1. **So sánh List, Tuple, Set và Dictionary trong ngôn ngữ python. Cho ví dụ từng loại.**

Trong ngôn ngữ Python, List, Tuple, Set và Dictionary là các cấu trúc dữ liệu phổ biến được sử dụng để lưu trữ và quản lý dữ liệu. Dưới đây là sự so sánh giữa chúng:

1. List:

- List là một danh sách có thứ tự các phần tử.

- Các phần tử trong List có thể thay đổi (mutable).

- Các phần tử trong List được phân tách bằng dấu phẩy và được đặt trong dấu ngoặc vuông [].

- Các phần tử trong List có thể trùng lặp.

- Ví dụ:

python

fruits = ['apple', 'banana', 'orange']

print(fruits) # Output: ['apple', 'banana', 'orange']

2. Tuple:

- Tuple cũng là một danh sách có thứ tự các phần tử.

- Các phần tử trong Tuple không thể thay đổi (immutable).

- Các phần tử trong Tuple được phân tách bằng dấu phẩy và được đặt trong dấu ngoặc đơn ().

- Các phần tử trong Tuple có thể trùng lặp.

- Ví dụ:

python

numbers = (1, 2, 3, 4, 5)

print(numbers) # Output: (1, 2, 3, 4, 5)

3. Set:

- Set là một tập hợp các phần tử không có thứ tự.

- Các phần tử trong Set không thể thay đổi (immutable).

- Các phần tử trong Set không được phân tách bằng dấu phẩy và được đặt trong dấu ngoặc nhọn {}.

- Các phần tử trong Set không thể trùng lặp.

- Ví dụ:

python

colors = {'red', 'green', 'blue'}

print(colors) # Output: {'blue', 'green', 'red'}

4. Dictionary:

- Dictionary là một cấu trúc dữ liệu không có thứ tự, được lưu trữ dưới dạng cặp key-value.

- Các phần tử trong Dictionary có thể thay đổi (mutable).

- Các phần tử trong Dictionary được phân tách bằng dấu phẩy và được đặt trong dấu ngoặc nhọn {}.

- Các key trong Dictionary là duy nhất và không thể trùng lặp.

- Ví dụ:

python

student = {'name': 'John', 'age': 20, 'grade': 'A'}

print(student) # Output: {'name': 'John', 'age': 20, 'grade': 'A'}

Như vậy, List, Tuple, Set và Dictionary đều có những đặc điểm riêng và được sử dụng tùy thuộc vào mục đích và yêu cầu của từng bài toán cụ thể.

1. **Viết chương trình nhập vào từ bàn phím một chuỗi S. Hãy tạo từ điển D trong đó key là các chữ cái xuất hiện trong S (chữ cái không trùng nhau), còn value tương ứng với số lần xuất hiện của các chữ cái trong chuỗi S.**

S = input("Nhập chuỗi S: ") # Nhập chuỗi S từ bàn phím

D = {} # Khởi tạo từ điển D

for char in S:

if char.isalpha(): # Kiểm tra nếu là chữ cái

if char not in D: # Nếu chữ cái chưa có trong từ điển

D[char] = 1 # Thêm key với value là 1

print("Từ điển D: ", D)

1. **Cho tập tin text.txt với nội dung: dsak1257S%3\*&4pk6819**

**Xây dựng hàm đọc nội dung tập tin text.txt. Thực hiện lọc và lấy ra các số lẻ đưa vào danh sách( List) SoLe và các số chẵn đưa vào danh sách (List) SoChan.**

def doc\_tap\_tin():

SoLe = []

SoChan = []

try:

with open("text.txt", "r") as file:

content = file.read() # Đọc nội dung của tập tin

# Lọc và lấy ra các số lẻ và số chẵn

for char in content:

if char.isdigit(): # Kiểm tra nếu là số

num = int(char)

if num % 2 == 0: # Nếu là số chẵn

SoChan.append(num)

else: # Nếu là số lẻ

SoLe.append(num)

print("Danh sách số lẻ:", SoLe)

print("Danh sách số chẵn:", SoChan)

except FileNotFoundError:

print("Không tìm thấy tập tin text.txt")

doc\_tap\_tin()

1. **Viết chương trình tính tần suất xuất hiện của các từ được nhập vào từ input. In ra màn hình danh sách các từ sau khi đã sắp xếp theo bảng chữ cái.**

def tinh\_tan\_suat():

tu\_dict = {}

# Nhập các từ từ input

chuoi = input("Nhập các từ: ")

tu\_list = chuoi.split()

