**1. Phân biệt toán tử định dạng chuỗi và hàm định dạng chuỗi trong Python:**

• Toán tử định dạng chuỗi (%):

Đây là cách định dạng chuỗi cũ trong Python. Nó sử dụng toán tử % để chèn giá trị vào chuỗi theo các ký hiệu định dạng.

• Hàm định dạng chuỗi (str.format()):

Đây là cách định dạng mới hơn, linh hoạt hơn, cung cấp nhiều tính năng để chèn giá trị vào chuỗi bằng cách sử dụng {} làm trình giữ chỗ.

• F-string (f""):

Đây là cách định dạng chuỗi mới nhất, có trong Python 3.6 trở đi, cho phép chèn trực tiếp các biểu thức vào chuỗi.

Ví dụ:

**1. Toán tử định dạng chuỗi (%):**

name = "Alice"

age = 25

print("My name is %s and I am %d years old." % (name, age))

**2. Hàm định dạng chuỗi (str.format()):**

name = "Alice"

age = 25

print("My name is {} and I am {} years old.".format(name, age))

3. F-string (f""):

name = "Alice"

age = 25

print(f"My name is {name} and I am {age} years old.")

**4. Căn chỉnh chuỗi với str.format():**

print("{:<10} | {:^10} | {:>10}".format("left", "center", "right"))

**5. Định dạng số thập phân với f-string:**

value = 123.456789

print(f"The value is {value:.2f}")

**2. Viết chương trình xuất ra số ngẫu nhiên trong một đoạn bất kỳ bất kỳ cho trước:**

import random

# Nhập đoạn bất kỳ

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

b = int(input("Nhập giá trị kết thúc: "))

# Xuất ra số ngẫu nhiên

so\_ngau\_nhien = random.randint(a, b)

print(f"Số ngẫu nhiên trong đoạn [{a}, {b}]: {so\_ngau\_nhien}")

**3. Khác biệt cơ bản giữa list và tuple**:

• list:

• Có thể thay đổi (mutable): Bạn có thể thêm, xóa, hoặc thay đổi các phần tử.

• Cú pháp: Được định nghĩa bằng dấu ngoặc vuông [].

• Hiệu suất: Chậm hơn tuple vì cần lưu trữ thêm thông tin về khả năng thay đổi.

• tuple:

• Không thể thay đổi (immutable): Sau khi tạo, bạn không thể sửa đổi nó.

• Cú pháp: Được định nghĩa bằng dấu ngoặc tròn ().

• Hiệu suất: Nhanh hơn list vì không cần quản lý việc thay đổi.

Ví dụ:

# List

my\_list = [1, 2, 3]

my\_list[0] = 10 # Có thể thay đổi

# Tuple

my\_tuple = (1, 2, 3)

# my\_tuple[0] = 10 # Sẽ báo lỗi vì tuple không thể thay đổi

**4. Ứng dụng kiểu dữ liệu tuple trong thực tế:**

• Bảo vệ tính bất biến: Dùng tuple khi bạn muốn đảm bảo rằng dữ liệu không bị thay đổi. Ví dụ, tọa độ điểm (x, y) thường được lưu trong tuple.

• Dùng làm khóa cho dictionary: Vì tuple là bất biến, chúng có thể được sử dụng làm khóa trong từ điển (dictionary), trong khi list thì không.

• Trả về nhiều giá trị từ hàm: Các hàm thường trả về nhiều giá trị dưới dạng tuple.

Ví dụ ứng dụng thực tế:

**1. Trả về nhiều giá trị:**

def get\_coordinates():

return (10, 20)

x, y = get\_coordinates()

print(f"x: {x}, y: {y}")

**2. Làm khóa cho từ điển:**

locations = {

(10.7769, 106.7009): "Ho Chi Minh City",

(21.0285, 105.8542): "Hanoi"

}