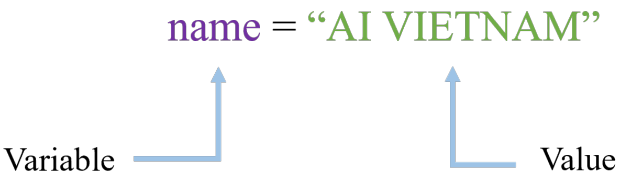
BASIC PYTHON - VARIABLE

Dinh-Tiem Nguyen và Quang-Vinh Dinh

1 Biến trong Python



Biến trong Python là tên được sử dụng để tham chiếu đến một vùng lưu trữ dữ liệu trong bộ nhớ, tên của biến là cách chúng ta đặt cho vùng lưu trữ đó để có thể truy cập và thao tác với dữ liệu trong chương trình của mình. Khi đặt tên biến sẽ có một số quy tắc mà bạn cần chú ý:

- Tên biến có thể chứa chữ cái, số, dấu gạch dưới "_"nhưng không được là số. Ví dụ: 'message_1', '_massage' nhưng không thể đặt là '1_message'.
- Tên biến có thể gồm nhiều từ, nhưng mỗi từ phải được viết liền, không được sử dụng khoảng trắng. Ví dụ 'fresh_apple' nhưng không thể đặt là 'fresh apple'
- Không được đặt tên biến trùng với những từ khóa và tên hàm trong Python. Ví dụ: Không nên đặt tên biến là 'list', 'for', 'from', 'if', 'is', 'False'...
- Tên biến chỉ cần ngắn thôi nhưng phải rõ ràng, nếu không thì có thể tạo tên dài hơn mà rõ ràng cũng được. Ví dụ: 'student_name' với 's_n' thì nên dùng biến tên 'student_name' vì khi đọc sẽ hiểu ngay.
- Bạn cũng nên cẩn thận khi sử dụng tên biến chứa chữ 'l' và 'o', vì nó giống số '1' và '0' nên có thể gây ra nhầm lẫn nếu quan sát không kỹ.

2 Bài tập

Viết chương trình với yêu cầu dưới đây, mỗi yêu cầu bạn sẽ viết nó tại một cell trong file jupyter notebook.

1. Bạn hãy tạo file và đặt tên file bất kỳ nhưng tên phải được viết thường và mỗi từ cách nhau bởi dấu gạch dưới '"_"', ví dụ như 'simple_message.ipynb'. Sau đó hãy lập trình in chuỗi '"Hello world!"' ra màn hình.
2. Gán một chuỗi '"Have a nice day!"' vào biến 'message'. Sau đó sử dụng câu lệnh 'print' để hiển thị biến này ra màn hình.
3. Bạn hãy tạo một chuỗi '"Let's have Tet holidays!"' và gán vào biến với tên '1_message'. Sau đó sử dụng câu lệnh 'print' để hiển thị biến này ra màn hình. Nếu kết quả hiển thị thông báo lỗi, hãy xác định lỗi này là gì và sửa như thế nào? Sau đó chạy lại chương trình.

BASIC PYTHON - STRING

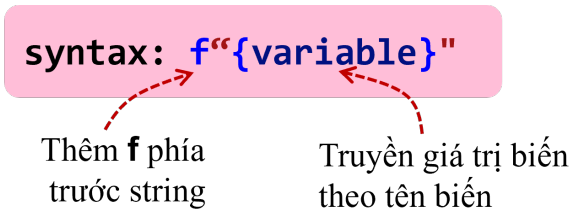
Dinh-Tiem Nguyen và Quang-Vinh Dinh

1 String

String trong Python là một chuỗi các ký tự, giống như các từ hoặc câu mà chúng ta viết. String có thể chứa chữ cái, số, ký tự đặc biệt và khoảng trắng. Để tạo một string trong Python, ta chỉ cần đặt các ký tự vào trong dấu ngoặc kép. Có thể dùng dấu ngoặc kép đơn (’), ngoặc kép (") hoặc dùng ba dấu nháy đối với chuỗi dài, nhiều dòng ("""hoặc ”’). Ví dụ:

```
page_name = "AI VIET NAM"
user_name = 'Tom'
greeting = """Xin chào Tom!
            Chào mừng bạn đến AI VIET NAM!"""
```

