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 ĐỒ DÙNG GIA ĐÌNH**

**Giảng viên: ThS. Lê Tuấn Anh**

**Sinh viên thực hiện:**

1. **Nguyễn Đức Cảnh – 1871020084**
2. **Trương Thanh An – 1771020010**

**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 ĐỒ DÙNG GIA ĐÌNH**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TT | Mã sv | Họ và Tên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 1 | 1871020084 | Nguyễn Đức Cảnh | 12/08/2006 |  |  |
| 2 | 1771020010 | Trương Thanh An | 05/12/2005 |  |  |

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

### 

**Hà Nội, năm 2025**

# LỜI NÓI ĐẦU

Trong bối cảnh xã hội ngày càng phát triển, nhu cầu quản lý tài sản, vật dụng trong gia đình một cách khoa học, hiệu quả đang trở thành mối quan tâm chung của nhiều hộ gia đình. Khi số lượng đồ dùng trong nhà tăng lên theo thời gian, việc ghi nhớ, theo dõi và kiểm soát thông tin về các vật dụng như thời gian mua, tình trạng sử dụng, giá trị tài sản,… ngày càng trở nên khó khăn nếu chỉ sử dụng các phương pháp ghi chép truyền thống bằng tay. Điều này dễ dẫn đến việc thất thoát, hư hỏng hoặc sử dụng không hiệu quả các đồ dùng hiện có.

Trong thời đại công nghệ thông tin phát triển mạnh mẽ, việc ứng dụng các công cụ lập trình để hỗ trợ quản lý đã và đang trở thành xu hướng tất yếu. Đặc biệt, Python – một ngôn ngữ lập trình hiện đại, dễ học, cú pháp rõ ràng và có khả năng mở rộng cao – đang ngày càng được sử dụng rộng rãi trong các lĩnh vực từ học thuật đến công nghiệp phần mềm. Nhận thấy được tiềm năng và sự phù hợp của Python với các bài toán xử lý dữ liệu thực tế, em đã chọn đề tài “Tạo chương trình quản lý đồ dùng gia đình” nhằm xây dựng một phần mềm đơn giản, trực quan, hỗ trợ người dùng ghi lại thông tin, tra cứu và quản lý hiệu quả các đồ dùng trong gia đình.

Chương trình được xây dựng nhằm phục vụ các chức năng cơ bản như thêm mới đồ dùng, chỉnh sửa thông tin, xóa, tìm kiếm, thống kê số lượng, phân loại theo từng nhóm,… Ngoài ra, đề tài còn hướng tới việc phát triển giao diện người dùng thân thiện, giúp người sử dụng không cần có kiến thức chuyên sâu về kỹ thuật vẫn có thể dễ dàng thao tác. Thông qua việc thực hiện đề tài này, em mong muốn vận dụng những kiến thức đã học về cấu trúc dữ liệu, xử lý tệp, lập trình hướng đối tượng và xây dựng giao diện đồ họa (GUI) trong Python để giải quyết một bài toán thực tế có ý nghĩa thiết thực.

**MỤC LỤC**

[LỜI NÓI ĐẦU 3](#_Toc198736954)

[CHƯƠNG 1. GIỚI THIỆU VỀ MỘT SỐ KHÁI NIỆM TRONG PYTHON 7](#_Toc198736955)

[1.1 Giới thiệu ngôn ngữ lập trình Python 7](#_Toc198736956)

[1.1.1 Một số đặc điểm nổi bật của Python 7](#_Toc198736957)

[1.1.2 Các lĩnh vực ứng dụng của Python 7](#_Toc198736958)

[1.2 Cấu trúc chương trình Python cơ bản 8](#_Toc198736959)

[1.2.1 Khai báo thư viện 8](#_Toc198736960)

[1.2.2 Định nghĩa hàm 9](#_Toc198736961)

[1.2.3 Phần xử lí chính 9](#_Toc198736962)

[1.2.4 Xử lí lỗi 9](#_Toc198736963)

[1.3 Biến và kiểu dữ liệu trong Python 9](#_Toc198736964)

[1.3.1 Biến 9](#_Toc198736965)

[1.3.2 Kiểu dữ liệu 10](#_Toc198736966)

[1.3.3 Các kiểu dữ liệu cấu trúc 10](#_Toc198736967)

[1.4 Câu lệnh và điều kiện vòng lặp 11](#_Toc198736968)

[1.4.1. Câu lệnh điều kiện rẽ nhánh 11](#_Toc198736969)

