo tên hoặc mã.
4. Cập nhật công việc: Cho phép thay đổi thông tin công việc (tên, trạng thái, ưu tiên,...).
5. Tìm kiếm công việc: Tìm công việc theo từ khóa (tên, trạng thái, hạn,...).
6. Lọc công việc: Hiển thị theo trạng thái, mức ưu tiên hoặc nhóm công việc.
7. Sắp xếp công việc: Sắp xếp theo ngày tạo, hạn hoàn thành, mức độ ưu tiên.
8. Sao lưu và phục hồi dữ liệu: Lưu và đọc dữ liệu từ tệp JSON.
9. Thống kê: Hiển thị tổng số công việc, số công việc hoàn thành/chưa hoàn thành.

## 2.5. Các yêu cầu phi chức năng

Ngoài các chức năng, hệ thống cũng cần đảm bảo một số yêu cầu về hiệu năng và trải nghiệm:

* Tính đơn giản: Giao diện dòng lệnh dễ dùng, dễ hiểu.
* Dễ bảo trì: Mã nguồn có cấu trúc rõ ràng, chia thành các hàm riêng biệt.
* Tính bền vững: Dữ liệu không bị mất khi thoát chương trình nhờ lưu vào tệp JSON.

## 2.6. Công cụ và ngôn ngữ sử dụng

Ngôn ngữ lập trình:

* Python 3.10+  
  Lý do chọn Python:
  + Dễ học, dễ đọc.
  + Có thư viện chuẩn mạnh mẽ hỗ trợ xử lý tệp, thời gian, dữ liệu.
  + Phù hợp với ứng dụng đơn giản nhưng giàu chức năng.

Thư viện sử dụng:

* json: xử lý lưu trữ dữ liệu.

Môi trường phát triển:

* Máy tính cá nhân, hệ điều hành Windows hoặc Linux.
* Trình soạn thảo: VS Code hoặc IDLE, hoặc bất kỳ editor nào hỗ trợ Python.

## 2.7. Mô hình dữ liệu và lưu trữ

Dạng dữ liệu sử dụng:

* Dữ liệu mỗi công việc được lưu dưới dạng dict, chứa các thông tin như:

{

    "Hoàn thành báo cáo tuần": {

        "Mã công việc": "CV001",

        "Mô tả": "Tổng hợp nội dung làm việc trong tuần và lập báo cáo",

        "Ngày tạo": "05/06/2025",

        "Hạn hoàn thành": "08/06/2025",

        "Trạng thái": "Đang thực hiện",

        "Người phụ trách": "Bảo"

    },

    "Dọn dẹp phòng ngủ": {

        "Mã công việc": "CV002",

        "Mô tả": "Quét, lau, sắp xếp lại đồ đạc trong phòng ngủ",

        "Ngày tạo": "06/06/2025",

        "Hạn hoàn thành": "06/06/2025",

        "Trạng thái": "Chưa bắt đầu",

        "Người phụ trách": "Trung"

    },

    "Mua quà sinh nhật cho mẹ": {

        "Mã công việc": "CV003",

        "Mô tả": "Chọn và mua quà sinh nhật",

        "Ngày tạo": "01/06/2025",

        "Hạn hoàn thành": "10/06/2025",

        "Trạng thái": "Chưa bắt đầu",

        "Người phụ trách": "Bảo"

    }

}

Tổ chức tổng thể:

* Danh sách các công việc được lưu trong một biến dạng dict lớn:

danh\_sach\_cong\_viec[ten\_cong\_viec] = {

            'Mã công việc': ma\_cong\_viec,

            'Mô tả': mo\_ta,

            'Ngày tạo': ngay\_tao,

            'Hạn hoàn thành': han\_hoan\_thanh,

            'Trạng thái': trang\_thai,

            'Người phụ trách': nguoi\_phu\_trach

        }

Lưu trữ dữ liệu:

* Dữ liệu được ghi vào tệp cong\_viec.json bằng thư viện json.
* Khi khởi động chương trình, dữ liệu sẽ được nạp lại từ file nếu có.

