***Phân biệt Toán tử Định dạng Chuỗi và Hàm Định dạng Chuỗi trong gói thư viện chuẩn Python?***

***Toán tử Định dạng Chuỗi (f-strings)***

**Đặc điểm:**

+ Được giới thiệu từ Python 3.6

+ Cú pháp trực quan, đặt chữ f trước chuỗi.

+ Thay thế giá trị biến trực tiếp trong chuỗi bằng dấu ngoặc nhọn {}.

+ Hỗ trợ các định dạng phức tạp như căn chỉnh, độ rộng, kiểu số, v.v.

**Ví dụ:**

name = "Alice"  
age = 30  
print(f"Hello, my name is {name} and I am {age} years old.")

Hàm định dạng chuỗi có sẵn

***Phương thức format()***

+ Cú pháp: chuỗi.format(arg1, arg2, ...)

+ Thay thế các dấu ngoặc nhọn {} bằng các đối số truyền vào.

+ Vị trí của các đối số tương ứng với thứ tự của dấu ngoặc nhọn.

**Ví dụ:**

name = "Bob"  
price = 19.99  
print("Hello, {0}. The price is ${1:.2f}".format(name, price))

***5 Ví Dụ*** :

**f-strings:**

user = {"name": "Charlie", "age": 25}  
print(f"User: {user['name']}, Age: {user['age']}")

**format():**

print("User: {name}, Age: {age}".format(name=user['name'], age=user['age']))

**Định dạng số:**

**f-strings:**

Python

pi = 3.14159  
print(f"Pi: {pi:.2f}") # In 2 chữ số thập phân

**format():**

print("Pi: {:.2f}".format(pi))

**Căn chỉnh:**

**f-strings:**

number = 123  
print(f"Number: {number:>10}") # Căn phải trong trường rộng 10 ký tự

**format():**

print("Number: {:>10}".format(number))

**Định dạng ngày tháng:**

**f-strings:**

import datetime today = datetime.date.today() print(f"Today: {today:%B %d, %Y}")

**format():**

print("Today: {:%B %d, %Y}".format(today))

**Nhúng biểu thức:**

**f-strings:**

Python

x = 10  
y = 5  
print(f"Tổng: {x + y}")

**format():**

Python

print("Tổng: {}".format(x + y))

Tuple trong Python là một kiểu dữ liệu vô cùng hữu ích, đặc biệt khi bạn cần một cấu trúc dữ liệu không thể thay đổi. Dưới đây là một số ứng dụng điển hình của tuple trong thực tế: 1. Lưu trữ dữ liệu không đổi: Thông tin về người: (tên, tuổi, địa chỉ) Tọa độ: (x, y, z) Màu sắc: (red, green, blue) Ngày tháng: (ngày, tháng, năm) Hằng số: (PI, e) 2. Làm khóa trong dictionary: Vì tuple là bất biến, nên chúng có thể được sử dụng làm khóa trong dictionary. Điều này rất hữu ích khi bạn muốn ánh xạ một tập hợp các giá trị không đổi với một giá trị khác. 3. Trả về nhiều giá trị từ một hàm: Một hàm có thể trả về nhiều giá trị dưới dạng một tuple. 4. Định nghĩa các cấu trúc dữ liệu đơn giản: Tuple có thể được sử dụng để định nghĩa các cấu trúc dữ liệu đơn giản như điểm, vector, phân số, v.v. 5. Làm đối số cho các hàm: Tuple có thể được sử dụng để truyền nhiều đối số cho một hàm. 6. Tăng tốc độ thực thi: Vì tuple là bất biến nên Python có thể tối ưu hóa việc truy cập và sử dụng chúng, dẫn đến hiệu suất tốt hơn so với list trong một số trường hợp. 7. Đảm bảo tính toàn vẹn dữ liệu: Bằng cách sử dụng tuple, bạn đảm bảo rằng dữ liệu sẽ không bị thay đổi vô tình, giúp tăng tính ổn định của chương trình.