**Câu 1:  
1. Toán tử định dạng chuỗi (%)**

Toán tử % được sử dụng để định dạng chuỗi theo cách tương tự như trong ngôn ngữ C. Đây là cách truyền thống, có thể không còn phổ biến bằng các phương pháp hiện đại nhưng vẫn được hỗ trợ.

**Cú pháp cơ bản:**

python

Copy code

formatted\_string = "format string % (values)"

**Ví dụ 1:**

python

Copy code

name = "Linh"

age = 19

result = "Name: %s, Age: %d" % (name, age)

print(result) # Output: Name: Linh, Age: 19

**Ví dụ 2:**

python

Copy code

pi = 3.14159

result = "Value of pi: %.2f" % pi

print(result) # Output: Value of pi: 3.14

**Ví dụ 3:**

python

Copy code

item = "banana"

count = 1

result = "I have %d %s(s)" % (count, item)

print(result) # Output: I have 1 banana

**Ví dụ 4:**

python

Copy code

name = "Bob"

score = 90

result = "Student %s scored %05d" % (name, score)

print(result) # Output: Student Bob scored 00090

**Ví dụ 5:**

python

Copy code

value = 1234.567

result = "Formatted value: %,.2f" % value

print(result) # Output: Formatted value: 1,234.57

**2. Hàm định dạng chuỗi (f-string và phương thức .format())**

**a. F-strings (Python 3.6+)**

F-strings là một cách mới và rất thuận tiện để định dạng chuỗi, giúp mã dễ đọc hơn và có hiệu suất tốt hơn.

**Cú pháp cơ bản:**

python

Copy code

formatted\_string = f"format string {expression}"

**Ví dụ 1:**

python

Copy code

name = "Linh"

age = 19

result = f"Name: {name}, Age: {age}"

print(result) # Output: Name: Linh, Age: 19

**Ví dụ 2:**

python

Copy code

pi = 3.14159

result = f"Value of pi: {pi:.2f}"

print(result) # Output: Value of pi: 3.14

**Ví dụ 3:**

python

Copy code

item = "banana"

count = 1

result = f"I have {count} {item}(s)"

print(result) # Output: I have 1 banana

**Ví dụ 4:**

python

Copy code

name = "Bob"

score = 90

result = f"Student {name} scored {score:05d}"

print(result) # Output: Student Bob scored 00090

**Ví dụ 5:**

python

Copy code

value = 1234.567

result = f"Formatted value: {value:, .2f}"

print(result) # Output: Formatted value: 1,234.57

**b. Phương thức .format()**

Phương thức .format() là cách định dạng chuỗi mạnh mẽ và linh hoạt hơn, hỗ trợ nhiều tùy chọn.

**Cú pháp cơ bản:**

python

Copy code

formatted\_string = "format string {}".format(values)

**Ví dụ 1:**

python

Copy code

name = "Linh"

age = 19

result = "Name: {}, Age: {}".format(name, age)

print(result) # Output: Name: Linh, Age: 19

**Ví dụ 2:**

python

Copy code

pi = 3.14159

result = "Value of pi: {:.2f}".format(pi)

print(result) # Output: Value of pi: 3.14

**Ví dụ 3:**

python

Copy code

item = "banana"

count = 1

result = "I have {} {}(s)".format(count, item)

print(result) # Output: I have 1 banana

**Ví dụ 4:**

python

Copy code

name = "Bob"

score = 90

result = "Student {0} scored {1:05d}".format(name, score)

print(result) # Output: Student Bob scored 00090

**Ví dụ 5:**

python

Copy code

value = 1234.567

result = "Formatted value: {:, .2f}".format(value)

print(result) # Output: Formatted value: 1,234.57

**Tổng kết**

* **Toán tử %**: Đây là phương pháp truyền thống để định dạng chuỗi, sử dụng cú pháp tương tự như trong ngôn ngữ C. Thích hợp cho các tình huống đơn giản nhưng có phần hạn chế và có thể khó đọc hơn với các định dạng phức tạp.
* **F-strings**: Cung cấp cách định dạng chuỗi rõ ràng và hiệu quả nhất trong Python, đặc biệt là với khả năng tích hợp biểu thức trực tiếp trong chuỗi định dạng.
* **Phương thức .format()**: Rất linh hoạt và mạnh mẽ, phù hợp với các tình huống cần định dạng phức tạp hoặc cần định dạng dựa trên các tham số không xác định trước.

Tùy thuộc vào yêu cầu của bạn và phiên bản Python bạn đang sử dụng, bạn có thể chọn phương pháp định dạng chuỗi phù hợp nhất.

**Câu 2:**import random

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

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

if start > end:

print("Giá trị bắt đầu phải nhỏ hơn hoặc bằng giá trị kết thúc.")

else:

random\_number = random.randint(start, end)

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

**Câu 3:  
1. Tính thay đổi**

* **list**: Thay đổi được (có thể thêm, xóa, thay đổi phần tử).
* **tuple**: Không thay đổi được (không thể thay đổi sau khi tạo).

**2. Cú pháp**

* **list**: Dùng dấu ngoặc vuông [].

python

Copy code

my\_list = [1, 2, 3]

* **tuple**: Dùng dấu ngoặc đơn ().

python

Copy code

my\_tuple = (1, 2, 3)

**3. Hiệu suất**

* **list**: Chậm hơn và sử dụng nhiều bộ nhớ hơn.
* **tuple**: Nhanh hơn và tiết kiệm bộ nhớ hơn.

**4. Phương thức**

* **list**: Nhiều phương thức như .append(), .remove().
* **tuple**: Ít phương thức như .count(), .index().

**5. Sử dụng**

* **list**: Khi cần dữ liệu có thể thay đổi.
* **tuple**: Khi cần dữ liệu không thay đổi hoặc dùng làm khóa trong từ điển.

**Câu 4:**

Lưu trữ giá trị cố định

Khóa trong từ điển

Trả về nhiều giá trị

Bảo vệ dữ liệu

Lưu trữ dữ liệu lặp lại