Softech®
Gather to thrive together

Aptech™
COMPUTER EDUCATION
Unleash your potential

BÀI 3

Câu lệnh If else, Vòng lặp While, for

Tóm Tắt Nội Dung

Trong bài học này chúng ta sẽ đi tìm hiểu lần lượt các nội dung

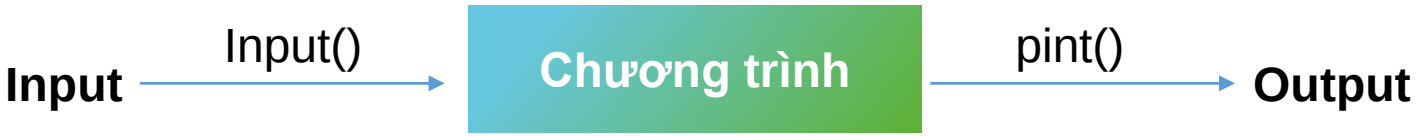
1. Tìm hiểu về lệnh input()
2. Tìm hiểu về toán tử Logical trong Python
3. Tìm hiểu về câu lệnh điều kiện rẻ nhánh if else
4. Tìm hiểu về vòng lặp while , for trong Python

3.1 Tìm hiểu về lệnh input()

Định nghĩa: Input là gì ?

Trong mọi chương trình hệ thống đều có sự tương tác dữ liệu 2 chiều: đầu vào (input) và đầu ra (output)

Trong các bài học trước thì chúng ta đã làm quen với lệnh **print** đóng vai trò là đâu ra của chương trình. Bài này chúng ta sẽ làm quen với input là đầu vào



Hàm **input()** dùng thực hiện việc **lấy dữ liệu nhập từ bàn phím** làm giá trị đầu vào cho chương trình

3.1 Tìm hiểu về lệnh input()

Cú pháp lệnh input()

tên_biến = input(‘Nội dung gợi ý’)

- **tên_biến** là tên mà ta muốn đặt cho biến
- **Nội dung gợi ý** là là câu từ mà mình hiển thị ra để người dùng biết nên nhập cái gì vào từ bàn phím

Ví dụ:

```
name  = input(‘Vui lòng nhập vào tên của bạn: ’)
print(‘Xin chào’ + name)
```

3.1 Tìm hiểu về lệnh input()

Ví dụ về toán tử với input()

Lấy giá trị 2 biến x và y nhập lên từ bàn phím là thực hiện các phép tính với chúng”

```
x  = input('Vui lòng nhập giá trị của biến x: ') #Ví dụ nhập vào: 5
y  = input('Vui lòng nhập giá trị của biến y: ') #Ví dụ nhập vào: 3

print(x + y) # Kết quả in ra Terminal: 53
```

Tại sao kết quả không phải là 8 mà là 53 ?

Giá trị nhận được từ input() luôn là một chuỗi

Do vậy với toán tử + trên nó trở thành lệnh nối chuỗi chứ không thực hiện một phép tính

3.1 Tìm hiểu về lệnh input()

Chuyển đổi dữ liệu

Để có được kiểu dữ liệu mình muốn, chúng ta có thể sử dụng các hàm int(), float(), str() để chuyển đổi kiểu dữ liệu:

Int() **Giá trị trả về = int(<giá trị>)**

```
x  =  '2‘ # x đang là str có giá trị = 2
print(int(x)) # x giờ là int = 2
```

float() **Giá trị trả về = int(<giá trị>)**

```
x  =  ‘3.14156‘ # x đang là str có giá trị = ‘3.14156’
print(float(x)) # x giờ là int = 3.14156
```

strt() Thì ngược lại, chuyển dữ liệu từ kiểu số sang kiểu chuỗi

3.1 Tìm hiểu về lệnh input()

Eval() tính toán giá trị từ một biểu thức trong chuỗi

Cú pháp **Giá trị trả về = eval("<biểu thức>")**

```
print(eval("12 + 7"))
# kết quả cho ra một số : 19
```

Tính toán giá trị từ biểu thức nhập vào bằng input()

```
x = eval(input("Nhập vào biểu thức toán học: "))
print("Giá trị của biểu thức x là", x)
```

Nhận biết đồng thời nhiều biểu thức trên một hang, cách bởi dấu ,

```
m, n, p = eval(input("Nhập vào các số m, n, p (cách nhau bởi dấu phẩy) "))
print("Các số đã nhập là", m,n, p)
```

3.2 Toán tử Logic trong Python

Dữ liệu Logic là gì ?

