Python cho Khoa học dữ liệu

Bài 6: Vòng lặp - Loops

Hà Minh Tuấn

Ngày 28 tháng 8 năm 2025

Khoa Toán - Tin học
Trường Đại học Khoa học Tự nhiên, ĐHQG-HCM

Nội dung bài học

1. Tổng quan về vòng lặp
2. Vòng lặp for
3. Vòng lặp for-else
4. Vòng lặp while
5. Phát biểu break - break statement
6. Phát biểu continue - Continue statement
7. Phát biểu pass - pass statement
8. Vòng lặp lồng - Nested Loops

Tổng quan về vòng lặp

Vòng lặp trong Python

- Cho phép thực thi một đoạn code nhiều lần.
- Bình thường, code chạy tuần tự (dòng 1, dòng 2, ...).
- Với vòng lặp, ta có thể lặp lại dựa trên điều kiện hoặc số lần xác định.

Các loại vòng lặp:

1. `while loop`: Lặp khi điều kiện còn đúng.
2. `for loop`: Duyệt qua một dãy (list, tuple, string, range...).
3. `nested loop`: Vòng lặp lồng nhau.

while loop

- Kiểm tra điều kiện trước khi chạy.
- Dùng khi số lần lặp chưa biết trước.

```python
count = 0
while count < 5:
    print("count =", count)
    count += 1
```

```
count = 0
count = 1
count = 2
count = 3
count = 4
```

for loop

- Dùng để duyệt qua một tập hợp (iterable).
- Gọn gàng, dễ đọc.

```python
for i in range(5):
    print("i =", i)
```

```
i = 0
i = 1
i = 2
i = 3
i = 4
```

Vòng lặp lồng nhau

```python
for i in range(1, 4):
    for j in range(1, 3):
        print(f"i={i}, j={j}")
```

```
i=1, j=1
i=1, j=2
i=2, j=1
i=2, j=2
i=3, j=1
i=3, j=2
```

Câu lệnh điều khiển vòng lặp

- `break`: Thoát vòng lặp ngay lập tức.
- `continue`: Bỏ qua phần còn lại của vòng lặp hiện tại, sang lần lặp tiếp theo.
- `pass`: Dùng như placeholder (không làm gì cả).

Ví dụ về break, continue, pass

```
for i in range(5):
    if i == 2:
        continue    # b? qua i=2
    if i == 4:
        break       # thoát khi i=4
    if i == 1:
        pass        # không làm gì
    print("i =", i)
```

```
i = 0
i = 1
i = 3
```

Vòng lặp for

`for` Loop in Python

- Dùng để lặp qua các phần tử của một **sequence** (list, tuple, string, range...).
- Cú pháp:

```
for variable in sequence:
    # khôi l�nh đ��c l�p
```

- `variable`: nhận giá trị của từng phần tử trong sequence.
- Lặp cho đến khi hết sequence.

For Loop với String

```python
zen = "Beautiful is better than ugly"
for char in zen:
    if char not in "aeiou":
        print(char, end="")
```

```
Btfl s bttr thn gly
```

For Loop với Tuple

```
numbers = (34,54,67,21,78)
total = 0
for num in numbers:
    total += num
print("Total =", total)
```

```
Total = 254
```

For Loop với List

```
numbers = [34,54,67,21,78]
for num in numbers:
    if num % 2 == 0:
        print(num)
```

34
54
78

For Loop với Range

```
for num in range(5):
    print(num, end=" ")

for num in range(10, 20, 3):
    print(num, end=" ")
```

```
0 1 2 3 4
10 13 16 19
```

For Loop với Dictionary

```
numbers = {10:"Ten", 20:"Twenty", 30:"Thirty"}

for k in numbers:
    print(k, ":", numbers[k])
```

```
10 : Ten
20 : Twenty
30 : Thirty
```

For Loop với Dictionary Items

```python
numbers = {10:"Ten", 20:"Twenty", 30:"Thirty"}

for item in numbers.items():
    print(item)
```

```
(10, 'Ten')
(20, 'Twenty')
(30, 'Thirty')
```

