Bài 26: VÒNG LẶP FOR TRONG PYTHON

Xem bài học trên website để ủng hộ Kteam: Vòng lặp For trong Python.

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Trong bài trước, Kteam đã giới thiệu đến bạn một cấu trúc vòng lặp, đó chính là VÒNG LẶP WHILE TRONG PYTHON.

Ở bài này Kteam sẽ giới thiệu với các bạn một công phu của vòng lặp nữa là **Vòng lặp For trong Python**.

Nội dung

Để đọc hiểu bài này tốt nhất bạn cần:

- Cài đặt sẵn MÔI TRƯỜNG PHÁT TRIỂN CỦA PYTHON.
- Xem qua bài CÁCH CHẠY CHƯƠNG TRÌNH PYTHON.
- Nắm CÁCH GHI CHÚ và BIẾN TRONG PYTHON.
- CÁC KIỂU DỮ LIỆU ĐƯỢC GIỚI THIỆU TRONG PYTHON
- CÂU ĐIỀU KIỆN IF TRONG PYTHON
- VÒNG LẶP WHILE TRONG PYTHON

Bạn và Kteam sẽ cùng tìm hiểu những nội dung sau đây

- Hạn chế của vòng lặp while
- Cấu trúc vòng lặp for và cách hoạt động

- Sử dụng vòng lặp để xử lí các iterator và Dict
- Câu lệnh break và continue
- Cấu trúc vòng lặp for-else và cách hoạt động

Hạn chế của vòng lặp while

Bạn có thể sử dụng vòng lặp while để có thể duyệt một List, chuỗi hoặc là một Tuple. Và thậm chí là một **iterator** (một object không hỗ trợ indexing) khi biết được số phần tử mà **iterator** đó chứa.

Ví dụ:

```
>>> length = 3
>>> iter_ = (x for x in range(length))
>>> c = 0
>>> while c < length:
...     print(next(iter_))
...     c += 1
...
0
1
2
```

Nếu bạn không biết trước được số phần tử mà **iterator** đó có thì cũng không sao. Python vẫn cho phép bạn làm được điều đó bằng **try-block** (Kteam sẽ giới thiệu ở một bài khác)

Ví dụ:

```
>>> iter_ = (x for x in range(3)) # giả sử ta không biết có 3 phần tử
>>> while 1: # 1 là một expression True
...     try:
...         print(next(iter_))
...     except StopIteration:
...         break
...
0
1
```

```
2
```

Nhưng "con trăn" Python không thích sự rườm rà. Xưa nay vốn được biết đến với danh hiệu **one-liner*** nên điều này không chấp nhận được.

Vậy nên Python có một một vòng lặp khác giúp làm chuyện này đơn giản và ngắn gọn hơn chính là **vòng lặp for**.

Chú thích **One-liner**: Nhiều thuật toán dài hàng chục dòng có thể được viết ngắn gọn trong Python chỉ bằng một dòng. Điều này khá phổ biến với nhiều ngôn ngữ scripting đặc biệt trong số đó là Python.

Cấu trúc vòng lặp for và cách hoạt động

Chúng ta sẽ cùng tìm hiểu phần cấu trúc trước:

```
for variable_1, variable_2, .. variable_n in sequence:
    # for-block
```

Sequence ở đây là một **iterable object** (có thể là iterator hoặc là một dạng object cho phép sử dụng indexing hoặc thậm chí không phải hai kiểu trên).

Lưu ý: Nếu sequence là một **iterator object** thì việc dùng vòng lặp duyệt qua cũng sẽ tương tự như bạn sử dụng hàm next.

Ở cấu trúc vòng lặp này, bạn có thể **for** bao nhiêu biến theo sau cũng được. Nhưng phải đảm bảm một điều rằng, nếu bạn **for** với n biến thì mỗi phần tử trong sequence cũng phải bao gồm n (không lớn hơn hoặc nhỏ hơn) giá trị để unpacking (gỡ) đưa cho n biến của bạn.

