

|  |
| --- |
| **ĐẠI HỌC DUY TÂN TRƯỜNG KHOA HỌC MÁY TÍNH**  **KHOA KỸ THUẬT MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**  *🙦*🕮🙤 |
| **ĐỒ ÁN CÁ NHÂN/ NHÓM**  **MÔN:** **PERL & PYTHON – MÃ MÔN: CS 466** |
| ***Đề tài:*** |
| **LẬP TRÌNH WEB BÁN QUẦN ÁO PYSHOP BẰNG NGÔN NGỮ PYTHON** |
| Nhóm SV thực hiện: Nhóm 17   1. Nguyễn Văn Thắng 2. Trần Phước Tín 3. Trang Minh Trí 4. Nguyễn Ngọc Trường 5. Võ Anh Tuấn   Lớp môn học: CS 466 A  GVHD: ThS. TRẦN HỮU MINH ĐĂNG |
| Đà Nẵng, 12/2023 |

# BẢNG PHÂN CÔNG CÔNG VIỆC

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **STT** | **Thành viên** | **Phân công nhiệm vụ** | **Phần trăm làm việc** | **Sinh viên (ký tên)** | **Điểm chi tiết** | | **Điểm Bình quân (3)** | | **Giảng viên bộ môn  (ký tên)** |
| **Cột điểm (1)** | **Cột điểm (2)** | **Điểm số** | **Điểm Chữ** |
| 1 | Nguyễn Văn Thắng | - Tìm kiếm thông tin thiết lập dự án PyShop.  - Cài đặt demo.  - Thuyết trình. | 100% |  |  |  |  |  |  |
| 2 | Trần Phước Tín | - Nghiên cứu tìm hiểu về String trong Python và các thuật toán liên quan.  - Làm word. | 100% |  |  |  |  |  |  |
| 3 | Trang Minh  Trí | - Nghiên cứu tìm hiểu về Python Puple, Dictionaryvà Flow Control.  - Làm word. | 100% |  |  |  |  |  |  |
| 4 | Nguyễn Ngọc Trường | - Nghiên cứu tìm hiểu về tổng quát Python, Application, DataTypes, List và Set.  - Làm word. | 100% |  |  |  |  |  |  |
| 5 | Võ Anh  Tuấn  (Nhóm trưởng) | - Nghiên cứu tìm hiểu về Python Function và Files  - Tập hợp thông tin, chỉnh sửa word.  - Làm Slide thuyết trình. | 100% |  |  |  |  |  |  |

**Chú thích:**

* + Cột điểm (1): Trình bày Slide & Ceminar
  + Cột điểm (2): Trình bày Word (quyển In nộp)
  + Cột điểm bình quân (3): (Cột điểm (1) + Cột điểm (2))/2

# NHẬN XÉT CHO ĐIỂM ĐÁNH GIÁ CỦA GIẢNG VIÊN















MỤC LỤC

[BẢNG PHÂN CÔNG CÔNG VIỆC 1](#_Toc154181588)

[NHẬN XÉT CHO ĐIỂM ĐÁNH GIÁ CỦA GIẢNG VIÊN 2](#_Toc154181589)

[MỞ ĐẦU 5](#_Toc154181590)

[Mục tiêu tìm hiểu: 5](#_Toc154181591)

[Đối tượng nghiên cứu: 5](#_Toc154181592)

[Phương pháp nghiên cứu: 5](#_Toc154181593)

[Cấu trúc đồ án nhóm/ cá nhân 6](#_Toc154181594)

[CHƯƠNG 1: TỔNG QUAN VỀ PYTHON. 7](#_Toc154181595)

[1.1. PYTHON INTRODUCTION. 7](#_Toc154181596)

[1.2. PYTHON APPLICATION. 7](#_Toc154181597)

[1.3. PYTHON DATATYPES. 7](#_Toc154181598)

[1.3.1. Python List. 12](#_Toc154181599)

[1.3.2. Python Set. 13](#_Toc154181600)

[1.3.3. Python Puple. 14](#_Toc154181601)

[1.3.4. Python Dictionary. 15](#_Toc154181602)

[1.3.5. String. 17](#_Toc154181603)

[1.4. PYTHON FLOW CONTROL. 23](#_Toc154181604)

[1.5. PYTHON FUCTION. 29](#_Toc154181605)

[1.6. PYTHON FILES. 33](#_Toc154181606)

[KẾT LUẬN CHƯƠNG 1 34](#_Toc154181607)

[CHƯƠNG 2: TÌM HIỂU VÀ LẬP TRÌNH WEB BÁN QUẦN ÁO PYSHOP BẰNG NGÔN NGỮ PYTHON 35](#_Toc154181608)

[2.1. THIẾT LẬP DỰ ÁN PYSHOP - DJANGO & VUE WEB SERVICE 35](#_Toc154181609)

[2.1.1. Ý Tưởng Dự Án 35](#_Toc154181610)

[2.1.2. Mục Tiêu 35](#_Toc154181611)

[2.2. DJANGO - BACKEND 35](#_Toc154181612)

[2.2.1. Cấu Trúc Dự Án Django 35](#_Toc154181613)

[2.2.2. Xử Lý Dữ Liệu: 36](#_Toc154181614)

[2.3. VUE - FRONTEND 38](#_Toc154181615)

[2.3.1. Tích Hợp Vue.js: 38](#_Toc154181616)

[2.3.2. Tối Ưu Hóa Hiệu Suất: 38](#_Toc154181617)

[2.3.3. Các bước thiết lập dự án Vue: 38](#_Toc154181618)

[2.4. SQLITE - DATABASE 39](#_Toc154181619)

[2.4.1. Schema Database: 39](#_Toc154181620)

[2.4.2. ER Diagram: 40](#_Toc154181621)

[KẾT LUẬN CHƯƠNG 2 40](#_Toc154181622)

[CHƯƠNG 3: TRIỂN KHAI CÀI ĐẶT VÀ THỰC HIỆN DỰ ÁN 41](#_Toc154181623)

[3.1. CÁC CHỨC NĂNG 41](#_Toc154181624)

[3.1.1. Chức năng hiển thị sản phẩm 41](#_Toc154181625)

[3.1.2. Chức năng xem chi tiết sản phẩm 41](#_Toc154181626)

[3.1.3. Chức năng phân loại 42](#_Toc154181627)

[3.1.4. Chức năng tìm kiếm 43](#_Toc154181628)

[3.1.5. Chức năng đăng kí - đăng nhập 43](#_Toc154181629)

[3.1.6. Chức năng thêm vào giỏ hàng. 44](#_Toc154181630)

[3.1.7. Chức năng đặt hàng - thanh toán. 45](#_Toc154181631)

[3.2. NHẬN XÉT & ĐÁNH GIÁ 46](#_Toc154181632)

[3.2.1. Ưu điểm 46](#_Toc154181633)

[3.2.2. Nhược điểm 46](#_Toc154181634)

[KẾT LUẬN 47](#_Toc154181635)

[A) KẾT QUẢ ĐẠT ĐƯỢC 47](#_Toc154181636)

[B) HẠN CHẾ CỦA ĐỀ TÀI 47](#_Toc154181637)

[C) HƯỚNG PHÁT TRIỂN ĐỀ TÀI 47](#_Toc154181638)

[TÀI LIỆU THAM KHẢO 48](#_Toc154181639)

# 

# MỞ ĐẦU

Trong thời đại ngày nay, khi cuộc sống xã hội ngày càng phức tạp và công nghệ phát triển với tốc độ nhanh chóng, việc mua sắm trực tuyến đã trở thành một xu hướng không thể phủ nhận. Tuy nhiên, với sự gia tăng về số lượng các trang web bán hàng, người tiêu dùng đang đối mặt với một thách thức mới: làm thế nào để lựa chọn được một trang web chất lượng, đảm bảo an toàn và đồng thời đáp ứng đầy đủ nhu cầu thời trang của họ.

Chính từ nhận thức về những thách thức này, chúng tôi quyết định mang đến một trải nghiệm mua sắm các loại quần áo trực tuyến theo mùa trong năm với dự án mang tên “WEB BÁN QUẦN ÁO PYSHOP” được viết hoàn toàn bằng ngôn ngữ Python.

Thông qua dự án, chúng tôi hy vọng được học hỏi, tìm kiếm và nâng cao hơn về cách thức thực hiện các trang web bán hàng được viết bằng ngôn ngữ Python nói riêng và ngành bán hàng online đầy tìm năng nói chung.

## Mục tiêu tìm hiểu:

* Biết cơ bản về ngôn ngữ lập trình Python, bao gồm cú pháp, cấu trúc dữ liệu, và các khái niệm quan trọng. Biết cách sử dụng framework web trong Python là **Django** để làm backend, sử dụng **Vue** cho frontend và **Sqlite** cho database.
* Phân tích yêu cầu chức năng và thiết kế của một trang web bán quần áo, bao gồm giao diện người dùng và cơ sở dữ liệu. Hiểu cách tổ chức dữ liệu liên quan đến quản lý sản phẩm, danh mục, đơn đặt hàng và tài khoản người dùng.
* Vận dụng vào lập trình một trang web bán quần áo theo mùa trong năm qua ngôn ngữ Python.

## Đối tượng nghiên cứu:

* Lập trình web bán quần áo bằng ngôn ngữ Python.
* Các công nghệ sử dụng để tạo nên trang web, gồm có:
  + Framework Django - Backend.
  + Thư viện Vue.js - Frontend.
  + Hệ quản trị cơ sở dữ liệu SQLite.

## Phương pháp nghiên cứu:

* Lý thuyết: Tham khảo và tổng hợp từ sách, giáo trình, … và các video hướng dẫn trên Youtube.



* Thực nghiệm: Cài đặt và chạy dự án qua..

## Cấu trúc đồ án nhóm/ cá nhân

Đồ án bao gồm phần mở đầu, ba chương nội dung và phần kết luận.

**Chương 1:** Tổng quan về Python. Chương này sẽ nói khái niệm python, ứng dụng của python, các kiểu dữ liệu, mô tả hoạt động của các chức năng, các hàm trong python, cũng như ưu và nhược điểm của nó. Cuối cùng là những bài code làm quen với ngôn ngữ python.

**Chương 2:** Tìm hiểu và lập trình web bán quần áo PyShop bằng ngôn ngữ Python. Chương này sẽ làm rõ ý tưởng, mô tả cách hoạt động khi ứng dụng các công nghệ liên qua đến Python vào dự án, đánh giá ưu và nhược điểm của nó.

**Chương 3:** Triển khai các chức năng có trong PyShop.

Kết luận và hướng phát triển của đề tài.

# CHƯƠNG 1: TỔNG QUAN VỀ PYTHON.

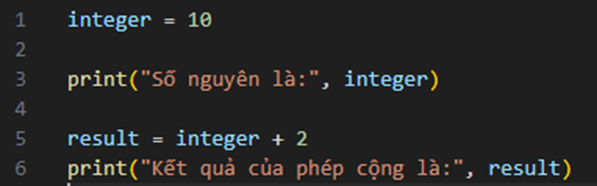
## PYTHON INTRODUCTION.

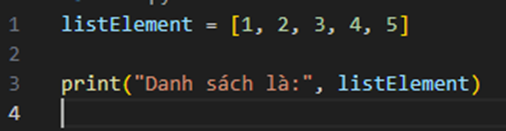
1. **Python là ngôn ngữ lập trình nào?**Ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng.
2. **Ai là tác giả chính của Python?** Guido van Rossum.
3. **Python được phát triển vào năm nào?** Năm 1991.
4. **Python là ngôn ngữ lập trình thông dịch (interpreted) hay biên dịch (compiled)?** Là ngôn ngữ lập trình thông dịch (interpreted).
5. **Python có ưu điểm gì nổi bật so với các ngôn ngữ lập trình khác?**  
   Ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ, đa mục đích, cộng đồng lớn, thư viện và framework đa dạng, tương thích nổi tiếng, hỗ trợ đối tượng, ngắn gọn và linh hoạt.

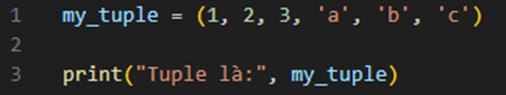
## PYTHON APPLICATION.

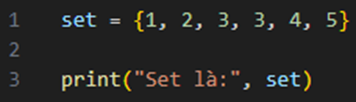
1. **Python được sử dụng trong những lĩnh vực ứng dụng nào trong phát triển web**  
   Framework Web, phát triển Backend, Microservices, RESTful APIs, Web Scraping, Real-time Applications, Content Management Systems (CMS).
2. **Python được áp dụng trong các ngành công nghiệp nào khác ngoài công nghệ thông tin và phát triển phần mềm**  
   Khoa học Dữ liệu và Trí tuệ Nhân tạo (AI), FinTech (Công nghệ Tài chính), Y tế và Sinh học, Quảng cáo và Tiếp thị, E-commerce, Giáo dục, Quản lý Hệ thống và Tự động hóa, Năng lượng và Môi trường

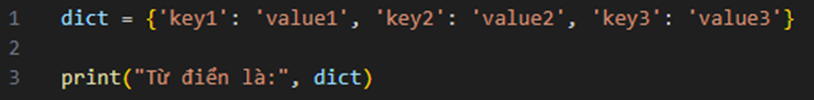
## PYTHON DATATYPES.

