**Huỳnh Diệu Mỹ - DHTMDT19D**

# 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 (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 cú pháp {biến}.
  + Hỗ trợ định dạng phức tạp như làm tròn, căn chỉnh, định dạng ngày tháng.
* **Ưu điểm:**
  + Dễ đọc, dễ viết.
  + Hiệu năng cao hơn so với các phương pháp cũ.
  + Hỗ trợ nhiều tính năng định dạng.

**Hàm Định dạng Chuỗi format()**

* **Đặc điểm:**
* Phương pháp định dạng chuỗi truyền thống.
* Sử dụng dấu ngoặc nhọn {} như các placeholder trong chuỗi.
* Truyền giá trị vào hàm format() theo thứ tự hoặc bằng cách đặt tên.
* **Ưu điểm:**
  + Linh hoạt trong việc sắp xếp các giá trị.

### So sánh và Ví dụ Minh Họa

|  |  |  |
| --- | --- | --- |
| **Tính năng** | **Toán tử f-strings** | **Hàm format()** |
| Cú pháp | f"chuỗi {biến}" | "chuỗi {}.format(biến)" |
| Đọc hiểu | Dễ đọc, trực quan | Cần hiểu rõ về vị trí của placeholder |
| Hiệu năng | Cao hơn | Thấp hơn |
| Định dạng phức tạp | Hỗ trợ tốt | Hỗ trợ, nhưng phức tạp hơn |

* **f-strings** là lựa chọn ưu tiên cho việc định dạng chuỗi trong Python hiện đại nhờ cú pháp đơn giản, hiệu năng cao và khả năng định dạng đa dạng.
* **Hàm format()** vẫn có thể sử dụng trong các trường hợp cần độ linh hoạt cao hoặc tương thích với các phiên bản Python cũ hơn.

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

import random

print(random.choice(range(100)))

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

**List** và **tuple** là hai kiểu dữ liệu được sử dụng để lưu trữ các phần tử có thứ tự trong Python. Tuy nhiên, chúng có những đặc điểm khác biệt quan trọng:

### 1. ****Tính biến đổi (mutability):****

* **List:** Có thể thay đổi sau khi được tạo. Bạn có thể thêm, xóa hoặc sửa đổi các phần tử trong một list.
* **Tuple:** Không thể thay đổi sau khi được tạo. Các phần tử của một tuple được cố định.

### 2. ****Cú pháp:****

* **List:** Được bao quanh bởi dấu ngoặc vuông [].
* **Tuple:** Được bao quanh bởi dấu ngoặc tròn ().

### 3. ****Sử dụng:****

* **List:** Thường được sử dụng khi bạn cần một cấu trúc dữ liệu có thể thay đổi linh hoạt. Ví dụ: lưu trữ một danh sách các số, một danh sách các từ, hoặc một danh sách các đối tượng.
* **Tuple:** Thường được sử dụng khi bạn cần một cấu trúc dữ liệu không đổi, như các hằng số, các khóa trong dictionary, hoặc khi bạn muốn đảm bảo rằng dữ liệu không bị thay đổi vô tình.

### Bảng so sánh:

|  |  |  |
| --- | --- | --- |
| **Tính năng** | **List** | **Tuple** |
| Tính biến đổi | Có thể thay đổi | Không thể thay đổi |
| Cú pháp | [] | () |
| Sử dụng | Danh sách động, có thể thay đổi | Danh sách tĩnh, không thể thay đổi |

# 4. Ứng dụng kiểu dữ liệu tuple trong thực tế?

**1. Lưu trữ dữ liệu không đổi:**

* **Thông tin cá nhân:** (tên, tuổi, ngày sinh) - Dữ liệu này thường không thay đổi thường xuyên.
* **Tọa độ:** (x, y) - Một điểm trên mặt phẳng có tọa độ cố định.
* **Màu sắc:** (R, G, B) - Giá trị màu sắc RGB không thay đổi.

**2. Làm khóa trong dictionary:**

* **Vì tuple là bất biến**, chúng có thể được sử dụng làm khóa trong dictionary. Điều này là rất hữu ích khi bạn muốn ánh xạ nhiều giá trị với một khóa phức tạp.

**3. Trả về nhiều giá trị từ hàm:**

* **Hàm có thể trả về nhiều giá trị dưới dạng một tuple.** Điều này tiện lợi khi bạn muốn trả về nhiều kết quả khác nhau từ một hàm.

**4. Định nghĩa các hằng số:**

* **Tuple có thể được sử dụng để định nghĩa các hằng số.** Việc sử dụng tuple giúp đảm bảo rằng các giá trị này không bị thay đổi vô tình.

**5. Làm tham số cho hàm:**

* **Tuple có thể được sử dụng làm tham số cho hàm.** Điều này giúp đảm bảo rằng hàm không thể thay đổi các giá trị được truyền vào.

**6. Lưu trữ dữ liệu cấu trúc:**

* **Tuple có thể được sử dụng để lưu trữ dữ liệu có cấu trúc.** Ví dụ: một danh sách các sinh viên có thể được biểu diễn dưới dạng một list của các tuple, mỗi tuple chứa thông tin về một sinh viên.

**7. Tăng tốc độ:**

* **Tuple thường nhanh hơn list khi truy cập các phần tử.** Điều này là do tuple là bất biến, nên Python có thể tối ưu hóa việc truy xuất dữ liệu từ tuple.