[Bài đọc] Cấu trúc lặp for

Giới thiệu

Vòng lặp được sử dụng lặp qua một tập hợp, một danh sách hoặc một chuỗi.

Nội dung bài đọc sẽ có các phần chính sau:

- Vòng lặp for
- Kết hợp hàm range trong vòng lặp
- Vòng lặp lồng nhau
- Một số keyword sử dụng trong vòng lặp

Vòng lặp for

Với vòng lặp for chúng ta có thể thực hiện 1 tập hợp các câu lệnh một lần cho từng mục trong danh sách hoặc chuỗi

Cú pháp:

```
for val in list :
```

statements

Ở đây

- val là biến nhận giá trị của từng mục trong chuỗi trên mỗi lần lặp.
- list là tập hợp các giá trị cần lặp
- statements là các dòng lệnh xử lý trong thân vòng lặp

Ví dụ 1: Lặp qua 1 chuỗi

```
for val in "apple":
    print(val)
```

Kết quả: a, p, p, l, e

Vòng lặp sẽ trả ra từng phần tử trong chuỗi đã cho

Ví dụ 2: Lặp qua 1 danh sách chuỗi

```
listFruit = [ "apple", "banana", "cherry"]
for fruit in  listFruit
       print(val)
```

Kết quả: apple, banana, cherry

Vòng lặp trả ra từng phần tử trong mảng danh sách đã cho

Ví dụ 3: Lặp qua 1 danh sách số

```
listNumber = [1, 2, 3, 4, 5, 6, 7, 8, 9]
       for number in listNumber
              print (number)
```

Kết quả: 1, 2, 3, 4, 5, 6, 7, 8, 9

Kết hợp hàm range() trong vòng lặp for

Hàm range sẽ trả về một mảng trong đó tổng số phần tử sẽ phụ thuộc vào các tham số truyền vào.

Cú pháp như sau:

```
range(start, end, step)
```

Trong đó:

- start: là giá trị bắt đầu
- end là giá trị kết thúc
- step là khoảng cách giữa các phần tử, hay còn gọi là bước nhảy

Trường hợp có một tham số

Nếu bạn chỉ truyền một tham số n thì nó sẽ tạo một mảng từ 0 -> n – 1.

```
for i in range(5):
    print(i, end=', ')
```

Kết quả sẽ tạo một mảng gồm 5 phần tử có giá trị lần lượt từ 0 -> 5.

0, 1, 2, 3, 4,

Trường hợp có hai tham số

Nếu bạn truyền 2 tham số thì sẽ tạo một mảng với bước nhảy là 1, phần tử đầu của mảng là start, phần tử cuối cùng của mảng là end – 1.

```
for i in range(5, 10):
    print(i, end=', ')
```

Kết quả sẽ tạo một mảng gồm 5 phần tử có giá trị lần lượt là 5 -> 9

5, 6, 7, 8, 9,

Trường hợp có ba tham số

Trường hợp này sẽ tạo một mảng như trường hợp 2 nhưng vì bước nhảy là step nên tổng số phần tử sẽ nhỏ hơn.

```
for i in range(1, 10, 2):
    print(i, end=', ')
```

Kết quả trả về một mảng 5 phần tử có giá lần lượt là 1, 3, 5, 7, 9 vì bước nhảy là 2.

1, 3, 5, 7, 9,

Vòng lặp for lồng nhau

Vòng lặp for lồng nhau là một vòng lặp bên trong một vòng lặp khác. Trong đó “Vòng lặp bên trong” sẽ thực hiện một lần sau mỗi lần lặp lại “vòng lặp bên ngoài”.