# Tính tần suất xuất hiện của các từ

for tu in tu\_list:

if tu not in tu\_dict:

tu\_dict[tu] = 1

else:

tu\_dict[tu] += 1

# Sắp xếp danh sách các từ theo bảng chữ cái

tu\_sorted = sorted(tu\_dict.keys())

# In ra danh sách các từ đã sắp xếp

print("Danh sách các từ đã sắp xếp:")

for tu in tu\_sorted:

print(tu, ":", tu\_dict[tu])

tinh\_tan\_suat()

1. **Viết chương trình cho phép nhập vào từ bàn phím họ và tên đầy đủ của các sinh viên trong lớp, mỗi sinh viên được nhập trên một dòng. Việc nhập sẽ kết thúc khi người dùng gõ vào dòng trống. Sau đó, hãy in ra tất cả các họ và tên không trùng nhau của sinh viên trong lớp.**

students = set()

while True:

name = input("Nhập họ và tên của sinh viên (nhập dòng trống để kết thúc): ")

if name == "":

break

students.add(name)

print("Các họ và tên không trùng nhau của sinh viên trong lớp:")

for name in students:

print(name)

1. **Viết chương trình nhập một mảng hai chiều các số thực A (m hàng, n cột) từ bàn phím**
2. **Tìm giá trị lớn nhất và nhỏ nhất trên mỗi cột.**
3. **Tìm phần từ lớn nhất và phần tử nhỏ nhất của mảng A cùng các chỉ số hàng và cột của 2 phần tử này.**
4. **Trong mảng A có bao nhiêu phần tử bằng phần tử lớn nhất.**

m = int(input("Nhập số hàng của mảng: "))

n = int(input("Nhập số cột của mảng: "))

# Nhập mảng từ bàn phím

A = []

for i in range(m):

row = []

for j in range(n):

num = float(input("Nhập phần tử A[{}][{}]: ".format(i, j)))

row.append(num)

A.append(row)

# A. Tìm giá trị lớn nhất và nhỏ nhất trên mỗi cột

max\_values = []

min\_values = []

for j in range(n):

column = [A[i][j] for i in range(m)]

max\_value = max(column)

min\_value = min(column)

max\_values.append(max\_value)

min\_values.append(min\_value)

print("Giá trị lớn nhất trên mỗi cột:", max\_values)

print("Giá trị nhỏ nhất trên mỗi cột:", min\_values)

# B. Tìm phần tử lớn nhất và nhỏ nhất của mảng A cùng chỉ số hàng và cột

max\_value = max(max\_values)

min\_value = min(min\_values)

max\_index = [(i, j) for i in range(m) for j in range(n) if A[i][j] == max\_value]

min\_index = [(i, j) for i in range(m) for j in range(n) if A[i][j] == min\_value]

print("Phần tử lớn nhất của mảng A:", max\_value, "với chỉ số hàng và cột:", max\_index)

print("Phần tử nhỏ nhất của mảng A:", min\_value, "với chỉ số hàng và cột:", min\_index)

# C. Đếm số phần tử bằng phần tử lớn nhất

count\_max = sum([1 for i in range(m) for j in range(n) if A[i][j] == max\_value])

print("Số phần tử bằng phần tử lớn nhất:", count\_max)

1. **Viết chương trình kiểm tra tính hợp lệ của mật khẩu mà người dùng nhập vào. Các tiêu chí kiểm tra mật khẩu bao gồm:**

**A.Ít nhất 1 chữ cái nằm trong [a-z]**

**B.Ít nhất 1 số nằm trong [0-9]ư**

**C.Ít nhất 1 kí tự nằm trong [A-Z]**

**D.Ít nhất 1 ký tự nằm trong [$#@]**

**E.Độ dài mật khẩu tối thiểu : 6**

**F.Độ dài mật khẩu tối đa : 12**

**Chương trình phải chấp nhận 1 chuỗi mật khẩu phân tách nhau bởi dấu phẩy và kiểm tra xem chúng có đáp ứng những tiêu chí trên hay không. Mật khẩu hợp lệ sẽ được in, mỗi mật khẩu cách nhau bởi dấu phẩy.**

