**Họ và tên:** Ngô Uyên Nhi

**Mssv:** 23682331

**CÂU HỎI TỔNG HỢP**

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 (“%”)
* **Chức năng**: Chèn giá trị vào chuỗi theo định dạng.
* **Cú pháp**: "% định\_dạng" % giá\_trị

Ví dụ: name = “Alice”

age = 30

Print(“Name: %s, Age: %d” %(name, age)

* Kết quả: Name: Alice, Age: 30

- Hàm Định dạng Chuỗi (format() và f-string)

* **Chức năng**: Cung cấp nhiều tùy chọn định dạng hơn và dễ đọc hơn.
* **Cú pháp**: format(): "chuỗi {} {}".format(giá\_trị1, giá\_trị2)

f-string: f"chuỗi {giá\_trị1} {giá\_trị2}"

Ví dụ: print("Name: {}, Age: {}".format(name, age))

* Kết quả: Name: Alice, Age: 30

print(f"Name: {name}, Age: {age}")

* Kết quả: Name: Alice, Age: 30

**Toán tử %**: Cú pháp cũ, ít tùy chọn hơn.

F**ormat() và f-string**: Cú pháp hiện đại, linh hoạt hơn.

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

import random

# Nhập giá trị thấp nhất và cao nhất của đoạn a,b

a = int(input("Nhập giá trị thấp nhất: "))

b = int(input("Nhập giá trị cao nhất: "))

# Chọn số ngẫu nhiên trong đoạn

ngaunhien = random.randint(a, b)

print(f"Số ngẫu nhiên trong đoạn [{a}, {b}] là: {ngaunhien }")

3. Khác biệt cơ bản giữa list và tuple?

* **list**: Có thể thay đổi (mutable), dùng [], cho phép thêm, xóa, và sửa phần tử, tốc độ chậm hơn. Dễ thay đổi, thêm/xóa phần tử.

**Minh họa:**

N = [1, 2, 3]

N[0] = 10 # Thay đổi giá trị phần tử đầu tiên

N.append(4) # Thêm phần tử mới

print(N) # Kết quả: [10, 2, 3, 4]

* **tuple**: Không thể thay đổi (immutable), dùng (), không cho phép thay đổi phần tử sau khi tạo, tốc độ nhanh hơn. Không thay đổi, cố định sau khi tạo.

**Minh họa:**

list = (1, 2, 3)

list[0] = 10 # Gây lỗi vì không thể thay đổi phần tử

print(list) # Kết quả: (1, 2, 3)

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

* **Khóa trong từ điển:** Dùng tuple làm khóa trong từ điển để lưu trữ và truy xuất dữ liệu dựa trên cặp giá trị.
* **Trả về nhiều giá trị từ hàm:** Trả về nhiều giá trị cùng lúc từ một hàm mà không cần sử dụng nhiều biến.
* **Lưu trữ dữ liệu cố định:** Sử dụng tuple để lưu trữ các giá trị không cần thay đổi trong suốt chương trình, như các hằng số.
* **Tăng tốc độ xử lý:** tuple có hiệu suất cao hơn list, thích hợp cho các chương trình yêu cầu tốc độ xử lý nhanh.
* **Nhóm các giá trị liên quan:** Dùng tuple để nhóm các giá trị liên quan, chẳng hạn như thông tin về một đối tượng (tên, tuổi, địa chỉ).
* **Đảm bảo tính an toàn của dữ liệu:** Bảo vệ dữ liệu khỏi bị thay đổi ngẫu nhiên hoặc không mong muốn vì tuple là bất biến (immutable).
* **Làm tham số hàm:** Sử dụng tuple để truyền nhiều giá trị cùng lúc vào một hàm, giúp hàm có thể xử lý nhiều dữ liệu mà không cần nhiều tham số riêng lẻ.
* **Định dạng không đổi trong cấu hình:** Dùng tuple để lưu các cấu hình hoặc tham số cần cố định và không thay đổi trong chương trình.