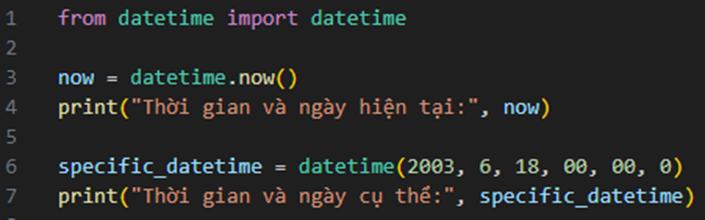
**a. Kiểu dữ liệu nào được sử dụng để lưu trữ số nguyên trong Python? cho ví dụ?**Kiểu dữ liệu “int” (integer).  
Ví dụ: ****

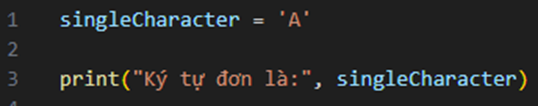
**b. Kiểu dữ liệu nào được sử dụng để lưu trữ danh sách các phần tử có thứ tự trong Python? cho ví dụ?**Kiểu dữ liệu “list”.  
Ví dụ: ****

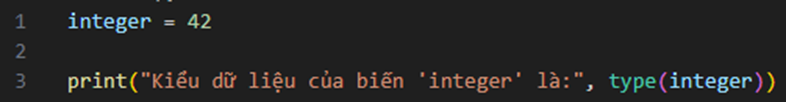
**c. Làm thế nào để tạo một tuple trong Python? cho ví dụ?** Để tạo một tuple bằng cách sử dụng dấu ngoặc đơn () và liệt kê các phần tử của tuple cách nhau bởi dấu phẩy. ****

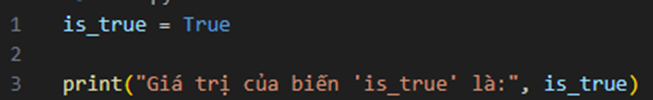
**d. Kiểu dữ liệu nào được sử dụng để lưu trữ tập hợp các giá trị duy nhất trong Python? cho ví dụ?**Kiểu dữ liệu “set”.  
Ví dụ: ****

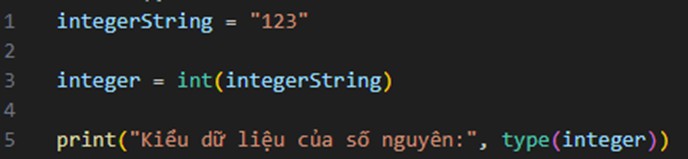
**e. Làm thế nào để tạo một từ điển (dictionary) trong Python? cho ví dụ?** Để tạo một từ điển bằng cách sử dụng dấu ngoặc nhọn “{}” và liệt kê các cặp “key-value” bên trong. Mỗi cặp “key-value” được phân cách bởi dấu hai chấm “:” và các cặp “key-value” được cách nhau bởi dấu phẩy “,”.  
 Ví dụ: ****

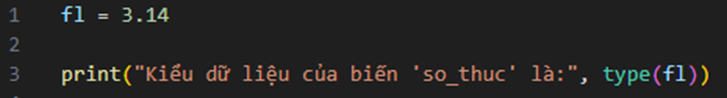
**f. Kiểu dữ liệu nào được sử dụng để lưu trữ thời gian và ngày trong Python? cho ví dụ?** Kiểu dữ liệu “datetime”. ****

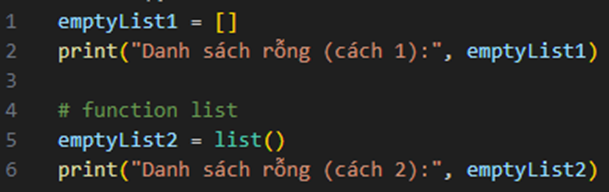
**g. Kiểu dữ liệu nào được sử dụng để biểu diễn một ký tự đơn trong Python? cho ví dụ?** Kiểu dữ liệu “str”.  
 Ví dụ: ****

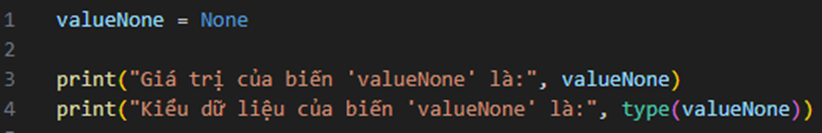
**h. Làm thế nào để kiểm tra kiểu dữ liệu của một biến trong Python? cho ví dụ?** Sử dụng hàm type().  
 Ví dụ: ****

**i. Kiểu dữ liệu nào được sử dụng để biểu diễn các giá trị logic True và False trong Python? cho ví dụ?** Kiểu dữ liệu “bool” (boolean).  
 Ví dụ: ****

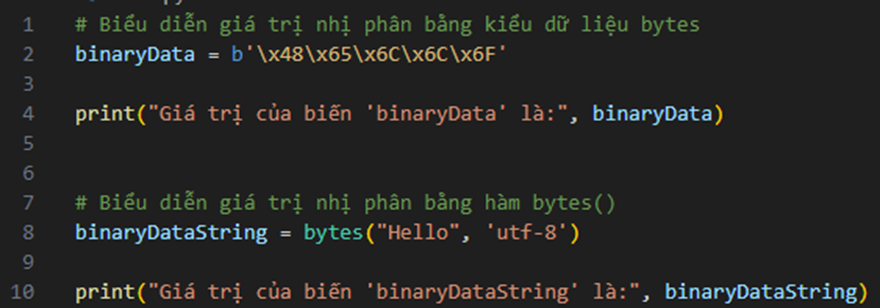
**j. Làm thế nào để chuyển đổi một chuỗi thành một số nguyên trong Python? cho ví dụ?** Sử dụng hàm “int()”.  
 Ví dụ: ****

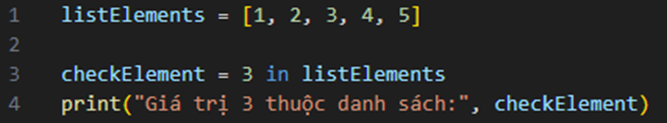
**k. Kiểu dữ liệu nào được sử dụng để lưu trữ các giá trị thực sự trong Python? cho ví dụ?** Kiểu dữ liệu “float”.  
 Ví dụ: ****

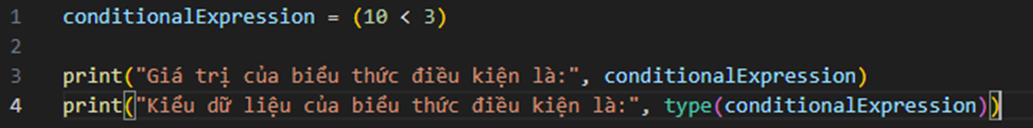
**l. Làm thế nào để tạo một danh sách rỗng trong Python? cho ví dụ?** Sử dụng dấu ngoặc vuông [] hoặc hàm list().  
 Ví dụ:  


**m. Kiểu dữ liệu nào được sử dụng để biểu diễn giá trị không tồn tại hoặc trống trong Python? cho ví dụ?** Kiểu dữ liệu “NoneType”, và giá trị là None.  
 Ví dụ:  


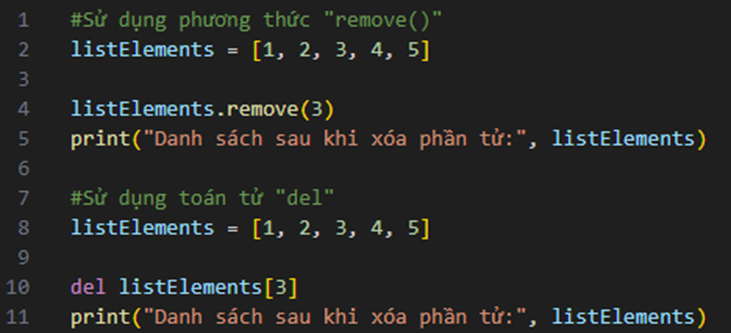
**n. Làm thế nào để thêm một phần tử vào danh sách trong Python? cho ví dụ?** Sử dụng phương thức append() hoặc insert() của danh sách.  
 Ví dụ:  


**o. Kiểu dữ liệu nào được sử dụng để biểu diễn các giá trị nhị phân trong Python? cho ví dụ?** Kiểu dữ liệu “bytes” hoặc “bytearray”.  
 Ví dụ:  


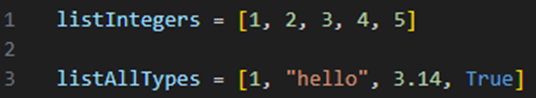
**p. Làm thế nào để kiểm tra xem một giá trị có thuộc một danh sách hay không trong Python? cho ví dụ?** Sử dụng toán tử “in”.  
 Ví dụ: ****

**q. Kiểu dữ liệu nào được sử dụng để biểu diễn các biểu thức điều kiện trong Python? cho ví dụ?** Kiểu dữ liệu “bool” (boolean).  
 Ví dụ: ****

**r. Làm thế nào để xóa một phần tử khỏi danh sách trong Python? cho ví dụ?**

Sử dụng phương thức “remove()” hoặc “del”.  
Ví dụ: ****

### Python List.

**a. List trong Python là gì và cách khai báo một List** - List là một kiểu dữ liệu được sử dụng để lưu trữ một tập hợp các phần tử có thứ tự. Danh sách có thể chứa các phần tử có kiểu dữ liệu khác nhau và được đặt trong dấu ngoặc vuông [].  
 - Khai báo một list: ****

**b. Làm thế nào để truy cập và lấy giá trị của một phần tử trong một List** Sử dụng chỉ số (index) của phần tử đó trong danh sách. Lưu ý rằng chỉ số của danh sách bắt đầu từ 0.

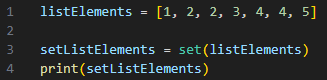
**c. Python cho phép thực hiện các thao tác nào để thêm và xóa phần tử trong một List** Thêm phần tử: Phương thức append(), phương thức insert(),  
 Xóa phần tử: Phương thức remove(), toán tử “del”.

**d. Làm thế nào để tìm độ dài (số lượng phần tử) của một List** Sử dụng hàm tích hợp len().

**e. Python cung cấp cách nào để lặp qua các phần tử của một List** Sử dụng vòng lặp for, sử dụng hàm tích hợp enumerate().

### Python Set.

1. **Python Set là gì và cách khai báo một Set trong Python.**

Set là một loại dữ liệu được sử dụng để lưu trữ các phần tử duy nhất không có thứ tự. Set không chứa các phần tử trùng lặp và không có thứ tự cụ thể.  
 Cách khai báo một Set trong Python:  
 

1. **Làm thế nào để kiểm tra xem một phần tử có tồn tại trong một Set hay không trong Python.**

Để kiểm tra xem một phần tử có tồn tại trong một set hay không trong Python, bạn có thể sử dụng toán tử in hoặc phương thức \_\_contains\_\_() của set.

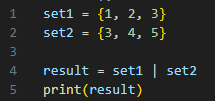
1. **Python Set có thể chứa các phần tử trùng lặp không? Nếu có, làm thế nào để xóa các phần tử trùng lặp trong Set.**

Không, Python Set không thể chứa các phần tử trùng lặp. Set trong Python là một tập hợp không thứ tự và không chứa các phần tử trùng lặp. Khi bạn thêm một phần tử vào set và phần tử đó đã tồn tại trong set, set sẽ không bị thay đổi và phần tử mới sẽ không được thêm vào.

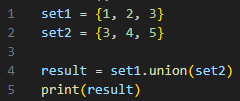
1. **Hãy nêu cách thực hiện các phép toán tập hợp cơ bản như hợp (union), giao (intersection) và hiệu (difference) trên các Set trong Python.**

**Hợp (Union):**

* Sử dụng toán tử |:

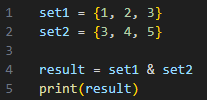


* Sử dụng phương thức union():

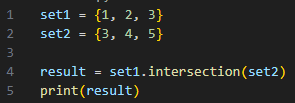


**Giao (Intersection):**

* Sử dụng toán tử &:

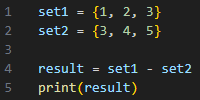


* Sử dụng phương thức intersection():

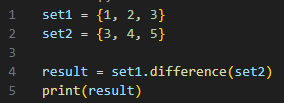


**Hiệu (Difference):**

* Sử dụng toán tử -:



* Sử dụng phương thức difference():



1. **Làm thế nào để thêm và xóa phần tử từ một Set trong Python.**

Thêm phần tử: Phương thức add(), Phương thức update()

Xóa phần tử: Phương thức remove(), Phương thức discard(), Phương thức pop(), Phương thức clear().

### Python Puple.

1. **Python Tuple là gì và cách khai báo một Tuple trong Python?**

* Trong Python, một tuple là một kiểu dữ liệu có thứ bậc (ordered) và là một dạng dữ liệu không thay đổi (immutable). Có thể chứa nhiều giá trị khác nhau. Một tuple được tạo ra bằng cách đặt các giá trị vào trong dấu ngoặc đơn và ngăn cách chúng bằng dấu phẩy.
* Cách khai báo: Sử dụng dấu ngoặc đơn và dấu phẩy.

my\_tuple = (1, 2, 3, 'a', 'b', 'c')

1. **Trong Python, Tuple có thể chứa các kiểu dữ liệu gì?**

* Một tuple có thể chứa các giá trị kiểu dữ liệu khác nhau, bao gồm số nguyên, số thực, chuỗi ký tự và ngay cả các tuple khác.

1. **Làm thế nào để truy cập các phần tử trong một Tuple?**

* Truy cập phần tử bằng chỉ mục:

my\_tuple = (1, 2, 3, 4, 5)

print(my\_tuple[0]) # Output: 1

1. **Python Tuple có khác biệt gì so với Python List?**

* Khả năng thay đổi:
* Tuple: là dạng dữ liệu không thay đổi, có nghĩa là sau khi tạo tuple, bạn không thể thay đổi giá trị của các phần tử trong tuple.
* List: là dạng dữ liệu thay đổi, cho phép bạn thêm, xóa, và sửa đổi các phần tử trong list.
* Cách khai báo và cú pháp:
* Tuple: được khai báo bằng cách đặt các giá trị vào trong dấu ngoặc đơn và ngăn cách chúng bằng dấu phẩy.
* List: được khai báo bằng cách đặt các giá trị vào trong dấu ngoặc vuông và ngăn cách chúng bằng dấu phẩy.