**Ví dụ mật khẩu đặng nhập là : ABd1234@LaF1#,2w3E\*,2We3345**

**Thì đầu ra là : ABd1234@1**

import re

def validate\_password(password):

if len(password) < 6 or len(password) > 12:

return False

if not re.search("[a-z]", password):

return False

if not re.search("[0-9]", password):

return False

if not re.search("[A-Z]", password):

return False

if not re.search("[$#@]", password):

return False

return True

passwords = input("Nhập chuỗi mật khẩu, phân tách nhau bởi dấu phẩy: ")

password\_list = passwords.split(",")

valid\_passwords = []

for password in password\_list:

if validate\_password(password):

valid\_passwords.append(password)

print("Mật khẩu hợp lệ:", ", ".join(valid\_passwords))

1. **Viết chương trình python nhập 1 dãy số nguyên cách nhau bởi khoảng trắng, sau đó kiểm tra xem nó có đối xứng không ? Nếu dãy số nhập vào không dối xứng, kiểm tra xem nó có thể biến đổi để trở nên đối xứng hay không ? Nếu có khả năng đối xứng thì biến đổi nó trở nên thành đối xứng, ngược lại xuất hiện ra màn hình không có khả năng đối xứng.**

def is\_palindrome(numbers):

# Kiểm tra tính đối xứng của dãy số

reversed\_numbers = numbers[::-1]

if numbers == reversed\_numbers:

return True

return False

def make\_palindrome(numbers):

# Biến đổi dãy số thành dãy số đối xứng

reversed\_numbers = numbers[::-1]

palindrome = numbers + reversed\_numbers[1:]

return palindrome

input\_numbers = input("Nhập dãy số nguyên, cách nhau bởi khoảng trắng: ")

numbers\_list = input\_numbers.split()

if is\_palindrome(numbers\_list):

print("Dãy số đã nhập là đối xứng.")

else:

palindrome = make\_palindrome(numbers\_list)

if is\_palindrome(palindrome):

print("Dãy số không đối xứng, nhưng có thể biến đổi thành đối xứng:", " ".join(palindrome))

else:

print("Dãy số không đối xứng và không thể biến đổi thành đối xứng.")

1. **Định nghĩa một hàm có thể tạo ra 1 dictionary chứa key là những số từu 1 đến 10 (bao gồm cả 1 và 10) và các value là các số nguyên tố khác nhau ( nếu người dùng nhập vào 1 số không phải số nguyên tố hoặc số nguyên tố đã được nhập trước đó, thì yêu cầu nhười dùng nhập lại đên khi nào hoàn tất việc nhập 10 value). In ra value lớn nhất của dictionary.**

def is\_prime(n):

# Kiểm tra xem một số có phải là số nguyên tố hay không

if n < 2:

return False

for i in range(2, int(n \*\* 0.5) + 1):

if n % i == 0:

return False

return True

def create\_prime\_dictionary():

# Tạo từ điển với key từ 1 đến 10 và các giá trị là các số nguyên tố khác nhau

prime\_dict = {}

count = 0

while count < 10:

number = int(input("Nhập một số nguyên tố: "))

if is\_prime(number) and number not in prime\_dict.values():

count += 1

prime\_dict[count] = number

else:

print("Số không hợp lệ hoặc đã được nhập trước đó. Vui lòng nhập lại.")

return prime\_dict

prime\_dictionary = create\_prime\_dictionary()

max\_value = max(prime\_dictionary.values())

print("Giá trị lớn nhất của từ điển:", max\_value)

1. **Hiện thực lớp đối tượng DIENTHOAI với các đặc điểm sau:**

**Thược tính:**

**-HANGSX: chuỗi**

**-MODEL: chuỗi**

**-GIATIEN: số nguyên**

**-MANHINH(kích thước màn hình): số thực**

**Phương thức:**

**-Thêm thuộc tin điện thoại**

**-Sửa thông tin điện thoại**

**-Xóa thông tin điện thoại**

**-Xuất thông tin điện thoại**

1. **Hãy xây dựng các hàm cần thiết để cho phép nhập vào từu bàn phím các thông tin để thực hiện danh sách (List) các đối tượng DIENTHOAI.**
2. **Sắp xếp và xuất ra danh sách (List) các đối tượng DIENTHOAI theo thứ tự kích thước màn hình lớn đều nhỏ.**

class DIENTHOAI:

def \_\_init\_\_(self, hangsx, model, giatien, manhinh):

self.HANGSX = hangsx

self.MODEL = model

self.GIATIEN = giatien

self.MANHINH = manhinh

def them\_thong\_tin(self):

self.HANGSX = input("Nhập hãng sản xuất: ")

self.MODEL = input("Nhập model: ")

