7. Điều kiện và lặp

Về cơ bản câu lệnh điều kiện If cũng mang cùng ý nghĩa như các ngôn ngữ lập trình khác, đó là cho phép bạn điều khiển luồng thực thi của chương trình hay cụ thể hơn là cho phép bạn thực thi một số câu lệnh khi một trong số các điều kiện đưa ra được ước lượng là true.

Hình thức chung của câu lệnh if trong Python giống như sau:

```
if condition_1:
    statement_block_1
elif condition_2:
    statement_block_2
else:
    statement_block_3
```

Nếu điều kiện "condition_1" là Đúng, các câu lệnh trong khối lệnh statement_block_1 sẽ được thực hiện. Nếu không, điều kiện “condition_2” sẽ được thực hiện. Nếu condition_2 ước lượng là True, statement_block_2 sẽ được thực hiện, nếu condition_2 là Sai, các câu lệnh trong statement_block_3 sẽ được thực thi.

Đánh giá “True” hoặc “False”:

Trong mỗi ngôn ngữ lại có cách đánh giá False khác nhau, ví dụ trong C hay C++ thì 0 được hiểu là False. Các đối tượng sau được đánh giá là False trong Python:

Số có giá trị là zero (0, 0L, 0.0, 0.0 + 0.0j),

Giá trị Boolean False

Chuỗi rỗng . e.g., “”

Danh sách rỗng hay tập rỗng. e.g., [],()

Dictionary rỗng. e.g., {}

Giá trị None.

Các đối tượng ngoài danh sách phía trên được đánh giá là True.

if Rút gọn:

C code	Rút gọn trong C	Rút gọn trong python

if (a > b)
  max=a;
else
  max=b;
	max = (a > b) ? a : b;	max = a if (a > b) else b;

Lặp (Loop), While

Có một cấu trúc mà làm cho đoạn mã có thể thực hiện một chuỗi các lệnh lặp đi lặp lại là Loop, điều này cần thiết ở nhiều thuật toán trong một ngôn ngữ lập trình. Mã bên trong vòng lặp, hay mã được thực hiện lặp đi lặp lại được gọi là phần thân của vòng lặp.

Python cung cấp hai loại vòng lặp khác nhau: vòng lặp while và vòng lặp for. Cấu trúc lặp có thể được khái quát như hình vẽ bên dưới.

Ví dụ:

```
>>> count = 10
>>> while count > 5:
...         count -=2
...
>>> print count
4
```

Tương tự như câu lệnh if, vòng lặp while của Python cũng có một phần tùy chọn khác. Đây là một cấu trúc không quen thuộc đối với nhiều lập trình viên của các ngôn ngữ lập trình truyền thống.

Các câu lệnh trong phần “else” được thực hiện, khi không thỏa điều kiện lặp nữa. Một số có thể tự hỏi mình bây giờ, lợi ích của phần bổ sung này là gì? Nếu các câu lệnh của phần bổ sung khác được đặt ngay sau vòng lặp while mà không có một else, chúng sẽ được thực hiện dù có hay không, đúng không. Sự khác biệt thuộc về xử lý sau while rơi vào trường hợp **vòng lặp bị thoát bởi ‘break’.** Cách thức hoạt động như hình phía bên phải. Bạn có thể để ý thấy có 2 câu lệnh đặc biệt được sử dụng trong vòng lặp while là break và continue. Chúng tôi sẽ nói rõ hơn ý nghĩa và sự khác biệt của chúng trong ví dụ phía dưới.

