**Câu hỏi tổng hợp**

**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?**

a. Toán tử định dạng chuỗi (%)

Toán tử % là một cách cũ để định dạng chuỗi, tương tự như cách định dạng chuỗi trong ngôn ngữ C. Nó có thể sử dụng các ký tự định dạng như %s, %d, %f, v.v.

Ví dụ 1:

ten = "Anhvy"

tuoi = 19

print("Tên của tôi là %s và tôi %d tuổi." % (ten, tuoi))

Kết quả:

Tên của tôi là Anhvy và tôi 19 tuổi.

b. Hàm định dạng chuỗi str.format()

Hàm str.format() là cách định dạng chuỗi cung cấp nhiều tính năng hơn so với toán tử %, chẳng hạn như khả năng sắp xếp và định dạng các tham số.

Ví dụ 2:

ten = "Duy"

age = 18

print("Tên của tôi là {} và tôi {} tuổi.".format(ten, tuoi))

Kết quả:

Tên của tôi là Duy và tôi 18 tuổi.

Ví dụ 3:

print("Tên của tôi là {0} và tôi {1} tuổi. {0} thật tuyệt!".format(ten, tuoi))

Kết quả:

Tên của tôi là Duy và tôi 18 tuổi. Duy thật tuyệt!

c. F-strings (chuỗi định dạng theo f-string)

F-strings cho phép bạn nhúng các biểu thức trực tiếp vào chuỗi bằng cách sử dụng dấu {} và tiền tố f trước chuỗi.

Ví dụ 4:

ten = "An"

tuoi = 20

print(f"Tên của tôi là {ten} và tôi {tuoi} tuổi.")

Kết quả:

Tên của tôi là An và tôi 20 tuổi.

Ví dụ 5:

chieu\_cao = 1.79

can\_nang = 70

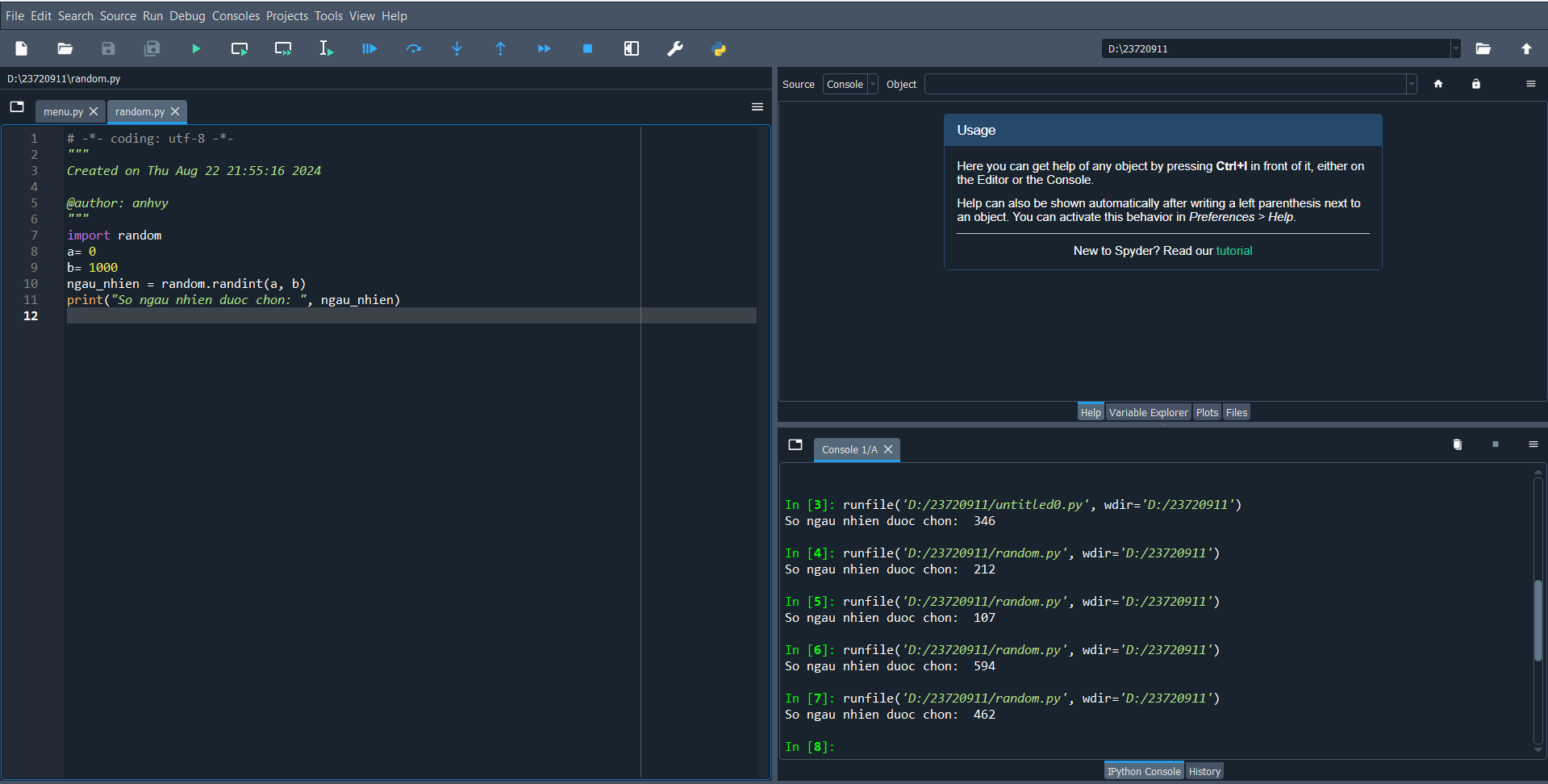
print(f"Chiều cao của tôi là {chieu\_cao:.2f} mét và cân nặng của tôi là {can\_nang} kg.")

