Câu 1:

Trong Python, chúng ta có nhiều cách để định dạng chuỗi, tạo ra các chuỗi mới dựa trên các giá trị biến. Hai cách phổ biến nhất là sử dụng toán tử % và phương thức format(). Mỗi cách có ưu điểm và nhược điểm riêng, phù hợp với những trường hợp khác nhau:

***Toán tử %***

· **Cú pháp:** chuỗi % (giá\_trị,)

· **Ưu điểm:**

* Đơn giản, dễ hiểu
* Được hỗ trợ từ các phiên bản Python cũ

· **Nhược điểm:**

* Cú pháp có thể trở nên phức tạp khi định dạng nhiều giá trị
* Ít linh hoạt so với phương thức format()

Ví dụ:

name = "Alice"

age = 30

print("Tên của tôi là %s và tôi %d tuổi." % (name, age))

***Phương thức format() (New-style formatting)***

Cú pháp: chuỗi.format(giá\_trị1, giá\_trị2, ...)

Ưu điểm:

Linh hoạt hơn, cho phép định dạng phức tạp

Hỗ trợ định dạng số, ngày tháng, ...

Có thể sử dụng đặt tên cho các giá trị

Nhược điểm:

Cú pháp có thể dài hơn trong một số trường hợp đơn giản

*Ví dụ:*

price = 19.99

quantity = 3

print("Tổng tiền là: {:.2f}$ cho {} sản phẩm.".format(price \* quantity, quantity))

# Sử dụng đặt tên cho các giá trị

person = {"name": "Bob", "age": 25}

print("Tên của người này là {name} và {age} tuổi.".format(\*\*person))

So sánh và lựa chọn

Nên sử dụng f-strings: Đây là cách định dạng chuỗi được khuyến nghị trong các phiên bản Python hiện đại, vì nó có cú pháp gọn gàng và hiệu năng tốt.

Sử dụng phương thức format(): Khi cần định dạng phức tạp hoặc tương thích với các phiên bản Python cũ hơn.

Tránh sử dụng toán tử %: Nên hạn chế sử dụng toán tử % vì cú pháp của nó có thể trở nên khó đọc và ít linh hoạt.

Câu 2:

import random

# Nhập đoạn [a, b] bất kỳ

a = int(input("Nhập giá trị đầu của đoạn (a): "))

b = int(input("Nhập giá trị cuối của đoạn (b): "))

# Kiểm tra để đảm bảo a <= b

if a > b:

a, b = b, a # Đảo ngược giá trị nếu a lớn hơn b

# Sinh số ngẫu nhiên trong đoạn [a, b]

random\_number = random.randint(a, b)

# Xuất kết quả

print(f"Số ngẫu nhiên được sinh ra trong đoạn [{a}, {b}] là: {random\_number}")

Câu 3:

List và tuple là hai kiểu dữ liệu được sử dụng để lưu trữ các phần tử trong Python. Tuy nhiên, chúng có những đặc điểm khác biệt quan trọng.

*Cú pháp:*

List: Được bao quanh bởi dấu ngoặc vuông [].

Tuple: Được bao quanh bởi dấu ngoặc tròn ().

*Sử dụng:*

List: Thường được sử dụng khi bạn cần một cấu trúc dữ liệu có thể thay đổi linh hoạt, như lưu trữ một danh sách các sản phẩm trong giỏ hàng, các số ngẫu nhiên.

Tuple: Thường được sử dụng khi bạn muốn bảo vệ dữ liệu khỏi bị thay đổi, như lưu trữ các thông tin không đổi như ngày tháng năm sinh, tọa độ địa lý.

*Tính biến đổi (mutability):*

List: Có thể thay đổi sau khi được tạo. Bạn có thể thêm, xóa, sửa đổi các phần tử trong list.

Tuple: Không thể thay đổi sau khi được tạo. Các phần tử của tuple là cố định.

Câu 4:

Tuple trong Python là một kiểu dữ liệu vô cùng hữu ích, đặc biệt khi bạn muốn bảo vệ dữ liệu khỏi bị thay đổi một cách không mong muốn. Dưới đây là một số ứng dụng điển hình của tuple trong thực tế:

Lưu trữ dữ liệu không thay đổi:

Tọa độ: Một tọa độ địa lý thường được biểu diễn dưới dạng một tuple (vĩ độ, kinh độ). Vì tọa độ thường không thay đổi nên việc sử dụng tuple là rất hợp lý.

Trả về nhiều giá trị từ một hàm:

Một hàm trong Python chỉ có thể trả về một giá trị. Tuy nhiên, bạn có thể trả về nhiều giá trị bằng cách đóng gói chúng vào trong một tuple.

Ví dụ: Một hàm tính toán diện tích và chu vi của một hình chữ nhật có thể trả về một tuple gồm hai giá trị.