Ưu điểm mô hình:

* Dễ mở rộng (có thể thêm thuộc tính mới mà không ảnh hưởng nhiều).
* Không cần cơ sở dữ liệu phức tạp.
* Dễ sao lưu, chia sẻ dữ liệu (tệp JSON có thể gửi qua email, sao chép,...)

# CHƯƠNG 3. PHÁT TRIỂN VÀ TRIỂN KHAI ỨNG DỤNG

## 3.1. Cấu trúc chương trình và cách tổ chức mã nguồn

Mã nguồn được tổ chức thành một file Python duy nhất, trong đó:

* Biến toàn cục danh\_sach\_cong\_viec lưu trữ dữ liệu công việc.
* Dữ liệu được lưu và đọc bằng file JSON (du\_lieu.json) thông qua thư viện json.
* Mỗi chức năng được đóng gói thành một hàm riêng biệt.
* Chương trình khởi chạy trong khối if \_\_name\_\_ == "\_\_main\_\_" với menu điều khiển chính.

Cách tổ chức này giúp mã nguồn gọn gàng, dễ đọc, thuận tiện cho việc mở rộng và bảo trì.

## 3.2. Khởi tạo và lưu trữ dữ liệu công việc

Dữ liệu công việc được tổ chức dưới dạng dict trong Python, có cấu trúc như sau:

{

  "Tên công việc": {

    "Mã công việc": "CV01",

    "Mô tả": "Chi tiết công việc",

    "Ngày tạo": "01/07/2025",

    "Hạn hoàn thành": "10/07/2025",

    "Trạng thái": "Chưa bắt đầu",

    "Người phụ trách": "Nguyễn Văn A"

  }

}

Việc đọc/ghi dữ liệu được thực hiện bằng:

* doc\_danh\_sach\_cong\_viec(): đọc từ JSON.

Hàm doc\_danh\_sach\_cong\_viec():

def doc\_danh\_sach\_cong\_viec():

    try:

        with open('D:\\Python\\Du\_an\_Gia\_Bao\\du\_lieu.json', 'r', encoding = "utf-8") as f:

            global danh\_sach\_cong\_viec

            danh\_sach\_cong\_viec = json.load(f)

    except FileNotFoundError:

        print("File không tồn tại. Vui lòng kiểm tra đường dẫn")

* luu\_danh\_sach\_cong\_viec(): ghi vào JSON

Hàm luu\_danh\_sach\_cong\_viec():

def luu\_danh\_sach\_cong\_viec():

    try:

        with open('D:\\Python\\Du\_an\_Gia\_Bao\\du\_lieu.json', 'w', encoding = "utf-8") as f:

            json.dump(danh\_sach\_cong\_viec, f, ensure\_ascii=False, indent=4)

    except Exception as e:

        print(f"Lỗi khi lưu danh sách công việc: {e}")

Điều này cho phép dữ liệu được lưu trữ vĩnh viễn giữa các lần sử dụng chương trình.

## 3.3. Thêm công việc mới

Hàm them\_cong\_viec() cho phép thêm nhiều công việc cùng lúc. Người dùng nhập::

* Tên công việc
* Mã công việc, mô tả
* Ngày tạo và hạn hoàn thành
* Trạng thái (có hướng dẫn chọn)
* Người phụ trách

Sau khi nhập, dữ liệu được cập nhật vào từ điển và lưu vào file JSON.

