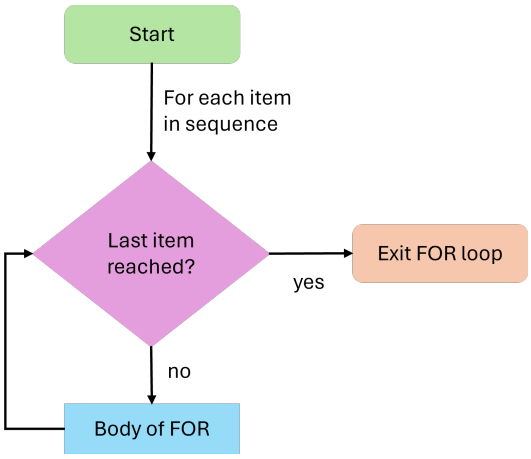
BASIC PYTHON - FOR LOOP

Dinh-Tiem Nguyen và Quang-Vinh Dinh

1 Giới thiệu

Trong hầu hết các ngôn ngữ lập trình, vòng lặp là một công cụ quan trọng giúp thực hiện lặp đi lặp lại các tác vụ mà không cần viết mã nhiều lần. Trong bài viết này chúng ta sẽ tìm hiểu cách sử dụng vòng lặp for từ cơ bản đến nâng cao, bao gồm cách sử dụng continue, break, vòng lặp lồng nhau và cách duyệt qua các cấu trúc dữ liệu như string, list, tuple và dictionary.



2 Vòng lặp for cơ bản

Sử dụng vòng lặp for là cách hiệu quả để xử lý các tác vụ lặp đi lặp lại, duyệt qua các phần tử của một danh sách, mảng, hoặc bất kỳ tập hợp nào khác. Cú pháp sử dụng như sau:

```
for variable in iterable:
    # Body of for loop
```

Trong đó:

- variable là biến sẽ nhận giá trị từng phần tử trong iterable qua mỗi lần lặp.
- iterable là một đối tượng có thể lặp, ví dụ như list, tuple, string, dictionary, hoặc range.

Ví dụ vòng lặp for sau sẽ thực hiện hiển thị giá trị của biến i 5 lần, mỗi lần lặp i sẽ nhận 1 giá trị trong range(5) từ 0 đến 4:

```
for i in range(5):
    print(i)



#Vòng lặp for với range
```

```
================= Output ================
0
1
2
3
4
=========================================
```

Trong chương trình trên, chúng ta bắt đầu vòng lặp với lệnh for, trong đó range(3) sẽ tạo ra một dãy số từ 0 đến 4 (không bao gồm 5). Tức là, dãy số này sẽ là: [0, 1, 2, 3, 4], i là biến đếm, nó sẽ lần lượt

nhận giá trị từ từng phần tử trong dãy số do range(5) tạo ra. Lệnh print(i) là khối lệnh thực thi trong vòng lặp. Với mỗi giá trị của i trong range(5), lệnh này sẽ in ra giá trị hiện tại của i.
Chi tiết hoạt động từng bước của chương trình trên ta có thể hình dung như sau:

- Bước đầu tiên: i nhận giá trị đầu tiên từ range(5), tức là 0, sau đó print(i) in ra giá trị 0.
- Bước thứ hai: i nhận giá trị tiếp theo từ range(5), tức là 1, sau đó print(i) in ra giá trị 1.
- Bước thứ 3 tương tự, i nhận giá trị 2 và in ra 2.
- Bước thứ 4 i nhận giá trị 3 và in ra 3
- Bước thứ 5 i nhận giá trị 4 và in ra 4 và kết thúc vòng lặp.

Như đã đề cập phía trên thì mọi kiểu dữ liệu thuộc loại interable đều có thể dùng vòng lặp for. Theo định nghĩa Iterable là bất kỳ đối tượng Python nào có khả năng trả về từng phần tử của nó cùng một lúc, cho phép nó được lặp lại trong vòng lặp for. Dưới đây, chúng ta sẽ sử dụng vòng lặp for với string, list, dictionary, tuple.
Ví dụ sử dụng vòng lặp for qua từng phần tử trong string:

```
for i in "AI VIETNAM":
    print(i)









#Vòng lặp for với string
```

```
================= Output ================
A
I

V
I
E
T
N
A
M
=========================================
```

Ví dụ sử dụng vòng lặp for qua từng phần tử trong list:

```
# Vòng lặp for với danh sách
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:
    print(fruit)
```

```
================= Output ================
apple
banana
cherry
=========================================
```

Ví dụ sử dụng vòng lặp for qua từng phần tử trong tuple:

```
# Vòng lặp for với tuple
numbers = (1, 2, 3)

for number in numbers:
    print(number)
```

```
================= Output ================
1
2
3
=========================================
```

Ví dụ sử dụng vòng lặp for qua từng phần tử trong dictionary:

```python
# Vòng lặp for với từ điển
student_scores = {
                    "Bông": 90,
                    "Hoa": 85,
                    "Mai": 78
                }
for student, score in student_scores.items():
    print(f"{student}: {score}")
```

```
================ Output ===============
Bông: 90
Hoa: 85
Mai: 78
=======================================
```

3 Vòng lặp for trong comprehension

Comprehension là một loại cú pháp để viết vòng lặp for ngắn gọn hơn để tạo ra list, dictionary, set mới. Chúng ta sẽ tìm hiểu về cách sử dụng comprehension với list, dictionary còn các kiểu dữ liệu khác thì tương tự.

