**Câu 1**

Toán tử % là cách định dạng chuỗi cổ điển trong Python, tương tự như cách định dạng chuỗi trong C.

**Cú pháp**: "chuỗi % kiểu" % giá\_trị

**Ví dụ 1**: Định dạng số nguyên

name = "Alice"

age = 30

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

print(result) # Output: Name: Alice, Age: 30

**Ví dụ 2**: Định dạng số thực

pi = 3.14159

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

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

**Ví dụ 3**: Định dạng chuỗi với padding

name = "Bob"

result = "Hello, %-10s!" % name

print(result) # Output: Hello, Bob !

**Ví dụ 4**: Định dạng ngày tháng

from datetime import datetime

now = datetime.now()

result = "Current date and time: %s" % now.strftime("%Y-%m-%d %H:%M:%S")

print(result) # Output: Current date and time: 2024-08-25 14:30:00

**Ví dụ 5**: Định dạng từ điển

info = {"name": "Charlie", "age": 28}

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

print(result) # Output: Name: Charlie, Age: 28

### Hàm format()

Hàm format() là phương pháp định dạng chuỗi được giới thiệu trong Python 2.7 và Python 3.0. Nó cung cấp nhiều tính năng hơn và linh hoạt hơn so với toán tử %

**Cú pháp**: "chuỗi {}".format(giá\_trị)

**Ví dụ 1**: Định dạng số nguyên

name = "Alice"

age = 30

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

print(result) # Output: Name: Alice, Age: 30

**Ví dụ 2**: Định dạng số thực

pi = 3.14159

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

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

**Ví dụ 3**: Định dạng với padding

name = "Bob"

result = "Hello, {:<10}!".format(name)

print(result) # Output: Hello, Bob !

**Ví dụ 4**: Định dạng ngày tháng

from datetime import datetime

now = datetime.now()

result = "Current date and time: {}".format(now.strftime("%Y-%m-%d %H:%M:%S"))

print(result) # Output: Current date and time: 2024-08-25 14:30:00

**Ví dụ 5**: Định dạng từ điển

info = {"name": "Charlie", "age": 28}

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

print(result) # Output: Name: Charlie, Age: 28

**Câu 3**

**List**: Là kiểu dữ liệu thay đổi (mutable). Bạn có thể thay đổi, thêm, hoặc xóa các phần tử của một danh sách sau khi nó được tạo ra.

**Tuple**: Là kiểu dữ liệu không thay đổi (immutable). Một khi một tuple đã được tạo, bạn không thể thay đổi, thêm hoặc xóa các phần tử của nó.

**Câu 4**

**-Lưu trữ dữ liệu không thay đổi**

-Nhóm dữ liệu liên quan

-Dùng làm khóa trong từ điển

-Trả về nhiều giá trị từ hàm

-Dùng làm dữ liệu không thay đổi trong cấu trúc dữ liệu

Tóm lại tuple là một kiểu dữ liệu hữu ích trong Python, đặc biệt trong các trường hợp cần lưu trữ dữ liệu không thay đổi, nhóm các giá trị liên quan, sử dụng làm khóa trong từ điển, trả về nhiều giá trị từ hàm, hoặc đảm bảo tính toàn vẹn và hiệu suất của dữ liệu. Việc sử dụng tuple giúp làm cho mã nguồn của bạn rõ ràng hơn và giảm nguy cơ lỗi do thay đổi dữ liệu không mong muốn.