Dữ liệu logic (bool) là loại dữ liệu chỉ có 2 giá trị Đúng (True) và Sai (False).
Dữ liệu logic được dùng khi mô tả các điều kiện hoặc phép so sánh số hoặc chữ

Hàm **bool()** sẽ kiểm tra xem giá trị của đối số truyền vào là True hay False

```
print(3 > 2)
# kết quả cho ra một số : True
```

```
x = 2 > 3
print(bool(x)) # kết quả cho ra một số : False
```

3.2 Toán tử Logic trong Python

Toán tử So Sánh

Toán tử	Ý Nghĩa	Ví dụ	Diễn giải Phép so sánh
==	So sánh bằng	x == y	x có bằng y hay không ?
!=	So sánh không bằng	x != y	x khác y hay không ?
>	So sánh lớn hơn	x > y	x lớn hơn y hay không
>=	Lớn hơn hoặc bằng	x >= y	x lớn hơn hoặc bằng y hay không
<	So sánh bé hơn	x < y	x bé hơn hoặc bằng y hay không
<=	Bé hơn hoặc bằng	x <= y	x bé hơn hoặc bằng y hay không

Toán tử so sánh trên sẽ tạo ra kiểu dữ liệu Logic: True hoặc False

```
print(3 > 2)
# kết quả cho ra một số : True
```

3.2 Toán tử Logic trong Python

Toán tử Logic

Ví dụ: Cho x = 5, y = 8

Toán tử	Ý Nghĩa	Ví dụ	Kết quả
and	và	x > 3 and y > 5	True and True = True
or	Hoặc	x > 3 or y > 9	True and False = False
not	Phủ định	not(x > 3)	not(True) = False

Bảng định nghĩa giá trị của toán tử **and** và **or**

x	and	y	Kết quả
True	and	True	True
True	and	False	False
False	and	False	False
False	and	True	False

x	or	y	Kết quả
True	or	True	True
True	or	False	True
False	or	False	False
False	or	True	True

3.3 Câu lệnh if, else, elif

Từ thực tế đến ngôn ngữ lập trình

> **"Nếu trời tạnh mưa thì tôi sẽ được đi chơi,
> Còn nếu không tôi sẽ ở nhà học bài"**

Trong phát biểu trên chúng ta có 2 hành động là "đi chơi" và "học bài". Việc thực hiện 2 hành động này phụ thuộc vào điều kiện của thời tiết sau từ khóa "nếu"

Trong các ngôn ngữ lập trình các lệnh thực hiện theo điều kiện như vậy sẽ được thể hiện bằng từ khóa **if**

Câu lệnh **if** mô tả bằng ngôn ngữ con người như sau :

Nếu <điều kiện đúng> thì
    <thực hiện khối lệnh này>

3.3 Câu lệnh if, else, elif

Cú pháp if ngắn

Câu lệnh "if" được sử dụng để kiểm tra một điều kiện và thực thi một khối mã chỉ khi điều kiện đó là đúng.

if <điều kiện kiểm tra>:

<khối lệnh thực thi khi điều kiện đúng>

Điều kiện kiểm tra: Trả về một giá trị logic (True/False) và kết thúc bằng dấu :

Khối lệnh của if: Thụt vào một khoảng bằng nhau tối thiểu một dấu cách so với if. Điều này rất quan trọng, nếu không đúng lệnh sẽ không chạy được.

```
x = 15
if x > 10:
    print(“x lớn hơn 10”)
#Output: x lớn hơn 10
```

```
x = 7
if x > 10:
    print(“x lớn hơn 10”)
```

3.3 Câu lệnh if, else, elif

Cú pháp if đầy đủ

if <điều kiện kiểm tra>:
<khối lệnh thực thi khi điều kiện đúng >
else:
<khối lệnh thực thi khi điều kiện sai>

Điều kiện kiểm tra: Trả về một giá trị logic (True/False) và kết thúc bằng dấu :

else: sau else có thêm dấu :

```
x = 15
if x > 10:
    print(“x lớn hơn 10”)
else:
    print(“x nhỏ hơn 10”)
```