[1.4.2. Toán tử so sánh và logic 11](#_Toc198736970)

[1.4.3 Vòng lặp trong Python 12](#_Toc198736971)

[1.4.4 Câu lệnh điều khiển vòng lặp 12](#_Toc198736972)

[1.5. Hàm và phạm vi biến trong Python 13](#_Toc198736973)

[1.5.1. Hàm trong Python 13](#_Toc198736974)

[1.5.2. Hàm có tham số mặc định 13](#_Toc198736975)

[1.5.3. Hàm có số lượng tham số linh hoạt 13](#_Toc198736976)

[1.5.4. Phạm vi biến (Variable Scope) 14](#_Toc198736977)

[1.6 Danh sách, tuple, set và dictionary trong Python 14](#_Toc198736978)

[1.6.1 Danh sách 14](#_Toc198736979)

[1.6.2 Bộ (tuple) 15](#_Toc198736980)

[1.6.3 Tập hợp (set) 15](#_Toc198736981)

[1.6.4 Từ điển (dictionary) 15](#_Toc198736982)

[1.7 Làm việc với tệp tin (File I/O) 16](#_Toc198736983)

[1.7.1. Mở tệp tin 17](#_Toc198736984)

[1.7.2 Đọc dữ liệu từ tệp 17](#_Toc198736985)

[1.7.3. Ghi dữ liệu vào tệp 17](#_Toc198736986)

[1.7.4 Sử dụng “with” để làm việc với tệp 18](#_Toc198736987)

[1.7.5. Ứng dụng trong chương trình quản lí đồ dùng 18](#_Toc198736988)

[1.7.6. Xử lý lỗi khi làm việc với tệp 18](#_Toc198736989)

[CHƯƠNG 2. GIỚI THIỆU, MÔ TẢ ĐỀ TÀI 20](#_Toc198736990)

[2.1. Mục tiêu và phạm vi của đề tài 20](#_Toc198736991)

[2.2. Yêu cầu chức năng của chương trình 21](#_Toc198736992)

[2.2.1. Thêm đồ dùng mới 21](#_Toc198736993)

[2.2.2. Hiển thị danh sách đồ dùng 21](#_Toc198736994)

[2.2.3. Tìm kiếm đồ dùng 22](#_Toc198736995)

[2.2.4. Chỉnh sửa thông tin đồ dùng 22](#_Toc198736996)

[2.2.5. Xóa đồ dùng 22](#_Toc198736997)

[2.2.6. Lưu dữ liệu vào tệp 22](#_Toc198736998)

[2.2.7. Đọc dữ liệu từ tệp khi khởi động 22](#_Toc198736999)

[2.2.8. Thoát chương trình an toàn 22](#_Toc198737000)

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

## 1.1 Giới thiệu ngôn ngữ lập trình Python

Python là một ngôn ngữ lập trình bậc cao, được Guido van Rossum phát triển vào cuối những năm 1980 và chính thức ra mắt vào năm 1991. Python được thiết kế với triết lí đơn giản, rõ ràng và dễ đọc, giúp lập trình viên dễ dàng học và sử dụng, đặc biệt là những người mới bắt đầu tiếp cận với lập trình.

Python là một ngôn ngữ đa mục đích, hỗ trợ nhiều phong cách lập trình như lập trình hướng đối tượng, lập trình thủ tục và lập trình hàm. Điều này cho phép Python được ứng dụng rộng rãi trong nhiều lĩnh vực như phát triển phần mềm, khoa học dữ liệu, trí tuệ nhân tạo, lập trình web, tự động hóa, và cả trong giáo dục.

### 1.1.1 Một số đặc điểm nổi bật của Python

Cú pháp đơn giản và dễ hiểu: Python có cú pháp gần với ngôn ngữ tự nhiên, giúp mã nguồn dễ đọc và bảo trì.

Thư viện phong phú: Python có hệ sinh thái thư viện rất lớn, hỗ trợ nhiều tác vụ như xử lí dữ liệu, giao diện đồ họa, lập trình web và nhiều lĩnh vực khác

Tính linh hoạt và đa nền tảng: Python có thể chạy trên nhiều hệ điều hành khác nhau như Windows, macOS và Linux mà không cần thay đổi mã nguồn.

Cộng đồng phát triển mạnh mẽ: Với cộng đồng lập trình viên lớn, người dùng Python có thẻ dễ dàng tìm thấy tài liệu học tập, diễn đàn hỗ trợ và các công cụ hữu ích.