1. **Làm thế nào để thay đổi một phần tử trong một Tuple?**

* Chuyển đổi tuple thành list.
* Thay đổi giá trị phần tử trong list.
* Chuyển đổi lại list thành tuple.
* Ví dụ:

my\_tuple = (1, 2, 3, 4, 5)

my\_list = list(my\_tuple) # Chuyển đổi tuple thành list

my\_list[2] = 6 # Thay đổi giá trị phần tử thứ 2 trong list

my\_tuple = tuple(my\_list) # Chuyển đổi lại list thành tuple

print(my\_tuple) # Output: (1, 2, 6, 4, 5)

### Python Dictionary.

1. **Python Dictionary là gì và có cấu trúc như thế nào**

* Python Dictionary là một cấu trúc dữ liệu trong Python được sử dụng để lưu trữ và quản lý dữ liệu dưới dạng cặp key-value (khóa-giá trị). Mỗi key trong Dictionary phải là duy nhất và không thay đổi (immutable), trong khi giá trị có thể là bất kỳ kiểu dữ liệu nào.
* Cấu trúc của một dictionary trong Python là một tập hợp các cặp key-value, trong đó key là duy nhất và không thay đổi, và giá trị của mỗi key được liên kết với nó.

1. **Làm thế nào để tạo một Dictionary trong Python và đưa các phần tử vào Dictionary**

* Tạo một Dictionary:
* Cách 1: Tạo Dictionary trống

my\_dict1 = {}

* Cách 2: Sử dụng dấu ngoặc nhọn và các cặp key-value

my\_dict2 = {"key1": "value1", "key2": "value2", "key3": "value3"}

* Cách 3: Sử dụng hàm dict()

my\_dict3 = dict(key1="value1", key2="value2", key3="value3")

* Đưa Các Phần Tử vào Dictionary:
* Tạo một Dictionary trống

my\_dict = {}

* Đưa các phần tử vào Dictionary

my\_dict["name"] = "John"

my\_dict["age"] = 30

my\_dict["city"] = "New York"

1. **Cách truy cập các giá trị trong một Dictionary trong Python**

* Sử Dụng Key

# Tạo một Dictionary

my\_dict = {"name": "John", "age": 30, "city": "New York"}

# Truy cập giá trị bằng cách sử dụng key

name\_value = my\_dict["name"]

age\_value = my\_dict["age"]

city\_value = my\_dict["city"]

print("Name:", name\_value)

print("Age:", age\_value)

print("City:", city\_value)

1. **Làm thế nào để thêm, sửa đổi hoặc xóa phần tử trong một Dictionary**

* Thêm Phần Tử:
* Sử Dụng Phương Thức update():

# Tạo một Dictionary

my\_dict = {"name": "John", "age": 30, "city": "New York"}

# Thêm nhiều phần tử mới

my\_dict.update({"occupation": "Engineer", "salary": 50000})

print("Updated Dictionary:", my\_dict)

* Sửa Đổi Phần Tử:

# Sửa đổi giá trị của một phần tử

my\_dict["age"] = 31

print("Updated Dictionary:", my\_dict)

* Xóa Phần Tử:
* Sử Dụng Key:

# Tạo một Dictionary

my\_dict = {"name": "John", "age": 30, "city": "New York"}

# Xóa một phần tử bằng key

del my\_dict["age"]

1. **Python Dictionary có những phương thức và tính năng gì giúp thực hiện các thao tác phổ biến**

* len(): Trả về số lượng cặp key-value trong dictionary.
* keys(): Trả về một danh sách chứa tất cả các keys trong dictionary.
* values(): Trả về một danh sách chứa tất cả các values trong dictionary.
* items(): Trả về một danh sách chứa tất cả các cặp key-value dưới dạng tuples.
* get(key, default): Trả về giá trị tương ứng với key, nếu key không tồn tại, trả về giá trị mặc định (default).
* pop(key, default): Loại bỏ phần tử có key chỉ định khỏi dictionary và trả về giá trị của key. Nếu key không tồn tại, trả về giá trị mặc định (default).
* update(other\_dict): Cập nhật dictionary với các cặp key-value từ dictionary khác.
* clear(): Xóa tất cả các cặp key-value khỏi dictionary

### String.

1. **Làm thế nào để khai báo một chuỗi (string) trong Python**  
   Để khai báo một chuỗi trong Python, bạn có thể sử dụng dấu nháy đơn (' '), dấu nháy kép (" "), hoặc cú pháp """ """ để bao quanh chuỗi.
2. **Python cung cấp những phương thức nào để cắt và nối chuỗi**

Cắt chuỗi: Sử dụng cú pháp chuỗi[start:end] để cắt một phần của chuỗi từ vị trí 'start' đến vị trí 'end' (không bao gồm 'end')

Nối chuỗi: Sử dụng toán tử ‘+’ hoặc phương thức join().

1. **Làm thế nào để kiểm tra độ dài của một chuỗi trong Python**

Sử dụng hàm len()

1. **Python cho phép định dạng chuỗi như thế nào, ví dụ: làm thế nào để chuyển một số thành chuỗi có định dạng cụ thể**

Trong Python, có thể định dạng chuỗi bằng cách sử dụng phương pháp format hoặc f-strings.

Sử dụng phương thức format:

so\_nguyen = 42

chuoi = "Số nguyên là: {}".format(so\_nguyen)

Sử dụng f-strings

chuoi = f"Số nguyên là: {so\_nguyen}"

1. **Làm thế nào để tìm kiếm và thay thế một phần của chuỗi trong Python**

Sử dụng phương thức .replace(phần cần tìm, phần cần thay thế)

chuoi = "Hôm nay là ngày thứ hai"

chuoi\_moi = chuoi.replace("hai", "ba")

print(chuoi\_moi) # Kết quả sẽ là: "Hôm nay là ngày thứ ba"

1. **Viết chương trình (hàm) tạo một xâu và in ra màn hình?**

def tao\_chuoi():

chuoi = "Đây là chuỗi mẫu được tạo bởi chương trình Python"

return chuoi

print(tao\_chuoi())

1. **Viết chương trình (hàm) tìm tần suất xuất hiện của một ký tự trong một xâu?**

def tinh\_tan\_suat(chuoi, ky\_tu):

tan\_suat = chuoi.count(ky\_tu)

return tan\_suat

# Sử dụng hàm để tìm tần suất xuất hiện của ký tự 'a' trong chuỗi 'Hello, ai lại đây!'

tan\_suat\_a = tinh\_tan\_suat('Hello, ai lại đây!', 'a')

print("Tần suất xuất hiện của ký tự 'a' là:", tan\_suat\_a)

1. **Viết chương trình (hàm) đảo ngược xâu?**

def dao\_nguoc\_chuoi(chuoi):

chuoi\_dao\_nguoc = chuoi[::-1]

return chuoi\_dao\_nguoc

# Sử dụng hàm để đảo ngược chuỗi "Hello, world!"

chuoi\_nguoc = dao\_nguoc\_chuoi("Hello, world!")

print("Chuỗi đảo ngược là:", chuoi\_nguoc)

1. **Viết chương trình (hàm) tìm xâu con dài nhất trong một xâu?**

def tim\_xau\_con\_dai\_nhat(chuoi):

chuoi = chuoi + " " # Thêm dấu cách ở cuối chuỗi để đảm bảo việc xác định xâu con hoàn chỉnh

xau\_con\_hien\_tai = '' # Biến lưu trữ xâu con hiện tại

xau\_con\_dai\_nhat = '' # Biến lưu trữ xâu con dài nhất

for ky\_tu in chuoi:

if ky\_tu != ' ':

xau\_con\_hien\_tai += ky\_tu

else:

if len(xau\_con\_hien\_tai) > len(xau\_con\_dai\_nhat):

xau\_con\_dai\_nhat = xau\_con\_hien\_tai

xau\_con\_hien\_tai = ''

return xau\_con\_dai\_nhat

# Sử dụng hàm để tìm xâu con dài nhất trong chuỗi "Python là ngôn ngữ lập trình tuyệt vời"

xau\_con\_dai\_nhat = tim\_xau\_con\_dai\_nhat("Python là ngôn ngữ lập trình tuyệt vời")

print("Xâu con dài nhất là:", xau\_con\_dai\_nhat)

1. **Viết chương trình (hàm) tìm xâu con xuất hiện nhiều nhất trong một xâu?**

def tim\_xau\_con\_nhieu\_nhat(xau):

so\_lan\_xuat\_hien = {}

max\_so\_lan\_xuat\_hien = 0

xau\_con\_nhieu\_nhat = []

# Duyệt qua từng xâu con có độ dài từ 1 đến độ dài của xâu

for i in range(len(xau)):

for j in range(i+1, len(xau)+1):

xau\_con = xau[i:j]

if xau\_con in so\_lan\_xuat\_hien:

so\_lan\_xuat\_hien[xau\_con] += 1

else:

so\_lan\_xuat\_hien[xau\_con] = 1

if so\_lan\_xuat\_hien[xau\_con] > max\_so\_lan\_xuat\_hien:

max\_so\_lan\_xuat\_hien = so\_lan\_xuat\_hien[xau\_con]

xau\_con\_nhieu\_nhat = [xau\_con]

elif so\_lan\_xuat\_hien[xau\_con] == max\_so\_lan\_xuat\_hien:

xau\_con\_nhieu\_nhat.append(xau\_con)

return xau\_con\_nhieu\_nhat, max\_so\_lan\_xuat\_hien

1. **Viết chương trình (hàm) loại bỏ khoảng trắng đầu và cuối của một xâu?**

def loai\_bo\_khoang\_trang(xau):

xau\_moi = xau.strip()

return xau\_moi

xau = " Hello, world! "

xau\_da\_loai\_bo = loai\_bo\_khoang\_trang(xau)

print(xau\_da\_loai\_bo)

1. **Viết chương trình (hàm) chuyển một xâu sang viết hoa hoặc viết thường?**

def chuyen\_doi\_chu\_hoa(xau):

return xau.upper()

def chuyen\_doi\_chu\_thuong(xau):

return xau.lower()

xau = "Hello, World!"

xau\_chu\_hoa = chuyen\_doi\_chu\_hoa(xau)

xau\_chu\_thuong = chuyen\_doi\_chu\_thuong(xau)

print("Xâu viết hoa:", xau\_chu\_hoa)

print("Xâu viết thường:", xau\_chu\_thuong)

1. **Viết chương trình (hàm) tách một xâu dựa trên ký tự hoặc xâu trắng?**

def tach\_xau(xau, ky\_tu\_tach=" "):

return xau.split(ky\_tu\_tach)

xau = "Hello, world! This is a string."

xau\_da\_tach = tach\_xau(xau)

print(xau\_da\_tach)

xau\_da\_tach\_theo\_dau\_phoi = tach\_xau(xau, ",")

print(xau\_da\_tach\_theo\_dau\_phoi)

1. **Viết chương trình (hàm) tìm số lần xuất hiện của một xâu con trong một xâu khác?**

def dem\_so\_lan\_xuat\_hien(xau\_cha, xau\_con):

so\_lan\_xuat\_hien = xau\_cha.count(xau\_con)

return so\_lan\_xuat\_hien

xau\_cha = "abababcbab"

xau\_con = "ab"

so\_lan\_xuat\_hien = dem\_so\_lan\_xuat\_hien(xau\_cha, xau\_con)

print("Số lần xuất hiện của xâu con trong xâu là:", so\_lan\_xuat\_hien)

1. **Viết chương trình (hàm) đếm số các kí tự là chữ số trong xâu S cho trước?**

def dem\_so\_chu\_so(xau):

dem = 0

for ky\_tu in xau:

if ky\_tu.isdigit():

dem += 1

return dem

xau = "Abc123Def456"

so\_chu\_so = dem\_so\_ch

1. **Viết chương trình (hàm) đếm số các kí tự là chữ cái tiếng Anh trong xâu S cho trước?**

def dem\_so\_chu\_cai(xau):

so\_chu\_cai = 0

for ky\_tu in xau:

if ky\_tu.isalpha():

so\_chu\_cai += 1

return so\_chu\_cai

xau = "Abc123Def456"

so\_chu\_cai = dem\_so\_chu\_cai(xau)

print("Số lượng ký tự là chữ cái trong xâu là:", so\_chu\_cai)

1. **Cho trước hai xâu kí tự s1, s2. Viết chương trình (hàm) in ra xâu kí tự bao gồm lần lượt các kí tự được lấy ra từ s1, s2. Nếu một trong hai xâu s1, s2 hết trước thì lấy tiếp từ xâu còn lại. Ví dụ nếu s1 = “012”, s2 = “abcde” thì xâu kết quả sẽ là “0a1b2cde”.**

def ghep\_xau(s1, s2):

xau\_ket\_qua = ""

i, j = 0, 0

while i < len(s1) and j < len(s2):

xau\_ket\_qua += s1[i] + s2[j]

i += 1

j += 1

xau\_ket\_qua += s1[i:] + s2[j:]

return xau\_ket\_qua

s1 = "012"

s2 = "abcde"

xau\_ket\_qua = ghep\_xau(s1, s2)

print(xau\_ket\_qua) # Kết quả sẽ là "0a1b2cde"