Cú pháp:

```
for [biến lặp đầu tiên] in [vòng lặp ngoài]:   # Vòng lặp ngoài
    for [biến lặp thứ hai] in [vòng lặp bên trong]:    # Vòng lặp bên
trong
```

Ví dụ :

```
num_list = [1, 2, 3]
alpha_list = ['a', 'b', 'c']

for number in num_list:
    print(number)
    for letter in alpha_list:
        print(letter)
```

Kết quả: 1 a b c 2 a b c 3 a b c

Một số keyword trong vòng lặp : Break, Continue,Pass and Else

Break: là từ khóa dùng để dừng và thoát khỏi vòng lặp

Ví dụ:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Kết quả: apple, banana

Continue : là từ khóa bỏ qua lần lặp 1 hàng hoặc các hàng cụ thể, và chuyển sang bước tiếp theo

Ví dụ:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

Kết quả: apple, cherry

Pass: Vòng lặp for không được để trống, nhưng trong trường hợp vì lý do nào đó vòng lặp không có nội dung, để tránh xảy ra lỗi thì ta dùng pass

Ví dụ:

```
for i in s:
    # Không có lỗi xảy ra
    pass
```

Ví dụ:

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

Kết quả: Current Letter : P

Current Letter : y

Current Letter : t

This is pass block

Current Letter : h

Current Letter : o

Current Letter : n

Good bye!

Else: là từ khóa trong vòng lặp chỉ định 1 đoạn lệnh sẽ thực hiện khi kết thúc vòng lặp.

Ví dụ:

```
for x in [1, 2 ,3]:
  print(x)
else:
  print("Finally finished!")
```

Kết quả: 1, 2 ,3 , Finally finished!

Chú ý : Else sẽ không thực hiện nếu vòng lặp kết thúc bằng break

Ví dụ:

```
for i in [1,2,3,4,5]:
    if i==3:
        break
    print (i)
else:
    print ("for loop is done")

print ("Outside the for loop")
```

Kết quả:

1

2

Outside the for loop

[Bài đọc] Sự khác nhau giữa while và for

Giới thiệu

Đối với vòng lặp for:

- Câu lệnh for lặp qua một tập hợp (collection), đối tượng có thể lặp (iterable object) hoặc kết quả của một hàm khởi tạo (generator function) (ví dụ range()).
- Vòng lặp for có cấu trúc đơn giản dễ viết, là cách viết tốt nhất khi chúng ta biết số lần lặp lại
- Khi chúng ta có một tập hợp được tạo sẵn: a set, tuple, list, map hoặc một chuỗi văn bản thì đơn giản, chúng ta sẽ dùng vòng lặp for.

Đối với vòng lặp while:

- Vòng lặp while là vòng lặp phụ thuộc điều kiện True hoặc False, vòng lặp chỉ dừng lại khi điều kiện là Sai
- Vòng lặp while sử dụng khi chúng ta không biết chính xác số lần lặp qua khối lệnh
- Nếu chúng ta không có cấu trúc dữ liệu gọn gàng để lặp lại hoặc chúng ta không có một hàm khởi tạo, bạn phải sử dụng vòng lặp while.

Ví dụ: Hiển thị các số từ 1 đến 11

Dùng vòng lặp for

```
for i in range(11)
print i
```

Dùng vòng lặp while

```
i = 0
while i <= 10:
    print (i)
    i += 1
```

Cách viết vòng lặp for đơn giản, dễ đọc hơn.

[Bài đọc] Vòng lặp lồng nhau

Giới thiệu

Vòng lặp lồng nhau là một vòng lặp xảy ra trong một vòng lặp khác, có cấu trúc tương tự như các câu lệnh if lồng vào nhau.

Cú pháp

```
for [first iterating variable] in [outer loop]:
    [do something]
    for [second iterating variable] in [nested loop]:
        [do something]
```

Chương trình khi chạy sẽ bắt đầu từ vòng lặp bên ngoài, thực hiện lần lặp đầu tiên. Lần lặp đầu tiên này sẽ kích hoạt vòng lặp nằm bên trong nó, vòng lặp này sẽ chạy cho đến khi hoàn thành. Sau đó chương trình sẽ quay lại vòng lặp phía ngoài cùng, hoàn thành lần lặp thứ 2 và kích hoạt lại lần nữa vòng lặp bên trong. Chương trình cứ lặp đi lặp lại thao tác này cho đến khi trình tự hoàn tất hoặc có câu lệnh ngắt trình tự này

