Tên: Trần Dương Bảo Trân

MSSV: 23658171

Câu 1:

Toán tử định dạng chuỗi và hàm định dạng chuỗi đều được sử dụng để chèn giá trị vào trong chuỗi nhưng chúng hoạt động theo cách khác nhau.

*So sánh chung:
- Toán tử % thích hợp cho các trường hợp đơn giản và mã ngắn.
- str.format() phù hợp hơn cho mã phức tạp, dễ đọc và bảo trì.
Toán tử định dạng chuỗi (%)
Toán tử % được sử dụng để chèn các giá trị vào trong chuỗi, dựa trên các định dạng đã chỉ định. Đây là cách định dạng chuỗi truyền thống trong Python.
• Ưu điểm:
- Đơn giản và nhanh chóng cho các trường hợp định dạng cơ bản.
- Được sử dụng nhiều trong các phiên bản Python cũ.
-Hiệu năng cao.
- Cú pháp trực quan rõ ràng.
• Nhược điểm:
- Khó đọc và duy trì với các chuỗi dài hoặc phức tạp.
- Không hỗ trợ sắp xếp thứ tự các giá trị định dạng một cách linh hoạt.
Ví dụ 1: 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=”Tran”
age=29
print(“My name is %s and I am %d years old.” %(name,age))

Hàm định dạng chuỗi (str.format())
Hàm str.format() cung cấp một cách định dạng chuỗi hiện đại hơn, linh hoạt và dễ đọc hơn so với toán tử %. Nó cho phép chèn các giá trị vào chuỗi với các cú pháp đặc biệt như {}.
• Ưu điểm:
- Linh hoạt hơn trong việc định dạng và sắp xếp thứ tự các giá trị.
-Dễ đọc và dễ bảo trì hơn, đặc biệt với các chuỗi phức tạp.

-Hỗ trợ định dạng nhiều kiểu dữ liệu khác nhau một cách rõ ràng.
• Nhược điểm:
-Cú pháp dài hơn so với %.
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= “Tran”
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=”Bao”
lastname=”Tran”
age=19
print(“My name is {} {}. I’m {} years old”.format(firstname, lastname, age))
```

Câu 2:

```
Import random

List=[]

So_luong_phan_tu=10

For i in range(so_luong_phan_tu):

        So_ngau_nhien=random.randint(1,100)

        list.append(so_ngau_nhien)

print(“Số ngẫu nhiên”, list)
```

Câu 3:

List và Tuple có những khác biệt cơ bản sau:

1. Tính thay đổi

* **List**: Là kiểu dữ liệu thay đổi (mutable), nghĩa là bạn có thể thay đổi nội dung của danh sách sau khi nó được tạo. Bạn có thể thêm, xóa hoặc sửa đổi các phần tử trong một list.
* **Tuple**: Là kiểu dữ liệu không thay đổi (immutable), nghĩa là một khi một tuple được tạo ra, bạn không thể thay đổi nội dung của nó. Bạn không thể thêm, xóa hoặc sửa đổi các phần tử trong một tuple.

2. Cú pháp:

* **List**: Được tạo ra bằng cách sử dụng dấu ngoặc vuông [].

Ví dụ: my_list = [a,b].

*Tuple: Được tạo ra bằng cách sử dụng dấu ngoặc đơn ().

Ví dụ: my_tuple = (a,b).

3.Hiệu suất:

- Tuple thường có hiệu suất tốt hơn so với list vì chúng không cần quản lý không gian cho các thay đổi. Điều này có nghĩa là chúng thường tiêu thụ ít bộ nhớ hơn so với list và có thể nhanh hơn trong một số trường hợp.

4.Sử dụng:

*** List**: Thích hợp hơn cho các tập hợp dữ liệu có thể thay đổi, chẳng hạn như quản lý danh sách của một sản phẩm, người dùng, hoặc bất kỳ đối tượng nào mà bạn có thể thêm, xóa hoặc cập nhật.

* **Tuple**: Thích hợp cho các tập hợp dữ liệu không thay đổi, chẳng hạn như tọa độ (x, y) trong không gian, ngày tháng, hoặc các giá trị mà bạn không muốn thay đổi.

5.Tính năng:

- Tuple có thể được sử dụng như các khóa trong từ điển (dictionary) vì chúng là bất biến, trong khi list thì không thể làm được.

6. Phương thức và thuộc tính

* **List**: Có nhiều phương thức hữu ích như append(), extend(), remove(), và pop() để thực hiện các thao tác thay đổi danh sách.
*__Tuple__: Có rất ít phương thức vì tính không thay đổi của nó; thường chỉ có các phương thức như count() và index().

Câu 4:

1. **Đại diện cho các tọa độ:**

- **Hình học:** Một tuple có thể biểu diễn một điểm trong không gian 2D hoặc 3D (ví dụ: (x, y), (x, y, z)).
- **Địa lý:** Một tuple có thể đại diện cho một vị trí trên bản đồ (ví dụ: (vĩ độ, kinh độ)).

2. **Lưu trữ thông tin không đổi:**

- **Thông tin cá nhân:** (tên, tuổi, ngày sinh)
- **Thông tin sản phẩm:** (mã sản phẩm, tên sản phẩm, giá)
- **Màu sắc:** (red, green, blue)

3, **Làm tham số cho các hàm:**

- **Truyền nhiều giá trị:** Truyền một tuple làm tham số cho một hàm để truyền nhiều giá trị cùng một lúc.
- **Định nghĩa các cấu trúc dữ liệu đơn giản:** Sử dụng tuple để định nghĩa các cấu trúc dữ liệu đơn giản như điểm, vector, khoảng thời gian.

4. **Trả về kết quả từ các hàm:**

- **Trả về nhiều giá trị:** Một hàm có thể trả về nhiều giá trị bằng cách đóng gói chúng trong một tuple.

5. **Xử lý dữ liệu:**

- **Tạo các cấu trúc dữ liệu phức tạp:** Kết hợp tuple với các kiểu dữ liệu khác để tạo ra các cấu trúc dữ liệu phức tạp hơn như list of tuples, dictionary of tuples.
- **Xử lý dữ liệu từ các file:** Đọc dữ liệu từ các file và lưu trữ dưới dạng tuple.