### 1.1.2 Các lĩnh vực ứng dụng của Python

Phát triển phần mềm: Python được sử dụng để xây dựng các ứng dụng từ đơn giản đến phức tạp, như phần mềm quản lý, phần mềm học tập, ứng dụng điều khiển thiết bị IoT, và nhiều loại công cụ tự động hoá. Với các thư viện như tkinter, PyQt, hoặc Kivy, Python có thể hỗ trợ việc phát triển giao diện người dùng một cách nhanh chóng và trực quan.

Khoa học dữ liệu và phân tích số liệu: Python là ngôn ngữ hàng đầu trong lĩnh vực khoa học dữ liệu. Các thư viện như pandas, numpy, matplotlib và seaborn cho phép xử lý, phân tích, trực quan hóa dữ liệu một cách dễ dàng và hiệu quả. Đây là một lợi thế lớn trong việc đưa ra quyết định dựa trên dữ liệu, được ứng dụng rộng rãi trong kinh doanh, tài chính, nghiên cứu và marketing.

Trí tuệ nhân tạo và học máy: Python là lựa chọn phổ biến để xây dựng các mô hình học máy và hệ thống thông minh. Các thư viện như TensorFlow, Keras, PyTorch, Scikit-learn hỗ trợ triển khai các thuật toán học máy, học sâu, xử lý ngôn ngữ tự nhiên, thị giác máy tính,… Python hiện đang chiếm ưu thế lớn trong việc phát triển các ứng dụng AI thực tiễn như chatbot, hệ thống gợi ý, phân tích hình ảnh,…

Phát triển web: Với các framework mạnh mẽ như Django, Flask, hoặc FastAPI, Python có thể được dùng để xây dựng các hệ thống web từ cơ bản đến phức tạp. Các ứng dụng này bao gồm trang quản lý, cổng thông tin điện tử, hệ thống đặt lịch, blog cá nhân hoặc thậm chí các nền tảng thương mại điện tử quy mô nhỏ.

Tự động hóa và viết script: Python là công cụ mạnh trong việc tự động hóa các tác vụ lặp đi lặp lại như xử lý file, đổi tên hàng loạt, gửi email tự động, kiểm tra hệ thống,… Những ứng dụng nhỏ này giúp tiết kiệm đáng kể thời gian và giảm lỗi thủ công trong công việc hàng ngày.

Lĩnh vực giáo dục và giảng dạy: Python thường được sử dụng làm ngôn ngữ đầu tiên cho người mới học lập trình vì tính dễ học, cú pháp rõ ràng và lượng tài liệu phong phú. Rất nhiều trường học và khoá học trực tuyến trên toàn thế giới chọn Python làm nền tảng để giảng dạy tư duy lập trình và khoa học máy tính cơ bản.

Phát triển trò chơi : Dù không phổ biến như C++ hay Unity, Python vẫn được sử dụng để phát triển game đơn giản, giáo dục hoặc thử nghiệm ý tưởng nhờ thư viện pygame.

## 1.2 Cấu trúc chương trình Python cơ bản

### 1.2.1 Khai báo thư viện

Trước khi sử dụng các hàm hoặc lớp (class) từ thư viện bên ngoài hoặc chuẩn, chương trình cần khai báo chúng bằng câu lệnh import. Ví dụ:

import json

import tkinter as tk

Việc sử dụng các thư viện giúp giảm thời gian viết lại các chức năng phổ biến như xử lý số liệu, tạo giao diện, làm việc với tệp tin, v.v.

### 1.2.2 Định nghĩa hàm

Hàm là thành phần quan trọng giúp chia nhỏ chương trình thành các phần xử lý riêng biệt, tăng tính tái sử dụng và dễ bảo trì. Một hàm trong Python được khai báo bằng từ khóa def:

def tinh\_tong(a, b):

    return a + b

Việc tổ chức chương trình theo hàm giúp giảm lặp mã và làm cho chương trình dễ kiểm soát hơn.

### 1.2.3 Phần xử lí chính

Phần này chứa các dòng mã điều khiển luồng thực thi chính của chương trình. Trong Python, người ta thường sử dụng mẫu if \_\_name\_\_ == "\_\_main\_\_" để đảm bảo chương trình chỉ chạy khi được thực thi trực tiếp:

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

    # Gọi các hàm chính

    main()

Việc sử dụng đoạn mã này là một chuẩn thực hành tốt, đặc biệt khi chương trình có nhiều module hoặc được tái sử dụng.