1. **Cho trước hai xâu kí tự s1, s2. Viết chương trình (hàm) tính in ra xâu kí tự là phần chung đầu tiên cực đại của hai xâu s1, s2. Ví dụ nếu s1 = “abcdeghik”, s2 = “abcmnopq” thì xấu cần tính là “abc”.**

def phan\_chung\_dau\_cuc\_dai(s1, s2):

xau\_chung = ""

i = 0

while i < len(s1) and i < len(s2) and s1[i] == s2[i]:

xau\_chung += s1[i]

i += 1

return xau\_chung

s1 = "abcdeghik"

s2 = "abcmnopq"

print(phan\_chung\_dau\_cuc\_dai(s1, s2)) # Kết quả sẽ là "abc"

1. **Cho trước xâu kí tự S. Viết chương trình (hàm) xoá đi các dấu cách thừa trong xâu S. Dấu cách thừa là các dấu cách ở đầu, cuối và giữa các từ. Quy định giữa các từ chỉ có một dấu cách.   
   Ví dụ nếu S = “ baby table cloud “ thì sau khi sửa S sẽ trở thành “baby table cloud”.**

def xoa\_dau\_cach\_thua(xau):

return ' '.join(xau.split())

xau = " baby table cloud "

xau\_da\_sua = xoa\_dau\_cach\_thua(xau)

print(xau\_da\_sua) # Kết quả sẽ là "baby table cloud"

1. **Viết chương trình (hàm) nhập một xâu là họ tên đầy đủ (gồm họ, đệm, tên) từ bàn phím, sau đó in ra màn hình lần lượt họ, đệm và tên vừa nhập. Ví dụ:  
    Nhập họ tên đầy đủ: Nguyễn Xuân Quang Lâm  
    Họ: Nguyễn  
    Đệm: Xuân Quang  
    Tên: Lâm**

def phan\_tach\_ho\_ten():

ho\_ten\_day\_du = input("Nhập họ tên đầy đủ: ")

cac\_tu = ho\_ten\_day\_du.split()

ho = cac\_tu[0]

ten = cac\_tu[-1]

dem = ' '.join(cac\_tu[1:-1])

print("Họ:", ho)

print("Đệm:", dem)

print("Tên:", ten)

phan\_tach\_ho\_ten()

1. **Viết chương trình (hàm) nhập n số nguyên từ bàn phim, các số cách nhau bởi dấu cách, sau đó in ra màn hình:**

* **Danh sách số đã nhập trên một hàng ngang.**
* **Số lớn nhất và chỉ số của số lớn nhất.**
* **Số nhỏ nhất và chỉ số của số nhỏ nhất.**

def thong\_tin\_so():

try:

danh\_sach = list(map(int, input("Nhập các số nguyên cách nhau bởi dấu cách: ").split()))

print("Danh sách số đã nhập:", danh\_sach)

max\_value = max(danh\_sach)

min\_value = min(danh\_sach)

print("Số lớn nhất là", max\_value, "ở vị trí", danh\_sach.index(max\_value))

print("Số nhỏ nhất là", min\_value, "ở vị trí", danh\_sach.index(min\_value))

except ValueError:

print("Vui lòng chỉ nhập số nguyên")

thong\_tin\_so()

1. **Cho trước xâu kí tự S và xâu substr. Viết chương trình (hàm) tính số lần lặp của xâu con substr trong xâu S, cho phép chồng lấn của các xâu con này.   
   Ví dụ nếu S = “12121332121432121”, substr = “121” thì số lần lặp là 4**

def dem\_so\_lan\_lap(xau\_cha, xau\_con):

count = start = 0

while True:

start = xau\_cha.find(xau\_con, start) + 1

if start > 0:

count += 1

else:

break

return count

xau = "12121332121432121"

substr = "121"

so\_lan\_lap = dem\_so\_lan\_lap(xau, substr)

print("Số lần lặp của xâu con trong xâu là:", so\_lan\_lap)

## PYTHON FLOW CONTROL.

1. **Flow control là gì trong Python? cho ví dụ.**

* Flow control trong Python là cách kiểm soát luồng thực hiện của chương trình dựa trên các điều kiện và quyết định. Nó cho phép chương trình thực hiện các khối mã khác nhau tùy thuộc vào giá trị của các biểu thức hoặc điều kiện.
* Có ba cấu trúc chính cho flow control trong Python: câu lệnh rẽ nhánh (conditional statements), vòng lặp (loops), và câu lệnh điều khiển (control statements).
* Ví dụ: câu lệnh rẽ nhánh (Conditional statements):

age = 18

if age >= 18:

print("Bạn đã đủ tuổi để lái xe.")

else:

print("Bạn chưa đủ tuổi để lái xe.")

1. **Python có bao nhiêu loại câu lệnh điều khiển cơ bản? cho ví dụ.**

* Python có ba loại câu lệnh điều khiển cơ bản: if, for, và while.
* Ví dụ:
* If:

x = 10

if x > 0:

print("x is positive")

elif x == 0:

print("x is zero")

else:

print("x is negative")

* For:

fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

print(fruit)

* While:

count = 0

while count < 5:

print(count)

count += 1

1. **Câu lệnh "if" được sử dụng để làm gì? cho ví dụ.**

- Câu lệnh if trong Python được sử dụng để kiểm tra một điều kiện và thực hiện một khối mã chỉ khi điều kiện đó là True. Nếu điều kiện là False, khối mã bên trong câu lệnh if sẽ được bỏ qua và chương trình sẽ tiếp tục thực hiện các dòng mã sau câu lệnh if.

- Ví dụ:

# Kiểm tra điều kiện và in ra kết quả tương ứng

x = 10

if x > 0:

print("x là số dương.")

1. **Câu lệnh "for" dùng để làm gì? cho ví dụ**

- Sử dụng để lặp qua một dãy giá trị hoặc các phần tử trong một chuỗi, danh sách, tuple, hoặc đối tượng lặp khác.

- Ví dụ:

fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

print(fruit)

1. **Câu lệnh "while" thực hiện việc gì? cho ví dụ.**

- Sử dụng để thực hiện một khối mã chỉ khi điều kiện là True, và nó sẽ tiếp tục thực hiện cho đến khi điều kiện trở thành False.

- Ví dụ:

count = 0

while count < 5:

print(cout)

count += 1

1. **Câu lệnh "elif" có ý nghĩa gì? cho ví dụ.**

- Câu lệnh elif là viết tắt của "else if" và được sử dụng trong cấu trúc điều khiển để kiểm tra điều kiện tiếp theo nếu điều kiện trong câu lệnh if là False. Nếu điều kiện trong câu lệnh if là True, câu lệnh elif sẽ được bỏ qua; ngược lại, nếu điều kiện trong câu lệnh if là False, điều kiện trong câu lệnh elif sẽ được kiểm tra.

- Ví dụ:

# Kiểm tra nhiều điều kiện sử dụng câu lệnh if và elif

x = 0

if x > 0:

print("x là số dương.")

elif x == 0:

print("x bằng 0.")

else:

print("x là số âm.")

1. Làm cách nào để ngắt một vòng lặp sớm hơn trong Python? cho ví dụ

* Sử dụng câu lệnh ‘break’
* VD:

# Sử dụng break để ngắt vòng lặp khi gặp giá trị nhất định

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for number in numbers:

if number == 5:

print("Đã tìm thấy giá trị 5. Ngắt vòng lặp.")

break

print(number)

1. **Câu lệnh "continue" trong Python có ý nghĩa gì? cho ví dụ**

- Câu lệnh continue trong Python được sử dụng để bỏ qua phần còn lại của một vòng lặp và tiếp tục với lần lặp tiếp theo. Khi câu lệnh continue được thực hiện, các câu lệnh bên dưới nó trong cùng một vòng lặp sẽ bị bỏ qua, và vòng lặp sẽ chuyển đến lần lặp tiếp theo.

- Ví dụ:

# Sử dụng continue để bỏ qua các số chẵn

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for number in numbers:

if number % 2 == 0:

# Bỏ qua các số chẵn và tiếp tục vòng lặp

continue

print(number)

1. **Python có hỗ trợ vòng lặp lồng nhau không? cho ví dụ**

- Có

- Ví dụ:

# Ví dụ về vòng lặp lồng nhau

for i in range(3): # Vòng lặp bên ngoài

for j in range(3): # Vòng lặp bên trong

print(f"i: {i}, j: {j}")

1. **Câu lệnh "else" trong câu lệnh "for" được thực hiện khi nào? cho ví dụ**

- Câu lệnh else trong câu lệnh for được thực hiện khi vòng lặp kết thúc mà không bị ngắt bởi câu lệnh break. Nó sẽ chạy sau khi vòng lặp hoàn thành việc lặp qua tất cả các phần tử trong chuỗi hoặc đối tượng mà vòng lặp đang thực hiện.

- Ví dụ:

fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

print(fruit)

else:

print("Không còn phần tử nào trong danh sách.")

1. **Làm thế nào để xác định số lần lặp trong vòng lặp "for"? cho ví dụ**

- Trong Python, bạn có thể sử dụng hàm range() để xác định số lần lặp trong vòng lặp for. Hàm range() cho phép bạn tạo một dãy số từ một giá trị đầu vào cho đến một giá trị kết thúc với một bước nhảy cố định.

- Ví dụ:

# Xác định số lần lặp trong vòng lặp for bằng hàm range()

for i in range(5):

print(f"Lần lặp thứ {i+1}")

1. **Làm thế nào để xác định số lần lặp trong vòng lặp "while"? cho ví dụ.**

- Trong vòng lặp while, số lần lặp được xác định dựa trên một điều kiện. Vòng lặp sẽ tiếp tục lặp cho đến khi điều kiện trở thành False.

- Ví dụ:

# Xác định số lần lặp trong vòng lặp while

count = 0

while count < 5:

print(f"Lần lặp thứ {count+1}")

count += 1

1. **Làm cách nào để xác định một phạm vi biến trong Python? cho ví dụ.**

- Phạm vi của một biến trong Python được xác định bởi nơi mà biến được khai báo và gán giá trị. Phạm vi có thể là toàn cục (global) hoặc cục bộ (local). Biến được khai báo ở ngoài bất kỳ hàm hay khối lệnh nào được coi là biến toàn cục, trong khi biến được khai báo trong một hàm hoặc khối lệnh cụ thể được coi là biến cục bộ.

- Ví dụ:

# Biến toàn cục

global\_variable = "Toàn cục"

def example\_function():

# Biến cục bộ

local\_variable = "Cục bộ"

print("Trong hàm:", local\_variable)

# Gọi hàm

example\_function()

# In giá trị của biến toàn cục

print("Ngoài hàm:", global\_variable)

1. **Câu lệnh "pass" trong Python làm gì? cho ví dụ**

- Trong Python, câu lệnh pass được sử dụng để tạo một khối mã không thực hiện bất kỳ hành động nào. Nó là một no-op (no operation), hay còn gọi là câu lệnh trống. Câu lệnh pass thường được sử dụng khi một khối mã là bắt buộc cú pháp, nhưng không cần thực hiện bất kỳ công việc gì. Câu lệnh pass giúp tránh lỗi cú pháp.

- Ví dụ:

# Câu lệnh pass trong một câu lệnh if

x = 10

if x > 5:

pass

# Không thực hiện bất kỳ hành động nào nếu x > 5

1. **Làm cách nào để xử lý ngoại lệ trong Python? cho ví dụ.**

- Trong Python, xử lý ngoại lệ được thực hiện bằng cách sử dụng các khối try, except. Mã trong khối try được thực hiện, và nếu có bất kỳ ngoại lệ nào xảy ra, mã trong khối except tương ứng sẽ được thực hiện để xử lý ngoại lệ đó. Điều này giúp chương trình tránh việc bị dừng đột ngột và cung cấp khả năng kiểm soát lỗi.

- Ví dụ:

def division\_example(x, y):

try:

result = x / y

print(f"Kết quả của {x} / {y} là {result}")

except ZeroDivisionError:

print("Không thể chia cho 0. Hãy nhập một số khác.")

except TypeError:

print("Có lỗi kiểu dữ liệu. Hãy nhập số nguyên hoặc số thực.")

except Exception as e:

print(f"Có lỗi không mong muốn: {e}")

else:

print("Khối mã trong try được thực hiện mà không có ngoại lệ.")

finally:

print("Khối mã trong finally sẽ luôn được thực hiện.")

# Thử nghiệm hàm với các giá trị khác nhau

division\_example(10, 2) # Không có ngoại lệ

division\_example(10, 0) # Ngoại lệ ZeroDivisionError

division\_example("abc", 2) # Ngoại lệ TypeError

1. **Python có hỗ trợ vòng lặp vô hạn không? cho ví dụ.**

- Trong Python, bạn có thể tạo vòng lặp vô hạn bằng cách sử dụng một điều kiện luôn trả về True. Một cách thông thường để thực hiện điều này là sử dụng từ khóa while True

- Ví dụ:

# Vòng lặp vô hạn sử dụng while True

while True:

user\_input = input("Nhập một số nguyên (nhập 'exit' để thoát): ")

if user\_input.lower() == 'exit':

print("Thoát khỏi vòng lặp vô hạn.")

break

try:

number = int(user\_input)

print(f"Bạn đã nhập số nguyên: {number}")

except ValueError:

print("Lỗi: Đây không phải là số nguyên.")

1. **Làm thế nào để lặp qua một dãy số từ 1 đến 10? cho ví dụ.**

- Để lặp qua một dãy số từ 1 đến 10 trong Python, bạn có thể sử dụng vòng lặp for kết hợp với hàm range().