Hàm them\_cong\_viec():

def them\_cong\_viec():

    so\_cong\_viec =int(input("Nhập số lượng công việc muốn thêm: "))

    for i in range(so\_cong\_viec):

        print(f"Nhập thông tin công việc thứ {i + 1}:")

        ten\_cong\_viec = input("Nhập tên công việc: ")

        ma\_cong\_viec = input("Nhập mã công việc: ")

        mo\_ta = input("Nhập mô tả công việc: ")

        ngay\_tao = input("Nhập ngày tạo công việc (dd/mm/yyyy): ")

        han\_hoan\_thanh = input("Nhập hạn hoàn thành công việc (dd/mm/yyyy): ")

        while True:

            print("Công việc có các trạng thái sau: ")

            print("1. Chưa bắt đầu")

            print("2. Đang thực hiện")

            print("3. Hoàn thành")

            trang\_thai = input("Nhập trạng thái công việc (1/2/3): ")

            if trang\_thai == '1':

                trang\_thai = 'Chưa bắt đầu'

                break

            elif trang\_thai == '2':

                trang\_thai = 'Đang thực hiện'

                break

            elif trang\_thai == '3':

                trang\_thai = 'Hoàn thành'

                break

            else:

                print("Chọn sai lựa chọn, vui lòng nhập lại")

                print("=" \* 20)

        nguoi\_phu\_trach = input("Nhập tên người phụ trách công việc: ")

        danh\_sach\_cong\_viec[ten\_cong\_viec] = {

            'Mã công việc': ma\_cong\_viec,

            'Mô tả': mo\_ta,

            'Ngày tạo': ngay\_tao,

            'Hạn hoàn thành': han\_hoan\_thanh,

            'Trạng thái': trang\_thai,

            'Người phụ trách': nguoi\_phu\_trach

        }

    luu\_danh\_sach\_cong\_viec()

## 3.4 Hiển thị danh sách công việc

Hàm xem\_danh\_sach\_cong\_viec() in toàn bộ danh sách ra màn hình, bao gồm tất cả thông tin chi tiết của từng công việc, có phân cách rõ ràng bằng dấu =.

Hàm xem\_danh\_sach\_cong\_viec():

def xem\_danh\_sach\_cong\_viec():

    if danh\_sach\_cong\_viec == {}:

        print("Danh sách công việc trống.")

    else:

        print("Danh sách công việc:")

        for cong\_viec, thong\_tin in danh\_sach\_cong\_viec.items():

            print(f"{cong\_viec}: ")

            for key, value in thong\_tin.items():

                print(f"{key}: {value}")

            print("=" \* 40)

## 3.5. Cập nhật thông tin công việc

Có 3 chức năng cập nhật:

* cap\_nhat\_ten\_cong\_viec(): đổi tên công việc (khóa của dict)
* cap\_nhat\_trang\_thai\_cong\_viec(): cập nhật trạng thái (có kiểm tra hợp lệ)
* cap\_nhat\_han\_hoan\_thanh\_cong\_viec(): thay đổi hạn hoàn thành

Tất cả các thay đổi đều được lưu lại ngay lập tức.

Hàm cap\_nhat\_ten\_cong\_viec():

def cap\_nhat\_ten\_cong\_viec():

    ten\_cong\_viec = input("Nhập tên công việc cần cập nhật: ")

    if ten\_cong\_viec not in danh\_sach\_cong\_viec:

        print("Công việc không tồn tại.")

        return

    else:

        ten\_moi = input("Nhập tên mới cho công việc: ")

        danh\_sach\_cong\_viec[ten\_moi] = danh\_sach\_cong\_viec.pop(ten\_cong\_viec)

        luu\_danh\_sach\_cong\_viec()

Hàm cap\_nhat\_trang\_thai\_cong\_viec():

def cap\_nhat\_trang\_thai\_cong\_viec():

    ten\_cong\_viec = input("Nhập tên công việc cần cập nhật trạng thái: ")

    if ten\_cong\_viec not in danh\_sach\_cong\_viec:

        print("Công việc không tồn tại.")

        return

    else:

        while True:

            print("Công việc có các trạng thái sau: ")

            print("1. Chưa bắt đầu")

            print("2. Đang thực hiện")

            print("3. Hoàn thành")

            trang\_thai = input("Nhập trạng thái công việc (1/2/3): ")

            if trang\_thai == '1':

                trang\_thai = 'Chưa bắt đầu'

                break

            elif trang\_thai == '2':

                trang\_thai = 'Đang thực hiện'

                break

            elif trang\_thai == '3':

                trang\_thai = 'Hoàn thành'

                break

            else:

                print("Chọn sai lựa chọn, vui lòng nhập lại")

                print("=" \* 20)

        danh\_sach\_cong\_viec[ten\_cong\_viec]['Trạng thái'] = trang\_thai

        luu\_danh\_sach\_cong\_viec()