### 1.2.4 Xử lí lỗi

Trong một số trường hợp, chương trình có thể xảy ra lỗi khi nhập dữ liệu sai, không tìm thấy tệp tin,… Để đảm bảo chương trình không bị dừng đột ngột, có thể dùng cấu trúc try-except:

try:

    file = open("du\_lieu.txt", "r")

except FileNotFoundError:

    print("Không tìm thấy tệp.")

Xử lý lỗi giúp chương trình hoạt động ổn định hơn và thân thiện với người dùng.

## 1.3 Biến và kiểu dữ liệu trong Python

### 1.3.1 Biến

Biến là tên đại diện cho một vùng nhớ dùng để lưu trữ giá trị trong quá trình thực thi chương trình. Python không yêu cầu khai báo kiểu dữ liệu của biến khi khởi tạo. Việc gán giá trị cho biến cũng đồng thời tạo ra biến đó:

ten = "Bàn học"

so\_luong = 3

co\_gia = True

Python là ngôn ngữ động kiểu (dynamically typed), nghĩa là kiểu dữ liệu của biến được xác định tự động dựa trên giá trị gán. Lập trình viên có thể thay đổi kiểu dữ liệu của một biến bằng cách gán lại giá trị mới mà không cần khai báo lại.

Một số quy tắc khi đặt tên biến:

* Tên biến phải bắt đầu bằng chữ cái hoặc dấu gạch dưới
* Không được chứa khoảng trắng hoặc ký tự đặc biệt
* Không được trùng với từ khóa của Python như if, while, class, ...

### 1.3.2 Kiểu dữ liệu

Kiểu số nguyên: Đại diện cho các số không có phần thập phân:

tuoi = 20

Kiểu số thực: Dùng để lưu trữ số có phần thập phân:

can\_nang = 55.8

Kiểu chuỗi: Đại diện cho dãy kí tự, thường dùng để lưu tên, mô tả

ten\_do\_dung = "Tủ quần áo"

Kiểu logic: Chỉ nhận hai giá trị true và false. Thường dùng trong điều kiện rẽ nhánh

co\_gia = True

### 1.3.3 Các kiểu dữ liệu cấu trúc

Danh sách (list): tập hợp các phần tử có thể thay đổi thứ tự và giá trị

do\_dung = ["Ghế", "Bàn", "Kệ sách"]

Bộ (tuple): giống list nhưng không thể thay đổi sau khi khai báo

kich\_thuoc = (50, 30, 20)

Tập hợp (set): tập hợp các phần tử không trùng nhau

loai\_do\_dung = {"Ghế", "Bàn", "Tủ"}

Từ điển (dict): cấu trúc lưu trữ theo cặp khóa – giá trị, rất hữu ích trong các chương trình quản lí.

do\_dung = {

    "ten": "Tủ lạnh",

    "so\_luong": 2,

    "gia": 4500000

}

## 1.4 Câu lệnh và điều kiện vòng lặp

### 1.4.1. Câu lệnh điều kiện rẽ nhánh

Câu lệnh điều kiện cho phép chương trình quyết định thực hiện các thao tác khác nhau tùy theo giá trị của biểu thức điều kiện. Python sử dụng các từ khóa if, elif và else:

so\_luong = 5

if so\_luong > 10:

    print("Số lượng lớn")

elif so\_luong == 10:

    print("Số lượng vừa đủ")

else:

    print("Số lượng ít")

Lưu ý: Python sử dụng thụt lề để xác định khối lệnh thuộc điều kiện. Đây là điểm khác biệt lớn so với nhiều ngôn ngữ khác.

### 1.4.2. Toán tử so sánh và logic

|  |  |
| --- | --- |
| **Toán tử so sánh** | **Ý nghĩa** |
| == | So sánh bằng |
| != | Khác nhau |
| > | Lớn hơn |
| < | Nhỏ hơn |
| >= | Lớn hơn hoặc bằng |
| <= | Nhỏ hơn hoặc bằng |

|  |  |
| --- | --- |
| **Toán tử logic** | **Ý nghĩa** |
| and | Và |
| or | Hoặc |
| not | Phủ định (không) |

### 1.4.3 Vòng lặp trong Python

Python hỗ trợ hai vòng lặp chính: for và while

a) Vòng lặp for

Vòng lặp for được dùng để duyệt qua các phần tử trong chuỗi, danh sách, từ điển, tập hợp, ...

do\_dung = ["Bàn", "Ghế", "Tủ"]

for ten in do\_dung:

    print("Đồ dùng:", ten)