Giả sử bạn có một sequence gồm 2 phần tử. Mỗi phần tử gồm 3 giá trị.

Bạn đưa vào vòng **for** gồm 3 biến h, k , t.

Bây giờ là nói về cách hoạt động của vòng lặp for này.

Bước 1: Vòng for sẽ bắt đầu bằng cách lấy giá trị đầu tiên của sequence.

Bước 2: Giá trị đầu tiên này có 3 giá trị. Bạn đưa vào 3 biến. Kiểm tra hợp lệ.

Bước 3: unpacking 3 giá trị này và lần lượt gán giá trị này cho ba biến h, k, t.

Dưới đây là một ví dụ unpacking:

```
>>> h = (1, 2, 3) # khởi tạo Tuple bình thường
>>> type(h)
<class 'tuple'>
>>>
>>> h, k, t = (1, 2, 3) # unpacking.
>>> h
1
>>> k
2
>>> t
3
```

Bước 4: Thực hiện nội dung for-block.

Bước 5: Lấy giá trị tiếp theo của sequence sau đó làm tương tự như Bước 2, 3, 4.

Bước 6: Lúc này, sequence đã hết giá trị. Kết thúc vòng lặp.

Sử dụng vòng lặp để xử lí các iterator và Dict

Lí thuyết là thế! Giờ chúng ta sẽ làm một vài ví dụ bằng cách bắt đầu với vấn đề lúc đầu:

```
>>> iter_ = (x for x in range(3))
>>> iter_ = (x for x in range(3))
>>> for value in iter_:
...     print('->', value)
...
-> 0
-> 1
-> 2
>>> value # biến value gián tiếp được khai báo
2
>>> next(iter_) # hãy học cách tiếp kiệp. Đây là object chỉ dùng một lần.
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

Tiếp đến chúng ta sẽ dùng vòng lặp này để duyệt một Dict. Nếu như một số ngôn ngữ khác phải có một vòng lặp mới for-reach thì với Python lại không cần.

Trước tiên hãy ôn lại bài cũ. Bạn còn chớ **phương thức items** của lớp Dict chứ? (nếu không, bạn có thể tham khảo lại trong bài KIỂU DỮ LIỆU DICT TRONG PYTHON)

```
>>> howkteam = {'name': 'Kteam', 'kter': 69}
>>> howkteam.items()
dict_items([('name', 'Kteam'), ('kter', 69)])
```

Dict-items không phải là một **iterator object**. Cũng không phải là một object cho phép bạn indexing. Nhưng nó vẫn là một **iterable**, nên ta có thể dùng

một constructor nào đó để biến đổi nó về một thứ dễ xem xét hơn. Chẳng hạn thế này.

```
>>> list_values = list(team.items())
>>> list_values
[('name', 'Kteam'), ('kter', 69)]
>>> list_values[0]
('name', 'Kteam')
>>> list_values[-1]
('kter', 69)
```

Từ đó, ta có thể dễ dàng suy ra cách để có thể có được một vòng lặp duyệt một Dict. Và đây là ví dụ:

```
>>> for key, value in team.items():
...     print(key, '=>', value)
...
name => Kteam
kter => 69
```

Câu lệnh break, continue

Những câu lệnh này có chức năng hoàn toàn tương tự như trong **vòng lặp while**.

Ví dụ về câu lệnh break trong vòng lặp **for**:

```
>>> s = 'How Kteam'
>>> for ch in s:
...     if ch == ' ':
...         break
...     else:
...         print(ch)
...
H
o
w
```

Ví dụ về câu lệnh continue trong vòng lặp for

```
>>> s = 'H o w K t e a m'
>>> for ch in s:
...     if ch == ' ':
...         continue
...     else:
...         print(ch)
...
H
o
w
K
t
e
a
m
```

Cấu trúc vòng lặp for-else và cách hoạt động

Cấu trúc:

