**Câu 1:Chuỗi định dạng toán tử phân biệt và chuỗi định dạng hàm trong Python**

Trong Python, có hai cách chính để định dạng chuỗi: sử dụng toán tử %(cũ hơn) và sử dụng phương thức format()(hiện đại hơn). Mặc dù cả hai đều đạt được mục tiêu tương tự, nhưng chúng có cú pháp và một số đặc điểm khác về tính năng.

**Toán tử** %**(Định dạng kiểu cũ)**

* **Cú pháp:** chuỗi % (giá\_trị,)
* **Cách thức:** Giá trị được đặt vào trong chuỗi bằng cách sử dụng dấu % như trình giữ chỗ.
* **Ví dụ:**

Trăn

name = "Alice"   
age = 30   
print("Hello, my name is %s and I am %d years old." % (name, age)) 

* **Ưu điểm:** Cú pháp đơn giản, dễ hiểu.
* **Nhược điểm:** Ít linh hoạt so với format(), đặc biệt khi định dạng phức tạp.

**Phương thức** format()**(Định dạng kiểu mới)**

* **Cú pháp:** chuỗi.format(giá\_trị1, giá\_trị2, ...)
* **Cách thức:** Giá trị được đặt vào trong chuỗi bằng cách sử dụng dấu ngoặc nền {}như trình giữ chỗ.
* **Ví dụ:**

Trăn

name = "Bob"   
price = 19.99   
print("The product costs {:.2f} dollars and is bought by {}".format(price, name)) 

* **Ưu điểm:** Linh hoạt hơn, hỗ trợ nhiều loại phức tạp định dạng, có thể sử dụng số hoặc tên để truyền tham số.
* **Nhược điểm:** Cú pháp có thể kéo dài dòng hơn trong một số trường hợp đơn giản.

**So sánh và minh họa ví dụ**

|  |  |  |
| --- | --- | --- |
| Tính năng | Toán tử% | Phương phápformat() |
| Cú pháp | chuỗi % (giá\_trị,) | chuỗi.format(giá\_trị1, giá\_trị2, ...) |
| Chỗ giữ chỗ | %s, %d, ... | {} |
| Linh hoạt | Thấp | Cao |
| Sử dụng số thứ tự | Không | Có |
| Sử dụng tên | Không | Có |

**Ví dụ minh họa:**

1. **Định dạng đơn giản:**

Trăn

# Toán tử %   
print("My name is %s" % "Alice")   
# Phương thức format()   
print("My name is {}".format("Alice")) 

1. **Định dạng nhiều giá trị:**

Trăn

# Toán tử %   
print("The sum of %d and %d is %d" % (3, 4, 3+4))   
# Phương thức format()   
print("The sum of {} and {} is {}".format(3, 4, 3+4)) 

1. **Định dạng số với độ chính xác cao:**

Trăn

# Phương thức format()   
pi = 3.14159   
print("Pi to two decimal places is {:.2f}".format(pi)) 

1. **Định dạng ngày tháng:**

Trăn

import datetime   
today = datetime.date.today()   
# Phương thức format()   
print("Today's date is: {:%B %d, %Y}".format(today)) 

1. **Format with tham số name:**

Trăn

# Phương thức format()   
person = {"name": "Bob", "age": 30}   
print("Hello, {name}. You are {age} years old.".format(\*\*person)) 

**Kết luận:**

* **Phương thức** format()khuyến nghị sử dụng trong các trường hợp phức tạp và khi cần linh hoạt cao.
* **Toán tử** %vẫn có thể được sử dụng trong các trường hợp đơn giản, nhưng nên hạn chế để tránh xung đột với các phiên bản Python mới hơn.

**Lưu ý:** Trong các phiên bản Python 3. 6 trở lên, có thêm một cách định dạng chuỗi mới được gọi là *f-strings* (chuỗi ký tự được định dạng), cung cấp một cú pháp gọn gàng và linh hoạt hơn nữa.

**Câu 2:**

import random

# Nhập giá trị cho đoạn [a, b]

a = float(input("Nhập giá trị bắt đầu của đoạn (a): "))

b = float(input("Nhập giá trị kết thúc của đoạn (b): "))

# Kiểm tra để đảm bảo a nhỏ hơn b

if a >= b:

    print("Giá trị bắt đầu của đoạn phải nhỏ hơn giá trị kết thúc.")

else:

    # Sinh số ngẫu nhiên trong khoảng [a, b)

    random\_number = random.uniform(a, b)

    # Xuất kết quả

    print(f"Số ngẫu nhiên trong khoảng [{a}, {b}): {random\_number:.3f}")

**Câu 3:**

**1. Tính biến đổi:**

* **List:** Là một đối tượng có thể thay đổi (có thể thay đổi). Bạn có thể thêm, xóa hoặc sửa đổi các phần tử trong danh sách sau khi nó được tạo.
* **Tuple:** Là một đối tượng bất biến (không thể thay đổi). Một khi bộ dữ liệu được tạo, bạn không thể thay đổi các phần tử trong đó.