Cũng có thể kết hợp với hàm range() để lặp số lần cố định

for i in range(3):

    print("Lần lặp:", i)

b) Vòng lặp while

Vòng lặp while được sử dụng khi chưa biết trước số lần lặp và chỉ dừng lại khi điều kiện không còn đúng:

so\_luong = 0

while so\_luong < 5:

    print("Số lượng hiện tại:", so\_luong)

    so\_luong += 1

### 1.4.4 Câu lệnh điều khiển vòng lặp

* break: Dừng vòng lặp ngay lập tức.
* continue: Bỏ qua lần lặp hiện tại và tiếp tục vòng lặp tiếp theo.
* pass: Câu lệnh rỗng, thường dùng như một trình giữ chỗ.

Ví dụ:

for i in range(10):

    if i == 5:

        break  # Dừng vòng lặp khi i = 5

    print(i)

Việc sử dụng thành thạo các cấu trúc điều kiện và vòng lặp giúp kiểm soát tốt luồng thực thi của chương trình. Trong chương trình quản lý đồ dùng gia đình, các cấu trúc này thường được sử dụng để kiểm tra điều kiện nhập liệu, duyệt danh sách đồ dùng, hoặc xác định hành động phù hợp tùy theo dữ liệu người dùng cung cấp.

## 1.5. Hàm và phạm vi biến trong Python

### 1.5.1. Hàm trong Python

Hàm là một khối mã thực hiện một chức năng cụ thể, giúp chia chương trình thành các phần nhỏ gọn, dễ tổ chức và tái sử dụng. Trong Python, hàm được định nghĩa bằng từ khóa def:

def tinh\_tong(a, b):

    return a + b

Để gọi hàm:

ket\_qua = tinh\_tong(5, 3)

print("Tổng là:", ket\_qua)

Hàm có thể nhận nhiều tham số (parameters), có thể có hoặc không trả về giá trị (return). Nếu không dùng return, hàm trả về giá trị mặc định là None.

### 1.5.2. Hàm có tham số mặc định

Python cho phép khai báo giá trị mặc định cho tham số, giúp đơn giản hóa việc gọi hàm:

def chao(ten="Người dùng"):

    print("Xin chào,", ten)

chao()          # Kết quả: Xin chào, Người dùng

chao("Lan")     # Kết quả: Xin chào, Lan

### 1.5.3. Hàm có số lượng tham số linh hoạt

\*args: cho phép truyền vào nhiều đối số dưới dạng tuple

\*\*kwargs: cho phép truyền vào nhiều đối số có tên dưới dạng dictionary.

def hien\_thi\_do\_dung(\*ten\_do\_dung):

    for ten in ten\_do\_dung:

        print("Đồ dùng:", ten)

### 1.5.4. Phạm vi biến (Variable Scope)

Phạm vi biến là vùng mà biến đó có thể được truy cập trong chương trình.

Biến cục bộ (Local variable): Chỉ tồn tại trong phạm vi của hàm:

def ham():

    x = 10  # biến cục bộ

    print(x)

Biến toàn cục (Global variable): Được khai báo bên ngoài tất cả các hàm, có thể dùng trong toàn chương trình:

x = 20  # biến toàn cục

def ham():

    print(x)  # truy cập biến toàn cục

Thay đổi biến toàn cục bên trong hàm: Dùng từ khóa global nếu muốn gán lại giá trị cho biến toàn cục:

x = 5

def thay\_doi():

    global x

    x = 10

## 1.6 Danh sách, tuple, set và dictionary trong Python

Python cung cấp nhiều kiểu dữ liệu cấu trúc để lưu trữ tập hợp các giá trị. Mỗi kiểu có đặc điểm riêng, phù hợp với các mục đích sử dụng khác nhau trong quá trình xử lý và quản lý dữ liệu.

### 1.6.1 Danh sách

Danh sách là một tập hợp các phần tử có thứ tự, có thể thay đổi (mutable) và có thể chứa các kiểu dữ liệu khác nhau.

Khai báo danh sách:

do\_dung = ["Bàn", "Ghế", "Tủ", "Kệ"]

Truy cập và thay đổi phần tử:

print(do\_dung[1])      # Kết quả: Ghế

do\_dung[2] = "Giường"  # Thay đổi phần tử thứ 3

Một số thao tác phổ biến:

do\_dung.append("Quạt")         # Thêm phần tử vào cuối

do\_dung.remove("Ghế")          # Xóa phần tử theo giá trị

do\_dung.pop(0)                 # Xóa phần tử theo chỉ số

print(len(do\_dung))            # Đếm số phần tử

### 1.6.2 Bộ (tuple)

Tuple giống như danh sách nhưng không thể thay đổi sau khi khai báo. Dùng khi dữ liệu cần đảm bảo tính cố định.