Cú pháp tổng quát của vòng lặp while giống như sau:

```
while condition:
statement_1
...
statement_n
else:
statement_1
...
statement_n
```

Phân tích ví dụ sau: Trong ví dụ này các bạn sẽ nhận thấy “ELSE” không được in ra do kết thúc “while” không phải do không thỏa mãn điều kiện “i<5” mà là do bị tác động bởi “break”.

```
i = 0
while i < 5:
    print i
    if i == 2:
        break
    i = i +1
else:
    print 'ELSE'

print 'The next statement'



Output:
```

```
0

1

2

The next statement
```

Ví dụ trên miêu tả kết thúc sớm của vòng lặp:

Thường, một vòng lặp chỉ kết thúc nếu điều kiện trong đầu vòng lặp được hoàn thành. Với sự trợ giúp của lệnh “break”, một vòng lặp trong quá trình thực thi có thể bị thoát sớm, nghĩa là ngay khi luồng của chương trình đi đến một điểm thỏa mãn “break” bên trong vòng lặp “while” (hoặc các vòng lặp khác) vòng lặp sẽ ngay lập tức bị “break”. **"break"** không nên nhầm lẫn với câu lệnh continue. **"continue"** sẽ bỏ qua các lệnh phía sau của vòng lặp hiện tại và bắt đầu lặp lại nếu vẫn thỏa mãn điều kiện lặp. Bây giờ đến điểm quan trọng: Nếu một vòng lặp gặp break, phần mã thuộc “else” không được thực hiện.

Ví dụ bài toán đoán số ngẫu nhiên tiếp theo là số nào:

```
import random
n = 20
to_be_guessed = int(n * random.random()) + 1
guess = 0
while guess != to_be_guessed:
    guess = input("Nhap mot so duong: ")
    if guess > 0:
        if guess > to_be_guessed:
            print "Can so nho hon"
        else:
            print "Can so lon hon"
    else:
        print "Ban bo cuoc "
        break
else:
    print "Congratulation. Doan chinh xac!!!"

Output:

Can so nho hon

Nhap mot so duong: 12

Can so nho hon

Nhap mot so duong: 8

Can so lon hon

Nhap mot so duong: 9

Can so lon hon

Nhap mot so duong: 10

Can so lon hon

Congratulation. Doan chinh xac!!!
```

Lặp (Loop), For

Cú pháp của cấu trúc For	Ví dụ về iterating trong một danh sách

for variable in sequence:
  Statement1
  Statement2
  ...
  Statementn
else:
  Else-Statement1
  Else-Statement2
  ...
  Else-Statementm
	for x in "Hoc ngon ngu python".split(" "):
	  print x
	
	Output:
	Hoc
	ngon
	ngu
	python

Tip: range() là hàm giúp ta có được một chuỗi số. Giải quyết câu hỏi làm sao viết được phép lặp n lần.

Đơn giản, ta có thể dùng range(n) để tạo ra một chuỗi số từ 0 đến n-1. Hoặc dùng range(x,y) để tạo ra một chuỗi số từ x đến y-1.

Ví dụ:

```
>>> for n in range(3):
...     print n
...
0
1
2
>>> for n in range(2,5):
...     print n
...
2
3
4
```

Kết Luận

Như vậy chúng tôi đã trình bày khá chi tiết về các câu lệnh điều kiện (If) và lặp (while, for). Với các bạn đã làm quen với các ngôn ngữ lập trình khác thì việc tiếp cận bài học này cũng không

phức tạp, các bạn chỉ cần chú ý những điểm khác biệt về cú pháp của các câu lệnh này trong Python. Điểm nhấn trong bài là cấu trúc tùy chọn while…else khá lạ và cách thức hoạt động của nó khi kết hợp với câu lệnh đặc biệt “**break**” và “**continue**”. Ngoài ra, Python không cung cấp các lệnh switch hoặc case như trong các ngôn ngữ lập trình khác, tuy nhiên bạn có thể sử dụng các lệnh if…elif để thực hiện vai trò như của switch hoặc case.

Lần tới chúng tôi sẽ tiếp tục giới thiệu đến các bạn về kiểu dữ liệu tuần tự, ngoài những kiểu dữ liệu mà chúng tôi đã giới thiệu trong bài số 3 về “Biến và Kiểu biến” trong chuỗi bài học Python cơ bản.