

---

# Biến, kiểu dữ liệu và toán tử

Khóa học: Python căn bản

---

# Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài “Thư viện Turtle”

Tóm tắt lại các phần đã học từ bài “Thư viện Turtle”

# Mục tiêu

---

- Trình bày được khái niệm biến
- Trình bày được cú pháp khai báo biến
- Trình bày được khái niệm kiểu dữ liệu
- Trình bày được các toán tử thông dụng
- Khai báo và sử dụng được biến
- Sử dụng được các kiểu dữ liệu
- Sử dụng được các toán tử cơ bản

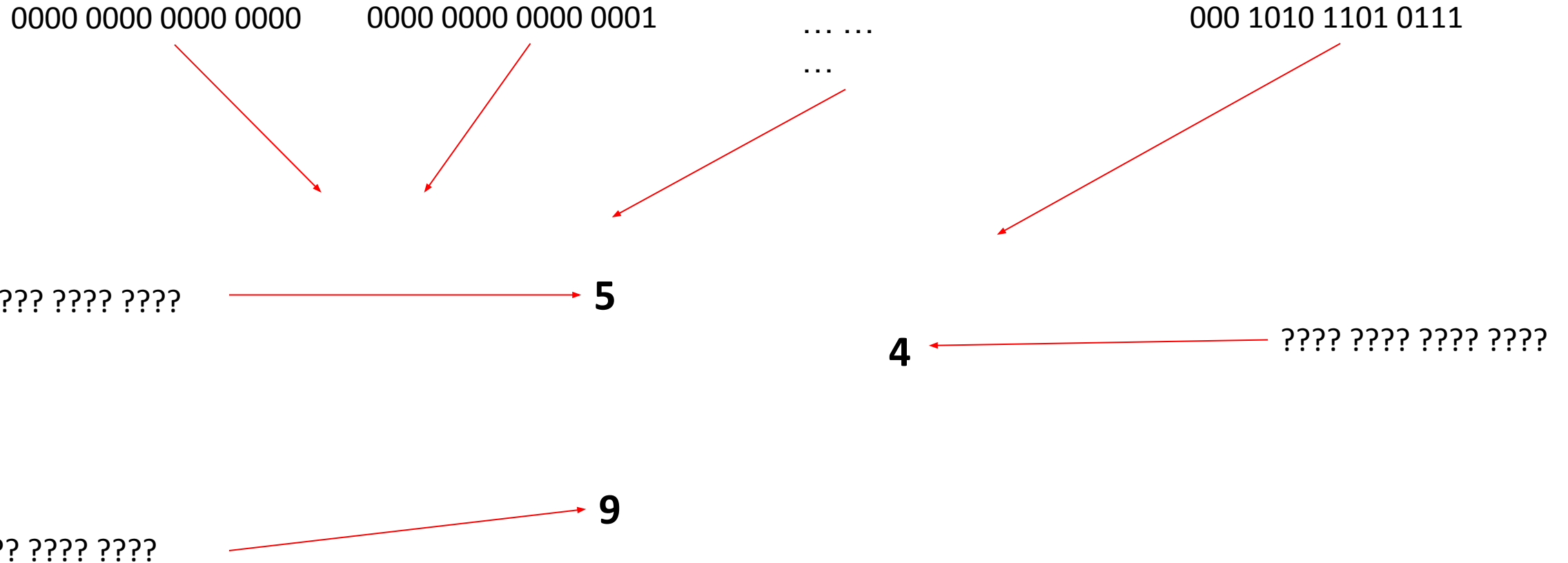
# Demo

Sử dụng code demo sẵn về hình mặt cười

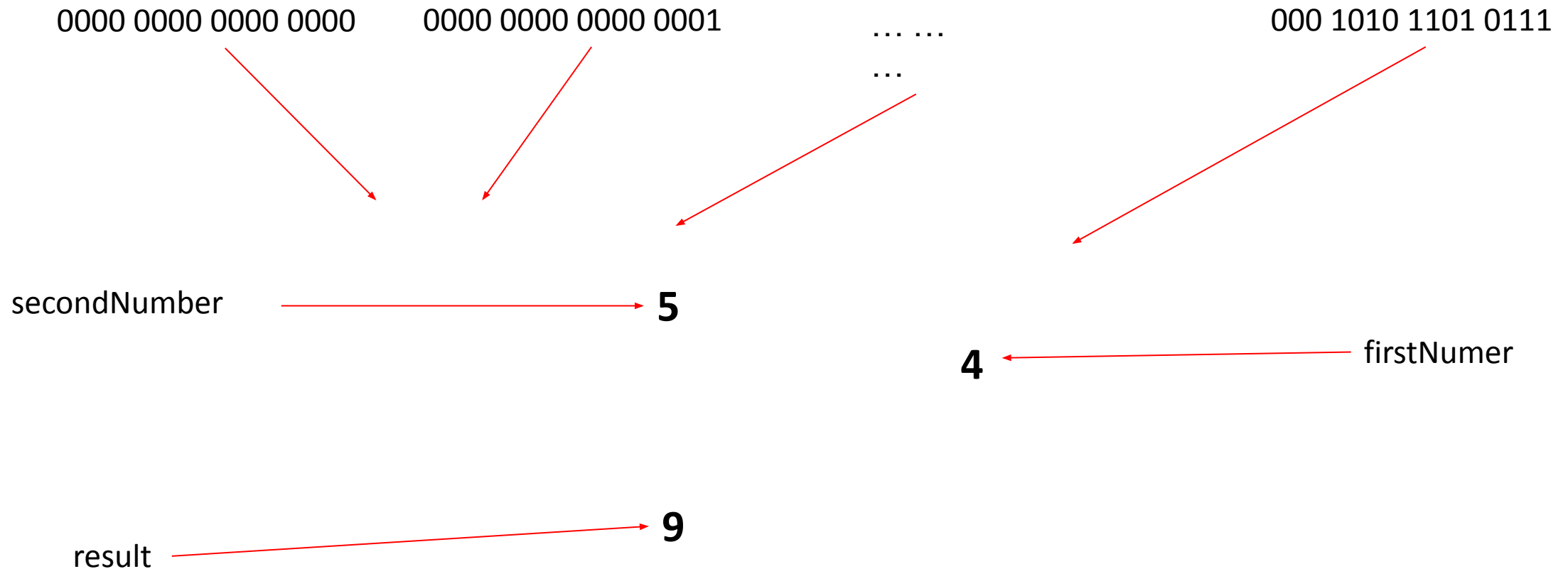
Thay đổi giá trị lặp lại bằng biến

Giải thích được vai trò của biến

# Lưu trữ dữ liệu trong bộ nhớ



# Lưu trữ dữ liệu trong bộ nhớ



# Biến (variable)

- **Biến** là một tên gọi được gán cho một vùng nhớ chứa dữ liệu
- Dữ liệu được lưu trữ trong vùng nhớ của biến được gọi là **giá trị** (value)
- Có thể truy nhập, gán hay thay đổi giá trị của biến
- Khi gán một giá trị mới thì giá trị cũ sẽ bị ghi đè lên
- Với hầu hết các ngôn ngữ lập trình thì cần phải khai báo biến trước khi sử dụng. Trong python bỏ qua phần khai báo và thực hiện luôn phần khởi tạo.
- Chẳng hạn:

tên\_biến = giá trị;

Ví dụ: name = "Nestech-Online"

# Ví dụ về biến

---

**BEGIN**

**DISPLAY** "Enter 2 numbers: "

**INPUT** A, B

$C = A + B$

**DISPLAY** C

**END**

- A, B và C là các biến trong đoạn mã giả trên
- Tên biến giúp chúng ta truy cập vào bộ nhớ mà không cần dùng địa chỉ của chúng
- Hệ điều hành đảm nhiệm việc cấp bộ nhớ còn trống cho những biến này