Khai báo tuple:

kich\_thuoc = (40, 60, 80)

Truy cập phần tử:

print(kich\_thuoc[0])  # Kết quả: 40

Ưu điểm của tuple:

* Tốc độ truy xuất nhanh hơn list.
* Đảm bảo dữ liệu không bị thay đổi ngoài ý muốn.

### 1.6.3 Tập hợp (set)

Set là một tập hợp không có thứ tự, không chứa phần tử trùng lặp, và có thể thay đổi.

Khai báo set:

loai\_do\_dung = {"Bàn", "Ghế", "Tủ", "Ghế"}

print(loai\_do\_dung)  # Chỉ in ra một "Ghế"

Một số thao tác phổ biến:

loai\_do\_dung.add("Quạt")

loai\_do\_dung.remove("Tủ")

Dùng set để:

* Loại bỏ các giá trị trùng nhau.
* Thực hiện các phép toán tập hợp (hợp, giao, hiệu,...).

### 1.6.4 Từ điển (dictionary)

Dictionary lưu trữ dữ liệu dưới dạng cặp khóa - giá trị (key - value). Mỗi khóa là duy nhất và được dùng để truy cập giá trị tương ứng.

Khai báo từ điển:

do\_dung = {

    "ten": "Bàn học",

    "so\_luong": 2,

    "co\_gia": True

}

Truy cập và thay đổi dữ liệu:

print(do\_dung["ten"])         # Kết quả: Bàn học

do\_dung["so\_luong"] = 3       # Cập nhật giá trị

do\_dung["mau\_sac"] = "Nâu"    # Thêm cặp mới

Một số thao tác phổ biến:

del do\_dung["co\_gia"]         # Xóa cặp khóa-giá trị

print(do\_dung.keys())         # Lấy danh sách khóa

print(do\_dung.values())       # Lấy danh sách giá trị

|  |  |  |  |
| --- | --- | --- | --- |
| **Kiểu dữ liệu** | **Có thứ tự** | **Thay đổi được** | **Phần tử trùng lặp** |
| List | Có | Có | Có |
| Tuple | Có | Không | Có |
| Set | Không | Có | Không |
| Dictionary | Có (từ khóa) | Có | Không trùng khóa |

Các kiểu dữ liệu này đóng vai trò quan trọng trong việc quản lý thông tin. Trong chương trình quản lý đồ dùng gia đình, chúng có thể được sử dụng để:

* Lưu danh sách các món đồ (list)
* Lưu thông tin cố định như kích thước (tuple)
* Ghi nhận loại đồ dùng không trùng lặp (set)
* Tổ chức thông tin chi tiết cho từng món đồ (dictionary)

## 1.7 Làm việc với tệp tin (File I/O)

Xử lý tệp tin (File I/O – Input/Output) là thao tác cơ bản trong lập trình, cho phép chương trình lưu trữ và truy xuất dữ liệu một cách lâu dài. Trong Python, làm việc với tệp rất linh hoạt và đơn giản.

### 1.7.1. Mở tệp tin

Python sử dụng hàm open() để mở tệp. Cú pháp:

f = open("ten\_tap\_tin.txt", "che\_do")

Một số chế độ phổ biến:

|  |  |
| --- | --- |
| **Chế độ** | **Ý nghĩa** |
| "r" | Mở để đọc (mặc định) |
| "w" | Mở để ghi (xóa nội dung cũ nếu có) |
| "a" | Mở để ghi tiếp vào cuối tệp |
| "r+" | Mở để đọc và ghi |

### 1.7.2 Đọc dữ liệu từ tệp

read(): Đọc toàn bộ nội dung tệp.

f = open("do\_dung.txt", "r")

noi\_dung = f.read()

print(noi\_dung)

f.close()

readline(): Đọc từng dòng.

f = open("do\_dung.txt", "r")

dong\_dau = f.readline()

print(dong\_dau)

f.close()

readlines(): Đọc toàn bộ tệp thành danh sách các dòng.

f = open("do\_dung.txt", "r")

cac\_dong = f.readlines()

for dong in cac\_dong:

    print(dong.strip())

f.close()

### 1.7.3. Ghi dữ liệu vào tệp

write(): Ghi nội dung vào tệp.

f = open("do\_dung.txt", "w")

f.write("Tên: Bàn học\nSố lượng: 2\n")

f.close()

writelines(): Ghi danh sách chuỗi vào tệp.

do\_dung = ["Bàn\n", "Ghế\n", "Tủ\n"]

f = open("do\_dung.txt", "w")

f.writelines(do\_dung)

f.close()

Lưu ý: Ghi tệp ở chế độ "w" sẽ xóa toàn bộ nội dung cũ trong tệp. Dùng "a" nếu muốn ghi thêm.

