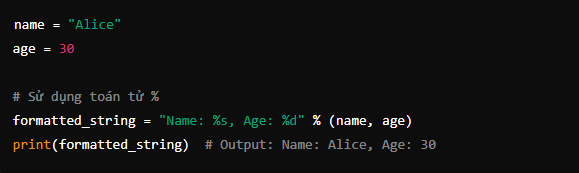
**Câu 1**

### 1. Toán tử định dạng chuỗi (%)

Toán tử định dạng chuỗi (%) là phương pháp định dạng chuỗi cổ điển trong Python. Nó cho phép bạn chèn giá trị vào chuỗi bằng cách sử dụng ký tự đặc biệt (%) để đánh dấu vị trí của giá trị.

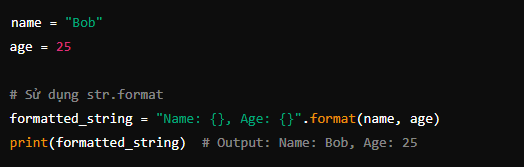
**Ví dụ:**

****

### 2. Hàm định dạng chuỗi (str.format)

Hàm str.format là phương pháp định dạng chuỗi hiện đại hơn, cung cấp nhiều tính năng linh hoạt và mạnh mẽ. Bạn có thể sử dụng các chỉ định vị trí và tên để làm cho chuỗi dễ đọc và bảo trì hơn.

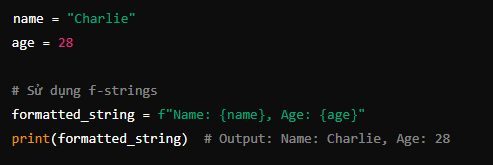
**Ví dụ:**

****

### 3. F-strings (Literal String Interpolation)

F-strings (chuỗi định dạng tĩnh) là một cách mới hơn và rất mạnh mẽ để định dạng chuỗi, được giới thiệu trong Python 3.6. F-strings cho phép chèn biểu thức trực tiếp vào chuỗi bằng cách đặt chúng trong dấu ngoặc nhọn {}.

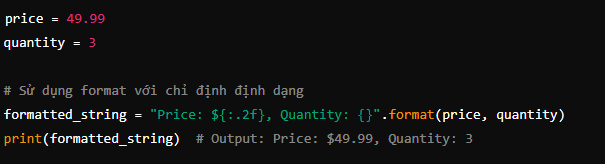
**Ví dụ:**

****

### 4. Hàm format của chuỗi (Phương thức format)

Ngoài việc sử dụng hàm str.format, bạn cũng có thể sử dụng các tham số và chỉ định định dạng cụ thể trong phương thức format.

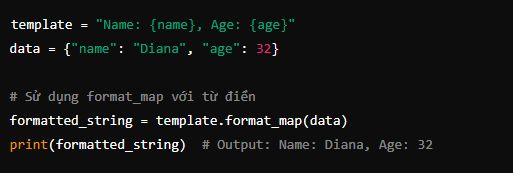
**Ví dụ:**

****

### 5. Định dạng chuỗi với format\_map

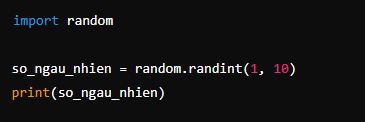
format\_map là một phương thức của chuỗi, tương tự như str.format, nhưng sử dụng một đối tượng ánh xạ như từ điển để thay thế các tham số.

**Ví dụ:**

****

**Câu 2.**

**Sinh số nguyên ngẫu nhiên trong khoảng từ 1 đến 10:**



**Câu 3.**

|  |  |  |
| --- | --- | --- |
| **TÍNH CHẤT** | **LIST** | **TUPLE** |
| **Có thể thay đổi** | Có | Không |
| **Sử dụng** | Lưu trữ dữ liệu có thể thay đổi, các phép toán trên danh sách | Lưu trữ dữ liệu không thay đổi, các hằng số, khóa trong dictionary |
| Thực hiện các phép toán trên danh sách như sắp xếp, tìm kiếm, cắt ghép. | Bảo vệ dữ liệu khỏi bị thay đổi vô tình. |
| **Ký hiệu** | [] | () |
| **Ví dụ** | [1, 2, 3] | (1, 2, 3) |

**Khi nào nên sử dụng list và tuple?**

- Sử dụng list khi:

* Cần một cấu trúc dữ liệu linh hoạt, có thể thay đổi.
* Cần thực hiện nhiều phép toán trên dữ liệu.

- Sử dụng tuple khi:

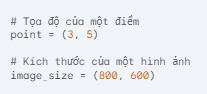
* Cần một cấu trúc dữ liệu không đổi để bảo vệ dữ liệu.
* Muốn sử dụng làm khóa trong dictionary.
* Muốn tăng tốc độ truy xuất dữ liệu (tuple thường nhanh hơn list).

**Câu 4.**

Tuple trong Python là một kiểu dữ liệu rất hữu ích và có nhiều ứng dụng thực tế khác nhau. Dưới đây là một số ví dụ cụ thể:

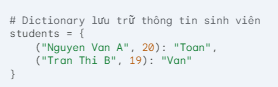
**1. Lưu trữ các cặp giá trị không đổi**

* Tọa độ: (x, y) đại diện cho một điểm trên mặt phẳng.
* Kích thước: (width, height) biểu diễn kích thước của một hình ảnh hoặc một cửa sổ.
* Màu sắc**:** (R, G, B) đại diện cho một màu trong hệ màu RGB.



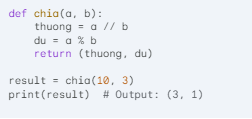
### 2. ****Làm khóa trong dictionary****

Vì tuple là bất biến, nên nó có thể được sử dụng làm khóa trong một dictionary. Điều này rất hữu ích khi bạn muốn ánh xạ nhiều giá trị với một cặp khóa duy nhất



### 3. ****Trả về nhiều giá trị từ một hàm****

Một hàm trong Python chỉ có thể trả về một giá trị. Tuy nhiên, bạn có thể trả về một tuple để thực hiện việc này.



### 4. ****Định nghĩa các hằng số****

Vì tuple không thể thay đổi, nên nó thường được sử dụng để định nghĩa các hằng số trong chương trình.



**5. Lưu trữ dữ liệu cấu trúc**

Tuple có thể được sử dụng để lưu trữ dữ liệu có cấu trúc, chẳng hạn như:

* Thông tin về người dùng: (tên, tuổi, địa chỉ)
* Thông tin về sản phẩm: (tên, giá, mô tả)

**6. Truyền nhiều đối số cho một hàm**

Bạn có thể sử dụng tuple để đóng gói nhiều đối số và truyền chúng cho một hàm.

