**CÂU HỎI TỔNG HỢP**

**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, hai cách phổ biến nhất là sử dụng toán tử định dạng (f-strings) và hàm định dạng (method format()). Mặc dù cả hai đều phục vụ mục đích tương tự, nhưng chúng có cú pháp và cách sử dụng khác nhau.

- Toán tử định dạng chuỗi (f-strings)

Cú pháp: Bắt đầu bằng chữ f trước dấu ngoặc kép hoặc ngoặc đơn, sau đó nhúng các biểu thức Python trực tiếp vào trong chuỗi bằng cách đặt chúng trong cặp dấu ngoặc nhọn {}.

Ví dụ:

name = "thuhang"

age = 19

print(f"Hello, my name is {name} and I am {age} years old.")

- Hàm định dạng chuỗi (method format())

Cú pháp: Gọi phương thức format() trên một chuỗi và truyền các giá trị cần thay thế vào trong đó.

Ví dụ:

name = "banhmi"

price = 15

print("Hello, my name is {} and the price is {:.2f}".format(name, price))

|  |  |  |
| --- | --- | --- |
|  | Toán tử định dạng (f-strings) | Hàm định dạng (format()) |
| Cú pháp | f"chuỗi {biểu\_thức}" | chuỗi.format(giá\_trị1, giá trị2,...) |
| Ví dụ: Định dạng đơn giản | f"Hello, {name}!" | "Hello, {}!".format(name) |
| Ví dụ: Định dạng số | f"Pi = {3.14159:.2f}" | "Pi = {:.2f}".format(3.14159) |
| Ví dụ: Căn chỉnh | f"{name:<10} is a programmer" | "{:<10} is a programmer".format(name) |
| Ví dụ: Biểu thức phức tạp | f"The sum of {x} and {y} is {x+y}" | "The sum of {} and {} is {}".format(x, y, x+y) |
| Ví dụ: Định dạng nhiều lần | Không trực tiếp | "Hello, {0}. You are {1} years old.".format(name, age) |

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

import random

def random\_in\_range(start, end):

"""Trả về một số nguyên ngẫu nhiên trong khoảng [start, end]"""

return random.randint(start, end)

def random\_float\_in\_range(start, end):

"""Trả về một số thực ngẫu nhiên trong khoảng [start, end]"""

return random.uniform(start, end)

def random\_choice(sequence):

"""Trả về một phần tử ngẫu nhiên từ một sequence (danh sách, tuple,...)"""

return random.choice(sequence)

# Ví dụ:

# Số nguyên ngẫu nhiên từ 1 đến 100

random\_number = random\_in\_range(1, 100)

print(random\_number)

# Số thực ngẫu nhiên từ 0 đến 1

random\_float = random\_float\_in\_range(0, 1)

print(random\_float)

# Lấy ngẫu nhiên một chữ cái từ chuỗi

random\_letter = random\_choice('ABCDEFGHIJKLMNOPQRSTUVWXYZ') print(random\_letter)

**3. 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:

- Tính biến đổi (mutability):

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

- Cú pháp:

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

- Sử dụng:

* List: Thường được sử dụng khi bạn cần một cấu trúc dữ liệu có thể thay đổi linh hoạt, như lưu trữ danh sách các sản phẩm, danh sách người dùng, v.v.
* Tuple: Thường được sử dụng khi bạn muốn bảo vệ dữ liệu khỏi bị thay đổi, như làm khóa cho dictionary, trả về nhiều giá trị từ một hàm, hoặc đại diện cho các dữ liệu không thay đổi như tọa độ (x, y).

Ví dụ:

# Tạo một list

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

my\_list[0] = 5 # Thay đổi giá trị phần tử đầu tiên

print(my\_list) # Output: [5, 2, 3, "apple", "banana"]

# Tạo một tuple

my\_tuple = (1, 2, 3) # my\_tuple[0] = 5 # Sẽ gây ra lỗi vì tuple không thể thay đổi

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

- Lưu trữ tọa độ:

Ví dụ: Một điểm trên đồ thị có thể được biểu diễn bằng một tuple (x, y).

point = (3, 4) # Điểm có tọa độ x=3, y=4

- Trả về nhiều giá trị từ hàm:

Ví dụ: Một hàm tính toán diện tích và chu vi của hình chữ nhật có thể trả về cả hai giá trị dưới dạng một tuple.

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

dien\_tich = chieu\_dai \* chieu\_rong

chu\_vi = 2 \* (chieu\_dai + chieu\_rong)

return dien\_tich, chu\_vi

- Đại diện cho các giá trị không đổi:

Ví dụ: Các ngày trong tuần có thể được biểu diễn bằng một tuple:

ngay\_trong\_tuan = ("Thứ Hai", "Thứ Ba", "Thứ Tư", ...)

- Sử dụng trong các cấu trúc dữ liệu khác:

Ví dụ: Tuple có thể được sử dụng làm phần tử của một list hoặc một set.

danh\_sach\_diem = [(1, 8), (2, 9), (3, 7)] # Mỗi phần tử là một tuple