**Câu 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, toán tử định dạng chuỗi và hàm định dạng chuỗi đều được sử dụng để tạo ra các chuỗi có định dạng đặc biệt, nhưng chúng hoạt động theo cách khác nhau.

Toán tử định dạng chuỗi (%): Toán tử % là một cách cũ để định dạng chuỗi. Bạn sử dụng nó để chèn các giá trị vào chuỗi với các định dạng cụ thể.

Ví dụ:

python

VD1:

ten = "kiet"

tuoi = 18

formatted\_str = "ten: %s, tuoi: %d" % (ten, tuoi)

print(formatted\_str)

Kết quả: ten: kiet, tuoi: 18

VD2:

pi = 3.14159

formatted\_str = "Pi xấp xỉ %.2f" % pi

print(formatted\_str)

Output: Pi xấp xỉ 3.14

VD3:

Mặt hàng = 7

Chi phí = 32

formatted\_str = "Bạn có %d mặt hàng có chi phí là $%.2f." % (Mặt hàng, Chi phí)

print(formatted\_str)

Kết quả: Bạn có 7 mặt hàng có chi phí là 32$

VD4:

tên = "kiet"

Chiều cao = 173

formatted\_str = "Chiều cao của %d là %.1f cm." % (tên, chiều cao)

print(formatted\_str)

Kết quả: Chiều cao của kiet là 173 cm.

VD5:

Đáp án = 4 / 2

formatted\_str = "Đáp án của phép chia 4 với 2 là " % result

print(formatted\_str)

Kết quả: Đáp án của phep chia 4 với 2 là 2

Hàm định dạng chuỗi (`str.format()`):

Hàm `str.format()` cung cấp một cách hiện đại và linh hoạt để định dạng chuỗi. Bạn có thể sử dụng các dấu ngoặc nhọn `{}` và truyền các giá trị vào chúng.

*So sánh:*

- Toán tử `%`: Cách cũ, ít linh hoạt hơn và khó bảo trì hơn.

- Hàm `str.format()`: Cung cấp nhiều tính năng hơn và khả năng định dạng mạnh mẽ hơn.

Lưu ý: Python 3.6 trở đi hỗ trợ định dạng chuỗi bằng f-strings, cung cấp một cách thức đơn giản và trực quan hơn để định dạng chuỗi.

**Câu 2 : Viết chương trình xuất ra số ngẫu nhiên trong một đoạn bất kỳ bất cho trước?**

import random

# Nhập giá trị thấp nhất và cao nhất của đoạn a,b

a = int(input("Nhập giá trị thấp nhất: "))

b = int(input("Nhập giá trị cao nhất: "))

# Chọn số ngẫu nhiên trong đoạn

ngaunhien = random.randint(a, b)

print(f"Số ngẫu nhiên trong đoạn [{a}, {b}] là: {ngaunhien }")

**Câu 3 :Khác biệt cơ bản giữa list và tuple?**

Khác biệt cơ bản giữa list và tuple là list có thể thay đổi (mutable) trong khi tuple không thể thay đổi (immutable). Bạn có thểthêm hoặc xóa các phần tử trong list, nhưng với tuple, các phần tử không thể được thay đổi sau khi tuple được tạo ra.

**Câu 4 : Ứng dụng kiểu dữ liệu tuple trong thực tế**

Tuple là kiểu dữ liệu không thay đổi (immutable) trong Python. Ví dụ:

1. Lưu trữ dữ liệu cố định: Tuple được dùng để lưu trữ các dữ liệu không cần thay đổi sau khi tạo. Ví dụ, các tọa độ địa lý (vĩ độ, kinh độ) có thể được lưu trong một tuple vì chúng không thay đổi.

Python

coordinates = (40.7128, -74.0060) # New York City coordinates

1. Hàm trả về nhiều giá trị:   
   Khi một hàm cần trả về nhiều giá trị, tuple thường được sử dụng để nhóm các giá trị này lại.   
   Python def min\_max(numbers):  
    return (min(numbers), max(numbers))  
    result = min\_max([1, 2, 3, 4, 5]) print(result)   
    Output: (1, 5)
2. Dữ liệu bất biến: Tuple được dùng trong các cấu trúc dữ liệu như dictionary hoặc set, nơi mà các phần tử không thể thay đổi. Điều này giúp đảm bảo tính toàn vẹn của dữ liệu.

python

location = {('Home', 1): 'John', ('Office', 2): 'Jane'}

1. Nhóm các giá trị liên quan: Tuple có thể dùng để nhóm các giá trị liên quan. Ví dụ, bạn có thể sử dụng tuple để nhóm thông tin liên quan đến một người, như tên và tuổi.

python

person = ('kiet', 18)

name, age = person

5. Sắp xếp và so sánh: Tuple hỗ trợ so sánh và sắp xếp dựa trên thứ tự của các phần tử, làm cho chúng hữu ích trong các tình huống cần sắp xếp dữ liệu theo nhiều tiêu chí.

python

data = [(1, 'a'), (3, 'c'), (2, 'b')]

sorted\_data = sorted(data)

print(sorted\_data)

Output: [(1, 'a'), (2, 'b'), (3, 'c')]

Tuple là một công cụ mạnh mẽ trong Python cho những tình huống cần lưu trữ dữ liệu bất biến và nhóm các giá trị liên quan.