Chương 4. Câu lệnh Rẽ nhánh If

NGUYỄN HOÀNG ANH

CNTT1. PTIT

Nội dung trình bầy

- Ví dụ
- Kiểm tra có điều kiện
- Câu lệnh If
- Câu lệnh If với danh sách

4.1. Ví dụ

Đoạn mã sau đây lặp lại danh sách tên xe và tìm kiếm giá trị 'bmw'. Bất cứ khi nào giá trị là 'bmw', nó sẽ được in bằng chữ hoa thay vì chữ hoa tiêu đề

```
cars = ['audi', 'bmw', 'subaru', 'toyota']
for car in cars:
    if car=='bmw':
        print(car.upper())
    else:
        print(car.title())
```

```
Audi
BMW
Subaru
Toyota
```

4.2. Kiểm tra có điều kiện

Ở trung tâm của mọi câu lệnh if là biểu thức mà có thể được đánh giá là True hoặc False và nó được gọi là kiểm tra có điều kiện.

Python sử dụng các giá trị True và False để quyết định xem mã trong câu lệnh if có được thực thi hay không.

Nếu một kiểm tra có điều kiện đánh giá là True, Python thực thi mã sau câu lệnh if. Nếu kiểm tra đánh giá là False, Python sẽ bỏ qua mã sau câu lệnh if.

Kiểm tra đẳng thức

Hầu hết các kiểm tra có điều kiện đều so sánh giá trị hiện tại của một biến với một giá trị cụ thể.

Câu kiểm tra có điều kiện đơn giản nhất là kiểm tra giá trị của một biến có bằng một giá trị cụ thể nào không.

```
>>> car = 'bmw'
>>> car == 'bmw'
True
```

```
>>> car = 'audi'
>>> car == 'bmw'
False
```

Trong ví dụ trên, dấu bằng '=' chính là một câu, thực hiện lệnh gán. Trong khi đó, dấu bằng kép '==' lại là khiến nó trở thành câu hỏi: "Liệu giá trị của biến car có bằng 'bmw' hay không?".

Bỏ qua chữ viết hoa khi kiểm tra đẳng thức

Kiểm tra đẳng thức là trường hợp phân biệt chữ hoa và chữ thường trong Python. Ví dụ, hai giá trị với chữ viết hoa khác nhau không được coi là bằng nhau

```
>>> car = 'Audi'
>>> car == 'audi'
False
```

Nếu chỉ muốn kiểm tra giá trị của một biến, ta có thể chuyển đổi giá trị của biến thành chữ thường trước khi thực hiện so sánh:

```
>>> car = 'Audi'
>>> car.lower() == 'audi'
True
```

Hàm lower() không thay đổi giá trị ban đầu được lưu trong biến car, do đó chúng ta có thể thực hiện loại so sánh này mà không làm ảnh hưởng tới giá trị ban đầu của biến.

```
>>> car = 'Audi'
>>> car.lower() == 'audi'
True
>>> car
'Audi'
```

Kiểm tra bất đẳng thức

Khi muốn kiểm tra hai giá trị không bằng nhau, kết hợp dấu chấm than với dấu bằng (!=).

Dấu chấm than có nghĩa là không(not) và được sử dụng trong nhiều ngôn ngữ lập trình

```
requested_topping = 'mushrooms'
if requested_topping != 'anchovies':
    print("Hold the anchovies!")
```

```
Hold the anchovies!
```

So sánh số

Mã nguồn sau đây kiểm tra xem một người đã đủ 18 tuổi hay chưa

```
>>> age = 18
>>> age == 18
True
```

có thể kiểm tra nếu hai số không bằng nhau

```
answer = 17
if answer != 42:
    print("That is not the correct answer. Please try again!")
That is not the correct answer. Please try again!
```