Hàm cap\_nhat\_han\_hoan\_thanh\_cong\_viec():

def cap\_nhat\_han\_hoan\_thanh\_cong\_viec():

    ten\_cong\_viec = input("Nhập tên công việc cần cập nhật hạn hoàn thành: ")

    if ten\_cong\_viec not in danh\_sach\_cong\_viec:

        print("Công việc không tồn tại.")

        return

    else:

        han\_hoan\_thanh = input("Nhập hạn hoàn thành mới (dd/mm/yyyy): ")

        danh\_sach\_cong\_viec[ten\_cong\_viec]['Hạn hoàn thành'] = han\_hoan\_thanh

        luu\_danh\_sach\_cong\_viec()

## 3.6. Xóa công việc

Chương trình hỗ trợ 3 phương án xóa:

* xoa\_cong\_viec\_theo\_ten(): xóa theo tên công việc
* xoa\_cong\_viec\_da\_hoan\_thanh(): xóa nếu trạng thái là “Hoàn thành”
* xoa\_toan\_bo\_cong\_viec(): xóa toàn bộ danh sách công việc

Các hàm đều có kiểm tra tồn tại để tránh lỗi.

Hàm xoa\_cong\_viec\_theo\_ten():

def xoa\_cong\_viec\_theo\_ten():

    ten\_cong\_viec = input("Nhập tên công việc cần xóa: ")

    if ten\_cong\_viec in danh\_sach\_cong\_viec:

        del danh\_sach\_cong\_viec[ten\_cong\_viec]

        luu\_danh\_sach\_cong\_viec()

        print(f"Công việc '{ten\_cong\_viec}' đã được xóa.")

    else:

        print("Công việc không tồn tại.")

Hàm xoa\_cong\_viec\_da\_hoan\_thanh():

def xoa\_cong\_viec\_da\_hoan\_thanh():

    ten\_cong\_viec = input("Nhập tên công việc đã hoàn thành cần xóa: ")

    if ten\_cong\_viec in danh\_sach\_cong\_viec:

        if danh\_sach\_cong\_viec[ten\_cong\_viec]['Trạng thái'] == 'Hoàn thành':

            del danh\_sach\_cong\_viec[ten\_cong\_viec]

            luu\_danh\_sach\_cong\_viec()

            print(f"Công việc '{ten\_cong\_viec}' đã được xóa.")

        else:

            print("Công việc chưa hoàn thành, không thể xóa.")

    else:

        print("Công việc không tồn tại.")

Hàm xoa\_toan\_bo\_cong\_viec():

def xoa\_toan\_bo\_cong\_viec():

    doc\_danh\_sach\_cong\_viec()

    global danh\_sach\_cong\_viec

    danh\_sach\_cong\_viec = {}

    luu\_danh\_sach\_cong\_viec()

    print("Đã xóa toàn bộ công việc.")

## 3.7. Lọc và tìm kiếm công việc

Tìm kiếm linh hoạt theo:

* ten\_cong\_viec: tim\_kiem\_cong\_viec\_theo\_ten()
* ma\_cong\_viec: tim\_cong\_viec\_theo\_ma()
* nguoi\_phu\_trach: tim\_kiem\_cong\_viec\_theo\_ten\_nguoi\_phu\_trach()
* trang\_thai: xem\_cong\_viec\_theo\_trang\_thai()
* han\_hoan\_thanh: xem\_cong\_viec\_theo\_han\_hoan\_thanh()