For Loop với Else

```python
for num in range(10, 15):
    for i in range(2, num):
        if num % i == 0:
            print(num, "=", i, "*", num//i)
            break
    else:
        print(num, "is a prime number")
```

```
10 = 2 * 5
11 is a prime number
12 = 2 * 6
13 is a prime number
14 = 2 * 7
```

Vòng lặp for-else

For-Else Loop in Python

- Python cho phép **else** đi kèm với vòng lặp `for`.
- Khối `else` chỉ được thực thi khi vòng lặp kết thúc **bình thường** (không bị dừng bởi `break`).
- Cú pháp:

```
for variable in iterable:
    # statements
else:
    # executed if loop ends normally
```

For-Else: Bình thường (không break)

```python
for count in range(5):
    print("Iteration", count)
else:
    print("Loop finished, now in else block")
```

```
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Loop finished, now in else block
```

For-Else: Không có break

```python
for i in ['T', 'P']:
    print(i)
else:
    print("For-else executed successfully")
```

```
T
P
For-else executed successfully
```

For-Else: Với break

```python
for i in ['T', 'P']:
    print(i)
    break
else:
    print("Else block skipped")
```

T

For-Else: Với break + điều kiện

```python
def positive_or_negative():
    for i in [5, 6, 7]:
        if i >= 0:
            print("Positive number")
        else:
            print("Negative number")
            break
    else:
        print("Loop-else executed")

positive_or_negative()
```

```
Positive number
Positive number
Positive number
Loop-else executed
```

Vòng lặp while

Python While Loop

- `while` loop lặp lại khi điều kiện (boolean expression) còn **True**.
- Khi điều kiện thành **False**, chương trình thoát khỏi vòng lặp.
- Nếu điều kiện không bao giờ False ⇒ **infinite loop**.

Cú pháp:

```
while condition:
    statement(s)
```

While Loop - Example 1

```python
count = 0
while count < 5:
    count += 1
    print("Iteration no.", count)

print("End of while loop")
```

```
Iteration no. 1
Iteration no. 2
Iteration no. 3
Iteration no. 4
Iteration no. 5
End of while loop
```

While Loop - User Input Example

```python
var = "0"
while var.isnumeric():
    var = input("Enter a number: ")
    if var.isnumeric():
        print("Your input:", var)
print("End of while loop")
```

```
Enter a number: 10
Your input: 10
Enter a number: qwer
End of while loop
```

Infinite While Loop

```
var = 1
while var == 1:   # infinite loop
    num = int(input("Enter a number: "))
    print("You entered:", num)

print("Good bye!")
```

Thoát vòng lặp: nhấn CTRL+C.

While-Else Loop

```python
count = 0
while count < 5:
    count += 1
    print("Iteration no.", count)
else:
    print("While loop over. Now in else block")
print("End of while loop")
```

```
Iteration no. 1
Iteration no. 2
Iteration no. 3
Iteration no. 4
Iteration no. 5
While loop over. Now in else block
End of while loop
```

Single Statement While

```python
flag = 0
while flag: print("Flag is true!")
print("Good bye!")
```

```
Good bye!
```

Phát biểu break - break statement

Python break Statement

- `break` dùng để **thoát khỏi vòng lặp ngay lập tức**.
- Sau khi thoát, chương trình tiếp tục thực thi câu lệnh sau vòng lặp.
- Có thể dùng trong cả `for` và `while`.
- Nếu trong vòng lặp lồng nhau ⇒ `break` chỉ dừng vòng lặp **gần nhất**.

Cú pháp:

```
loop:
    if condition:
        break
```

break với for loop

```python
for letter in 'Python':
    if letter == 'h':
        break
    print("Current Letter:", letter)
print("Good bye!")
```

```
Current Letter : P
Current Letter : y
Current Letter : t
Good bye!
```

break với while loop

```python
var = 10
while var > 0:
    print("Current value:", var)
    var -= 1
    if var == 5:
        break
print("Good bye!")
```

```
Current value: 10
Current value: 9
Current value: 8
Current value: 7
Current value: 6
Good bye!
```

break trong Nested Loops

```
no = 33
numbers = [11,33,55,39,75,21,23]
for num in numbers:
    if num == no:
        print("Number found!")
        break
else:
    print("Number not found!")
```

```
Number found!
```

Lưu ý: Nếu vòng lặp kết thúc bình thường (không break) ⇒ `else` block được chạy.

