**1.Phân biệt toán từ 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 định dạng chuỗi theo kiểu cũ. Bạn có thể sử dụng nó để chèn giá trị vào một chuỗi định dạng trước.

Cú pháp: "%<định\_dạng>" % (giá trị)

Ưu điểm:

- Đơn giản và dễ sử dụng cho các trường hợp định dạng cơ bản.

Nhược điểm:

- Khó đọc và bảo trì khi có nhiều giá trị cần định dạng.

- Ít linh hoạt hơn so với các phương pháp khác.

\*Hàm định dạng chuỗi Hàm định dạng chuỗi là phương pháp định dạng chuỗi hiện đại hơn.

Nó cho phép bạn chèn giá trị vào chuỗi bằng cách sử dụng các dấu ngoặc nhọn.

Cú pháp: "{<định\_dạng>}".format(giá trị)

Ưu điểm:

- Dễ đọc và dễ bảo trì hơn, đặc biệt khi có nhiều giá trị cần định dạng.

- Cho phép định dạng các đối tượng phức tạp hơn, và hỗ trợ sử dụng chỉ số hoặc tên để định dạng.

Nhược điểm:

- Có thể phức tạp hơn đối với các trường hợp đơn giản so với toán tử .

5 ví dụ đơn giản và dễ hiểu về việc sử dụng toán tử định dạng chuỗi và hàm định dạng có sẵn trong gói thư viện chuẩn

**Ví dụ 1: định dạng chuỗi**

name = "John"

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

print(formatted\_string)

**Kết quả:**Hello, John!

**Ví dụ 2: Định dạng số nguyên**

apples = 3

formatted\_string = "I have %d apples." % apples

print(formatted\_string)

**Kết quả:** I have 3 apples.

**Ví dụ 3: Định dạng số thực**

price = 19.99

formatted\_string = "The price is $%f." % price

print(formatted\_string)

**Kết quả:** The price is $19.990000.

**Ví dụ 4: Định dạng số thực với 2 chữ số thập phân**

price = 19.99

formatted\_string = "The price is $%.2f." % price

print(formatted\_string)

**Kết quả:**The price is $19.99.

**Ví dụ 5: Kết hợp nhiều định dạng**

name = "Anna"

age = 28

formatted\_string = "My name is %s and I am %d years old." % (name, age)

print(formatted\_string)

**Kết quả:**My name is Anna and I am 28 years old.

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

import random

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

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

print("Số ngẫu nhiên là:", random.randint(a, b - 1))

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

1.Tính bất biến:

* List: Làm việc với danh sách, bạn có thể thay đổi các phần tử của nó sau khi đã tạo ra. Ví dụ, bạn có thể thêm, xóa hoặc thay đổi các phần tử trong danh sách.
* Tuple: Làm việc với tuple, bạn không thể thay đổi các phần tử sau khi đã tạo ra. Tuple là bất biến, nghĩa là nó không thể bị thay đổi.

2.Cú pháp:

* List: Được khai báo bằng dấu ngoặc vuông []. Ví dụ: my\_list = [1, 2, 3].
* Tuple: Được khai báo bằng dấu ngoặc đơn (). Ví dụ: my\_tuple = (1, 2, 3).

3.Hiệu suất:

* List: Có thể kém hiệu quả hơn tuple vì tính linh hoạt của nó, như việc thay đổi kích thước và giá trị.
* Tuple: Thường có hiệu suất tốt hơn so với list khi không cần thay đổi nội dung, vì tuple là bất biến và có cấu trúc đơn giản hơn.

4.Sử dụng:

* List: Thường được sử dụng khi bạn cần một cấu trúc dữ liệu có thể thay đổi.
* Tuple: Thường được sử dụng khi bạn cần một tập hợp các giá trị không thay đổi, hoặc khi bạn muốn nhóm các giá trị có ý nghĩa liên quan.

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

* Lưu trữ dữ liệu không thay đổi: Tuple thường được sử dụng để lưu trữ dữ liệu mà bạn không muốn thay đổi trong suốt quá trình chạy chương trình, ví dụ như tọa độ địa lý (latitude, longitude).
* Khóa trong từ điển: Tuple có thể được sử dụng làm khóa trong từ điển vì chúng không thể thay đổi. \* Trả về nhiều giá trị từ hàm: Tuple thường được sử dụng để trả về nhiều giá trị từ một hàm.