**1. Phân biệt toán tử định dạng chuỗi và hàm định dạng chuỗi trong Python**

**Toán tử định dạng chuỗi (f-string)** là một cách mới và hiệu quả để nhúng các biểu thức Python vào trong chuỗi. Nó được giới thiệu từ Python 3.6 trở đi và được đánh giá cao về tính đọc và hiệu năng.

**Hàm định dạng chuỗi format()** là một phương thức của đối tượng chuỗi, cho phép bạn định dạng các giá trị bằng cách chèn chúng vào các vị trí giữ chỗ trong chuỗi.

**Ví dụ minh họa:**

|  |  |
| --- | --- |
| **f-string** | **Hàm format()** |
| name = "Alice" <br> age = 30 <br> print(f"Hello, my name is {name} and I am {age} years old.") | name = "Alice" <br> age = 30 <br> print("Hello, my name is {} and I am {} years old.".format(name, age)) |
| pi = 3.14159 <br> print(f"The value of pi is approximately {pi:.2f}") | pi = 3.14159 <br> print("The value of pi is approximately {:.2f}".format(pi)) |
| x = 10 <br> y = 20 <br> print(f"The sum of {x} and {y} is {x+y}") | x = 10 <br> y = 20 <br> print("The sum of {} and {} is {}".format(x, y, x+y)) |
| person = {'name': 'Bob', 'age': 25} <br> print(f"His name is {person['name']} and he is {person['age']} years old.") | person = {'name': 'Bob', 'age': 25} <br> print("His name is {name} and he is {age} years old.".format(name=person['name'], age=person['age'])) |
| list1 = [1, 2, 3] <br> print(f"The first element is {list1[0]}") | list1 = [1, 2, 3] <br> print("The first element is {}".format(list1[0])) |

**Ưu điểm của f-string:**

**Dễ đọc hơn:** Cú pháp trực quan và gần gũi với cách viết bình thường.

**Hiệu năng cao hơn:** Nhanh hơn so với format() trong nhiều trường hợp.

**Tính năng phong phú:** Hỗ trợ các biểu thức phức tạp, định dạng số, căn chỉnh, v.v.

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

+) Hiểu rõ yêu cầu

Số ngẫu nhiên: Một số được chọn một cách ngẫu nhiên từ một tập hợp các số.

Đoạn bất kỳ: Một khoảng giá trị xác định, ví dụ: từ 1 đến 100.

Sử dụng thư viện random trong Python

Python cung cấp thư viện random để tạo các số ngẫu nhiên. Hàm randint(a, b) sẽ trả về một số nguyên ngẫu nhiên nằm trong khoảng từ a đến b (bao gồm cả a và b).

Ví dụ chương trình:

Python

import random

def random\_number(min, max):

"""Trả về một số ngẫu nhiên trong khoảng từ min đến max (bao gồm cả min và max)."""

return random.randint(min, max)

min\_value = int(input("Nhập giá trị nhỏ nhất: "))

max\_value = int(input("Nhập giá trị lớn nhất: "))

random\_num = random\_number(min\_value, max\_value)

print("Số ngẫu nhiên:", random\_num)

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

**Tính bất biến:**

**List:** Có thể thay đổi các phần tử sau khi tạo.

**Tuple:** Không thể thay đổi các phần tử sau khi tạo.

**\*)Sử dụng:**

**List:** Thường được dùng để lưu trữ một tập hợp các phần tử có thể thay đổi.

**Tuple:** Thường được dùng để lưu trữ các dữ liệu không thay đổi, như các hằng số, các cặp khóa-giá trị.

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

Ứng dụng của kiểu dữ liệu tuple trong thực tế

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

1. Lưu trữ dữ liệu không đổi:

Tọa độ: Một điểm trên mặt phẳng có thể được biểu diễn bằng một tuple (x, y).

Kích thước: Kích thước của một hình ảnh hoặc màn hình có thể được biểu diễn bằng một tuple (width, height).

Màu sắc: Màu sắc thường được biểu diễn dưới dạng một tuple (R, G, B) với các giá trị tương ứng cho màu đỏ, xanh lá và xanh dương.

2. Làm tham số cho hàm:

Truyền nhiều giá trị: Bạn có thể truyền nhiều giá trị cho một hàm bằng cách sử dụng một tuple. Ví dụ:

Python

def tinh\_dien\_tich\_hinh\_chu\_nhat(chieu\_dai, chieu\_rong):

return chieu\_dai \* chieu\_rong

kich\_thuoc = (5, 3)

dien\_tich = tinh\_dien\_tich\_hinh\_chu\_nhat(\*kich\_thuoc)

Đảm bảo tính không đổi của dữ liệu: Khi truyền một tuple làm tham số, bạn đảm bảo rằng hàm không thể vô tình thay đổi giá trị của các phần tử trong tuple.

3. Trả về nhiều giá trị từ hàm:

Hàm tìm giá trị lớn nhất và nhỏ nhất:

Python

def tim\_gia\_tri\_lon\_nhat\_va\_nho\_nhat(numbers):

return min(numbers), max(numbers)

Hàm này trả về một tuple chứa giá trị nhỏ nhất và lớn nhất trong danh sách.

4. Làm khóa cho dictionary:

Vì tuple là immutable nên chúng có thể được sử dụng làm khóa cho dictionary. Điều này rất hữu ích khi bạn muốn ánh xạ các giá trị với các cặp khóa phức tạp.

Python

person = {

('Alice', 30): {'city': 'New York'},

('Bob', 25): {'city': 'Los Angeles'}

}

5. Các ứng dụng khác:

Định nghĩa các cấu trúc dữ liệu đơn giản: Tuple có thể được sử dụng để tạo các cấu trúc dữ liệu đơn giản như điểm, khoảng thời gian, ngày tháng.

Trả về kết quả từ các truy vấn cơ sở dữ liệu: Nhiều hệ thống cơ sở dữ liệu trả về các kết quả truy vấn dưới dạng các tuple.

Xử lý các dữ liệu không đồng nhất: Tuple có thể chứa các phần tử có kiểu dữ liệu khác nhau, điều này rất hữu ích khi bạn cần lưu trữ các thông tin liên quan nhưng có kiểu dữ liệu khác nhau.

Ưu điểm của tuple:

Hiệu suất: Tuple thường nhanh hơn list vì chúng là immutable.

An toàn: Việc tuple không thể thay đổi giúp tránh được các lỗi không mong muốn khi làm việc với dữ liệu.

Đọc được: Tuple làm cho code dễ đọc hơn vì chúng biểu diễn các giá trị liên quan một cách rõ ràng.