Vòng lặp trong python

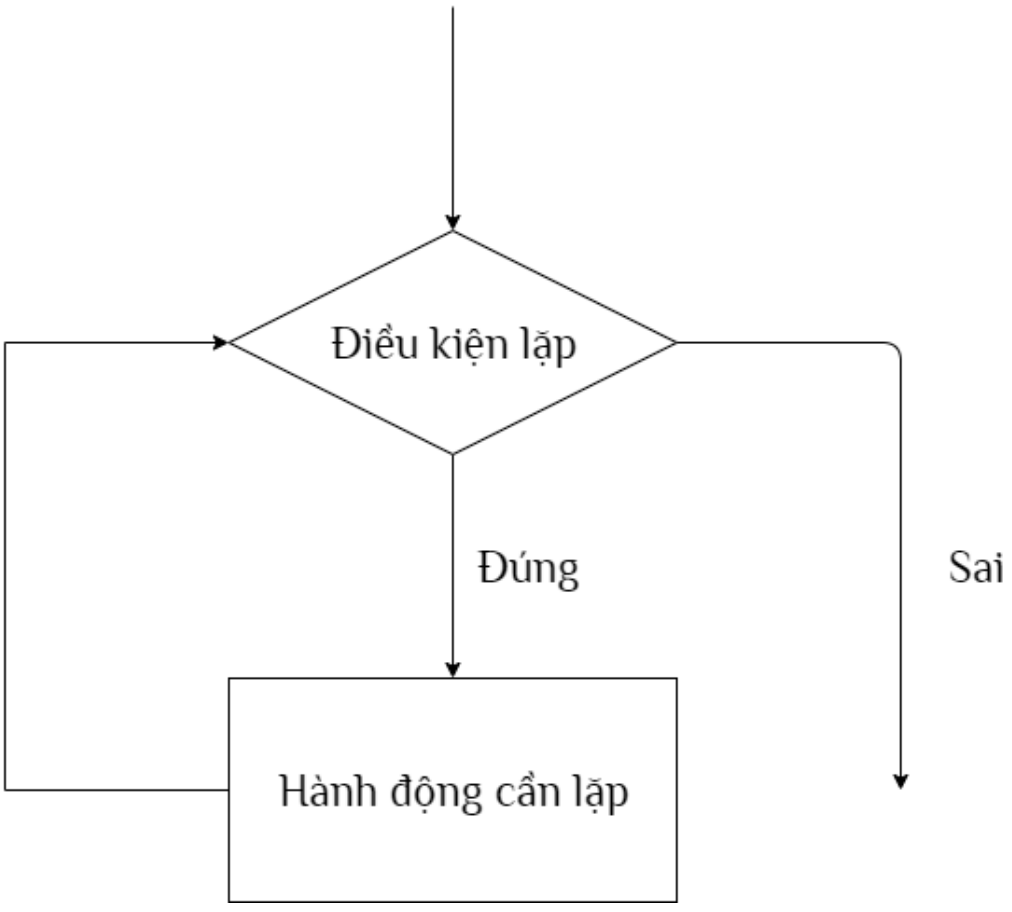
24-12-2020

1 Giới thiệu

Trong việc lập trình, chúng ta thường gặp phải vấn đề phải thực hiện một hành động lặp đi lặp lại, chẳng hạn như việc tính tổng $S = 1 + 2 + 3 + ... + 100$.

Trong lập trình, người ta sử dụng cấu trúc lặp để chỉ cho máy tính biết phải thực hiện một hành động lặp đi lặp lại.

Sơ đồ của cấu trúc lặp như sau :



Sơ đồ lặp

Ví dụ như trong việc tính tổng ở trên, bạn có thể thực hiện nó theo cách sau : 1. Cho biến `i` bằng 0, cho biến `S` bằng 0. 2. Nếu `i` > 100, kết thúc vòng lặp. 3. Gán `S` mới bằng `S` cũ cộng thêm `i`, tăng biến `i` thêm 1 rồi quay lại bước 2. 4. Xuất ra kết quả.