2 F-string



F-string là một cách đặc biệt để viết các chuỗi ký tự trong Python. Nó giúp chúng ta dễ dàng chèn các biến hoặc kết quả của các phép toán vào trong chuỗi. Chỉ cần đặt chữ ’f’ hoặc ’F’ trước dấu ngoặc kép của chuỗi, rồi đặt các biểu thức hoặc biến trong cặp dấu ngoặc nhọn .

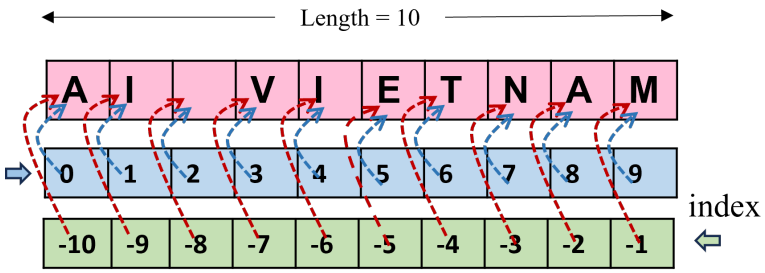
```
student_name = "Tom"
class_name = "AI VIETNAM"
message = f"{student_name} học lập trình tại {class_name}"
print(message)
```

Trong ví dụ này, student_name sẽ được thay thế bằng giá trị của biến student_name là "Tom", và class_name sẽ được thay thế bằng giá trị của biến class_name là "AI VIETNAM". Khi chúng ta chạy chương trình, kết quả sẽ là:

```
Tom học lập trình tại AI VIETNAM
```

3 Chiều dài, chỉ số trong chuỗi

Chúng ta có thể sử dụng hàm len() để đếm số ký tự trong một chuỗi. Hàm này sẽ trả về số ký tự có trong chuỗi đó. ví dụ:




```python
class_name = "AI VIETNAM"
length = len(class_name)
print(f"Số ký tự trong chuỗi là: {length}")
```

"Trong ví dụ này, class_name là chuỗi 'Hello'. Khi chúng ta sử dụng len(class_name), Python sẽ đếm số ký tự trong chuỗi này. Kết quả sẽ là:

```
Số ký tự trong chuỗi là: 10
```

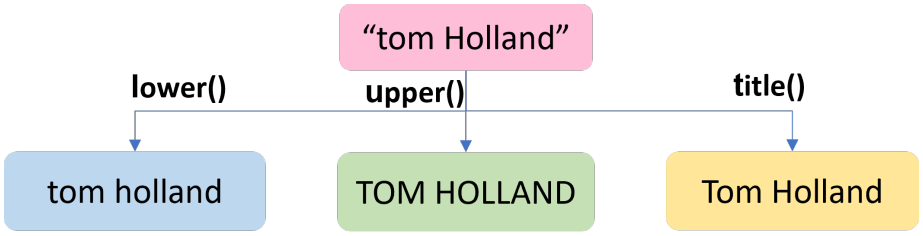
Tương tự, các bạn có thể đếm số ký tự trong bất kỳ chuỗi nào theo cách trên. Tiếp theo chúng ta sẽ học cách truy cập phần tử trong một chuỗi thông qua chỉ số index. Theo chiều từ trái sang phải chỉ số bắt đầu từ 0, có nghĩa là ký tự đầu tiên của chuỗi sẽ có chỉ số là 0. Đối với chiều ngược lại, chỉ số bắt đầu từ -1, có nghĩa là ký tự cuối cùng của chuỗi sẽ có chỉ số là -1. Ví dụ:

```python
class_name = "AI VIETNAM"
print(f"Ký tự đầu tiên trong chuỗi là: {class_name[0]}")
print(f"Ký tự cuối cùng trong chuỗi là: {class_name[-1]}")
```