3.1 List comprehension

List comprehension cho phép chúng ta tạo ra một danh sách mới bằng cách áp dụng một biểu thức cho mỗi phần tử trong một iterable.

```python
[expression for item in iterable if condition]
```

Trong đó:

- expression: Biểu thức được áp dụng cho mỗi phần tử.
- item: Phần tử hiện tại từ iterable.
- iterable: Bất kỳ đối tượng nào có thể lặp (như danh sách, chuỗi, range, v.v.).
- condition: (Tùy chọn) Điều kiện để lọc các phần tử.

Ví dụ: Tạo danh sách bình phương của các số chẵn từ 0 đến 9

```python
squares = [x ** 2 for x in range(10) if x % 2 == 0]
print(squares)
```

```
================ Output ===============
[0, 4, 16, 36, 64]
=======================================
```

3.2 Dictionary comprehension

Dictionary comprehension cho phép ta tạo ra một từ điển mới bằng cách áp dụng một biểu thức cho mỗi phần tử trong một iterable.

```python
{key_expression: value_expression for item in iterable if condition}
```

Trong đó:

- key_expression: Biểu thức cho khóa.
- value_expression: Biểu thức cho giá trị.
- item, iterable, và condition: Tương tự như trong list comprehension.

Ví dụ: Tạo dictionary với khóa là số và giá trị là bình phương của số đó.

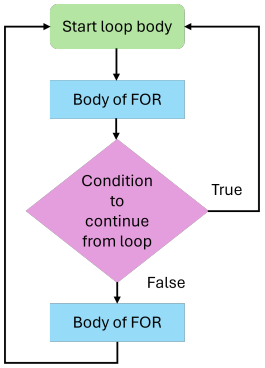
```
1
2 squares_dict = {x: x ** 2 for x in range
      (10)}
3 print(squares_dict)
```

```
================ Output ===============
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6:
    36, 7: 49, 8: 64, 9: 81}
=======================================
```

4 Vòng lặp for với continue

Lệnh continue trong vòng lặp for được sử dụng để bỏ qua các lần lặp cụ thể và tiếp tục với lần lặp tiếp theo. Khi gặp lệnh continue, vòng lặp sẽ ngay lập tức bỏ qua các lệnh còn lại trong lần lặp hiện tại và chuyển sang lần lặp tiếp theo. Điều này hữu ích khi ta muốn bỏ qua một số điều kiện nhất định mà không cần kết thúc hoàn toàn vòng lặp.

Ví dụ dưới đây minh họa việc sử dụng lệnh continue trong vòng lặp for để bỏ qua việc in ra số 2:




```
1 # Vòng lặp for với lệnh continue
2
3 for i in range(5):
4     if i == 2:
5         continue
6     print(i)
```

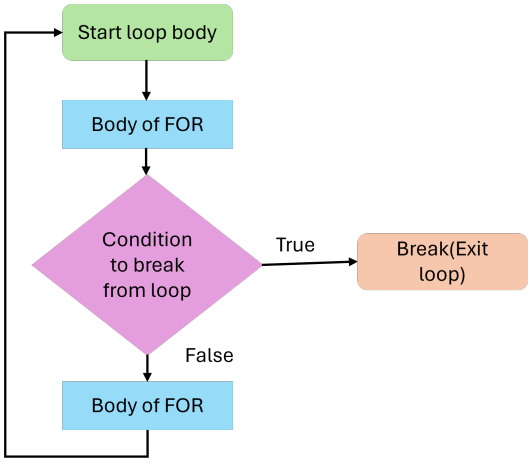
```
================ Output ===============
0
1
3
4
=======================================
```

Trong chương trình trên, vòng lặp for được khởi tạo với i nhận các giá trị từ range(5), tức là [0, 1, 2, 3, 4]. Khi i là 2, lệnh if i == 2 sẽ được thực thi vì điều kiện lúc đó là True, lệnh continue sẽ được thực thi, bỏ qua phần còn lại của vòng lặp cho giá trị i hiện tại và chuyển sang giá trị tiếp theo của i.
Dưới đây là chi tiết từng bước thực hiện:

- Khi i là 0, điều kiện i == 2 là sai, lệnh print(i) sẽ in ra 0.
- Khi i là 1, điều kiện i == 2 là sai, lệnh print(i) sẽ in ra 1.
- Khi i là 2, điều kiện i == 2 là đúng, lệnh continue sẽ được thực thi và vòng lặp bỏ qua việc in ra số 2.
- Khi i là 3, điều kiện i == 2 là sai, lệnh print(i) sẽ in ra 3.
- Khi i là 4, điều kiện i == 2 là sai, lệnh print(i) sẽ in ra 4.
- Sau khi đã duyệt hết các giá trị trong range(5), vòng lặp kết thúc.

