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 (%):** là một phương pháp cũ để chèn các giá trị vào trong chuỗi. Nó tương tự như cách định dạng trong C.

**Cú pháp:** "format string" % (values)

Ví dụ 1: Định dạng số nguyên:

age = 25

formatted\_string = "I am %d years old." % age

print(formatted\_string)

Ví dụ 2: Định dạng chuỗi:

name = "Alice"

formatted\_string = "Hello, %s!" % name

print(formatted\_string)

Ví dụ 3: Định dạng số thực:

pi = 3.14159

formatted\_string = "The value of pi is %.2f" % pi

print(formatted\_string)

Ví dụ 4: Định dạng số nguyên và chuỗi:

item = "apples"

quantity = 5

formatted\_string = "I have %d %s." % (quantity, item)

print(formatted\_string)

Ví dụ 5: Định dạng phần trăm:

percentage = 50

formatted\_string = "The task is %d%% complete." % percentage

print(formatted\_string)

* **Hàm định dạng chuỗi “str.format()”:** là phương pháp hiện đại hơn, cho phép định dạng chuỗi một cách linh hoạt hơn. Nó cho phép chèn các giá trị vào chuỗi với cú pháp gọn gàng và dễ đọc.

**Cú pháp:** "format string".format(values)

Ví dụ 1: Định dạng số nguyên:

age = 25

formatted\_string = "I am {} years old.".format(age)

print(formatted\_string)

Ví dụ 2: Định dạng chuỗi:

name = "Alice"

formatted\_string = "Hello, {}!".format(name)

print(formatted\_string)

Ví dụ 3: Định dạng số thực:

pi = 3.14159

formatted\_string = "The value of pi is {:.2f}".format(pi)

print(formatted\_string

Ví dụ 4: Định dạng số nguyên và chuỗi:

item = "apples"

quantity = 5

formatted\_string = "I have {} {}.".format(quantity, item)

print(formatted\_string)

Ví dụ 5: Định dạng phần tram:

percentage = 50

formatted\_string = "The task is {}% complete.".format(percentage)

print(formatted\_string)

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

import random

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

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

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

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

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

* **Tính bất biến (Mutable vs Immutable):**

List: Có thể thay đổi (mutable) giá trị của các phần tử trong list sau khi đã tạo ra list đó, như thêm, xóa, hoặc sửa phần tử.

Tuple: Không thể thay đổi (immutable) các phần tử bên trong nó sau khi một tuple được tạo.

* **Cú pháp:**

List: Được tạo bằng cặp dấu ngoặc vuông [].

Tuple: Được tạo bằng cặp dấu ngoặc tròn ().

* **Sử dụng:**

List: Thường được sử dụng khi cần một cấu trúc dữ liệu mà có thể thay đổi trong suốt quá trình thực thi chương trình.

Tuple: Thường được sử dụng khi có một tập hợp các giá trị cố định, không thay đổi, như tọa độ (x, y) hoặc các hằng số.

* **Hiệu năng:**

List: Do có thể thay đổi được nên list tiêu tốn nhiều tài nguyên hơn và thường chậm hơn khi so sánh với tuple.

Tuple: Do không thể thay đổi nên tuple thường có hiệu năng tốt hơn list, đặc biệt trong các trường hợp cần lưu trữ và truy cập dữ liệu không đổi.

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

* **Lưu trữ các giá trị không đổi (Immutable Data):** thường được sử dụng để lưu trữ các dữ liệu không thay đổi trong suốt quá trình thực thi chương trình. Ví dụ, các cặp giá trị như tọa độ (x, y), thông số cấu hình, hoặc các hằng số.
* **Làm khóa trong từ điển (Dictionary Key):** do tính chất không thay đổi của tuple, nó có thể được sử dụng làm khóa trong từ điển trong khi list thì không.
* **Trả về nhiều giá trị từ một hàm (Function Return Multiple Values):** hàm trong Python có thể trả về nhiều giá trị dưới dạng một tuple, cho phép dễ dàng trả về nhiều kết quả mà không cần sử dụng cấu trúc dữ liệu phức tạp.
* **Làm giá trị không thay đổi trong các cấu trúc dữ liệu khác:** có thể được sử dụng bên trong list, set, hoặc các cấu trúc dữ liệu khác khi cần đảm bảo một nhóm các phần tử không bị thay đổi.
* **Lưu trữ dữ liệu có cấu trúc cố định:** thường được sử dụng để lưu trữ các tập hợp dữ liệu có cấu trúc cố định, ví dụ như các trường thông tin của một bản ghi trong cơ sở dữ liệu.
* **Phân loại nhiều giá trị (Packing/Unpacking):** cho phép gán nhiều giá trị cho nhiều biến cùng một lúc, hoặc chuyển đổi nhiều biến thành một tuple.
* **Sử dụng trong hàm yêu cầu các đối số không thay đổi:** trong một số trường hợp, khi muốn đảm bảo rằng các đối số truyền vào hàm không thể thay đổi có thể sử dụng tuple thay vì list.
* **Dùng trong các tình huống yêu cầu hiệu suất cao:** tuple có thể được sử dụng thay cho list trong các tình huống yêu cầu hiệu suất cao, như các phép toán hoặc xử lý dữ liệu yêu cầu tốc độ.