



Lập trình Python

Bài 3: Vòng lặp trong python

Tài liệu này phân phối dưới giấy phép Creative Commons Attribution 4.0
(bất kỳ ai cũng đều có quyền tự do sử dụng, chia sẻ, sao chép, phân phối, phân phối lại, áp dụng, trích xuất, tùy biến, mở rộng, thương mại hóa,... miễn là ghi nhận công của các tác giả ban đầu của tài liệu)



Tóm tắt nội dung bài trước

- Định nghĩa và viết hàm trong python rất đơn giản: dùng từ khóa **def**, chỉ cần đặt tên và liệt kê danh sách tham số
 - Không hạn chế kiểu dữ liệu trả về từ hàm
 - Cho phép tham số mặc định
 - Hạn chế việc nạp chồng hàm
 - Có cơ chế cho phép số tham số không hạn chế
- Phép toán **if** là phép toán 3 ngôi, một dạng lệnh lựa chọn ngắn gọn, đơn giản
 - Tiện hơn lệnh **if** ở chỗ phép toán có thể tham gia vào biểu thức
- Python dùng lệnh lựa chọn **if-elif-else**
 - Lệnh lựa chọn duy nhất, dùng cho cả trường hợp chọn giữa hai nhánh và chọn giữa nhiều nhánh



Nội dung

1. Vòng lặp “while”
2. Vòng lặp “for”
 - Lặp “for” duyệt một danh sách
 - Lặp “for” duyệt một miền số nguyên
3. Bài tập



Phần 1

Vòng lặp “while”



Vòng lặp while

```
while expression:
```

```
# while-block
```

```
while expression:
```

```
#while-block-1
```

```
continue / break
```

```
#while-block-2
```

```
while expression:
```

```
# while-block
```

```
else:
```

```
# else-block
```

- Vòng **while** thực hiện lặp lại khối lệnh chừng nào biểu thức điều kiện còn đúng
 - Phát biểu **continue** trong khối lệnh sẽ ngắt khối lệnh hiện tại và bắt đầu một vòng lặp mới
 - Phát biểu **break** sẽ kết thúc vòng lặp ngay lập tức
- Khối **else** sẽ được thực hiện sau khi toàn bộ vòng lặp đã chạy xong, không bắt buộc phải có khối này
 - Khối này sẽ không chạy nếu vòng lặp bị “break”



Vòng lặp while đơn giản

```
# có 10 triệu đồng, gửi ngân hàng với lãi suất 5,1% hàng năm
# tính xem sau bao nhiêu năm thì bạn có ít nhất 50 triệu
# cách giải sử dụng vòng lặp
so_tien = 1e7
lai_suat = 5.1 / 100
so_nam = 0
# chừng nào số tiền chưa đủ 50 triệu thì gửi thêm 1 năm nữa
while so_tien < 5e7:
    so_nam += 1
    so_tien = so_tien * (1 + lai_suat)
    print("Số tiền sau", so_nam, "năm:", so_tien)
# in kết quả
print("Sau", so_nam, "bạn sẽ có ít nhất 50 triệu.")
```



Vòng lặp while kết hợp điều kiện if

```
# In ra các số tự nhiên chia hết cho 7 nhỏ hơn 1000
```

```
n = 0
```

```
while n < 1000:
```

```
    if (n % 7) == 0:
```

```
        print(n)
```

```
    n += 1
```

```
# Tính tổng các số nhỏ hơn 1000 và không chia hết cho 3
```

```
t = 0
```

```
n = 0
```

```
while n < 1000:
```

```
    if (n % 3) != 0:
```

```
        t = t + n
```

```
    n += 1
```

```
print(t)
```



Vòng lặp while với break

```
# bài tập buổi trước: kiểm tra xem một số dương N có phải số  
# fibonacci hay không?  
n = int(input("Nhập số dương N: "))  
a, b = 0, 1          # kiểu gán trong python: a = 0, b = 1  
while b != n:  
    if b > n:        # nếu b vượt quá n thì dừng  
        break  
    a, b = b, a + b  # a = b, b = a + b  
print('Fibonacci' if b == n else 'Non-fibonacci')
```




Vòng lặp while với continue

tính tổng các số fibonacci chia hết cho 3 nhỏ hơn N

```
n = int(input("Nhập số dương N: "))
```

```
tong, a, b = 0, 0, 1
```

```
while b < n:
```

```
    a, b = b, a + b
```

```
    if 0 != a % 3: # bỏ qua nếu không chia hết cho 3
```

```
        continue
```

```
    tong += a
```

```
print('Tổng là:', tong)
```