So sánh 2 số

Có thể bao gồm các phép so sánh toán học khác nhau trong các câu lệnh, chẳng hạn như nhỏ hơn, nhỏ hơn hoặc bằng, lớn hơn, và lớn hơn hoặc bằng

```
>>> age = 19
>>> age < 21
True
>>> age <= 21
True
>>> age > 21
False
>>> age >= 21
False
```

Kiểm tra nhiều điều kiện

SỬ DỤNG AND

Kiểm tra xem hai điều kiện có đồng thời là True hay không, hãy sử dụng từ khóa and để kết hợp hai kiểm tra điều kiện

```
>>> age_0 = 22
>>> age_1 = 18
>>> age_0 >= 21 and age_1 >= 21
False
>>> age_1 = 22
>>> age_0 >= 21 and age_1 >= 21
True
```

Có thể sử dụng dấu ngoặc đơn xung quanh biểu thức kiểm tra, nhưng chúng không bắt buộc

```
(age_0 >= 21) and (age_1 >= 21)
```

Kiểm tra nhiều điều kiện – Dùng **OR**

Từ khóa Or cho phép chúng ta kiểm tra nhiều điều kiện, nhưng nó vượt qua khi một trong hai hoặc cả hai bài kiểm tra riêng lẻ vượt qua. Một biểu thức Or chỉ thất bại khi cả hai bài kiểm tra riêng lẻ đều thất bại.

①
```
>>> age_0 = 22
>>> age_1 = 18
```
②
```
>>> age_0 >= 21 or age_1 >= 21
True
```
③
```
>>> age_0 = 18
>>> age_0 >= 21 or age_1 >= 21
False
```

Kiểm tra xem một giá trị có trong danh sách hay không

Để tìm hiểu xem một giá trị cụ thể đã có trong danh sách hay chưa, hãy sử dụng từ khóa **In**.

```
>>> requested_toppings = ['mushrooms', 'onions', 'pineapple']
>>> 'mushrooms' in requested_toppings
True
>>> 'pepperoni' in requested_toppings
False
```

Từ khóa **in** yêu cầu Python kiểm tra sự tồn tại của 'mushrooms' và 'pepperoni' trong danh sách requested_toppings. Kỹ thuật này là khá mạnh mẽ vì chúng ta có thể tạo danh sách các giá trị thiết yếu và sau đó dễ dàng kiểm tra xem giá trị ta đang kiểm tra có khớp với một trong các giá trị trong danh sách hay không.

Kiểm tra một giá trị không nằm trong danh sách

sử dụng từ khóa *not in* kiểm tra một phần tử không nằm trong danh sách

```
banned_users = ['andrew', 'carolina', 'david']
user = 'marie'
if user not in banned_users:
    print(f"{user.title()}, you can post a response if you wish.")
```

```
Marie, you can post a response if you wish.
```

4.3. Câu lệnh If

Khi hiểu các câu kiểm tra điều kiện, ta có thể bắt đầu viết câu lệnh if. Một số loại câu lệnh if khác nhau tồn tại và sự lựa chọn sử dụng phụ thuộc vào số lượng điều kiện ta cần kiểm tra.

Câu lệnh if đơn giản

Loại câu lệnh if đơn giản nhất có một kiểm tra và một hành động

```
if conditional_test:
    do something
```

```
    age = 19
①   if age >= 18:
②       print("You are old enough to vote!")
```

```
You are old enough to vote!
```

Chúng ta có thể có bao nhiêu dòng mã tùy thích trong khối theo sau câu lệnh if.

```
age = 19
if age >= 18:
    print("You are old enough to vote!")
    print("Have you registered to vote yet?")
```

```
You are old enough to vote!
Have you registered to vote yet?
```

Câu lệnh if-else

Một khối if-else tương tự như một câu lệnh if đơn giản, nhưng câu lệnh else cho phép ta xác định một hành động hoặc một tập hợp các hành động được thực thi khi kiểm tra điều kiện không thành công.

```
    age = 17
①   if age >= 18:
        print("You are old enough to vote!")
        print("Have you registered to vote yet?")
②   else:
        print("Sorry, you are too young to vote.")
        print("Please register to vote as soon as you turn 18!")
```

```
Sorry, you are too young to vote.
Please register to vote as soon as you turn 18!
```