- Ví dụ:

# Lặp qua dãy số từ 1 đến 10 sử dụng vòng lặp for và range

for i in range(1, 11):

print(i)

1. **Câu lệnh "else" có thể được sử dụng trong vòng lặp "while" không? cho ví dụ.**

- Trong Python, câu lệnh else có thể được sử dụng trong vòng lặp while. Khối mã bên trong else sẽ được thực hiện khi điều kiện của vòng lặp trở thành False. Điều này có thể hữu ích khi bạn muốn thực hiện một số công việc khi vòng lặp kết thúc mà không phải do break.

- Ví dụ:

# Ví dụ sử dụng else trong vòng lặp while

count = 0

while count < 5:

print(f"Lần lặp thứ {count+1}")

count += 1

else:

print("Vòng lặp đã kết thúc do điều kiện trở thành False.")

1. Làm thế nào để ngăn vòng lặp vô hạn? cho ví dụ.

- Để ngăn chặn vòng lặp vô hạn, bạn có thể sử dụng một điều kiện dừng bên trong vòng lặp. Thông thường, điều kiện này sẽ dựa trên một số điều kiện logic hoặc sự tương tác của người dùng.

- Ví dụ:

# Ngăn chặn vòng lặp vô hạn bằng cách sử dụng điều kiện dừng

while True:

user\_input = input("Nhập 'exit' để thoát: ")

if user\_input.lower() == 'exit':

print("Thoát khỏi vòng lặp vô hạn.")

break

else:

print("Vòng lặp vô hạn đang tiếp tục.")

1. **Có bao nhiêu cách để thực hiện kiểm tra điều kiện nhiều lựa chọn trong Python? cho ví dụ.**

- Trong Python, có một số cách để thực hiện kiểm tra điều kiện nhiều lựa chọn. Ba cách phổ biến là sử dụng câu lệnh if-elif-else, sử dụng từ khóa in, và sử dụng câu lệnh switch-case thông qua mô phỏng bằng từ điển.

- Ví dụ:

# Sử dụng câu lệnh if-elif-else

score = 75

if score >= 90:

grade = 'A'

elif score >= 80:

grade = 'B'

elif score >= 70:

grade = 'C'

elif score >= 60:

grade = 'D'

else:

grade = 'F'

print(f"Điểm {score} tương ứng với hạng {grade}")

## PYTHON FUCTION.

1. **Làm thế nào để định nghĩa một hàm trong Python, cho ví dụ**

* Để định nghĩa một hàm trong Python, bạn sử dụng từ khóa def sau đó là tên của hàm, danh sách tham số và dấu hai chấm. Sau đó, bạn thực hiện các công việc cần thiết bên trong hàm.
* VD:

def greet(name):

print(f"Hello, {name}!")

1. **Làm thế nào để gọi một hàm trong Python, cho ví dụ**

* Để gọi một hàm, bạn chỉ cần sử dụng tên của hàm cùng với các đối số cần thiết (nếu có).
* VD: hàm greet trong python  
  greet("Alice")

1. **Hàm có thể trả về giá trị trong Python không, cho ví dụ**

* Có, hàm có thể trả về giá trị bằng từ khóa return.
* VD:

def add(x, y):

return x + y

1. **Làm thế nào để truyền đối số cho một hàm, cho ví dụ**

* Đối số được truyền bằng cách đặt chúng trong dấu ngoặc đơn sau tên hàm.
* VD:

def multiply(a, b):

return a \* b

result = multiply(3, 4)

1. **Làm thế nào để định nghĩa tham số mặc định trong hàm, cho ví dụ**

* Tham số mặc định được định nghĩa bằng cách gán giá trị mặc định cho tham số đó.
* VD:

def power(base, exponent=2):

return base \*\* exponent

1. **Làm thế nào để truy cập biến toàn cục trong hàm, cho ví dụ**

* Sử dụng từ khóa global để truy cập biến toàn cục.
* VD:

global\_var = 10

def access\_global():

global global\_var

print(global\_var)

access\_global()

1. **Hàm lambda là gì, cho ví dụ**

* Hàm lambda là một dạng của hàm vô danh trong ngôn ngữ lập trình Python. Nó là một cách ngắn gọn để định nghĩa hàm mà không cần phải sử dụng từ khóa ‘def’.
* Cú pháp:

lambda arguments: expression

* VD:

multiply = lambda x, y: x \* y

1. **Làm thế nào để gọi hàm lambda, cho ví dụ:**

* Gọi lại hàm lamda ở ví dụ ở câu g với các giá trị cụ thể cho các tham số:

result = multiply(3, 4)

# In kết quả

print(result) # Output: 12

1. **Làm thế nào để xác định số lượng đối số biến đổi (variable-length arguments) trong hàm, cho ví dụ**

* Sử dụng \*args để xác định số lượng đối số biến đổi.
* VD:

def variable\_args(\*args):

for arg in args:

print(arg)

variable\_args(1, 2, 3)

1. **Làm thế nào để xác định số lượng đối số keyword trong hàm, cho ví dụ**

* Sử dụng \*\*kwargs để xác định số lượng đối số keyword.
* VD:

def keyword\_args(\*\*kwargs):

for key, value in kwargs.items():

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

keyword\_args(name="Alice", age=25)

1. **Có thể gọi chính nó không, cho ví dụ?**

* Có, hàm có thể gọi chính nó, được gọi là đệ quy.
* VD:

def factorial(n):

if n == 0 or n == 1:

return 1

else:

return n \* factorial(n-1)

1. **Làm thế nào để khai báo hàm biến thể (nested function) trong Python, cho ví dụ**

* Bạn có thể định nghĩa một hàm bên trong một hàm khác.
* VD:

def outer\_function(x):

def inner\_function(y):

return x + y

return inner\_function

1. **Làm thế nào để trả về nhiều giá trị từ hàm, cho ví dụ**

* Hàm có thể trả về nhiều giá trị dưới dạng tuple.
* VD:

def get\_coordinates():

return 3, 4

x, y = get\_coordinates()

1. **Làm thế nào để xác định một biến là hàm, cho ví dụ**

* Sử dụng hàm callable() để kiểm tra xem một biến có phải là hàm hay không.
* VD:

def my\_function():

print("Hello!")

is\_function = callable(my\_function)

1. **Làm thế nào để gửi một hàm như một đối số cho một hàm khác, cho ví dụ**

* Hàm có thể được truyền như đối số cho một hàm khác.
* VD:

def apply\_operation(x, y, operation):

return operation(x, y)

result = apply\_operation(3, 4, lambda a, b: a \* b)

1. **Làm thế nào để xác định tên của một hàm, cho ví dụ**

* Sử dụng thuộc tính \_\_name\_\_ để lấy tên của một hàm.
* VD:

def my\_function():

print("Hello!")

function\_name = my\_function.\_\_name\_\_

1. **Làm thế nào để xác định tài liệu của một hàm, cho ví dụ**

* Sử dụng hàm help() để xem tài liệu của một hàm.
* VD:

def my\_function():

"""This is the documentation of my\_function."""

print("Hello!")

help(my\_function)

1. **Anh/ chị hãy cho biết làm thế nào để bỏ qua lỗi trong hàm, cho ví dụ**

* Sử dụng try và except để bắt lỗi và thực hiện các hành động khác nếu có lỗi.
* VD:

def divide(x, y):

try:

result = x / y

except ZeroDivisionError:

result = "Cannot divide by zero"

return result

1. **Làm thế nào để đóng gói nhiều hàm thành một module, cho ví dụ**

* Tạo một tập tin mới với phần mở rộng .py và đặt các hàm vào đó, sau đó sử dụng import để nhập chúng vào chương trình chính.
* VD:
* module\_example.py

def add(x, y):

return x + y

def subtract(x, y):

return x - y

* main\_program.py

import module\_example

result\_add = module\_example.add(3, 4)

result\_subtract = module\_example.subtract(7, 2)

1. **Làm thế nào để nhập một module vào một chương trình Python, cho ví dụ**

* Sử dụng import để nhập một module vào chương trình.
* VD:

import math

result = math.sqrt(16)

## 

## PYTHON FILES.

1. **Viết một chương trình Python để đọc nội dung của một file văn bản (ví dụ: "CS466.txt") và in nội dung ra màn hình.**

file\_path = "CS466.txt"

with open(file\_path, "r") as file:

content = file.read()

print(content)

1. **Viết một chương trình Python để đọc file văn bản và đếm số dòng trong file.**

file\_path = "CS466.txt"

with open(file\_path, "r") as file:

line\_count = sum(1 for line in file)

print(f"Number of lines in the file: {line\_count}")

1. **Viết một chương trình Python để tạo một file mới và ghi nội dung vào file đó.**

file\_path = "new\_file.txt"

content\_to\_write = "Hello, this is the content of the new file."

with open(file\_path, "w") as file:

file.write(content\_to\_write)

1. **Anh/ chị hãy viết một chương trình Python để đọc file văn bản và in ra tất cả các từ (tách bằng khoảng trắng) trong file.**

file\_path = "CS466.txt"

with open(file\_path, "r") as file:

words = file.read().split()

for word in words:

print(word)

1. **Viết một chương trình Python để sao chép nội dung từ một file gốc ("input.txt") sang một file đích ("output.txt")**

source\_path = "input.txt"

destination\_path = "output.txt"

with open(source\_path, "r") as source\_file, open(destination\_path, "w") as destination\_file:

content = source\_file.read()

destination\_file.write(content)

1. **Viết một chương trình Python để đọc file văn bản, tìm một từ cụ thể và thay thế nó bằng một từ khác. Sau đó, ghi nội dung mới vào file.**

file\_path = "CS466.txt"

search\_word = "Python"

replace\_word = "Java"

with open(file\_path, "r") as file:

content = file.read()

updated\_content = content.replace(search\_word, replace\_word)

with open(file\_path, "w") as file:

file.write(updated\_content)



## KẾT LUẬN CHƯƠNG 1

Python là một ngôn ngữ lập trình mạnh mẽ, linh hoạt, và có nhiều ứng dụng. Các khái niệm cơ bản như kiểu dữ liệu, kiểm soát luồng, hàm, và xử lý file là quan trọng để hiểu khi bắt đầu lập trình bằng Python. Những khái niệm này tạo nên cơ bản cho việc phát triển ứng dụng Python hiệu quả và chuyên nghiệp.

# CHƯƠNG 2: TÌM HIỂU VÀ LẬP TRÌNH WEB BÁN QUẦN ÁO PYSHOP BẰNG NGÔN NGỮ PYTHON

Chương này sẽ làm rõ ý tưởng, mô tả hoạt động của dự án web bán quần áo trực tuyến PyShop được viết bằng ngôn ngữ Python.

Nội dung tìm hiểu:

2.1. Thiết lập dự án

2.2. Django - Backend

2.3. Vue - Frontend

2.4. SQLite - Database

## THIẾT LẬP DỰ ÁN PYSHOP - DJANGO & VUE WEB SERVICE

Trong phần này, chúng ta sẽ tìm hiểu về quá trình thiết lập dự án PyShop. Điều này bao gồm việc xác định ý tưởng cụ thể của dự án, mô tả hoạt động của nó, và quyết định các công nghệ chính được sử dụng.

### Ý Tưởng Dự Án

Trang web PyShop được xây dựng với mục đích giúp khách hàng dễ dàng lựa chọn sản phẩm quần áo phù hợp với mọi tầm giá. Dự án này cung cấp trải nghiệm mua sắm trực tuyến thuận lợi, giúp khách hàng tiếp cận những sản phẩm chất lượng một cách dễ dàng.

### Mục Tiêu

* Xây dựng trang web bán quần áo trực tuyến với giao diện thân thiện và dễ sử dụng.
* Tích hợp các tính năng tìm kiếm và lọc sản phẩm để người dùng có thể dễ dàng tìm kiếm sản phẩm mong muốn.
* Hỗ trợ đa ngôn ngữ và đa tầm giá để mở rộng tới đối tượng người dùng đa dạng.

## DJANGO - BACKEND

Django được chọn làm framework backend cho PyShop với những ưu điểm về hiệu suất, tính bảo mật cao, và khả năng mở rộng dễ dàng. Trong phần này, chúng ta sẽ tìm hiểu về cách Django được sử dụng để xây dựng logic của trang web.

### Cấu Trúc Dự Án Django

Dự án Django được tổ chức thành các ứng dụng nhỏ, mỗi ứng dụng đảm nhận một phần cụ thể của logic và chức năng. Điều này giúp quản lý mã nguồn một cách hiệu quả và dễ dàng mở rộng.

Các bước thiết lập:

* Thiết lập environment:  
  + tạo environment: **virtualenv environment\_3\_8\_2**

+ active environment: **environment\_3\_8\_2\Scripts\activate**

* Cài django và các dependencies (các thư viện phụ thuộc) vào dự án:

install django in project :

**pip install django**

install django dependencies :

**pip install django-rest-framework**

**pip install django-cors-headers** (xử lý bảo mật api)

**pip install djoser** (thư viện xử lý authen)

**pip install pillow** (thư viện xử lý hình ảnh)

**pip install stripe** (thư viện xử lý thanh toán trực tuyến)

### Xử Lý Dữ Liệu:

Django sử dụng ORM (Object-Relational Mapping) để tương tác với cơ sở dữ liệu SQLite. Quản lý người dùng, sản phẩm, đơn đặt hàng và các tính năng khác được triển khai thông qua mô hình Django.