**2. Cú pháp:**

* **List:** Được bao quanh bởi các dấu vuông []. Ví dụ: my\_list = [1, 2, 3, "apple"]
* **Tuple:** Được bao quanh bởi các dấu ngoặc tròn (). Ví dụ: my\_tuple = (1, 2, 3, "apple")

**3. Sử dụng:**

* **Danh sách:** Thường được sử dụng khi bạn cần một dữ liệu cấu trúc có thể thay đổi động. Ví dụ: lưu trữ danh sách các sản phẩm trong một giỏ hàng, các ngẫu nhiên, v. v.
* **Tuple:** Thường được sử dụng để lưu trữ dữ liệu không thay đổi, chẳng hạn như các hằng số, các cặp khóa-giá trị trong một từ điển, hoặc để trả về nhiều giá trị từ một hàm.

**4. Hiệu suất:**

* **Tuple:** Thường nhanh hơn liệt kê một chút vì tính chất bất biến của nó. Python không cần phải quản lý các thay đổi đối với bộ tuple.

**Câu 4:**

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

* **Tốc độ:** Một điểm trên Hiện vật có thể được biểu diễn bằng một bộ dữ liệu (x, y).
* **Kích thước:** Kích thước của một hình ảnh hoặc màn hình có thể được biểu thị bằng một bộ dữ liệu (chiều rộng, chiều cao).
* **Ngày tháng:** Một ngày trong tháng có thể được biểu diễn bằng một bộ dữ liệu (năm, tháng, ngày).

Trăn

# Tọa độ   
point = (3, 5)   
# Kích thước màn hình   
screen\_size = (1920, 1080)   
# Ngày sinh   
birth\_date = (1990, 12, 25) 

Sử dụng mã [một cách thận trọng.](https://word-edit.officeapps.live.com/faq#coding)

**2. Trả về nhiều giá trị từ hàm:**

* **Hàm tìm giá trị lớn nhất và nhỏ nhất:**

Trăn

def find\_min\_max(numbers):   
    return min(numbers), max(numbers) 

Sử dụng mã [một cách thận trọng.](https://word-edit.officeapps.live.com/faq#coding)

* **Hàm chia một số cho một số khác:**

Trăn

def divide(x, y):   
    quotient = x // y   
    remainder = x % y   
    return quotient, remainder 

Sử dụng mã [một cách thận trọng.](https://word-edit.officeapps.live.com/faq#coding)

**3. Làm khóa cho từ điển:**

* Các phần tử trong bộ dữ liệu có thể băm được, nghĩa là chúng có thể được sử dụng làm khóa trong từ điển. Điều này hữu ích khi bạn cần một từ khóa tổng hợp.
* Ví dụ:

Trăn

person = {('Alice', 'Bob'): 'married', ('Charlie', 'David'): 'friends'} 

Sử dụng mã [một cách thận trọng.](https://word-edit.officeapps.live.com/faq#coding)

**4. Sử dụng trong các dữ liệu cấu trúc khác:**

* **Set:** Các phần tử của tập hợp phải có thể băm được, nên bộ tuple có thể được sử dụng làm phần tử của tập hợp.
* **List:** Tuple có thể là một phần tử của danh sách.

**5. Các ứng dụng khác:**

* **Định nghĩa các cấu trúc đơn giản:** Ví dụ: một thẻ trong bộ bài có thể được biểu thị bằng một bộ dữ liệu (xếp hạng, chất).
* **Truyền nhiều đối số cho hàm:** Tuple can be used to close package many đối số thành một đối tượng duy nhất.
* **Trả về kết quả của các phép toán được phép:** Ví dụ, hàm divmod()trong Python trả về một bộ chứa số thương và số dư của phép chia.

**Ưu điểm khi sử dụng tuple:**

* **Đã đọc:** cấu trúc rõ ràng của dữ liệu.
* **An toàn:** Bảo vệ dữ liệu khỏi tình trạng vô hiệu hóa.
* **Hiệu quả:** Tuple thường nhanh hơn danh sách một chút.