**Bài làm**

**Bài 1**

Trong Python, có hai cách phổ biến để định dạng chuỗi: sử dụng **toán tử định dạng chuỗi** (%) và **hàm định dạng chuỗi** (str.format()). Dưới đây là sự phân biệt giữa hai cách này và các ví dụ minh họa.

**1.Toán tử định dạng chuỗi (%)**

Toán tử % là cách cổ điển để định dạng chuỗi trong Python. Bạn sử dụng ký tự đặc biệt để xác định kiểu dữ liệu sẽ được chèn vào chuỗi

 **%d**: số nguyên

 **%s**: chuỗi ký tự

 **%f**: số thực

**Ví dụ1:Định dạng chuỗi với chuỗi kí tự**

name= “NgocNi”

print(“My name is %s .“%(name))

**Ví dụ 2:Định dạng chuỗi với số nguyên**

age=19

print(“I’m %d years old.”%(age))

**Ví dụ 3: Định dạng chuỗi với số thực**

pi=3.14159

print(“Giá trị của pi là: %.2f “%(pi))

**Ví dụ 4:Định dạng nhiều biến trong một chuỗi**

Name=”Ni”

age=29

print(“My name is %s and I am %d years old.” %(name,age))

**2.Hàm định dạng chuỗi (str.format())**

Hàm str.format() là một cách định dạng chuỗi hiện đại hơn, linh hoạt hơn và dễ đọc hơn so với toán tử %.

**Ví dụ 5: Định dạng ngày tháng trong chuỗi**

day=21

month=8

year=2024

print(”Today’s date is {}/{}/{}.”.format(day,month,year)

**Ví dụ 6:Định dạng chuỗi với một biến**

name=”Ni”

print(“My name is {}”.format(name))

**Ví dụ 7:Định dạng chuỗi với nhiều biến**

name= “Ni”

age=19

print(”My name is {} and I {} years old”.format(name,age))

**Ví dụ 8:Định dạng số thập phân**

pi= 3.14159

print(“Giá trị của pi là {:.2f}”.format(pi))

**Ví dụ 9: Định dạng và sắp xếp thứ tự các biến**

firstname=”Ngọc”

lastname=”Ni”

age=19

print(“My name is {} {}. I’m {} years old”.format(firstname, lastname, age))

**Bài 2**

import random

a=int(input(“Nhập số nguyên a: “)

b=int(input(“Nhập số nguyên b: “)

#Kiểm tra để đảm bảo a<b

if a>b:

print(“Giá trị a phải nhỏ hơn hoặc bằng b.Mời nhập lại”)

else:

#Tạo số ngẫu nhiên trong đoạn

random\_number=random.randint(a,b)

print(f”Số ngẫu nhiên được tạo ra trong đoạn là : {random\_number}”)

**Bài 3:** List và tuple đều là các kiểu dữ liệu trong Python dùng để lưu trữ tập hợp các giá trị. Tuy nhiên, chúng có một số khác biệt cơ bản:

1.**Khả năng thay đổi (Mutability):**

**List**: Có thể thay đổi. Bạn có thể thêm, xóa hoặc thay đổi các phần tử của list sau khi nó đã được tạo.

**Tuple**: Không thể thay đổi (immutable). Khi một tuple được tạo ra, bạn không thể thay đổi, thêm hoặc xóa các phần tử của nó.

**2. Cú pháp:**

**List**: Được khai báo bằng dấu ngoặc vuông [], ví dụ: [1, 2, 3].

**Tuple**: Được khai báo bằng dấu ngoặc đơn (), ví dụ: (1, 2, 3).

**3. Hiệu suất:**

**List**: Do khả năng thay đổi, list có thể chậm hơn một chút so với tuple trong một số trường hợp nhất định.

**Tuple**: Vì là immutable, tuple thường có hiệu suất tốt hơn và tiêu tốn ít bộ nhớ hơn.

**4. Ứng dụng:**

**List**: Thích hợp khi bạn cần một tập hợp các phần tử có thể thay đổi hoặc khi bạn cần thực hiện các thao tác như thêm, xóa, sắp xếp phần tử.

**Tuple**: Thích hợp cho các tập hợp dữ liệu không thay đổi, như các giá trị cấu hình, hoặc khi bạn muốn đảm bảo rằng dữ liệu không bị thay đổi.

**5. Hàm hỗ trợ:**

**List**: Có nhiều phương thức như .append(), .remove(), .extend(), .pop(), etc.

**Tuple**: Hỗ trợ ít phương thức hơn, chủ yếu là .count() và .index().

**Bài 4**

Tuple có nhiều ứng dụng trong thực tế nhờ vào tính chất không thể thay đổi của chúng. Dưới đây là một số ví dụ về cách sử dụng tuple:

**1.Nhóm giá trị không thay đổi:**

* Khi bạn cần lưu trữ một tập hợp các giá trị mà không muốn thay đổi sau khi tạo, chẳng hạn như tọa độ của một điểm trong không gian (x, y, z).

point = (10, 20, 30)

**2.Key trong từ điển:**

* Tuple có thể được sử dụng làm key trong từ điển (dictionary) vì chúng là immutable, trong khi list không thể.
* locations = { (40.7128, -74.0060): "New York", (34.0522, -118.2437): "Los Angeles" }

**3.Trả về nhiều giá trị từ hàm:**

* Bạn có thể trả về nhiều giá trị từ một hàm bằng tuple. Điều này giúp dễ dàng gói các kết quả liên quan lại với nhau.

def get\_person\_info(): return ("John", "Doe", 30) first\_name, last\_name, age = get\_person\_info()

**4.Gán giá trị:**

* Tuple có thể được sử dụng để gán nhiều biến cùng một lúc, điều này giúp mã nguồn ngắn gọn và dễ hiểu hơn.

a, b, c = 1, 2, 3

**5.Lưu trữ cấu hình:**

* Tuple có thể dùng để lưu trữ các giá trị cấu hình không thay đổi của chương trình, chẳng hạn như các tham số cấu hình.

CONFIG = ("debug", 30, "localhost")

**6.Nhóm các đối tượng không thay đổi:**

* Khi bạn cần nhóm các đối tượng không thay đổi lại với nhau để sử dụng trong các thuật toán, chẳng hạn như các phần tử của một phép toán ma trận.

matrix = ((1, 2, 3), (4, 5, 6), (7, 8, 9))

**7.Danh sách các giá trị liên quan:**

* Tuple giúp nhóm các giá trị liên quan nhưng không cần thay đổi sau khi tạo, chẳng hạn như thông tin của một sinh viên bao gồm tên, tuổi, và điểm trung bình.

student = ("Alice", 21, 88.5)