Hàm tim\_kiem\_cong\_viec\_theo\_ten():

def tim\_kiem\_cong\_viec\_theo\_ten():

    ten\_cong\_viec = input("Nhập tên công việc cần tìm: ")

    if ten\_cong\_viec in danh\_sach\_cong\_viec:

        for cong\_viec, thong\_tin in danh\_sach\_cong\_viec.items():

            if ten\_cong\_viec == cong\_viec:

                print(f"{cong\_viec}: ")

                for key, value in thong\_tin.items():

                    print(f"{key}: {value}")

                print("=" \* 40)

Hàm tim\_kiem\_cong\_viec\_theo\_ma():

def tim\_cong\_viec\_theo\_ma():

    ma\_cong\_viec = input("Nhập mã công việc cần tìm: ")

    for cong\_viec, thong\_tin in danh\_sach\_cong\_viec.items():

        if thong\_tin['Mã công việc'] == ma\_cong\_viec:

            print(f"{cong\_viec}")

            for key, value in thong\_tin.items():

                print(f"{key}: {value}")

            print("=" \* 40)

Hàm tim\_kiem\_cong\_viec\_theo\_ten\_nguoi\_phu\_trach():

def tim\_kiem\_cong\_viec\_theo\_ten\_nguoi\_phu\_trach():

    ten\_nguoi\_phu\_trach = input("Nhập tên người phụ trách công việc cần tìm: ")

    print(f"Các công việc do {ten\_nguoi\_phu\_trach} phụ trách:")

    for cong\_viec, thong\_tin in danh\_sach\_cong\_viec.items():

        if thong\_tin['Người phụ trách'] == ten\_nguoi\_phu\_trach:

            print(f"{cong\_viec}: ")

            for key, value in thong\_tin.items():

                print(f"{key}: {value}")

            print("=" \* 40)

Hàm xem\_cong\_viec\_theo\_han\_hoan\_thanh():

def xem\_cong\_viec\_theo\_han\_hoan\_thanh():

    han\_hoan\_thanh = input("Nhập vào hạn hoàn thành công việc (dd/mm/yyyy): ")

    print(f"Các công việc có hạn hoàn thành quá ngày {han\_hoan\_thanh}: ")

    for cong\_viec, thong\_tin in danh\_sach\_cong\_viec.items():

        if thong\_tin['Hạn hoàn thành'][0:2] >= han\_hoan\_thanh[0:2] and thong\_tin['Hạn hoàn thành'][3:5] >= han\_hoan\_thanh[3:5] and thong\_tin['Hạn hoàn thành'][6:10] >= han\_hoan\_thanh[6:10]:

            print(f"{cong\_viec}: ")

            for key, value in thong\_tin.items():

                print(f"{key}: {value}")

            print("=" \* 40)

## 3.8. Sắp xếp và thống kê

Chức năng sắp xếp:

* sap\_xep\_cong\_viec\_theo\_han\_hoan\_thanh()
* sap\_xep\_cong\_viec\_theo\_ngay\_tao()

Sử dụng sorted() với lambda để sắp xếp dữ liệu theo thời gian. Kết quả được hiển thị rõ ràng theo thứ tự ưu tiên.

Hiện tại chưa có thống kê, tuy nhiên dễ dàng bổ sung để thống kê số công việc đang làm, hoàn thành, chưa bắt đầu.

Hàm sap\_xep\_cong\_viec\_theo\_han\_hoan\_thanh():

def sap\_xep\_cong\_viec\_theo\_han\_hoan\_thanh():

    doc\_danh\_sach\_cong\_viec()

    danh\_sach\_cong\_viec\_sorted = dict(sorted(danh\_sach\_cong\_viec.items(), key=lambda item: item[1]['Hạn hoàn thành']))

    print("Danh sách công việc đã sắp xếp theo hạn hoàn thành:")

    for cong\_viec, thong\_tin in danh\_sach\_cong\_viec\_sorted.items():

        print(f"{cong\_viec}: ")

        for key, value in thong\_tin.items():

            print(f"{key}: {value}")

        print("=" \* 40)