Chuỗi if-elif-else

hãy xem xét một công viên giải trí tính phí các mức giá khác nhau cho các nhóm tuổi khác nhau

- Miễn phí vé vào cửa cho bất kỳ ai dưới 4 tuổi.
- Vé vào cửa cho bất kỳ ai trong độ tuổi từ 4 đến 18 là $ 25.
- Vé vào cửa cho bất kỳ ai từ 18 tuổi trở lên là $ 40.

```
    age = 12
①   if age < 4:
        print("Your admission cost is $0.")
②   elif age < 18:
        print("Your admission cost is $25.")
③   else:
        print("Your admission cost is $40.")
```

```
Your admission cost is $25
```

Chuỗi if-elif-else (t)

Thay vì in giá vào cửa trong khối if-elif-else, sẽ ngắn gọn hơn nếu chỉ đặt giá bên trong chuỗi if-elif-else và sau đó có một lệnh gọi print() đơn giản chạy sau khi chuỗi đã được đã đánh giá

```
    age = 12
    if age < 4:
①       price = 0
    elif age < 18:
②       price = 25
    else:
③       price = 40
④   print(f"Your admission cost is ${price}.")
```

Sử dụng nhiều khối elif

Có thể sử dụng bao nhiêu khối elif trong mã tùy thích

Giả sử rằng bất kỳ ai 65 tuổi trở lên trả một nửa số tiền vào cửa thông thường, hoặc $ 20:

```
age = 12
if age < 4:
    price = 0
elif age < 18:
    price = 25
① elif age < 65:
    price = 40
② else:
    price = 20
print(f"Your admission cost is ${price}.")
```

Bỏ qua khối else

Python không yêu cầu một khối else ở cuối chuỗi if-elif.

Đôi khi một khối else hữu ích; đôi khi việc sử dụng một khối elif khác khiến nó rõ ràng hơn

```
age = 12
if age < 4:
    price = 0
elif age < 18:
    price = 25
elif age < 65:
    price = 40
① elif age >= 65:
    price = 20
print(f"Your admission cost is ${price}.")
```

Khối elif bổ sung ở ① ấn định giá 20 đô la khi người đó 65 tuổi hoặc già hơn, rõ ràng hơn một chút so với khối else. Với sự thay đổi này, mọi khối mã phải vượt qua một bài kiểm tra riêng biệt để được thực thi.

Kiểm tra nhiều điều kiện

Đôi khi điều quan trọng là phải kiểm tra tất cả các điều kiện của quan tâm. Trong trường hợp này, ta nên sử dụng một loạt các câu lệnh if đơn giản với không elif hoặc các khối khác

```
①  requested_toppings = ['mushrooms', 'extra cheese']
②  if 'mushrooms' in requested_toppings:
       print("Adding mushrooms.")
③  if 'pepperoni' in requested_toppings:
       print("Adding pepperoni.")
④  if 'extra cheese' in requested_toppings:
       print("Adding extra cheese.")
   print("\nFinished making your pizza!")
```

```
Adding mushrooms.
Adding extra cheese.

Finished making your pizza!
```

Kiểm tra nhiều điều kiện (t)

Mã này sẽ không hoạt động đúng nếu chúng ta sử dụng khối if-elif-else, bởi vì mã sẽ ngừng chạy sau khi chỉ có một thử nghiệm vượt qua.

```
requested_toppings = ['mushrooms', 'extra cheese']
if 'mushrooms' in requested_toppings:
    print("Adding mushrooms.")
elif 'pepperoni' in requested_toppings:
    print("Adding pepperoni.")
elif 'extra cheese' in requested_toppings:
    print("Adding extra cheese.")
print("\nFinished making your pizza!")
```

```
Adding mushrooms.

Finished making your pizza!
```

4.4. Câu lệnh If với danh sách

Ta có thể thực hiện một số công việc thú vị khi kết hợp danh sách và câu lệnh if.

