Họ và tên: Lê Thị Phương Anh

MSSV: 23711171

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?

*** Toán tử định dạng chuỗi:**

- Cú pháp : `f"chuỗi {biến1} {biến2} ..."`

- Đặc điểm:

+ Đơn giản, trực quan: Cú pháp giống như viết bình thường, chỉ cần thêm chữ f trước dấu ngoặc kép.

+ Hiệu năng cao: Được biên dịch tại thời điểm biên dịch, nên tốc độ thực thi nhanh hơn các phương pháp khác.

+ Tính năng phong phú: Hỗ trợ định dạng số, gọi hàm, biểu thức ngay trong chuỗi.

- Ví dụ:

```
name = "phuonganh"
age = 19
print(f"Tên tôi là {name} và tôi {age} tuổi.")
```

*** Hàm định dạng chuỗi:**

- Cú pháp: `"chuỗi {} {} ...".format(biến1, biến2, ...)`

- Đặc điểm:

+ Linh hoạt: Cho phép định dạng phức tạp hơn bằng cách sử dụng chỉ số hoặc tên biến trong cặp ngoặc nhọn.

+ Khả năng tương thích: Có mặt từ các phiên bản Python cũ hơn.

- Ví dụ:

```
Python
price = 19.99
quantity = 3
print("Tổng tiền là: {:.2f} cho {} sản phẩm.".format(price * quantity, quantity))
```

*** So sánh và ví dụ minh họa:**

Tính năng	**Toán tử định dạng chuỗi**	**Hàm định dạng chuỗi**
Cú pháp	f"chuỗi {biến}"	"chuỗi {}".format(biến)
Hiệu năng	Nhanh	Chậm hơn
Tính năng	Gọi hàm, biểu thức	Chỉ số, tên biến
Ví dụ	print(f"Pi xấp xỉ {math.pi:.3f}")	print("Pi xấp xỉ {:.3f}".format(math.pi))
Ví dụ	print(f"{2**3} mũ 3")	print("{} mũ 3".format(2**3))
Ví dụ	name = "Bob"; print(f"Xin chào, {name.upper()}!")	name = "Bob"; print("Xin chào, {}!".format(name.upper()))
Ví dụ	name = "Bob"; print(f"Xin chào, {name.upper()}!")	print("Ngày hôm nay là ngày {:%d/%m/%Y}".format(datetime.date.today()))

Ví dụ	print(f"Số pi có {len(str(math.pi))} chữ số")	print("Số pi có {} chữ số".format(len(str(math.pi))))

Câu 2: Viết chương trình xuất ra số ngẫu nhiên trong một đoạn bất kỳ bất cho trước?
Python
```
import random

# Nhập khoảng giá trị từ người dùng
min_value = int(input("Nhập giá trị nhỏ nhất: "))
max_value = int(input("Nhập giá trị lớn nhất: "))

# Sinh số ngẫu nhiên và in ra
random_number = random.randint(min_value, max_value)
print("Số ngẫu nhiên:", random_number)
```

* Giải thích code:

1. import random: Nhập thư viện random để sử dụng các hàm liên quan đến số ngẫu nhiên.
2. Nhập khoảng giá trị:
    - min_value: Giá trị nhỏ nhất của khoảng.
    - max_value: Giá trị lớn nhất của khoảng.
3. Sinh số ngẫu nhiên:
    - random.randint(min_value, max_value): Tạo một số nguyên ngẫu nhiên nằm trong khoảng từ min_value đến max_value.
4. In kết quả: In ra số ngẫu nhiên vừa sinh.

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

List và tuple là hai kiểu dữ liệu được sử dụng để lưu trữ một tập hợp các giá trị trong Python. Tuy nhiên, chúng có những đặc điểm khác biệt quan trọng:

1. Tính biến đổi (mutability):

- **List:** Có thể thay đổi sau khi được tạo. Bạn có thể thêm, xóa, sửa đổi các phần tử trong một list.
- **Tuple:** Không thể thay đổi sau khi được tạo. Các phần tử của một tuple được cố định.

2. Cú pháp:

- **List:** Được bao quanh bởi dấu ngoặc vuông `[]`.
- **Tuple:** Được bao quanh bởi dấu ngoặc tròn `()`.

3. Sử dụng:

- **List:** Thường được sử dụng khi bạn cần một cấu trúc dữ liệu có thể thay đổi linh hoạt. Ví dụ: lưu trữ danh sách các sản phẩm, danh sách các người dùng.
- **Tuple:** Thường được sử dụng khi bạn muốn bảo vệ dữ liệu khỏi bị thay đổi vô tình hoặc khi bạn cần một cấu trúc dữ liệu không đổi để làm khóa cho dictionary. Ví dụ: lưu trữ các tọa độ (x, y), các ngày trong tuần.

- Ví dụ:

```
# List

my_list = [1, 2, 3, "apple", "banana"]

my_list.append(4)  # Thêm phần tử vào list

print(my_list)  # Output: [1, 2, 3, 'apple', 'banana', 4]



# Tuple

my_tuple = (10, 20, "hello")

# my_tuple[0] = 30  # Sẽ gây ra lỗi vì tuple không thể thay đổi

print(my_tuple)  # Output: (10, 20, 'hello')
```

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

Tuple trong Python là một kiểu dữ liệu vô cùng hữu ích, đặc biệt khi bạn cần lưu trữ dữ liệu có thứ tự và không muốn chúng bị thay đổi. Dưới đây là một số ứng dụng thực tế phổ biến của tuple:

1. Lưu trữ dữ liệu có thứ tự cố định:

- **Tọa độ:** Một điểm trên đồ thị có thể được biểu diễn bằng một tuple (x, y).
- **Ngày tháng năm:** (ngày, tháng, năm)
- **Màu sắc:** (red, green, blue)
- **Card trong bộ bài:** (suit, rank)

- **Thông tin về người dùng:** (tên, tuổi, email)

2. Làm khóa cho dictionary:

- Vì tuple là bất biến, nên chúng có thể được sử dụng làm khóa trong dictionary. Điều này đặc biệt hữu ích khi bạn cần ánh xạ nhiều giá trị với một khóa phức tạp.
- Ví dụ:

Python

student_grades = {('Alice', 'Math'): 95, ('Bob', 'Science'): 88}

3. Trả về nhiều giá trị từ một hàm:

- Một hàm Python chỉ có thể trả về một giá trị. Tuy nhiên, bằng cách trả về một tuple, bạn có thể trả về nhiều giá trị cùng một lúc.
- Ví dụ:

Python

def find_min_max(numbers):

return min(numbers), max(numbers)

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

- Tuple có thể được sử dụng để định nghĩa các cấu trúc dữ liệu đơn giản như điểm, vector, phân số.

5. Làm tham số cho các hàm:

- Bạn có thể truyền một tuple làm tham số cho một hàm để truyền nhiều giá trị cùng một lúc.

6. Unpacking tuple:

- Bạn có thể "giải nén" một tuple thành nhiều biến riêng biệt.

Ví dụ:

point = (3, 4)

x, y = point

7. Tạo các danh sách không thể thay đổi:

- Nếu bạn muốn tạo một danh sách mà các phần tử của nó không thể bị thay đổi, bạn có thể sử dụng một tuple của các tuple.