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?

Trong Python, có nhiều cách để định dạng chuỗi (string formatting), nhưng hai cách phổ biến nhất là sử dụng ****toán tử %**** và ****phương thức format()****. Mỗi cách có ưu nhược điểm riêng và phù hợp với các trường hợp sử dụng khác nhau.

### **Toán tử % (Old-style formatting)**

****- Cú pháp:**** chuỗi % (giá trị)

****- Ưu điểm:**** Đơn giản, dễ hiểu

****- Nhược điểm:**** Cú pháp hơi rườm rà, ít linh hoạt so với phương thức format()

****Ví dụ:****

Python

name = "Alice"

age = 30

print("Hello, my name is %s and I am %d years old." % (name, age))

### **Phương thức format() (New-style formatting)**

****- Cú pháp:**** chuỗi.format(giá trị1, giá trị2, ...)

****- Ưu điểm:**** Linh hoạt, dễ đọc, hỗ trợ nhiều tính năng hơn

****- Nhược điểm:**** Cú pháp hơi dài dòng trong một số trường hợp đơn giản

****Ví dụ:****

Python

name = "Bob"

price = 19.99

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

### **So sánh và ví dụ minh họa chi tiết**

|  |  |  |
| --- | --- | --- |
| Tính năng | Toán tử % | Phương thức format() |
| Cú pháp | chuỗi % (giá trị) | chuỗi.format(giá trị1, giá trị2, ...) |
| Linh hoạt | Ít | Cao |
| Đọc hiểu | Dễ (đối với người mới bắt đầu) | Dễ (khi quen) |
| Tính năng bổ sung | Hạn chế | Nhiều (định dạng số, căn chỉnh, ...) |

****Ví dụ minh họa:****

****- Định dạng đơn giản:****

Toán tử %: print("Giá trị là: %d" % 42)

format(): print("Giá trị là: {}".format(42))

****- Định dạng nhiều giá trị:****

Toán tử %: print("Tên: %s, Tuổi: %d" % ("Alice", 30))

format(): print("Tên: {0}, Tuổi: {1}".format("Alice", 30))

****- Định dạng số:****

Toán tử %: print("Pi = %.2f" % 3.14159)

format(): print("Pi = {:.2f}".format(3.14159))

****- Căn chỉnh:****

format(): print("{:<10} | {:>10}".format("Trái", "Phải"))

****- Định dạng ngày tháng:****

format(): import datetime; print("Hôm nay là: {:%Y-%m %d}".format(datetime.datetime.now()))

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

trước?

### **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)

1. Khác biệt cơ bản giữa list và tuple?

****List**** và ****tuple**** là hai kiểu dữ liệu được sử dụng để lưu trữ một tập hợp các giá trị trong Python. Tuy nhiên, chúng có những đặc điểm khác biệt quan trọng:

### **1. Khả năng thay đổi (mutability):**

****- List:**** Có thể thay đổi sau khi được tạo. Bạn có thể thêm, xóa, sửa các phần tử trong một list.

****- Tuple:**** Không thể thay đổi sau khi được tạo. Các phần tử của tuple là cố định.

**2. Cách khai báo:**

****- List:**** Được bao quanh bởi dấu ngoặc vuông [].

Python

my\_list = [1, 2, 3, "apple", "banana"]

****- Tuple:**** Được bao quanh bởi dấu ngoặc tròn ().

Python

my\_tuple = (1, 2, 3, "apple", "banana")

### **3. Sử dụng:**

****- List:****

+ Lưu trữ các phần tử có thể thay đổi theo thời gian.

+ Làm việc với các cấu trúc dữ liệu động.

+ Thường được sử dụng trong các vòng lặp, phép toán trên các phần tử.

****- Tuple:****

+ Lưu trữ các giá trị không đổi, như các hằng số.

+ Làm khóa cho dictionary (vì tuple là immutable).

+ Trả về nhiều giá trị từ một hàm.

+ Đảm bảo tính toàn vẹn của dữ liệu.

### **4. Hiệu suất:**

****Tuple**** thường nhanh hơn list một chút trong các phép toán vì chúng là immutable. Tuy nhiên, trong hầu hết các trường hợp, sự khác biệt về hiệu suất không đáng kể.

### **Ví dụ minh họa:**

Python

fruits = ["apple", "banana", "cherry"]

fruits.append("orange") print(fruits)

coordinates = (10, 20)# coordinates.append(30)

print(coordinates)

****Bạn có muốn tìm hiểu thêm về các ứng dụng cụ thể của list và tuple không?****

4. Ứng dụng 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.

****- 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:****

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.

**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.