self.GIATIEN = int(input("Nhập giá tiền: "))

self.MANHINH = float(input("Nhập kích thước màn hình: "))

def sua\_thong\_tin(self):

print("Thông tin hiện tại:")

self.xuat\_thong\_tin()

self.HANGSX = input("Nhập hãng sản xuất mới: ")

self.MODEL = input("Nhập model mới: ")

self.GIATIEN = int(input("Nhập giá tiền mới: "))

self.MANHINH = float(input("Nhập kích thước màn hình mới: "))

print("Thông tin sau khi sửa:")

self.xuat\_thong\_tin()

def xoa\_thong\_tin(self):

self.HANGSX = ""

self.MODEL = ""

self.GIATIEN = 0

self.MANHINH = 0.0

def xuat\_thong\_tin(self):

print("Hãng sản xuất:", self.HANGSX)

print("Model:", self.MODEL)

print("Giá tiền:", self.GIATIEN)

print("Kích thước màn hình:", self.MANHINH)

danh\_sach\_dien\_thoai = []

def nhap\_danh\_sach\_dien\_thoai():

so\_luong = int(input("Nhập số lượng điện thoại: "))

for i in range(so\_luong):

print("Nhập thông tin điện thoại thứ", i + 1)

dien\_thoai = DIENTHOAI("", "", 0, 0.0)

dien\_thoai.them\_thong\_tin()

danh\_sach\_dien\_thoai.append(dien\_thoai)

def sap\_xep\_va\_xuat\_danh\_sach():

sorted\_danh\_sach = sorted(danh\_sach\_dien\_thoai, key=lambda x: x.MANHINH)

for dien\_thoai in sorted\_danh\_sach:

dien\_thoai.xuat\_thong\_tin()

nhap\_danh\_sach\_dien\_thoai()

sap\_xep\_va\_xuat\_danh\_sach()

1. **Hiện thực lớp đối tượng SINHVIEN với các đặc điểm sau:**

**Thuộc tính:**

**-MSSV:chuỗi**

**-HOTEN:chuỗi**

**-DIEMKT1:số thực**

**-DIEMKT2:số thực**

**-DIEMTB: (DIEMKT1+ DIEMKT2)/2**

**Phương thức:**

**-Thêm thông tin sinh viên**

**-sửa thông tin sinh viên**

**-Xuất thông tin sinh viên**

1. **Hãy xây dựng các hàm cần thiết để cho phép nhập vào từu bàn phím các thông tin để thực hiện danh sách (List) các đối tượng SINHVIEN.**
2. **Sắp xếp và xuất ra danh sách (List) các đối tượng SINHVIEN theo thứ tự DIEMTB từ cao đến thấp.**

class SINHVIEN:

def \_\_init\_\_(self, mssv, hoten, diemkt1, diemkt2):

self.MSSV = mssv

self.HOTEN = hoten

self.DIEMKT1 = diemkt1

self.DIEMKT2 = diemkt2

self.DIEMTB = (diemkt1 + diemkt2) / 2

def them\_thong\_tin(self):

self.MSSV = input("Nhập MSSV: ")

self.HOTEN = input("Nhập họ tên: ")

self.DIEMKT1 = float(input("Nhập điểm KT1: "))

self.DIEMKT2 = float(input("Nhập điểm KT2: "))

self.DIEMTB = (self.DIEMKT1 + self.DIEMKT2) / 2

def sua\_thong\_tin(self):

print("Thông tin hiện tại:")

self.xuat\_thong\_tin()

self.MSSV = input("Nhập MSSV mới: ")

self.HOTEN = input("Nhập họ tên mới: ")

self.DIEMKT1 = float(input("Nhập điểm KT1 mới: "))

self.DIEMKT2 = float(input("Nhập điểm KT2 mới: "))

self.DIEMTB = (self.DIEMKT1 + self.DIEMKT2) / 2

print("Thông tin sau khi sửa:")

self.xuat\_thong\_tin()

def xuat\_thong\_tin(self):

print("MSSV:", self.MSSV)

print("Họ tên:", self.HOTEN)

print("Điểm KT1:", self.DIEMKT1)

print("Điểm KT2:", self.DIEMKT2)

print("Điểm TB:", self.DIEMTB)

danh\_sach\_sinh\_vien = []

def nhap\_danh\_sach\_sinh\_vien():

so\_luong = int(input("Nhập số lượng sinh viên: "))

for i in range(so\_luong):

print("Nhập thông tin sinh viên thứ", i + 1)

sinh\_vien = SINHVIEN("", "", 0.0, 0.0)

sinh\_vien.them\_thong\_tin()

danh\_sach\_sinh\_vien.append(sinh\_vien)

def sap\_xep\_va\_xuat\_danh\_sach():

sorted\_danh\_sach = sorted(danh\_sach\_sinh\_vien, key=lambda x: x.DIEMTB, reverse=True)

for sinh\_vien in sorted\_danh\_sach:

sinh\_vien.xuat\_thong\_tin()

nhap\_danh\_sach\_sinh\_vien()

sap\_xep\_va\_xuat\_danh\_sach()