Tùy thuộc vào điều kiện, bạn có thể chia thành 2 dạng lặp: * Lặp theo số lần biết trước. * Lặp theo điều kiện.

2 Lặp theo số lần biết trước trong python

2.1 Cấu trúc cơ bản

Để lặp lại một hành động nào đó với số lần là biết trước, bạn dùng cú pháp sau :

```
for <biến_đếm> in range(<số_lần_lặp>):
    <hành_động>
```

Trong đó : - `biến_đếm` chỉ cần khai báo lúc viết vòng `for`. - `hành_động` có thể phụ thuộc vào `biến_đếm` hoặc không.

Cú pháp trên sẽ thực hiện các hành động sau : 1. Gán `biến_đếm` bằng 0. 2. Thực hiện `hành_động`. 3. Tăng `biến_đếm` thêm 1 đơn vị. Nếu `biến_đếm` bằng số lần lặp thì dừng lại, nếu không quay lại bước 2.

```
In [1]: # in câu `Hello World!` 5 lần
        # đây là hành động không phụ thuộc vào biến đếm
        for i in range(5):
            print('Hello World!')
```

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

```
In [2]: # hành động phụ thuộc biến đếm
        for k in range(5):
            print(k)
```

```
0
1
2
3
4
```

```
In [3]: # tính tổng S = 1 + 2 + 3 + ... + 100
        # đây là hành động phụ thuộc vào biến đếm
        S = 0
        for i in range(101):
            S = S + i
        print(S)
```

```
5050
```

2.2 Hàm `range()`

Hàm `range()` thường được dùng để tạo một đối tượng đếm cho vòng `for`.

Dạng đầy đủ của hàm `range()` là :

```
range([bắt_đầu], <kết_thúc>, [bước_nhảy])
```

Trong đó : - `bắt_đầu`, `kết_thúc` và `bước_nhảy` là số nguyên. - Có thể bỏ qua `bắt_đầu`, python sẽ hiểu là `bắt_đầu` là 0. - Có thể bỏ qua `bước_nhảy`, python sẽ hiểu là `bước_nhảy` là 1.

```
In [4]: # bước nhảy dương
        for i in range(1, 10, 2) :
            print(i)
```

```
1
3
5
7
9
```

```
In [5]: # bước nhảy âm
        for i in range(10, 1, -2) :
          print(i)
```

```
10
8
6
4
2
```

Bài tập : Viết chương trình tính tổng những số tự nhiên chia hết cho 3 và nhỏ hơn 1000.

2.3 Lặp qua các phần tử của một danh sách

Bài tập: Cho danh sách `a = [100, 104, 34, 95]`, in ra màn hình tổng của các phần tử trong danh sách `a`?

Để lặp qua các phần tử của một danh sách `a` nào đó, người ta có thể viết

```
for i in range(len(a)):
    <hành_động_với_a[i]>
```

Tuy nhiên, trong python, bạn có thể viết như sau:

```
for i in a:
    <hành_động_với_i>
```

```
In [6]: s = [100, 104, 34, 95]
        t = 0
        for i in s:
            t = t + i
        print(t)
```

```
333
```

Bạn có thể dùng cú pháp tương tự để lặp qua một `set` hoặc lặp qua từng ký tự của một chuỗi.

```
In [7]: # lặp qua một set
        s = {1, 2, 3, 4}
        for i in s:
            print(i)

1
2
3
4


In [8]: # lặp qua các ký tự của một chuỗi
        s = 'abcxyz'
        for c in s:
            print(c)

a
b
c
x
y
z
```

2.4 Lặp qua một dictionary

Nhắc lại, `dictionary` là tập hợp của một cặp khóa và giá trị, nên khi lặp qua `dictionary`, người ta thường dùng cú pháp:

```
for k, v in <tên_dict>.items():
    <hành_động>
```

```
In [9]: d = {'a': 102, 'b': 103}
        for k, v in d.items():
            print(k, ':', v)

a : 102
b : 103
```

3 Lặp theo điều kiện trong python

Bên cạnh việc lặp theo số lần biết trước, bạn còn có một cấu trúc để lặp theo điều kiện, tức là khi nào điều kiện còn đúng thì bạn còn phải thực hiện hành động.