Trong ví dụ này, class_name[0] sẽ lấy ký tự đầu tiên trong chuỗi, và class_name[-1] sẽ lấy ký tự cuối cùng trong chuỗi. Kết quả sẽ là:

```
Ký tự đầu tiên trong chuỗi là: A
Ký tự cuối cùng trong chuỗi là: M
```

4 Một số phương thức biến đổi chuỗi



Trong phần này, chúng ta sẽ học về ba phương thức phổ biến để biến đổi chuỗi trong Python: lower(), upper(), và title(). Chúng ta sẽ dùng chuỗi "tom Holland"để minh họa.

4.1 lower()

Phương thức lower() sẽ chuyển tất cả các ký tự trong chuỗi thành chữ thường.

```python
name = "tom Holland"
lowercase_name = name.lower()
print(lowercase_name)
```

```
================= Output ================
tom holland
=========================================
```

Khi chúng ta dùng name.lower(), tất cả các ký tự sẽ được chuyển thành chữ thường. Kết quả sẽ là "tom holland"

4.2 upper()

Phương thức upper() sẽ chuyển tất cả các ký tự trong chuỗi thành chữ hoa.

```
name = "tom Holland"
uppercase_name = name.upper()
print(uppercase_name)
```

```
================= Output ================
TOM HOLLAND
=========================================
```

Trong ví dụ này, khi chúng ta dùng name.upper(), tất cả các ký tự sẽ được chuyển thành chữ hoa

4.3 title()

Phương thức title() sẽ chuyển chữ cái đầu của mỗi từ thành chữ hoa và các ký tự còn lại thành chữ thường.

```
name = "tom Holland"
titlecase_name = name.title()
print(titlecase_name)
```

```
================= Output ================
Tom Holland
=========================================
```

Trong ví dụ này, khi chúng ta dùng name.title(), chữ cái đầu của mỗi từ sẽ được chuyển thành chữ hoa và các chữ cái còn lại thành chữ thường.

5 Bài Tập

Câu 1: Tạo một biến name gán giá trị là tên một người, sau đó hiển thị ra màn hình một thông báo chứa tên đó sử dụng f-string. Ví dụ name = “Alice”, mình sẽ in ra màn hình nội dung là “Alice is a great teacher!”
Câu 2: Tạo một biến và gán giá trị “ms Taylor” cho nó, sau đó thực hiện in ra màn hình giá trị biến đó được viết hoa ký tự đầu tiên mỗi từ hoặc viết toàn bộ bằng chữ chữ hoa, hoặc chữ thường.

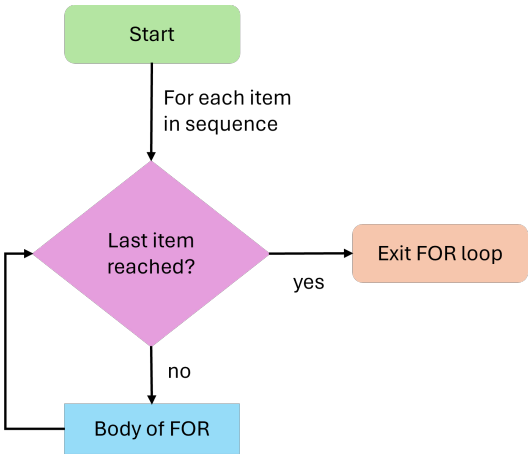
```
Input: name = "ms Taylor"
Output:
    Title case: Ms Taylor
    Upper case: MS TAYLOR
    Lower case: ms taylor
```

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.