### 1.7.4 Sử dụng “with” để làm việc với tệp

Sử dụng từ khóa “with” giúp tự động đóng tệp sau khi hoàn tất thao tác, an toàn và gọn gàng hơn:

with open("do\_dung.txt", "r") as f:

    for dong in f:

        print(dong.strip())

### 1.7.5. Ứng dụng trong chương trình quản lí đồ dùng

Việc đọc và ghi dữ liệu ra tệp giúp lưu trữ thông tin đồ dùng một cách lâu dài, thay vì chỉ tồn tại tạm thời trong bộ nhớ:

* Ghi danh sách đồ dùng vào tệp sau mỗi lần thêm mới.
* Đọc thông tin từ tệp khi khởi động chương trình.
* Cập nhật hoặc tìm kiếm đồ dùng bằng cách xử lý nội dung tệp.

### 1.7.6. Xử lý lỗi khi làm việc với tệp

Nên sử dụng khối try-except để tránh lỗi nếu tệp không tồn tại hoặc xảy ra lỗi khi thao tác:

try:

    with open("do\_dung.txt", "r") as f:

        print(f.read())

except FileNotFoundError:

    print("Tệp không tồn tại.")

Làm việc với tệp là kỹ năng thiết yếu trong lập trình ứng dụng. Trong đề tài “Tạo chương trình quản lý đồ dùng gia đình”, việc lưu và truy xuất dữ liệu từ tệp giúp chương trình hoạt động bền vững, hỗ trợ người dùng quản lý dữ liệu một cách hiệu quả và thuận tiện hơn.

# CHƯƠNG 2. GIỚI THIỆU, MÔ TẢ ĐỀ TÀI

## 2.1. Mục tiêu và phạm vi của đề tài

Mục tiêu của đề tài

Đề tài "Tạo chương trình quản lý đồ dùng gia đình" được xây dựng nhằm đạt được các mục tiêu chính sau:

* Hỗ trợ lưu trữ thông tin đồ dùng trong gia đình: Cho phép người dùng thêm, sửa, xóa và xem danh sách các đồ dùng một cách dễ dàng.
* Tăng hiệu quả trong việc quản lý vật dụng: Giúp theo dõi số lượng, tình trạng và phân loại đồ dùng theo các tiêu chí như tên, loại, vị trí, ngày mua,...
* Tạo ra một ứng dụng đơn giản, thân thiện: Sử dụng giao diện dòng lệnh (hoặc giao diện đơn giản nếu có nâng cấp) để người dùng phổ thông cũng có thể sử dụng.
* Áp dụng kiến thức Python vào thực tế: Kết hợp giữa lý thuyết đã học và kỹ năng lập trình để xây dựng một phần mềm có tính ứng dụng.
* Lưu trữ dữ liệu lâu dài: Ứng dụng sử dụng tệp văn bản hoặc định dạng lưu trữ cơ bản để đảm bảo dữ liệu không bị mất khi thoát chương trình.

Phạm vi của đề tài

Đề tài tập trung phát triển một ứng dụng nhỏ với các chức năng chính, phù hợp cho mục đích quản lý cá nhân trong phạm vi gia đình. Cụ thể:

* Đối tượng sử dụng: Người dùng cá nhân, không yêu cầu kiến thức chuyên sâu về máy tính.
* 1 số chức năng gợi í:
  + Thêm mới đồ dùng.
  + Hiển thị danh sách đồ dùng.
  + Tìm kiếm theo tên hoặc loại.
  + Chỉnh sửa thông tin đồ dùng.
  + Xóa đồ dùng không còn sử dụng.
  + Lưu và đọc dữ liệu từ tệp.
* Không bao gồm:
  + Giao diện đồ họa phức tạp.
  + Tích hợp cơ sở dữ liệu nâng cao như MySQL, SQLite.
  + Đa người dùng hoặc chức năng bảo mật.