1. **Hiện thực lớp đối tượng QUEUE với các đặc điểm sau:**

**Thuộc tính :**

**- elements: danh sách (List)**

**- front : số nguyên**

**Phương thức:**

**- Kiểm tra hàng đợi rỗng**

**- Trả về phần tử ở đầu hàng đợi**

**- Thêm 1 phần tử vào hàng đợi**

**- Xóa phàn tử ở đầu hàng đợi**

**- Xuất hàng đợi ra màn hình**

1. **Hãy xây dựng các hàm cần thiết bắt buộc người dùng nhập vào 1 số nguyên tố ( nếu giá trị nhập vào không phải số nguyên tố thì yêu cầu nhập lại). Thêm các số nguyên tố được nhập vào hàng đợi Q là đối tượng kế thừa từ lớp QUEUE. Xuất ra màn hình hàng đợi Q.**
2. **Tuần tự thực hiện các thao tác sau cho đến khi hàng đợi Q rỗng. Tìm và xuất ra màn hình phần từ đầu danh sách Q. Xóa phần tử đầu danh sách Q và xuất ra màn hình hàng đợi Q.**

class QUEUE:

def \_\_init\_\_(self):

self.elements = []

self.front = -1

def is\_empty(self):

return self.front == -1

def get\_front(self):

if not self.is\_empty():

return self.elements[self.front]

else:

return None

def enqueue(self, item):

self.elements.append(item)

if self.is\_empty():

self.front = 0

def dequeue(self):

if not self.is\_empty():

self.front += 1

if self.front == len(self.elements):

self.elements = []

self.front = -1

else:

print("Hàng đợi rỗng.")

def display(self):

if not self.is\_empty():

for item in self.elements[self.front:]:

print(item, end=" ")

print()

else:

print("Hàng đợi rỗng.")

# Hàm kiểm tra số nguyên tố

def is\_prime(num):

if num < 2:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

# Tạo một đối tượng QUEUE

Q = QUEUE()

# A. Thêm các số nguyên tố vào hàng đợi Q

while True:

try:

num = int(input("Nhập một số nguyên tố (hoặc -1 để kết thúc): "))

if num == -1:

break

if is\_prime(num):

Q.enqueue(num)

else:

print("Số bạn nhập không phải là số nguyên tố. Hãy thử lại.")

except ValueError:

print("Vui lòng nhập một số nguyên tố hợp lệ.")

# Hiển thị hàng đợi Q

print("Hàng đợi Q:")

Q.display()

# B. Thực hiện các thao tác trên hàng đợi Q

while not Q.is\_empty():

front = Q.get\_front()

if front is not None:

print("Phần tử ở đầu hàng đợi Q:", front)

Q.dequeue()

print("Hàng đợi Q sau khi xóa phần tử đầu:")

Q.display()

else:

print("Hàng đợi Q rỗng.")

1. **Viết chương trình python theo OOP:**

**- Class InHoaDon( in hóa đơn) bán hàng Rau Quả trong đó các thuộc tính : Ngay, LoaiRauQua, SoLuong, DonGia, ThanhTien.**

class InHoaDon:

def \_\_init\_\_(self, ngay, loai\_rau\_qua, so\_luong, don\_gia):

self.ngay = ngay

self.loai\_rau\_qua = loai\_rau\_qua

self.so\_luong = so\_luong

self.don\_gia = don\_gia

self.thanh\_tien = self.tinh\_thanh\_tien()

def tinh\_thanh\_tien(self):

return self.so\_luong \* self.don\_gia

def in\_hoa\_don(self):

print("Hóa đơn bán hàng Rau Quả")

print("Ngày:", self.ngay)

print("Loại Rau Quả:", self.loai\_rau\_qua)

print("Số Lượng:", self.so\_luong)

print("Đơn Giá:", self.don\_gia)

print("Thành Tiền:", self.thanh\_tien)

