Hoàng Nguyên Phong -23692131

1) Trong Python, có nhiều cách để định dạng chuỗi, bao gồm sử dụng toán tử định dạng chuỗi và các hàm định dạng chuỗi có sẵn. Dưới đây là sự khác biệt giữa chúng:

1. Toán tử định dạng chuỗi (%)

Toán tử % được sử dụng để định dạng chuỗi bằng cách thay thế các ký tự đặc biệt trong chuỗi bằng các giá trị cụ thể.

2. Hàm định dạng chuỗi (str.format())

Hàm str.format() là một phương thức của đối tượng chuỗi, cho phép định dạng chuỗi bằng cách sử dụng các dấu ngoặc nhọn {} để chèn các giá trị.

Hàm format() cung cấp nhiều tùy chọn hơn và linh hoạt hơn so với toán tử %.

3. F-string (Python 3.6+)

F-string là một cách định dạng chuỗi mới được giới thiệu trong Python 3.6, sử dụng ký tự f trước chuỗi và các dấu ngoặc nhọn {} để chèn các giá trị trực tiếp.

F-string được coi là cách định dạng chuỗi dễ đọc và hiệu quả nhất trong Python hiện nay.

**\*\*\* 5 ví dụ minh họa tương ứng**

# Ví dụ 1: Định dạng chuỗi với %s và %d

name = "Alice"

age = 30

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

print(formatted\_string)

# Ví dụ 2: Định dạng chuỗi với str.format()

name = "Bob"

age = 25

formatted\_string = "My name is {} and I am {} years old.".format(name, age)

print(formatted\_string)

# Ví dụ 3: Định dạng chuỗi với F-string

name = "Charlie"

age = 35

formatted\_string = f"My name is {name} and I am {age} years old."

print(formatted\_string)

# Ví dụ 4: Định dạng chuỗi với nhiều biến

first\_name = "David"

last\_name = "Smith"

age = 40

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

print(formatted\_string)

# Ví dụ 5: Định dạng chuỗi với định dạng số

price = 49.99

quantity = 3

total = price \* quantity

formatted\_string = "The price is ${:.2f} and the total cost for {} items is ${:.2f}.".format(price, quantity, total)

print(formatted\_string)

2) Để viết một chương trình xuất ra số ngẫu nhiên trong một đoạn bất kỳ cho trước, bạn có thể sử dụng thư viện random trong Python. Dưới đây là một ví dụ minh họa:

import random

def generate\_random\_number(start, end):

return random.randint(start, end)

# Đoạn bất kỳ cho trước

start = 1

end = 100

# Xuất ra số ngẫu nhiên trong đoạn [start, end]

random\_number = generate\_random\_number(start, end)

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

Trong chương trình này:

* Thư viện random được sử dụng để tạo ra số ngẫu nhiên.
* Hàm random.randint(start, end) trả về một số nguyên ngẫu nhiên trong đoạn từ start đến end (bao gồm cả start và end).
* Hàm generate\_random\_number(start, end) được định nghĩa để dễ dàng tái sử dụng.

3) List và tuple là hai kiểu dữ liệu tập hợp (collection) trong Python, nhưng chúng có một số khác biệt cơ bản:

### 1. Tính thay đổi (Mutability)

* **List**: Có thể thay đổi (mutable). Bạn có thể thêm, xóa hoặc sửa đổi các phần tử trong list sau khi nó được tạo.
* **Tuple**: Không thể thay đổi (immutable). Sau khi một tuple được tạo, bạn không thể thay đổi các phần tử của nó.

### 2. Cú pháp

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

### 3. Hiệu suất

* **List**: Do có thể thay đổi, list thường có hiệu suất chậm hơn tuple khi thực hiện các thao tác như duyệt qua các phần tử.
* **Tuple**: Do không thể thay đổi, tuple thường có hiệu suất nhanh hơn list trong các thao tác tương tự.

### 4. Sử dụng

* **List**: Thích hợp cho các tập hợp dữ liệu cần thay đổi hoặc cập nhật thường xuyên.
* **Tuple**: Thích hợp cho các tập hợp dữ liệu cố định, không cần thay đổi, như các hằng số.

### 5. Các phương thức hỗ trợ

* **List**: Có nhiều phương thức hỗ trợ như append(), remove(), pop(), sort(), v.v.
* my\_list.append(4)
* my\_list.remove(2)
* **Tuple**: Có ít phương thức hơn, chủ yếu là count() và index().
* my\_tuple.count(1)
* my\_tuple.index(2)

4) Tuple là một kiểu dữ liệu rất hữu ích trong Python, với nhiều ứng dụng trong thực tế nhờ vào tính chất bất biến (immutable) và khả năng chứa nhiều loại dữ liệu khác nhau. Dưới đây là một số ứng dụng thực tế phổ biến của kiểu dữ liệu tuple:

1. Lưu trữ dữ liệu không thay đổi

Tuple thường được sử dụng để lưu trữ các dữ liệu không cần thay đổi sau khi tạo ra.

2. Trả nhiều giá trị từ một hàm

Tuple cho phép hàm trả về nhiều giá trị cùng một lúc, giúp làm cho mã nguồn dễ đọc và tổ chức hơn:

3. Sử dụng làm khóa trong từ điển

Vì tuple là bất biến, chúng có thể được sử dụng làm khóa trong từ điển, trong khi list thì không:

4. Lưu trữ các cấu trúc dữ liệu cố định

Tuple rất hữu ích khi bạn cần lưu trữ dữ liệu có cấu trúc cố định:

- Thay thế cho các lớp dữ liệu đơn giản:

5. Nhóm các đối tượng có liên quan

Tuple có thể được sử dụng để nhóm các đối tượng liên quan lại với nhau:

6. Đóng gói và giải nén

Tuple cho phép đóng gói nhiều giá trị vào một đối tượng và sau đó giải nén chúng dễ dàng:

7. Sử dụng với các cấu trúc dữ liệu có sẵn

Sắp xếp danh sách dựa trên nhiều tiêu chí