5 Vòng lặp for với break

Lệnh break trong vòng lặp for được sử dụng để kết thúc vòng lặp ngay lập tức, ngay cả khi chưa hoàn thành vòng lặp. Khi gặp lệnh break, chương trình sẽ thoát khỏi vòng lặp và tiếp tục thực hiện các lệnh sau vòng lặp. Chúng ta sẽ sử dụng nó khi muốn dừng vòng lặp với một điều kiện cụ thể.
Ví dụ dưới đây minh họa việc sử dụng lệnh break trong vòng lặp for để kết thúc vòng lặp khi gặp số 2:




```python
# Vòng lặp for với lệnh break
for i in range(5):
    if i == 2:
        break
    print(i)
```

```
================= Output ================
0
1

=========================================
```

Vòng lặp for được khởi tạo với i nhận các giá trị từ range(5), tức là [0, 1, 2, 3, 4]. Khi i là 2, lệnh if i == 2 sẽ được thực thi và lệnh break xảy ra để kết thúc vòng lặp ngay lập tức và không in ra giá trị 2 hay các giá trị sau đó.

- Khi i là 0, điều kiện i == 2 là sai, lệnh print(i) sẽ in ra 0.
- Khi i là 1, điều kiện i == 2 là sai, lệnh print(i) sẽ in ra 1.
- Khi i là 2, điều kiện i == 2 là đúng, lệnh break sẽ được thực thi và vòng lặp kết thúc.

6 Vòng lặp for lồng

Vòng lặp lồng nhau là vòng lặp nằm bên trong một vòng lặp khác. Nó cho phép ta thực hiện các tác vụ phức tạp hơn, chẳng hạn như duyệt qua các ma trận hoặc thực hiện các phép toán trên nhiều chiều dữ liệu.
Ví dụ dưới đây minh họa việc sử dụng vòng lặp for lồng nhau để duyệt qua các phần tử của một danh sách 2 chiều và in ra vị trí cũng như giá trị của từng phần tử:

```python
# List 2 chiều
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Vòng lặp for lồng nhau để duyệt và in ra vị trí và giá trị phần tử
for i in range(len(matrix)):
    for j in range(len(matrix[i])):
        print(f"Vị trí: ({i}, {j}), Giá trị: {matrix[i][j]}")
```

```
================= Output ================
Vị trí: (0, 0), Giá trị: 1
Vị trí: (0, 1), Giá trị: 2
Vị trí: (0, 2), Giá trị: 3
Vị trí: (1, 0), Giá trị: 4
Vị trí: (1, 1), Giá trị: 5
Vị trí: (1, 2), Giá trị: 6
Vị trí: (2, 0), Giá trị: 7
Vị trí: (2, 1), Giá trị: 8
Vị trí: (2, 2), Giá trị: 9

=========================================
```

Đầu tiên là khởi tạo vòng lặp ngoài duyệt qua từng hàng của ma trận matrix, với i là chỉ số của hàng. Tiếp theo là vòng lặp trong với mỗi hàng i, vòng lặp for j in range(len(matrix[i])) sẽ duyệt qua từng phần tử trong hàng đó, với j là chỉ số của cột. Với mỗi phần tử matrix[i][j], lệnh print(f"Vị trí: (i, j), Giá trị: matrix[i][j]") sẽ in ra vị trí và giá trị của phần tử đó.
Chi tiết từng bước được mô tả như sau:

- Khi i là 0, j sẽ lần lượt là 0, 1, 2, kết quả in ra là

```
==================================== Output ====================================
Vị trí: (0, 0), Giá trị: 1
Vị trí: (0, 1), Giá trị: 2
Vị trí: (0, 2), Giá trị: 3
================================================================================
```

- Khi i là 1, j vẫn lần lượt là 0, 1, 2

```
==================================== Output ====================================
Vị trí: (1, 0), Giá trị: 4
Vị trí: (1, 1), Giá trị: 5
Vị trí: (1, 2), Giá trị: 6
================================================================================
```

- Khi i là 2, j vẫn lần lượt là 0, 1, 2

```
==================================== Output ====================================
Vị trí: (2, 0), Giá trị: 7
Vị trí: (2, 1), Giá trị: 8
Vị trí: (2, 2), Giá trị: 9
================================================================================
```

7 Kết luận

Trong bài viết này chúng ta đã tìm hiểu cách sử dụng vòng lặp for từ cơ bản đến nâng cao. Đây là kiến thức cơ bản nhưng rất quan trọng mà chúng ta cần phải thành thạo nó. Mặc dù bài viết đã cố gắng bao quát các trường hợp sử dụng vòng lặp for, tuy nhiên khi các bạn lập trình sẽ gặp nhiều trường hợp hơn nữa như việc kết hợp với thư viện tqdm, enumerate... nhưng về bản chất thì nó vẫn không thay đổi. Hy vọng các bạn đã nắm được vòng lặp for và sẽ sử dụng nó hiệu quả trong quá trình viết code của mình.