Vòng lặp while sử dụng else

```
# nhập số n và kiểm tra xem nó có phải số nguyên tố hay không
# chú ý: nếu không sử dụng else, ta sẽ phải khai báo thêm
# biến phụ và chương trình dài hơn vài dòng
n = int(input("Nhập số N: "))
x = 2
while x < n:
    if (n % x) == 0:
        print("N không phải số nguyên tố")
        break;
    x = x + 1
else:
    print("N là số nguyên tố")
```



Phần 2

Vòng lặp “for”



Vòng lặp for duyệt một danh sách

- Cú pháp:

```
for <biến> in <danh-sách>:  
    # khối for  
else  
    # khối else
```

- Vòng **for** cho phép sử dụng một <biến> lần lượt duyệt các giá trị trong <danh-sách>
- Tương tự như while, có thể sử dụng **break** và **continue**
- Khối **else** thực hiện sau khi toàn bộ vòng lặp đã chạy xong
 - Khối này sẽ không chạy nếu vòng lặp bị “break”
 - Không bắt buộc phải có khối này
 - Cách làm việc tương tự như ở vòng lặp while



Vòng lặp for duyệt một danh sách

```
X = ['chó', 'mèo', 'lợn', 'gà']
```

```
# In ra các loài vật trong danh sách
```

```
for w in X:  
    print(w)
```

```
# In ra các loại vật, ngoại trừ loài 'mèo'
```

```
for x in X:  
    if x == 'mèo':  
        continue  
    print(x)
```

```
# In ra các loại vật, nếu gặp loài 'mèo' thì dừng luôn
```

```
for z in X:  
    if z == 'mèo':  
        break  
    print(z)
```

Vòng lặp for duyệt một miền số nguyên



- Cú pháp vòng `for` rất phù hợp với việc duyệt một tập hợp ít phần tử
 - Vì ta phải liệt kê mọi phần tử trong tập
- Nhưng nếu muốn duyệt tập rất nhiều phần tử thì sao?
 - Chẳng hạn muốn duyệt các số nguyên từ 1 đến 1.000.000?
- Python cung cấp hàm `range` để tạo một dãy số:
 - Hàm `range(n)`: tạo dãy số nguyên từ 0 đến $n-1$
 - Hàm `range(n, m)`: tạo dãy số nguyên từ n đến $m-1$
 - Hàm `range(n, m, k)`: tạo dãy số nguyên từ n đến trước m với bước nhảy k (một lần giá trị tăng k đơn vị)
 - Chú ý: giá trị k có thể âm, trong trường hợp này dãy số sinh ra sẽ giảm dần

Vòng lặp for duyệt một miền số nguyên



```
# Trường hợp một khoảng số khá lớn, không thể liệt kê được
# Ta sử dụng hàm range để tạo ra khoảng số
# In các số từ 10 đến 19: khoảng 10 đến 20, bước nhảy 1
for d in range(10, 20):
    print(d)
# In các số từ 20 đến 11: khoảng 20 đến 10, bước nhảy -1
for d in range(20, 10, -1):
    print(d)
# In các số lẻ từ 1 đến 100: khoảng 1 đến 100, bước nhảy 2
for d in range(1, 101, 2):
    print(d)
```



Phần 4

Bài tập



Bài tập

1. Viết hàm `isPrime` kiểm tra xem N có phải là số nguyên tố hay không?
2. Viết chương trình nhập hai số A và B , in ra tất cả các số nguyên tố nằm trong khoảng $[A, B]$
3. Nhập 2 số nguyên dương A và B , tính và in ra màn hình ước số chung lớn nhất và bội số chung nhỏ nhất của hai số đó
4. Viết chương trình cho phép người dùng nhập vào liên tiếp một dãy số tự nhiên (không biết trước độ dài), việc nhập dãy sẽ kết thúc khi người dùng nhập một số âm nào đó



Bài tập

5. Viết chương trình cho phép người dùng nhập vào liên tiếp một dãy số tự nhiên (không biết trước độ dài), việc nhập dãy sẽ kết thúc khi người dùng nhập một số âm hoặc số 0.

Sau khi kết thúc, in ra màn hình ước số chung lớn nhất và bội số chung nhỏ nhất của tất cả các số vừa nhập vào.

6. Hàm fibo mở rộng được định nghĩa như sau

$$fibo(0 \leq n < k, k) = n$$

$$fibo(n \geq k, k) = \sum_{i=k}^1 fibo(n - i, k)$$

Viết chương trình nhập n và k sau đó tính fibo(n, k)