Bài tập tổng hợp

1. **Phân biệt toán tử định dạng chuỗi và hàm định dạng chuỗi trong Python, cho 5 ví dụ:**

* Toán tử định dạng chuỗi (%)

Toán tử % là cách định dạng chuỗi truyền thống trong Python. Bạn sử dụng nó để thay thế các ký tự đặc biệt trong chuỗi bằng các giá trị cụ thể. Đây là cách làm cũ hơn và đã được thay thế bởi các phương pháp mới hơn trong nhiều trường hợp.

Cú pháp: "chuỗi mẫu %s" % giá\_trị

Ví dụ:

* Định dạng số nguyên:

age = 25

print("I am %d years old" % age)

# Output: I am 25 years old

* Định dạng số thực với số chữ số thập phân cụ thể:

pi = 3.14159

print("Pi is approximately %.2f" % pi)

# Output: Pi is approximately 3.14

* Định dạng chuỗi:

name = "Alice"

print("Hello, %s!" % name)

# Output: Hello, Alice!

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

Phương pháp này mới hơn và được khuyến khích sử dụng trong các phiên bản Python hiện đại (2.7+ và 3.0+). Nó cung cấp nhiều tính năng hơn và linh hoạt hơn so với toán tử %.

Cú pháp: "chuỗi mẫu {}".format(giá\_trị)

Ví dụ:

1. Định dạng số nguyên:

age = 25

print("I am {} years old".format(age))

# Output: I am 25 years old

1. Định dạng số thực với số chữ số thập phân cụ thể:

pi = 3.14159

print("Pi is approximately {:.2f}".format(pi))

# Output: Pi is approximately 3.14

1. Định dạng chuỗi:

name = "Alice"

print("Hello, {}!".format(name))

# Output: Hello, Alice!

Toán tử %: Cách định dạng chuỗi truyền thống, có vẻ cũ hơn và ít linh hoạt hơn.

Hàm định dạng chuỗi: Cung cấp khả năng định dạng tốt hơn và linh hoạt hơn, được khuyến khích sử dụng trong mã Python hiện đại.

1. **Chương trình xuất ra số ngẫu nhiên trong một đoạn bất kỳ cho trước:**

import random

a = int(input("Nhập giá trị bắt đầu: "))

b = int(input("Nhập giá trị kết thúc: "))

print("Số ngẫu nhiên là:", random.randint(a, b - 1))

1. **Khác biệt cơ bản của list và tuple:**

1. Cú pháp:

Cú pháp của tập hợp danh sách kiểu List và Tuple là hơi khác nhau. Đối tượng kiểu dữ liệu List được bao quanh bởi dấu ngoặc vuông [] và Tuples được bao quanh bởi dấu ngoặc tròn ().

Ví dụ:

list\_num = [1,2 ,3, 4]

tup\_num = (1,2 ,3 ,4)

print (list\_num)

print (tup\_num)

Trong đoạn mã bên trên, chúng ta đã định nghĩa một biến được gọi là list\_num chứa một tập hợp kiểu List bao gồm các giá trị số từ 1 đến 4. Danh sách List được bao quanh bởi dấu ngoặc vuông []. Ngoài ra, chúng ta thực hiện định nghĩa một biến tup\_num; trong đó chứa một tập hợp kiểu tuple bao gồm các giá trị số từ 1 đến 4. Tập hợp kiểu tuple được bao quanh bởi dấu ngoặc tròn ().

2. Tính bất biến:

Đối tượng kiểu List có tính chất có thể thay đổi, tức là đối tượng kiểu List có thể được thay đổi hoặc sửa đổi sau khi tạo tùy theo nhu cầu trong khi đối tượng kiểu  tuple có tính chất bất biến, tức là không thể thay đổi được hoặc sửa đổi tuple sau khi tạo.

3. Các thao tác:

Đối tượng kiểu List có nhiều hàm bên trong hơn so với đối tượng kiểu tuple. Chúng ta có thể sử dụng hàm dir([object]) để lấy ra tất cả các hàm liên quan cho list và tuple.Có rất nhiều hàm bổ sung được liên kết với một đối tượng kiểu List so với một đối tượng kiểu Tuple. Chúng ta có thể thực hiện thao tác chèn và xóa và sắp xếp các phần tử trong đối tượng kiểu List bằng các hàm không có trong đối tượng kiểu Tuple.

4. Kích thước:

Các thao tác trên đối tượng kiểu Tuples có kích thước nhỏ hơn so với thao tác trong các đối tượng kiểu List, điều này làm cho nó nhanh hơn một chút nhưng không phải là quá nhiều.

5. Cách sử dụng:

List: Thường được sử dụng khi bạn cần một cấu trúc dữ liệu có thể thay đổi.

Tuple: Thường được sử dụng khi bạn cần một tập hợp các giá trị không thay đổi, hoặc khi bạn muốn nhóm các giá trị có ý nghĩa liên quan

1. **Ứng dụng kiểu dữ liệu tuple trong thực tế:**

Lưu trữ dữ liệu không thay đổi: Tuple thường được sử dụng để lưu trữ dữ liệu mà bạn không muốn thay đổi trong suốt quá trình chạy chương trình, ví dụ như tọa độ địa lý (latitude, longitude).

Khóa trong từ điển: Tuple có thể được sử dụng làm khóa trong từ điển vì chúng không thể thay đổi.

Trả về nhiều giá trị từ hàm: Tuple thường được sử dụng để trả về nhiều giá trị từ một hàm.

Ví dụ

location = (10.762622, 106.660172)

print(f"Tọa độ địa lý: {location}")