# Sử dụng lớp InHoaDon

ngay\_mua = "2023-09-07"

loai\_rau\_qua = "Dưa hấu"

so\_luong = 5

don\_gia = 25

hoa\_don = InHoaDon(ngay\_mua, loai\_rau\_qua, so\_luong, don\_gia)

hoa\_don.in\_hoa\_don()

1. **Cho tập tin thống kê môn học với tên k2021.csv.**
2. **Viết câu lệnh đọc file k2021.csv.**
3. **In 5 dòng đầu tiên và 5 dòng cuối cùng của dữ liệu và in ra màn hình.**
4. **Thống kê xem lớp có bao nhiêu bạn trượt môn ( điểm dưới 4 hoặc không có điểm).**
5. **Thống kê xem lớp có bao nhiêu bạn điểm loại giỏi (điểm từ 8 trở lên).**

import pandas as pd

# Đọc file CSV vào DataFrame

df = pd.read\_csv('k2021.csv')

# In 5 dòng đầu tiên

print("5 dòng đầu tiên:")

print(df.head(5))

# In 5 dòng cuối cùng

print("\n5 dòng cuối cùng:")

print(df.tail(5))

# Đếm số bạn trượt môn

so\_ban\_truot = len(df[(df['Diem'] < 4) | (df['Diem'].isna())])

print(f"Số bạn trượt môn: {so\_ban\_truot}")

# Đếm số bạn điểm loại giỏi

so\_ban\_loai\_gioi = len(df[df['Diem'] >= 8])

print(f"Số bạn điểm loại giỏi: {so\_ban\_loai\_gioi}")

1. **Dữ liệu: winequality-red. Csv có các thuộc tính sau :**

**Fixed acidity nồng độ axit tataric**

**Volatile acidity Tính axit**

**citric acid Nồng độ axit citric**

**Residual sugar Nồng độ đường dư**

**Chlorides Nồng độ clo**

**Free sulfur dioxide Nồng độ acid sulfurus tự do**

**Total sulfur dioxide Nồng độ acid sulfurus**

**Density Mật độ ( khối lượng đơn vị thể tích)**

**pH Độ pH**

**Sulphates Nồng độ sunfat**

**Alcohol Nồng độ chất alcohol**

**Quality Chất lượng**

**Viết chương trình python:**

1. **In ra xem bao nhiêu dong vào bao nhiêu cột trong file.**
2. **Vẽ biểu đồ minh họa Dataset với thuộc tính alcohol và điểm của quality.**
3. **Sử dụng hồi quy tuyến tính để xây dựng tương quan giữa thuộc tính alcohol và quality :**

**-In ra độ lệch chuẩn (căn bậc 2 phương sai).**

**-Hệ số hồi quy.**

**-Sai số.**

**-Dự báo về chất lượng rượu khi cho nồng độ alcohol thay đổi (Nhập).**

import pandas as pd

# Đọc dữ liệu từ tệp CSV

data = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/winequality-red.csv', delimiter=';')

# Lấy số dòng và số cột

num\_rows, num\_columns = data.shape

print(f"Số dòng: {num\_rows}")

print(f"Số cột: {num\_columns}")

import matplotlib.pyplot as plt

# Vẽ biểu đồ

plt.scatter(data['alcohol'], data['quality'], alpha=0.5)

plt.title('Biểu đồ Alcohol và Quality')

plt.xlabel('Alcohol')

plt.ylabel('Quality')

plt.show()

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import numpy as np

# Chọn thuộc tính 'alcohol' và 'quality'

X = data[['alcohol']]

y = data['quality']

# Xây dựng mô hình hồi quy tuyến tính

model = LinearRegression()

model.fit(X, y)

# In ra các thông số của mô hình

coeff = model.coef\_[0]

intercept = model.intercept\_

mse = mean\_squared\_error(y, model.predict(X))

std\_deviation = np.sqrt(mse)

print(f"Hệ số hồi quy: {coeff:.2f}")

print(f"Sai số (intercept): {intercept:.2f}")

print(f"Độ lệch chuẩn (căn bậc 2 phương sai): {std\_deviation:.2f}")

# Dự báo chất lượng rượu khi cho nồng độ alcohol thay đổi (Nhập)

new\_alcohol = float(input("Nhập nồng độ alcohol: "))

predicted\_quality = model.predict([[new\_alcohol]])

print(f"Dự báo chất lượng rượu: {predicted\_quality[0]:.2f}")