**CÂU HỎI TỔNG HỢP**

***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 hoạt tương tự?***

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

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

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

BÀI LÀM

**Câu 1:** Trong Python, có hai cách phổ biến để định dạng chuỗi: sử dụng toán tử định dạng chuỗi (%) và sử dụng phương thức định dạng chuỗi .format(). Ngoài ra, từ Python 3.6 trở đi, Python còn giới thiệu f-string (formatted string literals) để định dạng chuỗi.

### 1. Toán tử định dạng chuỗi (%)

Toán tử % được sử dụng để định dạng chuỗi bằng cách thay thế các placeholder (ký tự % theo sau bởi một ký tự định dạng) trong chuỗi với các giá trị tương ứng.

+ Cú pháp: “Format\_string” % (values)

+ Ví dụ:

# Ví dụ 1: Định dạng số nguyên

age = 19

print(“Tôi năm nay %d tuổi.” % age)

# Ví dụ 2: Định dạng chuỗi

name = kieu

print("Tên tôi là %s" % name)

# Ví dụ 3: Định dạng số thực

pi =3.4

print("Điểm trung bình là %.2f" % pi)

# Ví dụ 4: Định dạng ký tự

initial = ‘B’

print("Ký tự đầu tiên là %c" % 'B')

# Ví dụ 5: Định dạng nhiều giá trị

name = kieu

age = 19

print("Tên: %s, Tuổi: %d" % (kieu, 19))

2. Định dạng chuỗi có sẵn trong gói thư viện chuẩn Python

Định dạng chuỗi trong Python là một kỹ thuật cho phép bạn chèn các giá trị biến vào trong một chuỗi văn bản, giúp tạo ra các chuỗi động và linh hoạt. Thư viện chuẩn Python cung cấp nhiều cách để thực hiện việc này, trong đó hai cách phổ biến nhất là sử dụng phương thức format() và sử dụng f-strings.

# Ví dụ 1:

name = "kieu"

age = 19

print("Xin chào, tôi là {} và tôi {} tuổi.".format(name, age))

#Ví dụ 2

name = "khai"

age = 20

print(f"Tên: {name}, Tuổi: {age}")

#Ví dụ 3

number = 1234567.23

print(f"Số được định dạng: {number:,.2f}")

#Ví dụ 4

name = "hoang"

age = 21

print(f"Tên: {name}, Tuổi: {age}")

#Ví dụ 5

name = "Nhan"

age = 19

print("Xin chào, tôi là {} và tôi {} tuổi.".format(name, age))

**Câu 2:**

import random

# Tạo một danh sách các quả

hoaqua = ["tao", "xoai", "man", "oi"]

# Chọn ngẫu nhiên một quả

random\_hoaqua = random.choice(hoaqua)

print("Quả ngẫu nhiên:", random\_hoaqua)

**Câu 3: List** và **tuple** là hai kiểu dữ liệu được sử dụng rất phổ biến trong Python để lưu trữ các phần tử. Tuy nhiên, chúng có những đặc điểm khác biệt cơ bản sau đây:

### **1. Tính biến đổi (mutability):**

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

### **2. Cú pháp:**

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

### **3. Sử dụng:**

* **List:**
  + Lưu trữ các phần tử có thể thay đổi theo thời gian.
  + Làm việc với các cấu trúc dữ liệu động.
  + Sử dụng trong các thuật toán và cấu trúc dữ liệu phức tạp.
* **Tuple:**
  + Lưu trữ các dữ liệu không thay đổi.
  + Làm khóa cho dictionary.
  + Trả về nhiều giá trị từ hàm.
  + Tăng tốc độ thực thi chương trình vì tính bất biến

### **Ví dụ minh họa:**

Python

# List

my\_list = [1, 2, 3, "chuoi", "xoai"]

my\_list[0] = 10 # Thay đổi phần tử đầu tiên

my\_list.append("orange") # Thêm phần tử vào cuối

# Tuple

my\_tuple = (1, 2, 3)

# my\_tuple[0] = 10 # Sẽ gây lỗi vì không thể thay đổi

**Câu 4:**

**Tuple** trong Python là một kiểu dữ liệu bất biến, có nghĩa là các phần tử trong tuple không thể thay đổi sau khi được tạo. Mặc dù có vẻ hạn chế so với list, nhưng tuple lại có những ưu điểm riêng và được ứng dụng rộng rãi trong nhiều tình huống khác nhau.

### **Lý do sử dụng tuple:**

* **Bảo vệ dữ liệu:** Vì tuple không thể thay đổi nên nó giúp bảo vệ dữ liệu khỏi bị sửa đổi một cách vô tình. Điều này rất hữu ích khi bạn muốn đảm bảo rằng một tập hợp các giá trị sẽ không bị thay đổi trong quá trình thực thi chương trình.
* **Tăng hiệu suất:** Tuple thường nhanh hơn list vì chúng không cần phải thực hiện các phép kiểm tra để đảm bảo tính toàn vẹn của dữ liệu khi bị thay đổi.
* **Làm khóa cho dictionary:** Tuple có thể được sử dụng làm khóa cho dictionary vì chúng là hashtable (có thể băm). Điều này có nghĩa là bạn có thể sử dụng tuple để tạo ra các cấu trúc dữ liệu phức tạp hơn.
* **Trả về nhiều giá trị từ hàm:** Một hàm có thể trả về nhiều giá trị bằng cách trả về một tuple.

### **Các ví dụ ứng dụng thực tế:**

1. **Lưu trữ các hằng số:**
   * Các ngày trong tuần: days = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
   * Các tháng trong năm: months = ("January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December")
2. **Lưu trữ các tọa độ:**
   * coordinates = (10, 20)

**3.Trả về nhiều giá trị từ hàm:** Python  
 def findMinMax(numbers):

return min(numbers), max(numbers)

**4. Làm khóa cho dictionary:**  
person = {(1, "Alice"), (2, "Bob")}

**5**.**Định nghĩa cấu trúc dữ liệu đơn giản:**

student = ("Alice", 20, "Computer Science")

**6. Sử dụng trong các thư viện và framework:**

Nhiều thư viện và framework trong Python sử dụng tuple để định nghĩa các cấu trúc dữ liệu cơ bản. Ví dụ, trong thư viện os.path, tuple thường được sử dụng để biểu diễn các đường dẫn đến file và thư mục.