Kết quả:

Chiều cao của tôi là 1.79 mét và cân nặng của tôi là 70 kg.

**2. Viết chương trình xuất ra số ngẫu nhiên trong một đoạn bất kỳ bất cho**

**trước?**



**3. Khác biệt cơ bản giữa list và tuple?**

Tính bất biến (Immutability)

* list: có thể thay đổi, thêm hoặc xóa phần tử.
* tuple: sau khi tạo, không thể thay đổi các phần tử.

Ví dụ:

# List

my\_list = [1, 2, 3]

my\_list[1] = 10 # Thay đổi giá trị

my\_list.append(4) # Thêm phần tử

# Tuple

my\_tuple = (1, 2, 3)

# my\_tuple[1] = 10 # Lỗi: 'tuple' object does not support item assignment

b. Cú pháp

* list: Dùng dấu ngoặc vuông [].
* tuple: Dùng dấu ngoặc đơn ().

Ví dụ:

# List

my\_list = [1, 2, 3]

# Tuple

my\_tuple = (1, 2, 3)

c. Hiệu suất

* list: Thường chậm hơn vì có thể thay đổi kích thước.
* tuple: Nhanh hơn vì không thay đổi kích thước.

d. Khả năng sử dụng làm khóa từ điển

* list: Không thể dùng làm khóa từ điển.
* tuple: Có thể dùng làm khóa từ điển nếu tất cả phần tử trong tuple cũng bất biến.

Ví dụ:

# Không thể dùng list làm khóa từ điển

# my\_dict = {my\_list: "value"} # Lỗi

# Có thể dùng tuple làm khóa từ điển

my\_dict = {my\_tuple: "value"}

e. Tính năng

* list: Nhiều phương thức tích hợp như .append(), .remove().
* tuple: Ít phương thức hơn, chủ yếu có .count() và .index().

Ví dụ:

# List

my\_list = [1, 2, 3]

my\_list.append(4) # Thay đổi danh sách

# Tuple

my\_tuple = (1, 2, 3)

print(my\_tuple.count(2)) # Đếm số lần xuất hiện của giá trị 2

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

- Nhóm dữ liệu liên quan: Tạo các nhóm giá trị mà không cần thay đổi.

Ví dụ:

thongtin\_nguoidung = ("Vy", 19, "Sinh viên") # (ten, tuoi, nghe\_nghiep)

- Trả về nhiều giá trị từ hàm: Đóng gói các giá trị trả về từ một hàm.

def thongtin():

ten = "Vy"

tuoi = 19

nghenghiep = "Sinh vien"

return (ten, tuoi, nghenghiep)

nguoi = thongtin()

print(nguoi) # Output: ('Vy', 19, 'Sinh vien')

- Khóa trong từ điển: Sử dụng làm khóa trong từ điển.

nguoi = { (1.69, 59): "Nguyen", (1.79, 69): "An"}

print(nguoi[(1.69, 59)]) # Output: Nguyen

- Lưu trữ các giá trị bất biến: Đảm bảo dữ liệu không bị thay đổi.

# Danh sách các tháng trong năm

thang = ("Thang 1", "Thang 2", "Thang 3", "Thang 4", "Thang 5", "Thang 6 ", "Thang 7 ", "Thang 8 ", "Thang 9 ", "Thang 10 ", "Thang 11 ", "Thang 12")

# Sử dụng tuple như một danh sách không thay đổi

print(thang[5]) # Output: Thang 6

- Thư viện và API: Được sử dụng trong các thư viện và API để đảm bảo tính nhất quán.

import os

# Lấy thông tin về một tệp tin

tep\_thongtin = os.stat('vidu.txt')

print(tep\_thongtin) # Trả về một tuple chứa thông tin về tệp tin

- Giữ cho các giá trị đồng nhất

Khi bạn cần đảm bảo rằng một nhóm các giá trị không bị thay đổi, tuple là lựa chọn phù hợp. Ví dụ, bạn có thể sử dụng nó để xác định các trạng thái của một đối tượng mà không muốn bị thay đổi.

Ví dụ:

# Trạng thái của một đối tượng

trang\_thai = ("hoat dong", "dung", "hoan thanh")

doituong = trang\_thai[2] # "hoan thanh"