Để thể hiện việc này trong python, bạn viết như sau :

```
while <điều_kiện> :
    <hành_động>
```

```
In [10]: # tính tổng S = 1 + 2 + 3 + ... + 100 bằng vòng while
         i = 0
         S = 0

         while i <= 100 :
             S = S + i
             i = i + 1

         print(S)
```

```
5050
```

Lưu ý: Khi thực hiện vòng `while`, phải đảm bảo rằng vòng lặp phải kết thúc sau một số hữu hạn bước.

```
In [11]: # đừng chạy cái này nhé!
         #while True:
         #  print('Hello')
```

Bài tập : Viết chương trình nhập vào một số. Nếu là số chẵn, in số đó ra, nếu là số lẻ, yêu cầu nhập lại.

4 Các từ khoá `break`, `continue`

Như các bạn có thể thấy trong sơ đồ trên, python sẽ thực hiện hành động cho đến khi điều kiện không được thoả.

Tuy nhiên, xem xét các tình huống sau: - Bạn tìm kiếm trong một tập hợp và bạn muốn chấm dứt vòng lặp khi tìm thấy. - Bạn muốn bỏ qua một số hành động khi gặp phải đối tượng nào đó.

Trong các tình huống trên, nếu thực hiện vòng lặp có thể gây lãng phí.

Vì vậy, python cung cấp 2 từ khóa `break` và `continue` để có thể thay đổi cách lặp.

4.1 Từ khoá `break`

Khi gặp từ khoá `break`, python sẽ lập tức thoát khỏi vòng lặp.

```
In [12]: # ví dụ break
         for i in range(10) :
             if i > 5 :
                 break
             print(i)
         print('Hết vòng lặp')
```

```
0
1
2
3
```

```
4
5
Hết vòng lặp
```

4.2 Từ khoá `continue`

Khi gặp từ khoá `continue`, python sẽ bỏ qua phần còn lại của vòng lặp, chỉ áp dụng cho lần lặp đang thực hiện. Vòng lặp vẫn sẽ được tiếp tục với giá trị kế tiếp.

```
In [13]: # ví dụ continue
         for i in range(10) :
             if i % 2 == 0 :
                 print('Cái này được in')
                 continue
             print(i)
```

```
Cái này được in
1
Cái này được in
3
Cái này được in
5
Cái này được in
7
Cái này được in
9
```

```
In [14]: n = 17
         for i in range(2, n) :
             if n % i == 0 :
                 print(n, 'không phải số nguyên tố')
                 break
```

4.3 Tạo danh sách bằng vòng lặp

Bài tập: Tạo ra danh sách bao gồm bình phương các số từ 0 đến 9?

Trong nhiều ngôn ngữ, bạn sẽ phải viết một đoạn chương trình tương đương với:

```
s = []
for i in range(10):
    s.append(i * i)
```

Tuy nhiên, trong python, có thể viết gọn lại như sau: `s = [i * i for i in range(10)]`

```
In [15]: s = [i * i for i in range(10)]
         print(s)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Tức là, bạn có thể viết gọn

```
s = []
for <điều_kiện>:
    s.append(<biểu_thức>)
```

thành `s = [<biểu_thức> for <điều_kiện>]`

```
In [16]: # tạo ra tập hợp các số chẵn từ 0 đến 10
         s = [i for i in range(0, 11, 2)]
         print(s)

[0, 2, 4, 6, 8, 10]
```

Bài tập: Cho dãy số $\{a_i\}_{i\in\mathbb{N}}$ được định nghĩa $a_0 = 1$, $a_1 = 1$, $a_{n+2} = a_{n+1} + a_n (n \geq 0)$. Viết chương trình nhập vào số n, in ra màn hình giá trị của a_n.

```
In [17]: n = int(input('Nhập số n: '))
         curr, ne = 1, 1
         for i in range(1, n):
             curr, ne = ne, curr + ne

         print(ne)

Nhập số n: 10
89
```