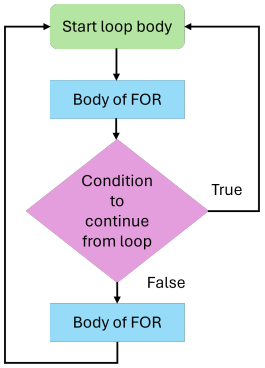
```
1
2 squares_dict = {x: x ** 2 for x in range
      (10)}
3 print(squares_dict)
```

```
================ 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)
```

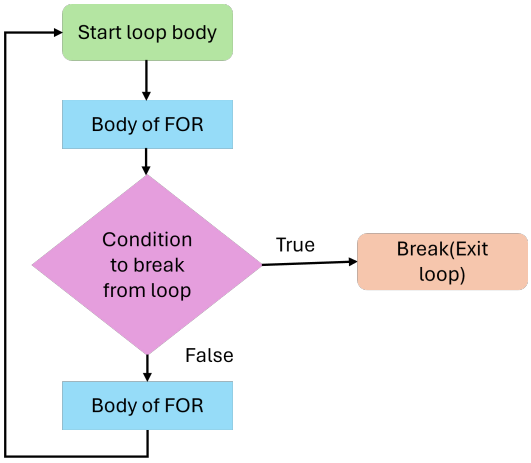
```
================ Output ===============
0
1
3
4
=======================================
```

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

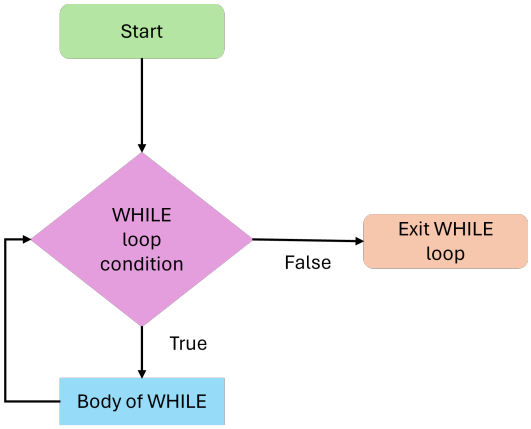
Trong bài viết này chúng ta đã tìm hiểu cách sử dụng vòng lặp for từ cơ bản đến nâng cao. Đây là kiến thức cơ bản nhưng rất quan trọng mà chúng ta cần phải thành thạo nó. Mặc dù bài viết đã cố gắng bao quát các trường hợp sử dụng vòng lặp for, tuy nhiên khi các bạn lập trình sẽ gặp nhiều trường hợp hơn 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.

BASIC PYTHON - WHILE LOOP

Dinh-Tiem Nguyen và Quang-Vinh Dinh

1 Giới thiệu

Trong hầu hết các ngôn ngữ lập trình, vòng lặp là một cấu trúc cho phép ta thực thi một khối code nhiều lần. Trong Python, có hai loại vòng lặp chính là for và while. Bài viết này sẽ tập trung vào vòng lặp while, giải thích cách hoạt động qua các ví dụ cụ thể để có thể dễ dàng hiểu và áp dụng.



1.1 Vòng lặp while là gì?

Vòng lặp while thực thi một khối mã liên tục miễn là điều kiện được chỉ định là đúng. Cú pháp của vòng lặp while như sau:

```
while điều_kiện:
    # Khối code trong while
```

Vòng lặp while bắt đầu bằng việc kiểm tra điều kiện. Nếu điều kiện là True, khối code bên trong vòng lặp sẽ được thực thi. Sau khi khối code kết thúc, điều kiện sẽ được kiểm tra lại. Quá trình này tiếp tục cho đến khi điều kiện trở thành False.
Ví dụ chương trình in ra màn hình các số từ 1 đến 5:

```
i = 1
while i <= 5:
    print(i)
    i += 1


#ai vietnam
```

```
================= Output ================
1
2
3
4
5
=========================================
```

Một ví dụ khác để hiểu vòng lặp while, hãy tưởng tượng bạn đang chơi một trò chơi bật nhảy với một quả bóng. Bạn sẽ tiếp tục nhảy và đập bóng xuống đất cho đến khi bạn mệt và không thể nhảy nữa. Ở đây, vòng lặp while hoạt động tương tự như việc bạn bật nhảy. Bạn sẽ tiếp tục thực hiện một hành động (nhảy và đập bóng) cho đến khi một điều kiện nào đó không còn đúng nữa (bạn cảm thấy mệt và không thể nhảy tiếp).

```python
has_energy = True
jump_count = 0

while has_energy:
    jump_count += 1
    print(f"Jump {jump_count} time(s)")
    # Giả sử sau 5 lần nhảy, bạn mệt và dừng lại
    if jump_count == 5:
        has_energy = False
```

