Câu 1

**. Toán tử định dạng chuỗi (f-strings):**

* **Cú pháp:** Bắt đầu bằng chữ f trước dấu ngoặc kép hoặc đơn, sau đó nhúng các biểu thức Python trực tiếp vào chuỗi bằng dấu ngoặc nhọn {}.
* **Ưu điểm:**
  + **Đọc được:** Cú pháp trực quan, gần giống cách viết bình thường.
  + **Hiệu quả:** Nhanh hơn so với các phương pháp khác.
  + **Linh hoạt:** Hỗ trợ nhiều tính năng như định dạng số, gọi hàm, biểu thức phức tạp.
* **Ví dụ:**

Python

name = "Alice"

age = 30

print(f"Hello, my name is {name} and I am {age} years old.")

Hãy [thận trọng](https://d.docs.live.net/faq#coding) khi sử dụng các đoạn mã.

**2. Hàm định dạng chuỗi format():**

* **Cú pháp:** Gọi phương thức format() trên một chuỗi, truyền các giá trị cần định dạng vào trong ngoặc.
* **Ưu điểm:**
  + **Linh hoạt:** Cung cấp nhiều tùy chọn định dạng hơn.
  + **Khả năng tương thích:** Có từ các phiên bản Python cũ hơn.
* **Ví dụ:**

Python

name = "Bob"

age = 25

print("Hello, my name is {} and I am {} years old.".format(name, age))

Hãy [thận trọng](https://d.docs.live.net/faq#coding) khi sử dụng các đoạn mã.

**3. So sánh:**

|  |  |  |
| --- | --- | --- |
| **Tính năng** | **Toán tử f-strings** | **Hàm format()** |
| Cú pháp | Trực quan, nhúng biểu thức | Cần gọi phương thức |
| Tốc độ | Nhanh hơn | Chậm hơn |
| Linh hoạt | Rất cao | Cao |
| Khả năng tương thích | Python 3.6 trở lên | Các phiên bản cũ hơn |

**Ví dụ minh họa**

**1. Sử dụng toán tử %**

 name = "Alice"

age = 30

formatted\_string = "My name is %s and I am %d years old." % (name, age)

print(formatted\_string) # Output: My name is Alice and I am 30 years old.

**2. Sử dụng hàm str.format()**

name = "Alice"

age = 30

formatted\_string = "My name is {} and I am {} years old.".format(name, age)

print(formatted\_string) # Output: My name is Alice and I am 30 years old.

**3. Sử dụng toán tử % với số thực**

pi = 3.14159

formatted\_string = "Value of pi is approximately %.2f." % pi

print(formatted\_string) # Output: Value of pi is approximately 3.14.

**4. Sử dụng hàm str.format() với số thực**

pi = 3.14159

formatted\_string = "Value of pi is approximately {:.2f}.".format(pi)

print(formatted\_string) # Output: Value of pi is approximately 3.14.

**5. Sử dụng toán tử % với nhiều đối số**

 name = "Alice"

age = 30

height = 1.65

formatted\_string = "Name: %s, Age: %d, Height: %.2f" % (name, age, height)

print(formatted\_string) # Output: Name: Alice, Age: 30, Height: 1.65

**6. Sử dụng hàm str.format() với nhiều đối số**

 name = "Alice"

age = 30

height = 1.65

formatted\_string = "Name: {}, Age: {}, Height: {:.2f}".format(name, age, height)

print(formatted\_string) # Output: Name: Alice, Age: 30, Height: 1.65

Câu 2

import random

def generate\_random\_number(start, end):

# Kiểm tra xem đoạn số có hợp lệ không

if start >= end:

return "Đoạn số không hợp lệ. Vui lòng nhập lại."