Với phạm vi và mục tiêu đã xác định rõ ràng, đề tài hướng đến việc tạo ra một sản phẩm đơn giản, dễ triển khai và có tính ứng dụng thực tế trong đời sống hàng ngày. Đây cũng là một bước thực hành quan trọng giúp củng cố kỹ năng lập trình Python và tư duy phát triển phần mềm.

## 2.2. Yêu cầu chức năng của chương trình

Chương trình quản lý đồ dùng gia đình được thiết kế với các chức năng chính nhằm hỗ trợ người dùng quản lý, tra cứu và cập nhật thông tin các vật dụng một cách dễ dàng. Các yêu cầu chức năng cụ thể bao gồm:

### 2.2.1. Thêm đồ dùng mới

* Cho phép người dùng nhập thông tin đồ dùng bao gồm:
  + Tên đồ dùng
  + Loại đồ dùng
  + Vị trí trong nhà
  + Số lượng
  + Tình trạng (mới, đã qua sử dụng, cần sửa chữa, v.v.)
  + Ngày mua
* Dữ liệu được lưu vào tệp để sử dụng lại khi khởi động chương trình.

### 2.2.2. Hiển thị danh sách đồ dùng

* In ra toàn bộ danh sách các đồ dùng đã được lưu.
* Hiển thị thông tin theo định dạng bảng gọn gàng, dễ đọc.
* Sắp xếp theo tên, loại, hoặc số lượng nếu cần.

### 2.2.3. Tìm kiếm đồ dùng

* Tìm kiếm theo:
  + Tên đồ dùng
  + Loại đồ dùng
  + Vị trí đặt
* Trả về danh sách kết quả phù hợp với từ khóa tìm kiếm.

### 2.2.4. Chỉnh sửa thông tin đồ dùng

* Cho phép người dùng sửa lại thông tin của một đồ dùng đã có:
  + Sửa tên, loại, vị trí, tình trạng hoặc số lượng.
* Cập nhật thông tin mới và ghi đè vào dữ liệu cũ.

### 2.2.5. Xóa đồ dùng

* Xóa một đồ dùng cụ thể khỏi danh sách theo tên hoặc mã định danh.
* Dữ liệu sau khi xóa được cập nhật lại trong tệp lưu trữ.

### 2.2.6. Lưu dữ liệu vào tệp

* Mỗi khi có thay đổi (thêm, sửa, xóa), chương trình cập nhật dữ liệu vào tệp văn bản hoặc định dạng .txt/.csv.
* Đảm bảo dữ liệu được lưu trữ lâu dài và có thể sử dụng lại trong các lần chạy sau.

### 2.2.7. Đọc dữ liệu từ tệp khi khởi động

* Khi khởi chạy chương trình, toàn bộ dữ liệu đồ dùng được đọc từ tệp lưu trữ và tải vào bộ nhớ để sử dụng.
* Giúp người dùng tiếp tục làm việc mà không cần nhập lại từ đầu.

### 2.2.8. Thoát chương trình an toàn

* Cung cấp tùy chọn để người dùng thoát chương trình khi cần.
* Trước khi thoát, chương trình kiểm tra và đảm bảo đã lưu đầy đủ dữ liệu vào tệp.

Các chức năng trên đảm bảo chương trình hoạt động hiệu quả, đơn giản và phù hợp với nhu cầu sử dụng trong phạm vi gia đình. Ngoài ra, các chức năng cũng được xây dựng sao cho dễ mở rộng trong tương lai nếu cần tích hợp giao diện đồ họa hoặc cơ sở dữ liệu nâng cao.

**CHƯƠNG 3. PHÁT TRIỂN VÀ TRIỂN KHAI ỨNG DỤNG**

**3.1. Tiểu mục 1**

***3.1.1. Tiêu mục con***

***3.1.2. Tiêu mục con***

**3.2. Tiểu mục 2**

***3.2.1. Tiêu mục con***

***3.2.2. Tiêu mục con***

**KẾT LUẬN**

**DANH MỤC TÀI LIỆU THAM KHẢO**

1. Fundamentals of Programming Python, Richard L. Halterman. Southern Adventist University 2019.
2. Nguyễn Ngọc Giang, Phan Xuân Vọng, Nguyễn Quốc Anh, Đường Vào Lập Trình Python, NXB Đại Học Quốc Gia, 2020
3. Wes McKinney, Python for Data Analysis, 3rd Edition, O'Reilly Media, Inc, 2022