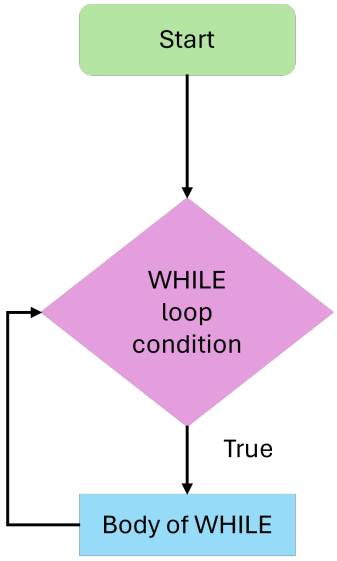
```
================ Output ===============
Jump 1 time(s)
Jump 2 time(s)
Jump 3 time(s)
Jump 4 time(s)
Jump 5 time(s)



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

has_energy là biến điều kiện. Mỗi lần vòng lặp chạy, bạn sẽ nhảy một lần và tăng số lần nhảy jump_count lên. Khi số lần nhảy đạt đến 5, bạn cảm thấy mệt và đặt has_energy thành False, vòng lặp kết thúc.

2 Vòng lặp vô hạn

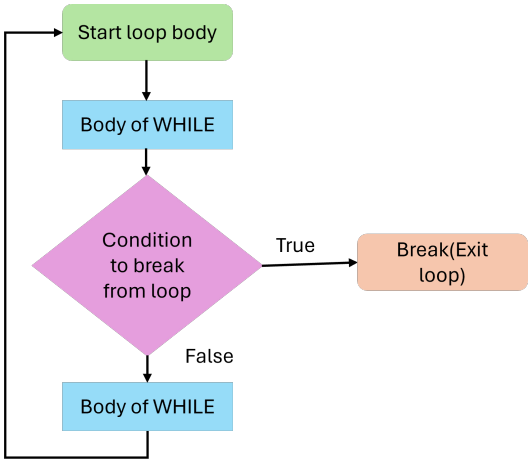


Một vòng lặp vô hạn là một vòng lặp tiếp tục thực thi câu lệnh trong vòng lặp đến mãi mãi, vì điều kiện của vòng lặp luôn là True. Mặc dù chúng thường là điều cần tránh, nhưng cũng có những trường hợp chúng ta có thể tạo ra một vòng lặp vô hạn một cách cố ý, chẳng hạn khi xây dựng các chương trình liên tục lắng nghe đầu vào hoặc chạy các tiến trình máy chủ.
Ví dụ Vòng lặp vô hạn:

```python
while True:
    print("This loop will run forever!")
```

Trong ví dụ trên, điều kiện của vòng lặp while được đặt là True, có nghĩa là vòng lặp sẽ chạy mãi mãi vì điều kiện luôn đúng. Bên trong vòng lặp, câu lệnh print("This loop will run forever!") được thực thi lặp đi lặp lại. Để thoát khỏi vòng lặp, chúng ta cần phải dừng chương trình bằng cách kết thúc quá trình thực thi của nó. Trong hầu hết các môi trường phát triển, chúng ta có thể làm điều này bằng cách nhấn Ctrl + C nếu chúng ta chạy chương trình bằng dòng lệnh hoặc nhấn nút stop trên công cụ lập trình.

3 Sử dụng break trong while



Chúng ta có thể sử dụng lệnh break để thoát khỏi vòng lặp ngay lập tức, bất kể điều kiện của vòng lặp là gì. Sử dụng break khi chúng ta muốn dừng vòng lặp dựa trên một điều kiện khác xảy ra trong quá trình thực thi.
Ví dụ sử dụng break để thoát khỏi vòng lặp:

```
#ai vietnam
i = 1
while i <= 10:
    print(i)
    if i == 5:
        break
    i += 1
```