```
for variable_1, variable_2, .. variable_n in sequence:
    # for-block
else:
    # else-block
```

Nếu bạn nắm rõ cách vòng lặp **while-else** hoạt động thì bạn cũng có thể tự đoán được cách mà **for-else** làm việc.

Cũng sẽ tương tự như **while-else**, vòng lặp hoạt động bình thường. Khi vòng lặp kết thúc, khối else-block sẽ được thực hiện. Và đương nhiên nếu trong quá trình thực hiện for-block mà xuất hiện câu lệnh break thì vòng lặp sẽ kết thúc mà bỏ qua else-block.

- **For-else** bình thường:

```
>>> for k in (1, 2, 3):
...     print(k)
... else:
...     print('Done!')
...
1
2
3
Done!
```

- **For-else** có break:

```
>>> for k in (1, 2, 3):
...     print(k)
...     if k % 2 == 0:
...         break
... else:
...     print('Done!')
...
1
2
```

Củng cố bài học

Đáp án bài trước

Bạn có thể tìm thấy câu hỏi của phần này tại CÂU HỎI CỦNG CỐ trong bài VÒNG LẶP WHILE TRONG PYTHON.

1.

```
five_even_numbers = []
k_number = 1

while len(five_even_numbers) < 5:
    if k_number % 2 == 0:
        five_even_numbers.append(k_number)
    k_number += 1
```

2.

```
with open('draft.txt') as f:
    # lấy nội dung của file dưới dạng một list
    data = f.readlines()

idx = 0 # mốc bắt đầu
length = len(data) # mốc kết thúc
new_content = '' # nội dung mới sẽ ghi vào file mới

while idx < length:
    # tách một dòng thành một list
    line_list = data[idx].split()
    idx_line = 0
    length_line = len(line_list)
    while idx_line < length_line:
        if line_list[idx_line] == 'Kteam':
            # thay thế chữ trước Kteam là How
            line_list[idx_line - 1] = 'How'
        idx_line += 1
    # nối lại thành một dòng chuỗi
    new_content += ' '.join(line_list) + '\n'
    idx += 1
```

```
with open('kteam.txt', 'w') as new_f:
    # ghi nội dung mới vào file kteam.txt
    new_f.write(new_content)
```

3.

```
lst = [56, 14, 11, 756, 34, 90, 11, 11, 65, 0, 11, 35]

idx = 0
max_idx = len(lst) - 1

max_jdx = len(lst)

while idx < max_idx:
    if lst[idx] == 11:
        idx += 1
        continue
    jdx = idx + 1
    while jdx < max_jdx:
        if lst[jdx] == 11:
            jdx += 1
            continue
        if lst[idx] > lst[jdx]:
            lst[idx], lst[jdx] = lst[jdx], lst[idx]
        jdx += 1
    idx += 1
```

Câu hỏi củng cố

1. Hãy dự đoán kết quả của hàm next dưới đây. Giải thích tại sao?

```
>>> iter_ = (x for x in range(3))
>>> for value in iter_:
...     print(non_exist_variable)
...
Traceback (most recent call last):
```

```
  File "<stdin>", line 2, in <module>
NameError: name 'non_exist_variable' is not defined
>>>
>>> next(iter_) # kết quả là gì?
```

2. Sử dụng vòng lặp để tính tổng các số trong set sau đây

```
>>> set_ = {5, 8, 1, 9, 4}
```

Đáp án của phần này sẽ được trình bày ở bài tiếp theo. Tuy nhiên, Kteam khuyến khích bạn tự trả lời các câu hỏi để củng cố kiến thức cũng như thực hành một cách tốt nhất!

Kết luận

Qua bài viết này, Bạn đã biết sơ lược về VÒNG LẶP FOR TRONG PYTHON.

Ở bài viết sau. Kteam sẽ tiếp tục đề cập đến VÒNG LẶP FOR TRONG PYTHON.

Cảm ơn bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".