1. **Phân biệt toán tử định dạng chuỗi và hàm định dạng chuỗi có sẵn trong gói thư viện chuẩn Python? Cho năm ví dụ minh họa tương ứng?**

* **Toán tử định dạng chuỗi**

- Toán tử % được sử dụng để chèn giá trị vào chuỗi bằng cách thay thế các ký tự đặc biệt với các giá trị tương ứng. Các chuỗi bao gồm %d ( chèn giá trị số nguyên ), %s ( chèn chuỗi ), %f ( chèn giá trị số thực )

Cú pháp: "format string" % (values)

* **Hàm định dạng chuỗi format{}**

- Phương thức format{} được giới thiệu trong Python 2.6 và 3.0. Nó cung cấp cách linh hoạt để định dạng chuỗi bằng cách sử dụng dấu ngoặc nhọn {}.

Cú pháp : "chuỗi định dạng {}".format(giá trị)

**VD minh họa: VD1 Toán tử định dạng chuỗi**

name = "Trí"

age = 19

formatted\_string = "Tên tôi là %s và tôi %d tuổi" % (name, age)

print(formatted\_string)

**VD 2 : Toán tử định dạng chuỗi**

pi = 3.14159

formatted\_string = "Giá trị của pi là %.2f" % pi

print(formatted\_string)

**VD 3 : Toán tử định dạng chuỗi**

name = "Trí"

age = 19

height = 1.75

formatted\_string = "Tên tôi là %s, tôi %d tuổi và cao %.2f mét" % (name, age, height)

print(formatted\_string)

**VD 4 : Hàm định dạng chuỗi**

name = "Trí"

age = 19

formatted\_string = "Tên tôi là {} và tôi {} tuổi".format(name, age)

print(formatted\_string)

**VD 5: Hàm định dạng chuỗi**

from datetime import datetime

today = datetime.now()

formatted\_string = "Ngày hôm nay là {:%d-%m-%Y}".format(today)

print(formatted\_string)

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

import random

# Nhập giá trị bắt đầu và kết thúc của đoạn

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

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

# Tạo số ngẫu nhiên trong đoạn [start, end]

random\_number = random.randint(start, end)

# Xuất ra số ngẫu nhiên

print(f"Số ngẫu nhiên trong đoạn [{start}, {end}] là: {random\_number}")

1. **Khác biệt cơ bản giữa list và tuple?**

- **List**: Là kiểu dữ liệu thay đổi được (mutable), nghĩa là bạn có thể thay đổi, thêm, hoặc xóa các phần tử sau khi danh sách đã được tạo ra.

- Cú pháp: List được khai báo bằng cặp dấu ngoặc vuông []

VD: my\_list = [1, 2, 3]

my\_list[0] = 10 # Thay đổi giá trị của phần tử đầu tiên

my\_list.append(4) # Thêm phần tử mới vào danh sách

print(my\_list) # Output: [10, 2, 3, 4]

- **Tuple**: Là kiểu dữ liệu không thay đổi được (immutable), nghĩa là sau khi tuple được tạo ra, bạn không thể thay đổi các phần tử, thêm hoặc xóa chúng.

- Cú pháp : Tuple được khai báo bằng cặp dấu ngoặc đơn ().

VD: my\_tuple = (1, 2, 3)

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

1. **Ứng dụng kiểu dữ liệu tuple trong thực tế?**

- **Nhóm các giá trị liên quan**: Khi bạn cần nhóm các giá trị liên quan với nhau nhưng không cần thay đổi chúng sau khi đã được tạo ra.

VD: Lưu trữ thông tin về một điểm địa lý (tọa độ)

location = (10.759, 106.662) # (vĩ độ, kinh độ)

- **Sử dụng làm khóa trong từ điển (Dictionary):**Vì tuple là không thay đổi được, nó có thể được sử dụng làm khóa trong từ điển, trong khi list thì không thể.

VD: Lưu trữ thông tin về số lượng sản phẩm trong kho theo các danh mục

inventory = {

('electronics', 'laptops'): 25,

('furniture', 'chairs'): 40,

('clothing', 't-shirts'): 100

}

- **Các phần tử cố định trong danh sách:**Khi bạn cần các phần tử cố định trong danh sách, ví dụ như các thiết lập cấu hình không thay đổi.

VD: Cấu hình của một ứng dụng

config = ('localhost', 8080) # (địa chỉ máy chủ, cổng)

- **Lưu trữ các phần tử trong một tập hợp (Set):** Vì tuple là immutable, nó có thể được lưu trữ trong một set, trong khi list thì không.

VD: Lưu trữ các tọa độ điểm trong một tập hợp để đảm bảo tính duy nhất

points = {(1, 2), (3, 4), (5, 6)}

**- Dữ liệu không thay đổi trong cấu trúc dữ liệu phức tạp:** Trong các cấu trúc dữ liệu phức tạp như đồ thị hoặc cây, tuple thường được sử dụng để đại diện cho các cạnh hoặc các nút.

VD: edges = [(1, 2), (2, 3), (3, 4)]