Hàm sap\_xep\_cong\_viec\_theo\_ngay\_tao():

def sap\_xep\_cong\_viec\_theo\_ngay\_tao():

    doc\_danh\_sach\_cong\_viec()

    danh\_sach\_cong\_viec\_sorted = dict(sorted(danh\_sach\_cong\_viec.items(), key=lambda item: item[1]['Ngày tạo']))

    print("Danh sách công việc đã sắp xếp theo ngày tạo:")

    for cong\_viec, thong\_tin in danh\_sach\_cong\_viec\_sorted.items():

        print(f"{cong\_viec}: ")

        for key, value in thong\_tin.items():

            print(f"{key}: {value}")

        print("=" \* 40)

## 3.9. Hàm chính và Menu chương trình

if \_\_name\_\_ == '\_\_main\_\_':

    doc\_danh\_sach\_cong\_viec()

    while True:

        print("==========Chương trình quản lí công việc==========")

        print("1. Thêm công việc")

        print("2. Xem danh sách công việc")

        print("3. Xem công việc theo trạng thái (chưa bắt đầu / đang làm / hoàn thành)")

        print("4. Xem công việc theo hạn hoàn thành")

        print("5. Tìm kiếm công việc theo tên")

        print("6. Tìm kiếm công việc theo mã")

        print("7. Tìm kiếm công việc theo tên người phụ trách")

        print("8. Cập nhật tên công việc")

        print("9. Cập nhật trạng thái công việc")

        print("10. Cập nhật hạn hoàn thành công việc")

        print("11. Xóa công việc theo tên")

        print("12. Xóa công việc đã hoàn thành")

        print("13. Sắp xếp công việc theo hạn hoàn thành")

        print("14. Sắp xếp công việc theo ngày tạo")

        print("15. Xóa toàn bộ công việc")

        print("0. Thoát chương trình")

        lua\_chon = input("Nhập lựa chọn của bạn ")

        if lua\_chon == '1':

            them\_cong\_viec()

            print("=" \* 20)

        elif lua\_chon == '2':

            xem\_danh\_sach\_cong\_viec()

        elif lua\_chon == '3':

            xem\_cong\_viec\_theo\_trang\_thai()

        elif lua\_chon == '4':

            xem\_cong\_viec\_theo\_han\_hoan\_thanh()

        elif lua\_chon == '5':

            tim\_kiem\_cong\_viec\_theo\_ten()

        elif lua\_chon == '6':

            tim\_cong\_viec\_theo\_ma()

        elif lua\_chon == '7':

            tim\_kiem\_cong\_viec\_theo\_ten\_nguoi\_phu\_trach()

        elif lua\_chon == '8':

            cap\_nhat\_ten\_cong\_viec()

        elif lua\_chon == '9':

            cap\_nhat\_trang\_thai\_cong\_viec()

        elif lua\_chon == '10':

            cap\_nhat\_han\_hoan\_thanh\_cong\_viec()

        elif lua\_chon == '11':

            xoa\_cong\_viec\_theo\_ten()

        elif lua\_chon == '12':

            xoa\_cong\_viec\_da\_hoan\_thanh()

        elif lua\_chon == '13':

            sap\_xep\_cong\_viec\_theo\_han\_hoan\_thanh()

        elif lua\_chon == '14':

            sap\_xep\_cong\_viec\_theo\_ngay\_tao()

        elif lua\_chon == '15':

            xoa\_toan\_bo\_cong\_viec()

        elif lua\_chon == '0':

            print("Cảm ơn bạn đã sử dụng chương trình!")

            break

## 3.10. Kiểm thử chương trình

### 3.10.1. Thêm công việc vào danh sách và hiển thị danh sách

**A screenshot of a computer program

AI-generated content may be incorrect.**

Hình 3. Kiểm thử chức năng 1 và 2

Kết quả:

**A screenshot of a computer

AI-generated content may be incorrect.**

Hình 3. Kết quả kiểm thử chức năng 1 và 2

### 3.10.2. Xem công việc có trạng thái chưa bắt đầu

**A screen shot of a computer

AI-generated content may be incorrect.**

Hình 3. Kết quả kiểm thử chức năng 3

### 3.10.3. Xem công việc theo trạng thái hoàn thành

**A screenshot of a computer

AI-generated content may be incorrect.**

Hình 3. Kết quả kiểm thử chức năng 4

### 3.10.4. Tìm công việc theo tên công việc

**A computer code with black text

AI-generated content may be incorrect.**

Hình 3. Kết quả kiểm thứ chức năng 5

### 3.10.5. Tìm công việc theo mã công việc

**A white text with black text

AI-generated content may be incorrect.**

Hình 3. Kết quả kiểm thử chức năng 6

### 3.10.6. Tìm công việc theo tên người phụ trách

**A white paper with black text

AI-generated content may be incorrect.**

Hình 3. Kết quả kiểm thử chức năng 7

### 3.10.7. Cập nhật tên công việc

**A close up of a text

AI-generated content may be incorrect.**

Hình 3. Kiểm thứ chức năng 8

Kết quả

**A white screen with black text

AI-generated content may be incorrect.**

Hình 3. Kết quả kiểm thử chức năng 8

### 3.10.8. Cập nhật trạng thái công việc

**A screenshot of a computer

AI-generated content may be incorrect.**

Hình 3. Kiểm thứ chức năng 9

Kết quả

**A white text with black text

AI-generated content may be incorrect.**

Hình 3. Kết quả kiểm thử chức năng 9

**KẾT LUẬN**

Đề tài “Quản lý công việc hàng ngày bằng Python” là một ứng dụng thực tiễn, giúp người dùng tổ chức, theo dõi và kiểm soát công việc cá nhân một cách hiệu quả. Trong quá trình thực hiện, đề tài đã khai thác tối ưu các đặc điểm của ngôn ngữ Python như cú pháp đơn giản, thư viện phong phú và khả năng thao tác với dữ liệu linh hoạt. Chương trình được thiết kế với giao diện dòng lệnh trực quan, có thể thực hiện nhiều chức năng cơ bản như thêm công việc mới, cập nhật thông tin, xóa công việc, tìm kiếm theo nhiều tiêu chí, sắp xếp theo ngày tạo hoặc hạn hoàn thành, và lưu trữ dữ liệu bằng định dạng JSON. Nhờ đó, người dùng có thể dễ dàng quản lý lịch trình làm việc cá nhân một cách có hệ thống, khoa học.

Ưu điểm nổi bật của đề tài là mã nguồn được tổ chức rõ ràng, dễ đọc và dễ mở rộng; dữ liệu được lưu trữ và khôi phục đơn giản mà không cần đến hệ quản trị cơ sở dữ liệu phức tạp; giao diện thân thiện, phù hợp với người dùng mới bắt đầu học lập trình hoặc muốn áp dụng công nghệ vào đời sống cá nhân. Ngoài ra, chương trình hoạt động ổn định, xử lý lỗi cơ bản, hỗ trợ người dùng nhập sai và đảm bảo tính toàn vẹn của dữ liệu trong suốt quá trình sử dụng.

Tuy nhiên, bên cạnh những ưu điểm, chương trình vẫn còn tồn tại một số nhược điểm nhất định. Cụ thể, việc nhập dữ liệu thủ công bằng dòng lệnh có thể gây bất tiện khi danh sách công việc lớn; chưa tích hợp các công nghệ hiện đại như giao diện đồ họa (GUI), cơ sở dữ liệu quan hệ hay điện toán đám mây; chưa có chức năng thống kê số lượng công việc theo trạng thái, chưa xử lý chính xác ngày tháng bằng thư viện chuyên dụng như datetime và cũng chưa có tính năng nhắc nhở công việc tự động hoặc hỗ trợ đa người dùng.

**DANH MỤC TÀI LIỆU THAM KHẢO**

1. Trần Đức Phương, Nhập môn lập trình với Python, NXB Thông tin và Truyền thông, 2022.
2. Đặng Kim Hoàng, Trần Đình Quang, Giáo trình Python Cơ Bản, NXB Đại Học Quốc Gia TP.HCM, 2021.