* Thiết lập database:

**python manage.py makemigrations**

**python manage.py migrate**

đăng ký superuser (admin):

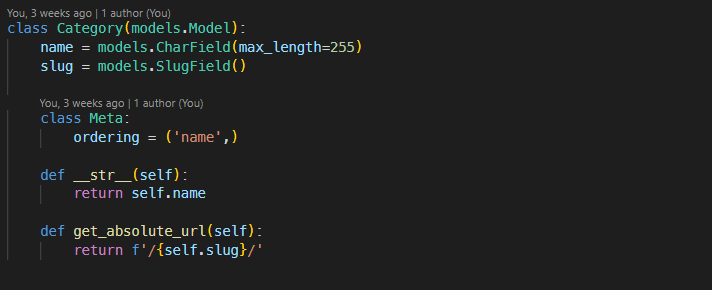
**python manage.py createsuperuser**

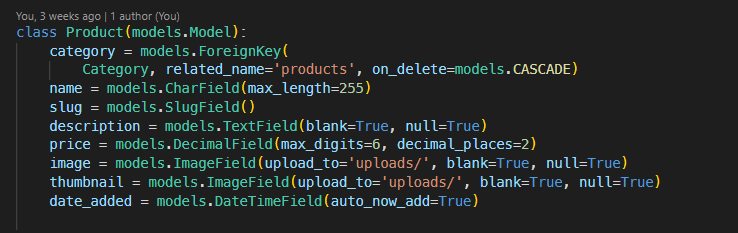
Run server với tài khoản admin (hoặc xem trực tiếp trên sqlite) :

**python manage.py runserver**

* Tạo các model để mapping và tạo các đối tượng trên tầng database:
* Tạo django app và models for product:

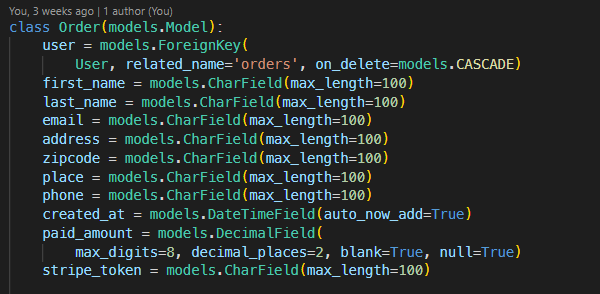
**python manage.py startapp product**

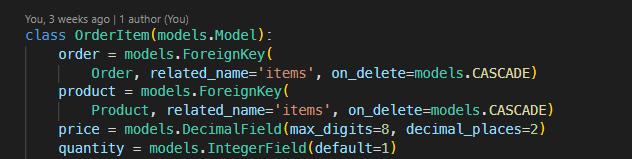
****

****

* Tạo django app và models for order:

**python manage.py startapp order**

****

****

* Sau khi tạo xong các model cho product và order chạy lại migrate để chạy lại database:

**python manage.py makemigrations**

**python manage.py migrate**

## VUE - FRONTEND

Phần frontend của PyShop được xây dựng bằng Vue.js để tạo ra giao diện người dùng đẹp mắt và tương tác mượt mà.

### Tích Hợp Vue.js:

Vue.js giúp tạo ra các thành phần giao diện linh hoạt và dễ quản lý. Sự tương tác giữa backend và frontend được thực hiện thông qua API, giúp trang web hoạt động mượt mà và nhanh chóng.

### Tối Ưu Hóa Hiệu Suất:

Vue.js cung cấp các công cụ tối ưu hóa hiệu suất, giúp trang web PyShop load nhanh và đáp ứng tốt trên nhiều thiết bị.

### Các bước thiết lập dự án Vue:

* Các các cli của vue và tạo dự án vue:

**npm install -g @vue/cli**

**vue create vue-restful**

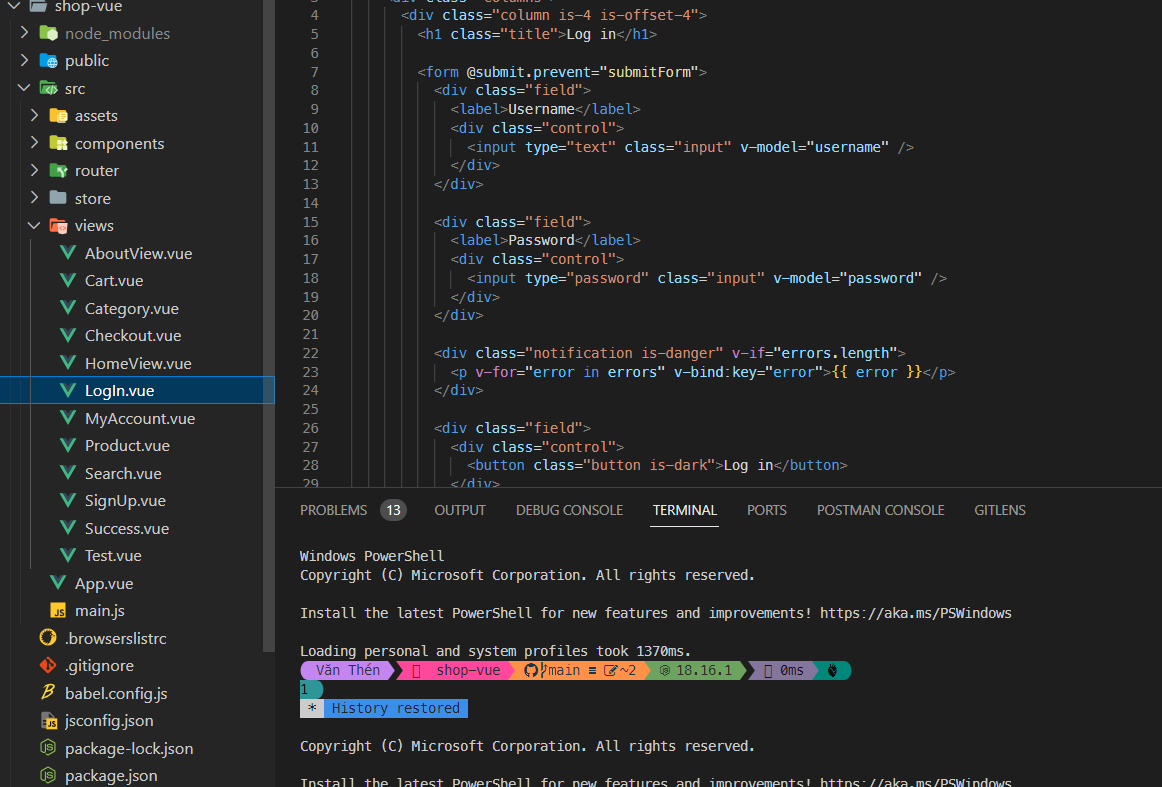
* Cài các dependencies (thư viện hỗ trợ fetch api và css):

**npm install axios**

**npm install bulma**

* Chạy dự án:

**npm run serve**

****

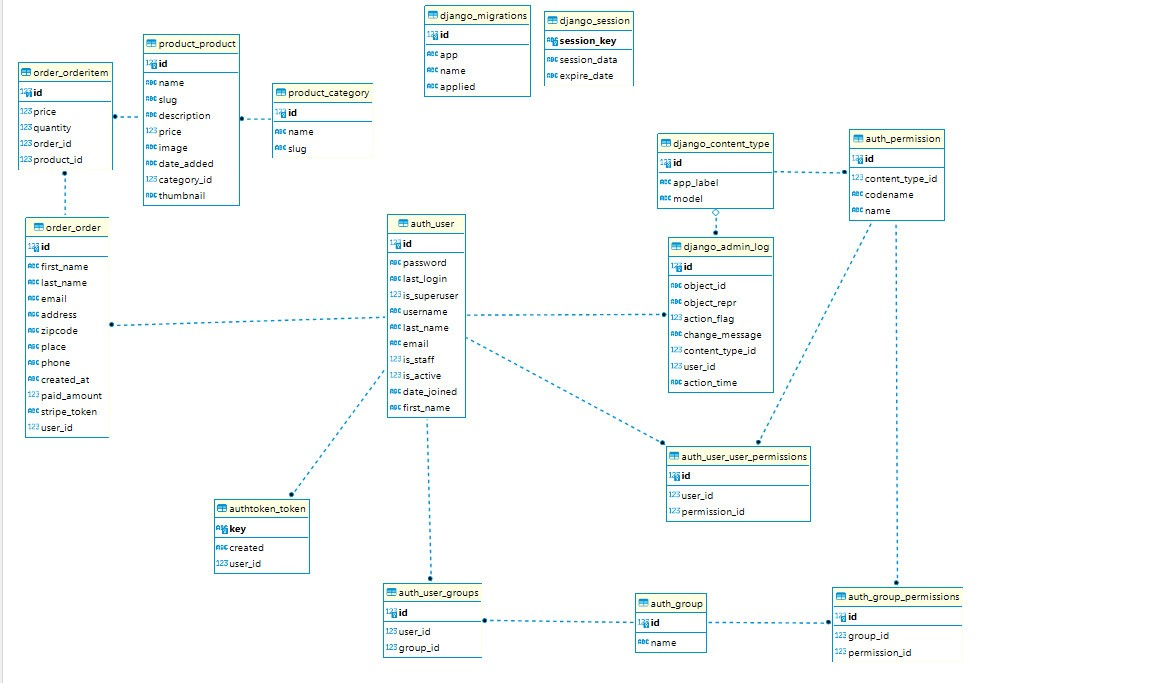
## SQLITE - DATABASE

Cơ sở dữ liệu SQLite được chọn cho PyShop do đặc tính nhẹ, linh hoạt và phù hợp cho các dự án nhỏ và trung bình. Trong phần này, chúng ta sẽ tìm hiểu về cách SQLite được tích hợp vào dự án PyShop để lưu trữ dữ liệu.

### Schema Database:

Cấu trúc của cơ sở dữ liệu được thiết kế để phản ánh cấu trúc logic của trang web PyShop, bao gồm bảng người dùng, sản phẩm, đơn đặt hàng, và các chi tiết khác.

### ER Diagram:



# 

## KẾT LUẬN CHƯƠNG 2

Trong chương này, chúng ta đã tìm hiểu về quá trình thiết lập dự án PyShop, sử dụng Django cho backend, Vue.js cho frontend, và SQLite cho cơ sở dữ liệu. Đồng thời, chúng ta đã đánh giá ưu và nhược điểm của cách triển khai này. Chương tiếp theo sẽ đi sâu vào các thành phần chi tiết của dự án

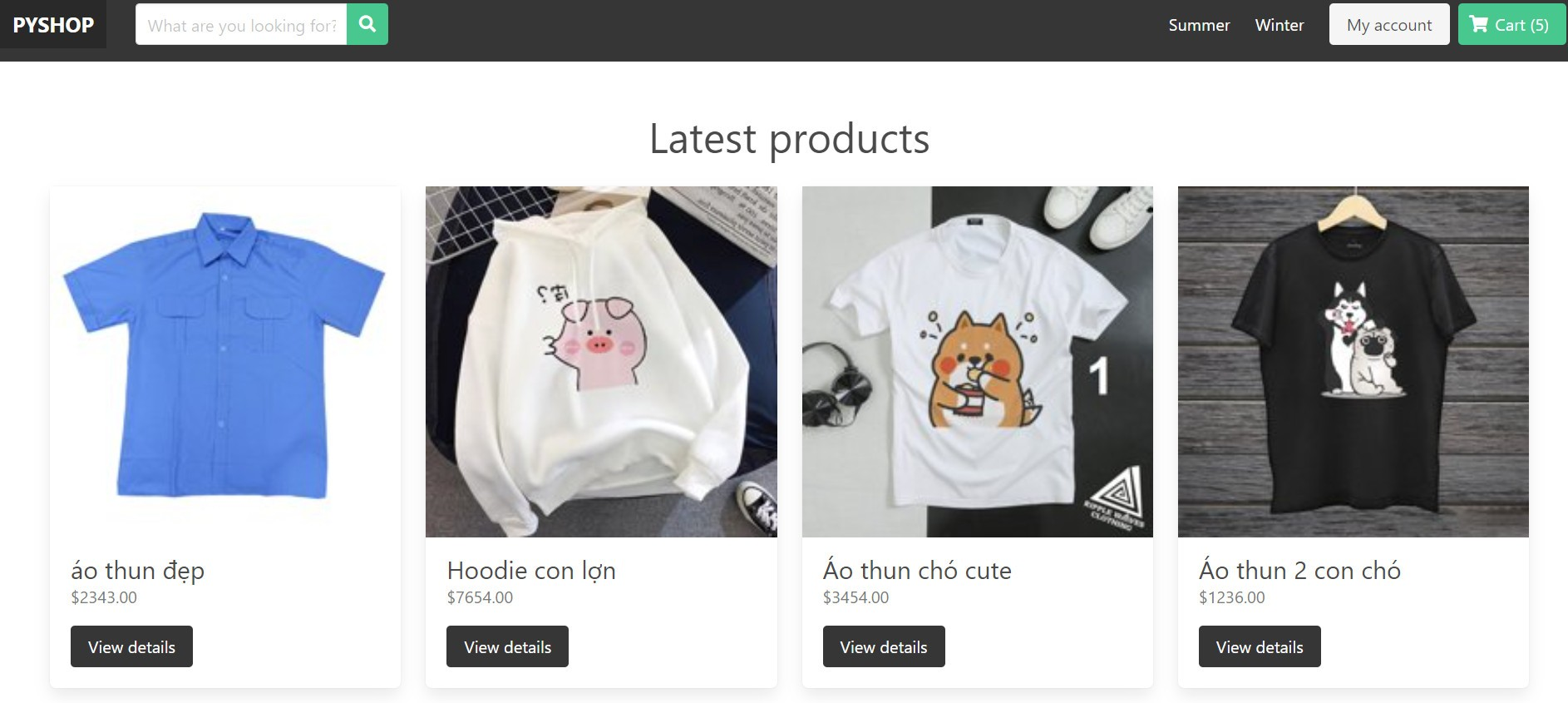
# CHƯƠNG 3: TRIỂN KHAI CÀI ĐẶT VÀ THỰC HIỆN DỰ ÁN

Chương này, sẽ tiến hành triển khai thực hiện các chức năng cho dự án pyshop. Các yêu cầu cụ thể từng chức năng sẽ được đề cập dưới đây.