# Sinh ra số ngẫu nhiên trong đoạn từ start đến end

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

return random\_number

# Nhập vào đoạn số

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

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

# Xuất ra số ngẫu nhiên

result = generate\_random\_number(start, end)

print("Số ngẫu nhiên trong đoạn [{} - {}]: {}".format(start, end, result))

Câu 3

Trong Python, list và tuple đều là các cấu trúc dữ liệu để lưu trữ tập hợp các đối tượng, nhưng chúng có một số khác biệt cơ bản như sau:

1. **Độ thay đổi (Mutability)**:
   * list: Là kiểu dữ liệu thay đổi (mutable), có nghĩa là bạn có thể thêm, xóa hoặc sửa đổi các phần tử bên trong danh sách sau khi nó đã được tạo.
   * tuple: Là kiểu dữ liệu không thay đổi (immutable), có nghĩa là một khi tuple đã được tạo, bạn không thể thay đổi, thêm hoặc xóa các phần tử trong nó.
2. **Cú pháp**:
   * list: Được định nghĩa bằng cách sử dụng dấu ngoặc vuông []. Ví dụ: my\_list = [1, 2, 3]
   * tuple: Được định nghĩa bằng cách sử dụng dấu ngoặc đơn (). Ví dụ: my\_tuple = (1, 2, 3)
3. **Hiệu suất**:
   * list: Thường chậm hơn tuple vì nó có thể thay đổi được. Khi bạn thay đổi kích thước của danh sách, Python cần quản lý bộ nhớ một cách linh hoạt hơn.
   * tuple: Nhanh hơn và tiêu thụ ít bộ nhớ hơn so với list, vì nó không thay đổi.
4. **Phân loại**:
   * list: Thích hợp cho các tập hợp dữ liệu mà bạn cần có khả năng thay đổi.
   * tuple: Thích hợp cho các tập hợp dữ liệu mà bạn muốn giữ nguyên không thay đổi, và có thể dùng làm khóa trong từ điển (dictionary) vì tính bất biến của nó.
5. **Chức năng**:
   * Một số phương thức như append(), remove() chỉ có sẵn cho list, trong khi tuple không có các phương thức này vì nó không thể thay đổi.

Câu 4

Tuple là một kiểu dữ liệu rất hữu ích trong lập trình, khi mà các phần tử trong tuple là không thể thay đổi (immutable) và có thể chứa nhiều loại dữ liệu khác nhau. Dưới đây là một số ứng dụng thực tế của tuple:

1. **Lưu trữ dữ liệu không thay đổi**: Tuple thường được sử dụng để lưu trữ các dữ liệu mà không cần thay đổi, chẳng hạn như các thông số cấu hình hoặc các tham số cố định trong một ứng dụng.
2. **Trả về nhiều giá trị**: Trong một số ngôn ngữ lập trình, tuple có thể được sử dụng để trả về nhiều giá trị từ một hàm. Điều này giúp cho việc truyền và nhận dữ liệu trở nên dễ dàng và gọn gàng hơn.
3. **Lưu trữ thông tin có liên quan**: Tuple có thể được sử dụng để lưu trữ các thông tin có liên quan với nhau. Ví dụ, khi lưu trữ thông tin về một người, ta có thể dùng tuple để lưu trữ tên, tuổi và địa chỉ.

person = ("John Doe", 30, "123 Main St")

1. **Sử dụng trong các cấu trúc dữ liệu phức tạp**: Trong các cấu trúc dữ liệu như dictionary, tuple có thể được sử dụng như là khóa để lưu trữ các thông tin liên kết hoặc bản đồ.
2. **Biểu diễn tọa độ**: Tuple thường được sử dụng để biểu diễn các tọa độ trong không gian 2D hoặc 3D. Ví dụ, tọa độ (x, y) hoặc (x, y, z) có thể được biểu diễn bằng tuple.

point\_2d = (10, 20)

point\_3d = (10, 20, 30)

1. **Lưu trữ dữ liệu trong cơ sở dữ liệu**: Tuple có thể được sử dụng để lưu trữ một hàng trong một bảng cơ sở dữ liệu. Mỗi tuple sẽ đại diện cho một bản ghi duy nhất với các trường dữ liệu tương ứng.
2. **Thay thế cho danh sách khi không cần thay đổi**: Khi bạn cần một danh sách mà không cần thực hiện các thao tác thay đổi (như thêm, xóa, sửa), tuple là lựa chọn tốt hơn vì nó giúp bảo vệ dữ liệu khỏi các thay đổi ngoài ý muốn.
3. **Tuple unpacking**: Cú pháp unpacking của tuple làm cho việc gán giá trị cho nhiều biến trở nên dễ dàng và nhanh chóng.

coordinates = (10, 20)

x, y = coordinates