3.3 Câu lệnh if, else, elif

Cú pháp if-elif-else

Câu lệnh "if-elif-else" được sử dụng để kiểm tra nhiều điều kiện khác nhau và thực thi các khối mã tương ứng. Cú pháp của câu lệnh if-elif-else như sau:

```
if <điều kiện 1>:
        <khối lệnh thực thi khi điều kiện 1 đúng >
elif <điều kiện 2>:
        <khối lệnh thực thi khi điều kiện 2 đúng >
…
else:
        <khối lệnh thực thi khi không có điệu kiện nào đúng>
```

3.3 Câu lệnh if, else, elif

Cú pháp if-elif-else

Câu lệnh "if-elif-else" được sử dụng để kiểm tra nhiều điều kiện khác nhau và thực thi các khối mã tương ứng. Cú pháp của câu lệnh if-elif-else như sau:

Ví dụ:

```
x = 10
if x < 5:
    print(“x nhỏ hơn 5”)
elif: x < 10:
    print(“x nhỏ hơn 10”)
else:
    print(“x lơn hơn hoặc bằng 10”)
```

Kết quả: x lớn hơn hoặc bằng 10

3.3 Câu lệnh if, else, elif

Câu lệnh pass trong Python

Từ khóa **pass** được sử dụng để đánh dấu một khối mã rỗng hoặc tạm thời không có nội dung. Nó không làm gì cả và được sử dụng để bỏ qua một phần của mã trong trường hợp bạn không muốn thực hiện bất kỳ hành động nào tại thời điểm đó, nhưng cú pháp yêu cầu phải có một khối mã

```
if x < 5:
    pass #Không muốn làm gì khi điều kiện đúng

for i in range(10):
    pass  # không có hành động cụ thể trong mỗi lần lặp


def my_function():
     pass #Không muốn làm gì khi gọi hàm
```

3.4 Vòng lặp trong Python

Câu lệnh range() trong Python

Hàm **range()** trả về một dãy số, bắt đầu từ 0 theo mặc định và tăng dần 1 (theo mặc định) và dừng trước một số đã chỉ định.

range([start], stop, [step])

start: là vị trí bắt đầu (Tùy chọn, mặc định là 0)
stop: vị trí kết thúc (Bắt buộc, và không bao gồm giá trị stop)
Step: bước nhảy (Tùy chọn, mặc định là 1, có thể số số nguyên âm)

```
x = range(3, 6)
for i in x :
   print(i)
#output:
3
4
5
```

Hàm **range(3,6)** cho ra một danh sách các số: 3, 4, 5 (stop -1)

Vòng lặp for, lặp qua danh sách đó, và in từng phần tử của danh sách số.

3.4 Vòng lặp trong Python

Câu lệnh range() trong Python

Một số trường hợp khi sử dụng Hàm **range()** trả về một dãy số:

Cú pháp	Ý nghĩa
range(0,n)	Cho ra dãy số từ 0,1…n-1
range(1,n,2)	Cho ra dãy số từ 0,3,5,7…n-1 (tăng dần 2 đơn vị)
range(n)	Cho ra dãy số từ 0,1…n-1 (lúc này n là stop, start là 0)
rangle(n,0,-1)	Cho ra dãy số n, n-1…1(chiều giảm dần)

Chúng ta sẽ tìm hiểu range rõ hơn qua một vài ví dụ về vòng lặp **for** sau đây

3.4 Vòng lặp trong Python

Thế nào là vòng lặp ?

Ví dụ trong đời sống có các phát biểu sau:

1. Bạn hãy nhảy lò cò **2 lần**
2. Chạy **5 lần** quanh sân bóng
3. Nói thật to **10 lần** 2 từ “Xin chào”

Thì có nghĩa bạn đang **lặp đi lặp lại** một hoặc nhiều hành động gì đó.

Thì trong ngôn ngữ lập trình, việc lặp lại đó được viết thành câu lệnh như là **for**, **while**

Vòng lặp là một khái niệm trong lập trình được sử dụng để lặp lại các hành động nhiều lần, mà không cần phải viết lại các lệnh nhiều lần.

3.4 Vòng lặp trong Python

Thế nào là vòng lặp ?

