**Tên: Huỳnh Văn Thiên**

**Mssv: 23730711**

**Câu 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?**

**1. Toán tử định dạng chuỗi (%)**

Toán tử % là một phương pháp truyền thống để định dạng chuỗi trong python

**Cú pháp cơ bản:**

"chuỗi định dạng" % (giá trị1, giá trị2, ...)

***Vd1:***

name = "Thiên"

age = 19

s = "Tên: %s, Tuổi: %d" % (name, age)

print(s)

# Output: Tên: Thiên, Tuổi: 19

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

Hàm str.format là một phương pháp mới hơn để định dạng chuỗi, cung cấp nhiều tính năng hơn và dễ sử dụng hơn so với toán tử %. Bạn có thể sử dụng các placeholder {} trong chuỗi định dạng và truyền các giá trị vào hàm format

**Cú pháp cơ bản**

"chuỗi định dạng {}".format(giá trị)

***Vd2:***

name = "Thiên"

age = 19

s = "Tên: {}, Tuổi: {}".format(name, age)

print(s)

# Output: Tên: Thiên, Tuổi: 19

**3.F-Strings (chuỗi định dạng cấp độ cao, từ Python 3.6+)**

F-strings là phương pháp mới nhất và được khuyến khích sử dụng trong các phiên bản Python từ 3.6 trở đi. Nó cho phép bạn chèn biểu thức trực tiếp vào trong chuỗi định dạng, làm cho mã dễ đọc và dễ viết hơn.

Cú pháp cơ bản

f"chuỗi định dạng {biểu thức}"

***Vd3:***

name = "Thiên"

age = 19

Thiên = f"Tên: {name}, Tuổi: {age}"

print(s)

# Output: Tên: Thiên, Tuổi: 28

**So sánh**

1. **Toán tử %:** Cú pháp cũ hơn, ít linh hoạt hơn, và có thể gây khó khăn khi làm việc với nhiều kiểu dữ liệu hoặc khi cần định dạng phức tạp.
2. **Hàm str.format():** Cung cấp cú pháp dễ đọc hơn và linh hoạt hơn so với toán tử %, cho phép sử dụng các chỉ định vị trí, tên, và các biểu thức.
3. **F-strings:** Cung cấp cú pháp rõ ràng và dễ đọc nhất, hỗ trợ định dạng linh hoạt và cho phép chèn biểu thức vào trong chuỗi.

***Vd4:***

price = 49.99

s= "Giá của sản phẩm là ${:.2f}".format(price)

print(s)

# Output: Giá của sản phẩm là $49.99

***Vd5:***

item = "apple"

soluong = 5

s = f"Bạn đã mua {soluong} quả {item}"

print(s)

# Output: Bạn đã mua 5 quả apple

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

import random

def random\_number(start, end):

"""Trả về một số ngẫu nhiên nguyên trong khoảng từ start đến end (bao gồm cả start và end)."""

return random.randint(start, end)

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

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

random\_value = random\_number(start, end)

print("Số ngẫu nhiên:", random\_value)

**Câu 3: Khác biệt cơ bản giữa list và tuple?**

Lists là các đối tượng có thể thay đổi, có nghĩa là bạn có thể sửa đổi một đối tượng list sau khi nó được tạo.

Mặt khác, các tuple là các đối tượng bất biến, có nghĩa là bạn không thể sửa đổi một đối tượng tuple sau khi nó được tạo.

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

**1. Lưu trữ dữ liệu không thay đổi:**

* **Tọa độ:** Một tuple có thể lưu trữ tọa độ (x, y) của một điểm trên mặt phẳng. Vì tọa độ thường không thay đổi nên tuple là một lựa chọn phù hợp.
* **Màu sắc:** Một tuple có thể biểu diễn một màu sắc trong hệ màu RGB (Red, Green, Blue) dưới dạng (R, G, B).
* **Ngày tháng:** Một tuple có thể lưu trữ ngày, tháng, năm dưới dạng (ngày, tháng, năm).

**2. Làm khóa trong từ điển:**

* Vì tuple là bất biến, nên nó có thể được sử dụng làm khóa trong từ điển. Điều này hữu ích khi bạn muốn ánh xạ một tập hợp các giá trị với một khóa phức tạp.

**3. Trả về nhiều giá trị từ hàm:**

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

**4. Định nghĩa các cấu trúc dữ liệu đơn giản:**

* Tuple có thể được sử dụng để tạo các cấu trúc dữ liệu đơn giản như một điểm trong không gian 3 chiều (x, y, z), một phân số (tử số, mẫu số),...

**5. Sắp xếp và tìm kiếm dữ liệu**:

* Tuples có thể được sử dụng trong các thuật toán sắp xếp và tìm kiếm, nơi mà các phần tử của tuple có thể là các giá trị cần so sánh