Ví dụ:

```
num_list = [1, 2, 3]
```

```
alpha_list = ['a', 'b', 'c']

for number in num_list:
    print(number)
    for letter in alpha_list:
       print(letter)
```

Kết quả: 1,a,b,c,2,a,b,c,3,a,b,c

Chương trình sẽ hoàn thành lần lặp đầu tiên của vòng lặp bên ngoài và in ra 1, sau đó kích hoạt vòng lặp bên trong in ra a,b,c . Khi vòng lặp bên trong hoàn thành, sẽ quay lại vòng lặp bên ngoài in ra 2, sau đó lại kích hoạt in ra vòng lặp a,b,c , cứ tiếp tục in ra kết quả cho đến khi kết thúc vòng lặp

Ví dụ: In ra từng phần tử trong mảng lồng nhau

```
list_of_lists = [['hammerhead', 'great white', 'dogfish'], [0, 1, 2],
[9.9, 8.8, 7.7]]

for list in list_of_lists:
    for item in list:
        print(item)
```

Kết quả:

```
hammerhead
great white
dogfish
0
1
2
9.9
8.8
7.7
```

Khi sử dụng vòng lặp for lồng nhau, ta có thể in ra từng phần tử riêng lẻ trong mảng

[Bài tập] Fizz Buzz (Python)

Mục tiêu

Luyện tập sử dụng khối lệnh for và if…elif…else trong lập trình

Mô tả

Fizz Buzz là một bài toán lập trình cổ điển. Đây là phiên bản sửa đổi nhẹ của nó.
Viết chương trình lấy đầu vào là hai số nguyên: đầu và cuối khoảng (cả hai số đều thuộc khoảng).
Chương trình sẽ in ra các số trong khoảng này. Nhưng nếu số đó chia hết cho 3, bạn sẽ in ra "Fizz" thay vì nó, nếu số đó chia hết cho 5, in ra "Buzz" và nếu số đó chia hết cho cả 3 và 5, hãy in ra "FizzBuzz".

Hướng dẫn

Bước 1: Cho người dùng nhập biến

- Dùng hàm input() để nhập hai số bất kì.
- Dùng hàm split() để tách chuỗi số vừa nhập để lấy được số bắt đầu và số kết thúc.
- Dùng hàm int() để chuyển số bắt đầu và kết thúc về kiểu int.

Bước 2: Kiểm tra biến nhập

- Dùng if…else để xử lý hai trường hợp có thể xảy ra:
  - Nếu số kết thúc bé hơn số bắt đầu thì thông báo số kết thúc cần lớn hơn số bắt đầu
  - Nếu số bắt đầu lớn hơn hoặc bằng số kết thúc thì bắt đầu xử lý việc in kết quả

Bước 3: Bắt đầu xử lý việc in kết quả

- Dùng for…in range(start, end) để khởi tạo vòng lặp.

Lưu ý: vòng lặp sẽ bắt đầu từ startNumber và kết thúc ở endNumber + 1 (để vòng lặp sẽ lặp qua được endNumber)

- Trong mỗi vòng lặp sẽ dùng khối lệnh if…elif…else để kiểm tra biến lặp và in ra kết quả phù hợp:
  - Chia hết cho 3 và 5 thì in ra "FizzBuzz"

- Hoặc chia hết cho 3 thì in ra “Fizz”
- Hoặc chia hết cho 5 thì in ra “Buzz”
- Không thuộc 3 trường hợp trên thì ra chính giá trị của biến lặp

[Bài tập] The 21 Game

Mục tiêu

Luyện tập sử dụng khối lệnh while và if…elif..else trong lập trình

Mô tả

Trò chơi 21 là trò chơi số đếm được bắt đầu từ 0, hai người chơi lần lượt có thể thêm 1, 2 hoặc 3, vào tổng số. Tổng số không được vượt quá 21 và người chơi đếm tới 21 sẽ thua. Ở bài tập này, chúng ta sẽ giả định cho máy làm người chơi thứ 2.

Hướng dẫn

Bước 1: Khởi tạo vòng lặp

- Dùng while để khởi tạo vòng lặp, điều kiện kiểm tra sẽ là True => trò chơi sẽ luôn được tiếp tục lại cho đến khi người dùng nhập giá trị để kết thúc.