```
================= Output ================
1
2
3
4
5
=========================================
```

Trong ví dụ này vòng lặp while sẽ chạy miễn là i nhỏ hơn hoặc bằng 10. Bên trong vòng lặp, giá trị của i được in ra và sau đó được tăng lên một đơn vị. Khi i đạt giá trị 5, lệnh if i == 5: break sẽ được thực thi, làm cho vòng lặp dừng lại ngay lập tức. Các giá trị từ 1 đến 5 sẽ được in ra, sau đó vòng lặp kết thúc khi i bằng 5.

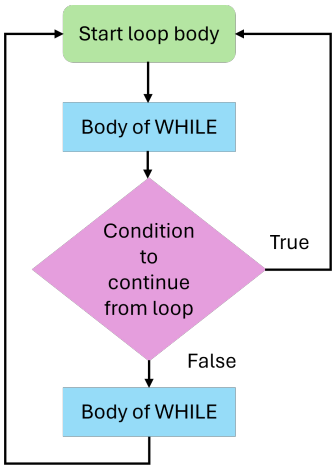
4 Sử dụng continue trong while

Chúng ta có thể sử dụng lệnh continue để bỏ qua các lệnh còn lại của khối code trong vòng lặp và bắt đầu lần lặp tiếp theo. Continue thường được sử dụng khi chúng ta muốn bỏ qua một số bước trong vòng lặp dựa trên một điều kiện nào đó.
Ví dụ sử dụng continue để bỏ qua lần lặp của những trường hợp i chia hết cho 2:

```
#ai vietnam
i = 0
while i < 10:
    i += 1
    if i % 2 == 0:
        continue
    print(i)
```

```
================= Output ================
1
3
5
7
9
=========================================
```

Trong ví dụ này vòng lặp while sẽ chạy miễn là i nhỏ hơn 10. Mỗi lần vòng lặp chạy, giá trị của i được tăng lên một đơn vị. Nếu i là số chẵn (i % 2 == 0), lệnh continue sẽ được thực thi, bỏ qua câu lệnh print(i) và bắt đầu lần lặp tiếp theo. Kết quả chỉ gồm các số lẻ từ 1 đến 9 được hiển thị ra màn hình.



5 Sử dụng while với list

Chúng ta cũng có thể sử dụng vòng lặp while để duyệt qua các phần tử của một danh sách. Cachs này thường dùng khi chúng ta cần thực hiện các thao tác lặp lại trên các phần tử của danh sách mà không sử dụng vòng lặp for.
Ví dụ sử dụng while để duyệt qua danh sách

```python
#ai vietnam
fruits = ["apple", "banana", "cherry"]
i = 0
while i < len(fruits):
    print(fruits[i])
    i += 1
```

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

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

Trong ví dụ trên, chúng ta có một danh sách fruits chứa các phần tử là tên các loại trái cây. Vòng lặp while sẽ chạy khi chỉ số i nhỏ hơn độ dài của danh sách fruits. Trong mỗi lần lặp, phần tử tại vị trí i trong danh sách fruits sẽ được in ra. Chỉ số i sau đó được tăng lên một đơn vị, giúp duyệt qua các phần tử tiếp theo của danh sách. Khi i bằng độ dài của danh sách, vòng lặp sẽ kết thúc. Ngoài ra các bạn có thể thử với dictionary, tuple...

6 Kết Luận

Vòng lặp while là một công cụ mạnh mẽ trong Python, cho phép ta thực thi một khối code nhiều lần dựa trên điều kiện nào đó. Tuy nhiên, chúng ta cần cẩn thận để tránh vòng lặp vô hạn và sử dụng các lệnh break và continue một cách hợp lý để kiểm soát luồng của vòng lặp.
Hy vọng rằng qua bài viết này, các bạn đã hiểu rõ hơn về vòng lặp while và cách sử dụng nó trong Python. Hãy thử áp dụng những gì các bạn đã học vào các bài tập và project trên lớp để nắm vững kiến thức hơn!