## CÁC CHỨC NĂNG

### Chức năng hiển thị sản phẩm

* Các sản phẩm sẽ hiển thị ngay tại trang chủ chính của web để dễ dàng tiếp cận khách hàng truy cập vào web.



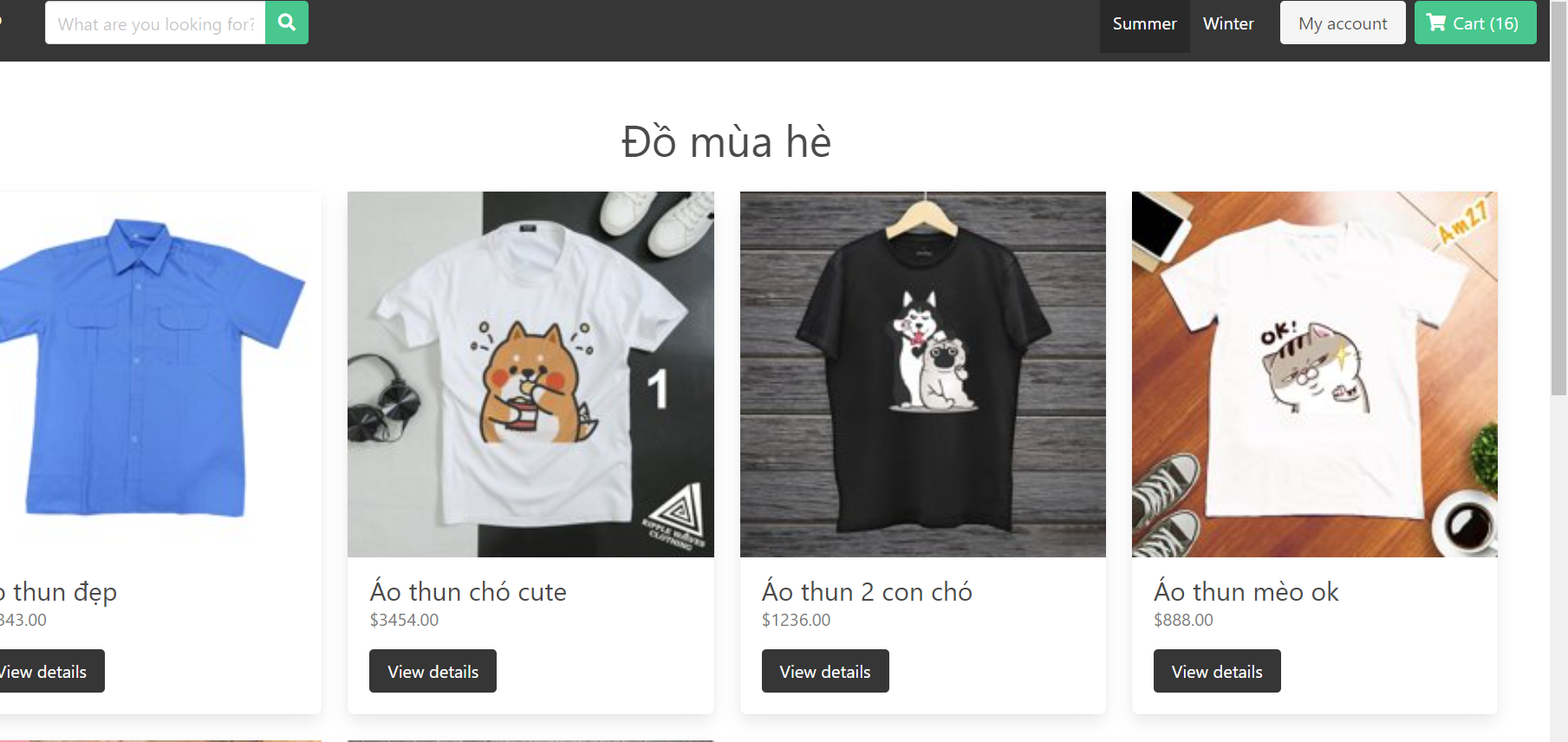
### Chức năng xem chi tiết sản phẩm

* Khi thấy một sản phẩm ưng ý và muốn mua thì có thể bấm vào “View details” để xem chi tiết sản phẩm như thế nào và có quyết định thêm vào giỏ hàng không?

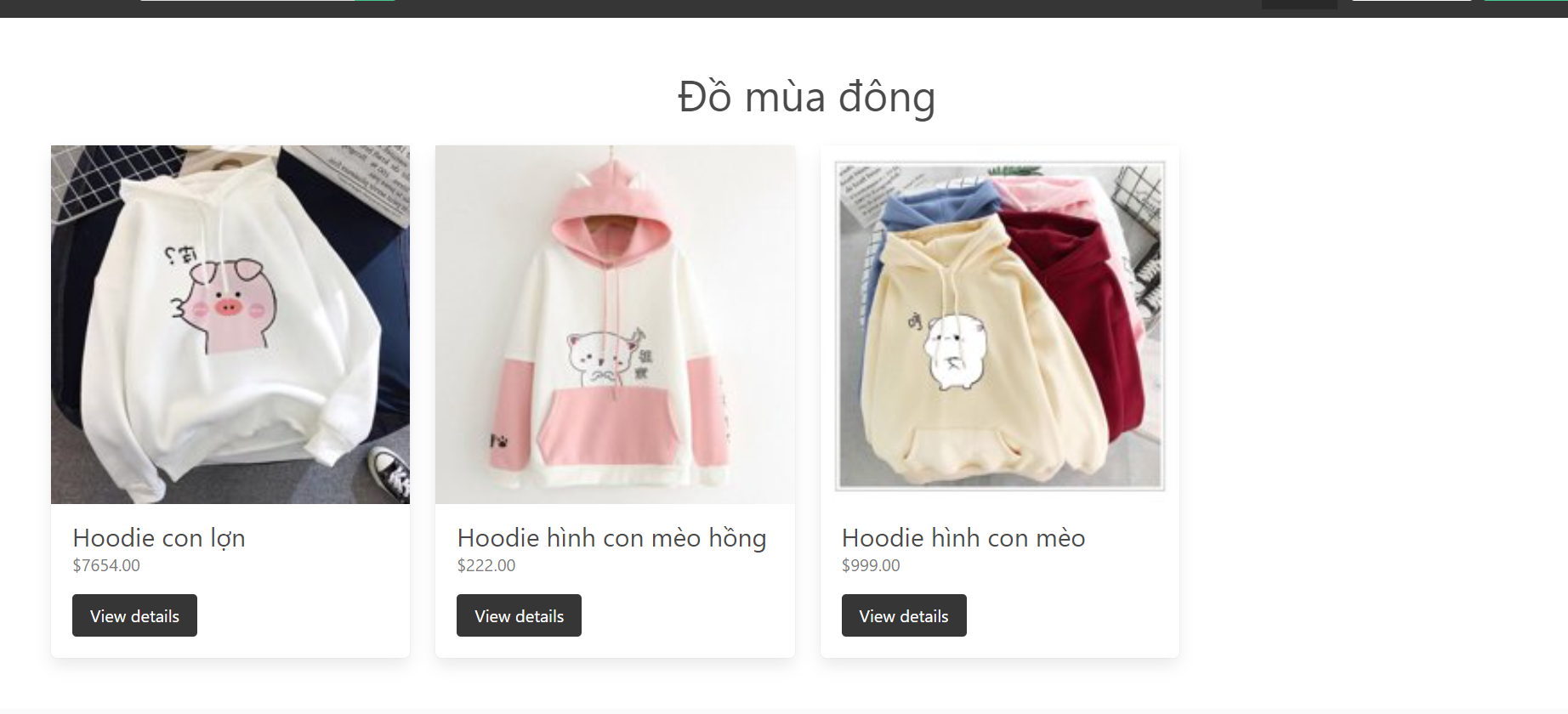


### Chức năng phân loại

* Có thể bấm vào 2 nút chức năng lọc sản phẩm theo 2 mùa là mùa hè - Summer và mùa đông - Winter.
* **Đồ mùa hè:**

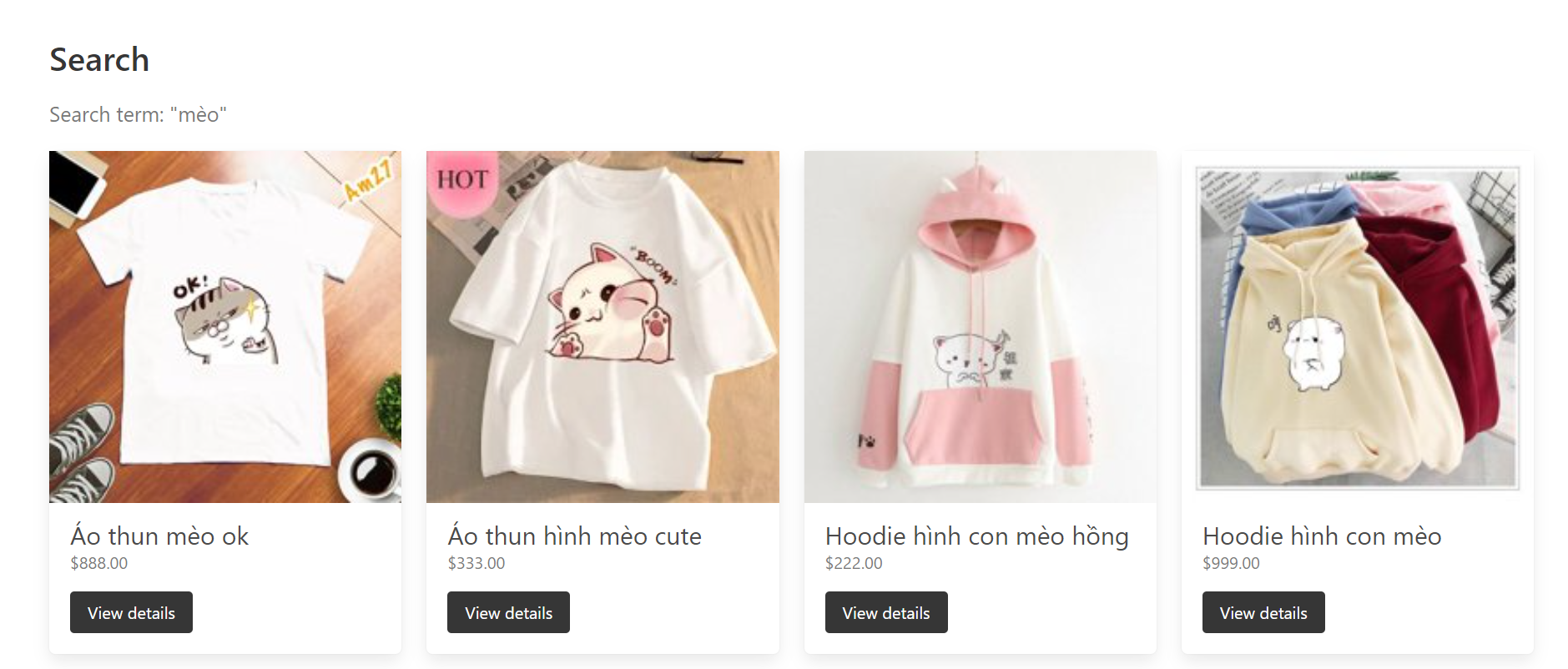
****

* **Đồ mùa đông:**

****

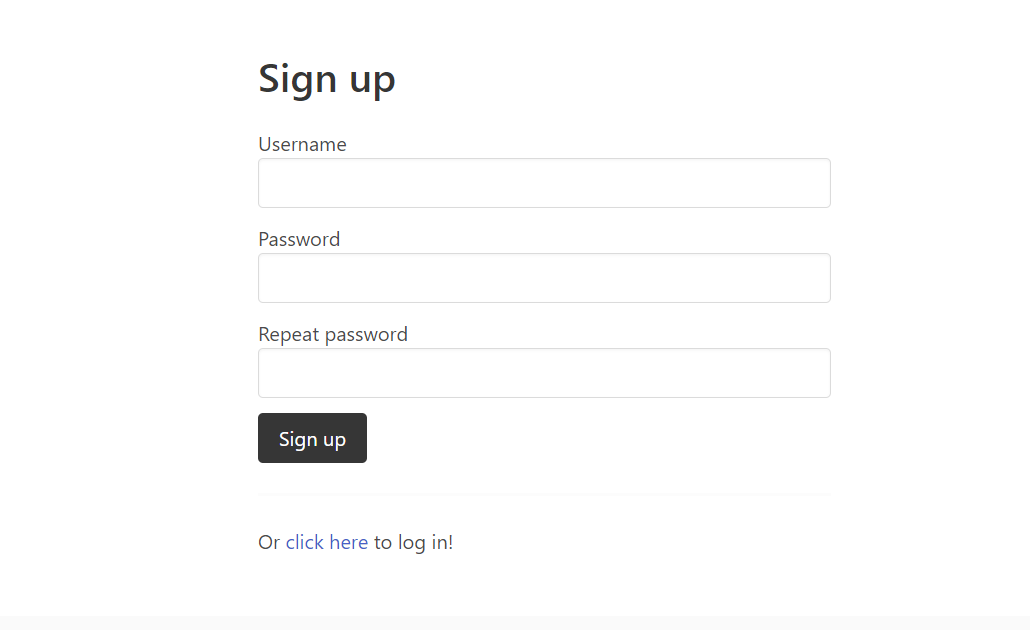
### Chức năng tìm kiếm

* Bạn có thể tìm kiếm tên theo tên sản phẩm khi gõ từ khóa sản phẩm cần tìm vào thanh tìm kiếm.
* Ví dụ các bạn muốn tìm kiểm sản phẩm có “Mèo”:

