1. ***Phân biệt toán tử định dạng chuỗi và hàm định dạng chuỗi có sẵn trong gói thư viện chuẩn Python?***

Trong Python, có hai cách chính để định dạng chuỗi:

* **Toán tử định dạng chuỗi (“**%”**)**: Đây là cách cũ hơn và thường được sử dụng trong các phiên bản Python trước. Toán tử % cho phép bạn chèn các giá trị vào trong chuỗi bằng cách sử dụng các ký tự định dạng đặc biệt.

Ví dụ:

name = "Xuan Ha"

age = 19

formatted\_string = "Name: %s, Age: %d" % (name, age)

print(formatted\_string)

Trong đó %s và %d là các ký tự định dạng tương ứng cho chuỗi và số nguyên.

* **Hàm định dạng chuỗi (“**str.format()”**)**: Phương pháp này là cách mới hơn và linh hoạt hơn, được giới thiệu trong Python . Nó sử dụng “{} “để xác định vị trí mà các giá trị sẽ được chèn vào và sử dụng phương thức “format()” để thực hiện định dạng.

Ví dụ

name = "Xuan Ha"

age = 19

formatted\_string = "Name: {}, Age: {}".format(name, age)

print(formatted\_string)

* **F-strings (chuỗi định dạng f)**: Đây là cách định dạng chuỗi mới nhất và được giới thiệu trong Python 3.6. F-strings sử dụng dấu f trước chuỗi và các biểu thức trong {} để định dạng.

Ví dụ:

name = "Alice"

age = 30

formatted\_string = f"Name: {name}, Age: {age}"

print(formatted\_string)

ví dụ:

**Ví dụ 1: Định dạng số nguyên và số thực**

print("Pi xấp xỉ %.2f" % 3.14159) ***Toán tử định dạng chuỗi***

print("Pi xấp xỉ {:.2f}".format(3.14159)) ***Hàm định dạng chuỗi***

### 2*. Viết chương trình Python cơ bản xuất ra số ngẫu nhiên trong một đoạn bất kỳ được cho trước?*

Để xuất ra một số ngẫu nhiên trong một đoạn cho trước, bạn có thể sử dụng thư viện random trong Python. Dưới đây là một ví dụ đơn giản:

import random

def generate\_random\_number(start, end):

return random.randint(start, end)

# Ví dụ sử dụng

start = 10

end = 50

random\_number = generate\_random\_number(start, end)

print(f"Số ngẫu nhiên trong khoảng từ {start} đến {end}: {random\_number}")

### 3. *Khác biệt cơ bản giữa list và tuple?*

**List**:

* + **Thay đổi được (Mutable)**: Bạn có thể thay đổi các phần tử trong một list sau khi nó đã được tạo ra.
  + **Cú pháp**: Danh sách được định nghĩa bằng dấu ngoặc vuông “[]”.
  + **Ví dụ**:” my\_list = [1, 2, 3, 4]”

**Tuple**:

* + **Không thay đổi được (Immutable)**: Một khi tuple đã được tạo, bạn không thể thay đổi các phần tử của nó.
  + **Cú pháp**: Tuple được định nghĩa bằng dấu ngoặc đơn ().
  + **Ví dụ**: “my\_tuple = (1, 2, 3, 4)”

**So sánh**:

* + **Danh sách**: Thích hợp khi bạn cần một tập hợp các phần tử có thể thay đổi.
  + **Tuple**: Thích hợp khi bạn cần một tập hợp các phần tử không thay đổi, hoặc để sử dụng như khóa trong từ điển (dict).

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

* **Như là khóa trong từ điển (dict)**: Tuple có thể được sử dụng làm khóa cho từ điển, trong khi danh sách thì không.

Code:my\_dict = { (1, 2): "value1", (3, 4): "value2" }

* **Nhóm dữ liệu liên quan**: Tuple thường được dùng để nhóm các giá trị liên quan với nhau, ví dụ: tọa độ (x, y).

Ví dụ:”coordinates = (10, 20)”

* **Trả về nhiều giá trị từ hàm**: Tuple có thể được sử dụng để trả về nhiều giá trị từ một hàm

Code python:

def min\_max(values):

return (min(values), max(values))

min\_val, max\_val = min\_max([1, 2, 3, 4, 5])

* **Như là đối số không thay đổi**: Tuple có thể được sử dụng khi bạn muốn đảm bảo rằng các giá trị không bị thay đổi trong một quá trình tính toán.

Ví dụ

def process\_data(data: tuple):

# data không thay đổi trong hàm này pass

* **Hợp tác với các thư viện và cấu trúc dữ liệu**: Một số thư viện và cấu trúc dữ liệu yêu cầu sử dụng tuple vì tính không thay đổi của nó.