3 Nói thật to **10 lần** 2 từ “Xin chào”

Chúng ta có thể mô tả phát biểu này trong lập trình Python như sau:

```
print(“Xin chào”)
print(“Xin chào”)
print(“Xin chào”)
print(“Xin chào”)
print(“Xin chào”)
print(“Xin chào”)
print(“Xin chào”)
print(“Xin chào”)
print(“Xin chào”)
print(“Xin chào”)
```

```
for i in range(1,11) :
    print(“Xin chào”, i)
```

Thay vì dùng 10 lần lệnh print() chúng ta có thể sử dụng 1 lệnh print() kết hợp với vòng lặp for như trên

Code vì thế trở nên ngắn gọn hơn

3.4 Vòng lặp trong Python

Vòng lặp for

Vòng lặp "for" được sử dụng để **lặp** qua một tập hợp các phần tử (như danh sách, chuỗi, tuple) và thực thi một khối mã cho mỗi phần tử trong tập hợp đó. Cú pháp của vòng lặp "for" như sau:

```
for phần_tử_lặp_qua in tập_hợp:
    # Các câu lệnh thực thi cho mỗi phần tử
```

Phần_tử_lặp_qua: là biến lưu trữ giá trị của phần tử trong mỗi lần lặp

Tập_hợp: là một tập hợp có thể lặp qua, chẳng hạn như danh sách, chuỗi, hoặc tuple

```
x = range(1,6)
for i in x:
    print(i)
```

3.4 Vòng lặp trong Python

Ví dụ về Vòng lặp for

Sử dụng với hàm **range()**

```
for i in range(1, 5):
    print(i)
```

Lặp qua một string

```
for x in “banana”:
    print(x)
```

Vòng lặp với câu lệnh **break** và **continue**

```
x = range(1, 5)
for i in x:
    if i == 3:
        continue
    print(i)
    if i == 4:
        break
```

Trong ví dụ trên, khi gặp phần tử có giá trị là “3", câu lệnh "continue" sẽ được thực thi, bỏ qua các câu lệnh phía dưới và chuyển đến phần tử tiếp theo trong vòng lặp. Khi gặp phần tử có giá trị là “4", câu lệnh "break" được thực thi, thoát khỏi vòng lặp ngay lập tức.

3.4 Vòng lặp trong Python

Vòng lặp while

Vòng lặp "while" được sử dụng để lặp lại một khối mã miễn là điều kiện **đúng** (True). Cú pháp của vòng lặp "while" như sau:

```
while <điều kiện kiểm tra>:
    # Các câu lệnh thực thi khi điều kiện đúng
```

"condition" là một biểu thức hoặc một điều kiện logic. Vòng lặp sẽ tiếp tục lặp lại cho đến khi điều kiện này trở thành sai (False).

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

vòng lặp "while" được sử dụng để lặp lại việc in ra giá trị của biến "count" từ 0 đến 4. Mỗi lần lặp, giá trị của "count" được tăng lên 1 và điều kiện "count < 5" được kiểm tra

3.4 Vòng lặp trong Python

Điều kiện dừng vòng lặp

Trong vòng lặp "while", điều kiện đúng (True) được kiểm tra ở đầu mỗi lần lặp. Nếu điều kiện đúng từ đầu, vòng lặp sẽ không được thực thi. Điều này có thể gây ra vòng lặp vô hạn nếu không có một cơ chế nào đảm bảo điều kiện sẽ trở thành sai (False) tại một thời điểm nào đó

```
count = 5
while count > 0:
        print(count)
```

Vòng lặp sẽ không bao giờ kết thúc vì điều kiện "count > 0" luôn đúng. Để tránh vòng lặp vô hạn, chúng ta cần đảm bảo rằng điều kiện cuối cùng sẽ trở thành sai tại một điểm nào đó trong vòng lặp

3.4 Vòng lặp trong Python

Nguồn tham chiếu

- Dữ liệu Logic
  https://www.w3schools.com/python/python_booleans.asp
- Tổng quan về if, else, elif
  https://www.w3schools.com/python/python_conditions.asp
- Vòng lặp for
  https://www.w3schools.com/python/python_for_loops.asp
- Vòng lặp While
  https://www.w3schools.com/python/python_while_loops.asp

Tổng kết lại bài 3

1. Nắm được cách sử dụng lệnh input()
2. Nắm được toán tử Logical trong Python
3. Cách sử dụng câu lệnh điều kiện rẻ nhánh if else
4. Nắm được cách sử dụng vòng lặp while trong Python
5. Nắm được cách sử dụng vòng lặp for trong Python