Phát biểu continue - Continue statement

Python `continue` Statement

- `continue` được dùng để **bỏ qua phần còn lại của vòng lặp hiện tại**.
- Sau khi gặp `continue`, chương trình sẽ **quay lại đầu vòng lặp** để bắt đầu lần lặp kế tiếp.
- Ngược lại với `break` (thoát vòng lặp hoàn toàn).

Cú pháp:

```
loop:
    if condition:
        continue
    # phân còn l□i b□ b□ qua
```

continue trong for loop

```python
for letter in 'Python':
    if letter == 'h':
        continue
    print("Current Letter:", letter)
print("Good bye!")
```

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : o
Current Letter : n
Good bye!
```

continue trong while loop

Ví dụ: tìm **thừa số nguyên tố** của một số.

```
num = 60
print("Prime factors for:", num)
d = 2
while num > 1:
    if num % d == 0:
        print(d)
        num = num / d
        continue
    d += 1
```

```
Prime factors for: 60
2
2
3
5
```

Phát biểu pass - pass statement

Python `pass` Statement

- `pass` là một **câu lệnh rỗng** (null statement).
- Dùng khi Python yêu cầu một câu lệnh về mặt cú pháp nhưng bạn **không muốn thực thi gì cả**.
- Thường sử dụng:
  - Trong **hàm / lớp** chưa triển khai.
  - Trong **cấu trúc điều khiển** như `if`, `for`, `while`.
  - Như **placeholder** để tránh lỗi `SyntaxError`.

Cú pháp:

```
pass
```

Ví dụ với `for loop`

```python
for letter in "Python":
    if letter == "h":
        pass
        print("This is pass block")
    print("Current Letter:", letter)
print("Good bye!")
```

```
Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : h
Current Letter : o
Current Letter : n
Good bye!
```

pass trong vòng lặp vô hạn

```python
while True:
    pass  # làm không gì c
```

- Tạo ra vòng lặp vô hạn nhưng không thực hiện công việc nào.
- Dùng `Ctrl + C` để dừng chương trình.

Dùng dấu ba chấm (`...`) thay cho `pass`

Python 3.x cho phép dùng `...` làm placeholder.

```python
def func1():
    ...   # thay cho pass

def func2(): ...

func1()
func2()   # không làm gì c
```

Vòng lặp lồng - Nested Loops

Python Nested Loops

- Khi viết một (hoặc nhiều) vòng lặp bên trong một vòng lặp khác → gọi là **nested loop**.
- Vòng lặp chính = **outer loop**, vòng lặp bên trong = **inner loop**.
- Có thể lồng bất kỳ loại vòng lặp nào:
  - `for` trong `for`
  - `while` trong `while`
  - `for` trong `while`, hoặc ngược lại

Nested for Loop

Cú pháp:

```
for var1 in sequence1:
    for var2 in sequence2:
        statements
    statements
```

Ví dụ:

```
months = ["jan", "feb", "mar"]
days = ["sun", "mon", "tue"]

for x in months:
    for y in days:
        print(x, y)

print("Good bye!")
```

Kết quả Nested `for` Loop

```
jan sun
jan mon
jan tue
feb sun
feb mon
feb tue
mar sun
mar mon
mar tue
Good bye!
```

Nested while Loop

Cú pháp:

```
while condition1:
    while condition2:
        statements
    statements
```

Ví dụ: Tìm số nguyên tố (2 → 25)

```
i = 2
while i < 25:
    j = 2
    while j <= (i/j):
        if not(i % j): break
        j = j + 1
    if j > i/j:
        print(i, "is prime")
    i = i + 1
print("Good bye!")
```

Kết quả Nested `while` Loop

```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
Good bye!
```