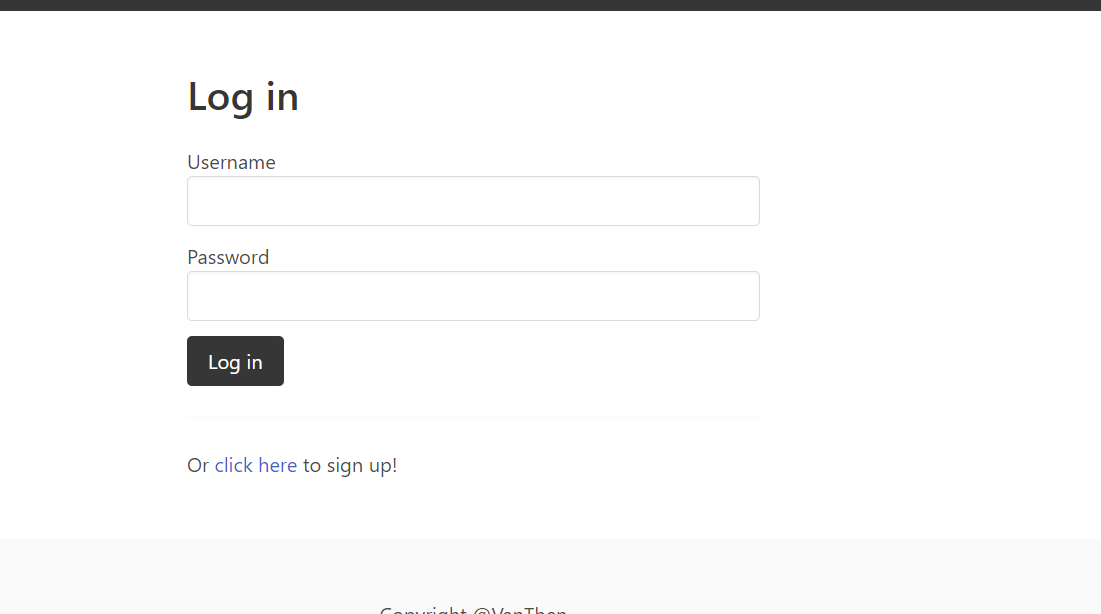


### Chức năng đăng kí - đăng nhập

* Bạn có thể Đăng kí nếu chưa có tài khoản và Đăng nhập nếu đã đăng kí thành công tài khoản. Việc có tài khoản có thể giúp bạn lưu lại các sản đã bỏ vào giỏ hàng cũng như mua hàng trực tuyến tại trang web.
* **Chức năng đăng kí**

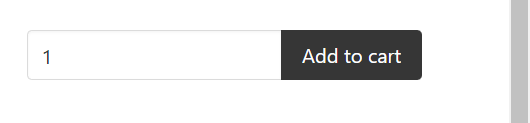


* **Chức năng đăng nhập**

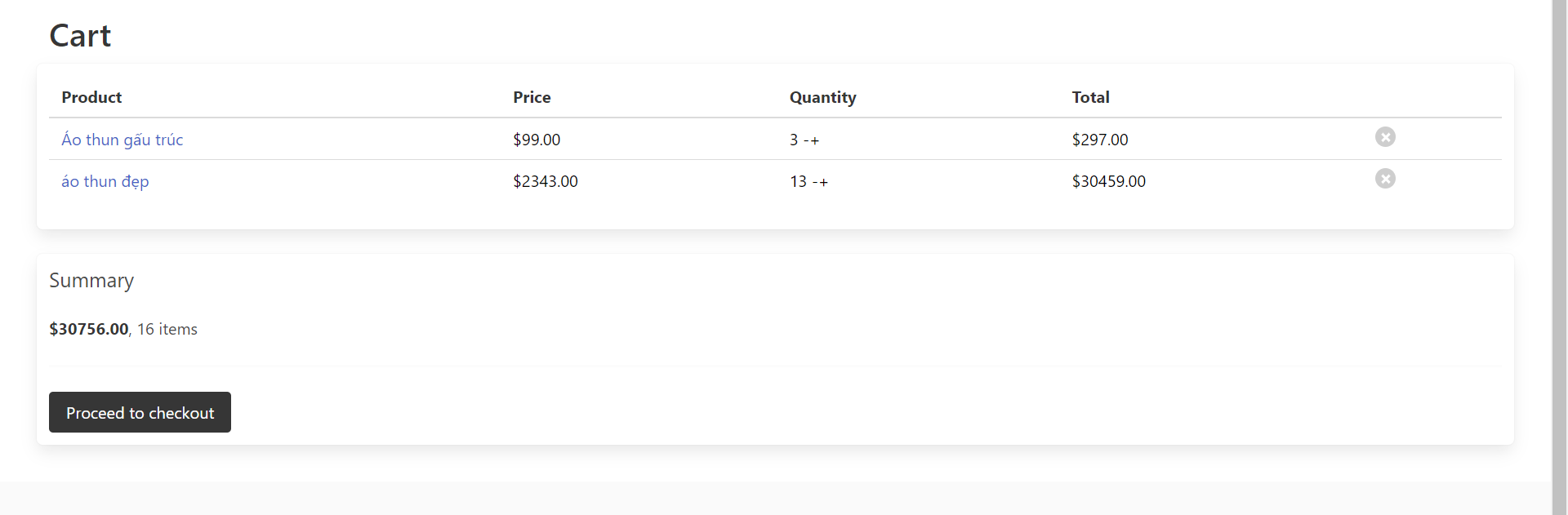


### Chức năng thêm vào giỏ hàng.

* Tại trang chi tiết sản phẩm, bạn có thể tùy chỉnh số lượng sản phẩm để thêm vào giỏ hàng.

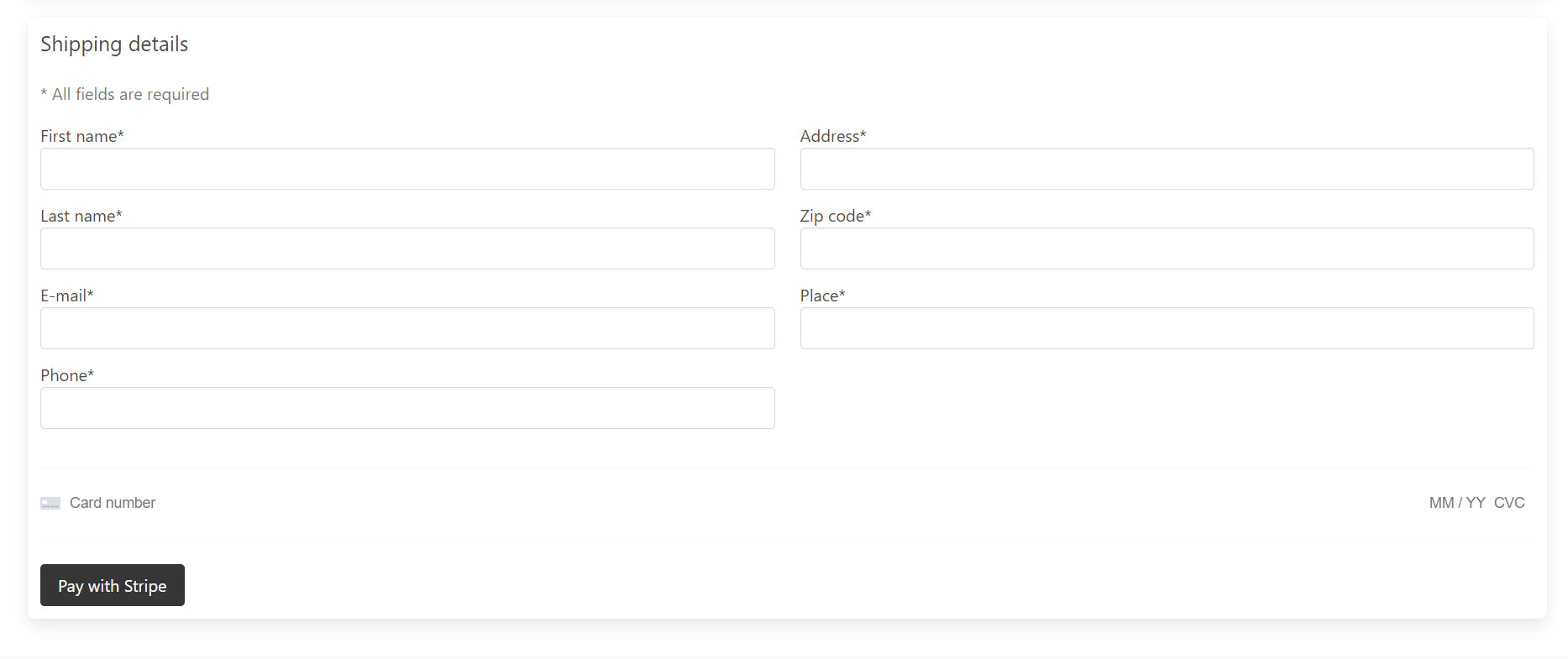


* Khi bấm nút thêm vào giỏ hàng - “Add to cart” thì các sản phẩm sẽ được chuyển tới Giỏ hàng - “Cart” của bạn.
* Ở đây các bạn có thể theo giỏi tiền của từng sản phẩm, số lượng, cũng như tổng số tiền các sản phẩm có trong giỏ hàng



### Chức năng đặt hàng - thanh toán.

* Khi đã sẵn sàng mua các mặt hàng có trong giỏ hàng, hệ thống sẽ điều hướng bạn tới trang “Shipping details” - chi tiết đặt hàng.
* Tại đây bạn bắt buộc phải điền các thông tin như Họ, Tên, Địa chỉ, Email, Zip code, Số điện thoại và Thẻ ngân hàng để chi trả tiền.
* Khi đã hoàn thành các bước trên, bấm nút “Pay with Stripe” để mua hàng.



## NHẬN XÉT & ĐÁNH GIÁ

### Ưu điểm

* Giao diện thân thiện dễ dùng.
* Có thể tìm kiếm sản phẩm theo từ khóa một cách dễ dàng.
* Có thể tìm kiếm sản phẩm theo 2 mùa trong năm.
* Tạo và nhập tài khoản nhanh chóng.
* Có thể lưu trữ các sản phẩm đã thêm vào giỏ hàng theo mỗi tài khoản.
* Có thể tính tổng tiền cần thanh toán cũng như mua các sản phẩm đã đặt vào giỏ.
* Chức năng đặt hàng - thanh toán online qua thẻ ngân hàng thuận tiện, nhanh chóng.

### Nhược điểm

* Giao diện còn quá đơn giản, không bắt mắt.
* Sản phẩm còn ít, chưa tiếp cận thêm nhiều nguồn hàng khác.
* Ngoài chức năng tìm kiếm sản phẩm theo từ khóa và theo mùa thì không còn chức năng lọc sản phẩm nào khác.
* Phải có tài khoản mới sử dụng được các chức năng như đặt sản phẩm vào giỏ hàng, mua hàng online

# KẾT LUẬN

## KẾT QUẢ ĐẠT ĐƯỢC

Dự án PyShop đã thành công trong việc:

* Cài đặt và cấu hình hệ thống một cách ổn định.
* Triển khai các chức năng hiển thị, xem chi tiết, phân loại, tìm kiếm, đăng ký, đăng nhập, thêm vào giỏ hàng và mua hàng.
* Tạo và chạy website với giao diện hấp dẫn và dễ sử dụng.

## HẠN CHẾ CỦA ĐỀ TÀI

Mặc dù dự án đã đạt được những kết quả tích cực, nhưng còn tồn tại một số hạn chế:

* Hiệu suất hệ thống: Mặc dù đã cố gắng tối ưu hóa, nhưng hiệu suất hệ thống vẫn cần được theo dõi và cải thiện nếu cần.
* Giao diện người dùng: Cần phát triển thêm về giao diện người dùng để đảm bảo trải nghiệm người dùng tốt nhất trên nhiều thiết bị và trình duyệt.

## HƯỚNG PHÁT TRIỂN ĐỀ TÀI

Để nâng cao chất lượng và mở rộng chức năng, dự án PyShop có thể phát triển theo các hướng sau:

* Tích hợp hệ thống thanh toán nâng cao: Nâng cấp hệ thống thanh toán để hỗ trợ nhiều phương thức thanh toán và đảm bảo tính bảo mật cao.
* Mở rộng danh mục sản phẩm: Thêm vào danh mục sản phẩm để mở rộng dải sản phẩm và thu hút đối tượng khách hàng đa dạng.
* Phát triển ứng dụng di động: Xây dựng ứng dụng di động để tăng cường trải nghiệm người dùng trên các thiết bị di động.
* Tối ưu hóa SEO: Tối ưu hóa trang web cho công cụ tìm kiếm để tăng khả năng xuất hiện trong các kết quả tìm kiếm.
* Mở rộng chức năng xã hội: Kết nối với các nền tảng mạng xã hội để tăng cường tương tác và quảng bá thương hiệu.

Những hướng phát triển này sẽ giúp dự án PyShop không chỉ duy trì sự hoạt động hiện tại mà còn đáp ứng được nhu cầu ngày càng đa dạng của khách hàng.



# TÀI LIỆU THAM KHẢO

**Tài liệu tiếng Việt:**

|  |  |
| --- | --- |
| [1] | Khóa học lập trình web python Django - Youtuber Son Nguyen - 20/11/2023 |
| [2] | Học VueJS 2 cơ bản - Youtuber toantran.mei - 22/11/2023 |
|  |  |

**Tài liệu tiếng Anh:**

|  |  |
| --- | --- |
| [3] | Python tutorial - w3school - 1/11/2023 |