Bước 2: Sử lý trong vòng lặp

Bước 2.1: Khai báo biến tổng số đếm với giá trị khởi tạo = 0 (tạm gọi là current_number)

Bước 2.2: Dùng randint(0, 1) lấy ra một con số ngẫu nhiên để quyết định ai sẽ là người chơi hiện tại. (tạm gọi biến là current_player)

- Nếu bằng 0 thì current_player = “human”,
- Nếu bằng 1 thì current_player = “computer”

Bước 2.3:

- Dùng while để tạo vòng lặp đếm số, điều kiện kiểm tra sẽ là current_number <= 21. Tiếp theo sẽ tạo hàm xử lý bên trong mỗi vòng lặp.

- In ra giá trị current_number cho người dùng
- Dùng if…else để bắt đầu kiểm tra lượt chơi:
- Nếu là lượt chơi của người
    - Khai báo biến player_choice mang giá trị khởi tạo là chuỗi rỗng, biến này sẽ mang giá trị mà người chơi nhập vào.

    - Dùng while để tạo vòng lặp kiểm tra giá trị player_choice có thuộc danh sách [‘1’, ‘2’, ‘3’] hay không, vòng lặp sẽ kết thúc khi người dùng nhập player_choice có giá trị phù hợp.
    - Dùng hàm int() để chuyển biến player_choice sang kiểu int
    - Giá trị current_number hiện tại sẽ được cộng thêm player_choice
    - Dùng hàm if để kiểm tra giá trị của current_number đã lớn hơn hoặc bằng 21 chưa:

- Nếu đúng thì in ra số hiện tại, thông báo rằng người dùng đã thua và kết thúc vòng lập.
- Nếu sai thì gán giá trị mới cho current_player = ‘computer’ => lượt chơi tiếp theo sẽ là máy.
- Nếu là lượt chơi của máy
    - Khai báo biến computer_choice, mang giá trị số mà máy sẽ chọn
    - Dùng hàm randint(1,3) để lấy giá trị ngẫu nhiên từ 1 đến 3 rồi gán cho biến computer_choice
    - In computer_choice ra để thông báo cho người chơi biết.
    - Giá trị current_number lúc này sẽ được cộng thêm computer_choice.
    - Dùng hàm if để kiểm tra giá trị của current_number đã lớn hơn hoặc bằng 21 chưa:
        - Nếu đúng thì in ra số hiện tại, thông báo rằng người chơi đã thắng và kết thúc vòng lập.
        - Nếu sai thì thì gán giá trị mới cho current_player = ‘human’ => lượt chơi tiếp theo sẽ là người dùng.

Bước 3: Khi vòng lặp kết thúc, Dùng input() để hỏi người chơi có muốn chơi lại không và gán giá trị cho biến play_again

Bước 4: Dùng if…else để kiểm tra giá trị play_again:

- Nếu play_again bắt đầu bằng “y” thì dùng continue cho vòng lặp tiếp tục
- Nếu không thì dùng break kết thúc vòng lặp.

[Bài tập] Sinh bảng cửu chương (Python)

Mục tiêu

Luyện tập sử dụng khối lệnh for…in và nested loops trong lập trình

Mô tả

Viết chương trình hiển thị tất cả các bảng cửu chương từ 1 đến 9.

Hướng dẫn

Bước 1: Dùng for…in và range(1, 10) để tạo vòng lặp, biến số của vòng lặp tượng cho cho bảng cửu chương của số sắp sửa được in.

for number in range(1,10)

Bước 2: Trong mỗi vòng lặp trên, tiếp tục sử dụng for…in và hàm range(1, 10) để bắt đầu tính và in ra tích của number và các số từ 1 đến 9.

```
for i in range(1,10):
    print(str(number) + ' x ' + str(i) + ' = ' + str(number*i))
```

- Để hoàn thành bài tập, học viên cần:
    - Đưa mã nguồn lên GitHub
    - Dán link của repository lên phần nộp bài