Ta có thể xem các giá trị đặc biệt cần được xử lý khác hơn so với các giá trị khác trong danh sách

```
requested_toppings = ['mushrooms', 'green peppers', 'extra cheese']

for requested_topping in requested_toppings:
    print(f"Adding {requested_topping}.")
print("\nFinished making your pizza!")
```

```
Adding mushrooms.
Adding green peppers.
Adding extra cheese.

Finished making your pizza!
```

Kiểm tra các phần tử đặc biệt

Một câu lệnh if bên trong vòng lặp for có thể xử lý tình huống nhà hàng hết ớt xanh một cách thích hợp:

```
requested_toppings = ['mushrooms', 'green peppers', 'extra cheese']
for requested_topping in requested_toppings:
①       if requested_topping == 'green peppers':
            print("Sorry, we are out of green peppers right now.")
②       else:
            print(f"Adding {requested_topping}.")
print("\nFinished making your pizza!")
```

```
Adding mushrooms.
Sorry, we are out of green peppers right now.
Adding extra cheese.

Finished making your pizza!
```

Kiểm tra một danh sách không rỗng

kiểm tra xem danh sách các lớp phủ được yêu cầu có trống trước khi xây dựng bánh pizza. Nếu danh sách trống, chúng ta sẽ nhắc người dùng và đảm bảo rằng họ muốn có một chiếc bánh pizza đơn giản.

```
①   requested_toppings = []
②   if requested_toppings:
        for requested_topping in requested_toppings:
            print(f"Adding {requested_topping}.")
        print("\nFinished making your pizza!")
③   else:
        print("Are you sure you want a plain pizza?")
```

```
Are you sure you want a plain pizza?
```

Sử dụng nhiều danh sách

Ví dụ sau xác định hai danh sách. Đầu tiên là danh sách các lớp phủ có sẵn tại tiệm bánh pizza và thứ hai là danh sách các lớp phủ mà người dùng có yêu cầu.

Lần này, mỗi mục trong request_toppings được kiểm tra dựa trên danh sách các lớp phủ có sẵn trước khi thêm vào bánh pizza:

```
① available_toppings = ['mushrooms', 'olives', 'green peppers', 'pepperoni', 'pineapple', 'extra cheese']
② requested_toppings = ['mushrooms', 'french fries', 'extra cheese']
③ for requested_topping in requested_toppings:
④     if requested_topping in available_toppings:
          print(f"Adding {requested_topping}.")
⑤     else:
          print(f"Sorry, we don't have {requested_topping}.")
print("\nFinished making your pizza!")
```

Kết quả:

```
Adding mushrooms.
Sorry, we don't have french fries.
Adding extra cheese.

Finished making your pizza!
```

Kết chương

Trong chương này, chúng ta đã học

- cách viết các bài kiểm tra có điều kiện, luôn luôn đánh giá True hay False.
- cách viết các câu lệnh if đơn giản, chuỗi if-else và chuỗi if-elif-else.
- sử dụng các cấu trúc này để xác định các điều kiện cụ thể cần để kiểm tra và biết khi nào các điều kiện đó đã được đáp ứng
- cách xử lý các phần tử nhất định trong danh sách khác với tất cả các mục khác trong khi tiếp tục sử dụng hiệu quả của vòng lặp for

Trong chương tiếp theo, ta sẽ tìm hiểu về từ điển của Python. Từ điển tương tự như một danh sách, nhưng nó cho phép ta kết nối các phần thông tin. Ta sẽ học cách xây dựng từ điển, lặp qua chúng và sử dụng chúng kết hợp với danh sách và câu lệnh if. Tìm hiểu về từ điển sẽ cho phép chúng ta có thể mô hình hóa nhiều tình huống trong thế giới thực hơn nữa.

Bài tập