- Để tham chiếu đến một giá trị cụ thể trong bộ nhớ, chúng ta chỉ cần dùng tên của biến



# Khởi tạo biến

- Khởi tạo biến sẽ bao gồm khai báo biến và gán giá trị cho biến
- x là tên biến
- Dấu bằng (=) được dùng để **gán giá trị** cho biến

Cú pháp:

*variableName = value*

Khai báo

Gán giá trị

Ví dụ:

```
name = "nестech-online"
```

# Đặt tên cho biến

---

- Tên biến phải bắt đầu bằng một ký tự alphabet (a-zA-z\_)
- Theo sau ký tự đầu có thể là các ký tự chữ, số ...
- Nên tránh đặt tên biến trùng tên các từ khoá
- Tên biến nên mô tả được ý nghĩa của nó
- Tránh dùng các ký tự gây nhầm lẫn
- Tên biến có phân biệt chữ hoa và chữ thường
- Nên áp dụng các quy ước đặt tên biến chuẩn khi lập trình

# Đặt tên cho biến: Ví dụ

---

- Ví dụ đặt tên biến đúng
  - arena
  - s\_count
  - marks40
  - class\_one
- Ví dụ đặt tên biến sai
  - 1sttest
  - oh!god
  - start... end

# Giá trị của biến

---

- Ví dụ
  - 5     **số / giá trị số nguyên** (integer)
  - 5.3   **số / giá trị số thập phân** (decimal)
  - "Black"   **Giá trị chuỗi** (string)
  - 'C'     **Giá trị ký tự** (character)
  - true và false là các **giá trị logic** (boolean)

# Demo

Biến, kiểu dữ liệu

Các kiểu dữ liệu nguyên thủy

Chuỗi

Giá trị mặc định

# Kiểu dữ liệu (Data Type)

- Kiểu dữ liệu là một cách phân loại dữ liệu cho trình biên dịch hoặc thông dịch hiểu các lập trình viên muốn sử dụng dữ liệu.
- Kiểu dữ liệu mô tả loại dữ liệu sẽ được lưu trong biến
- Các kiểu dữ liệu khác nhau được lưu trữ trong biến là:
  - Số (numbers)
    - Số nguyên: 10 hay 83839
    - Số thực: 15.33 hay 23.6677
    - Số dương: 3, 4
    - Số âm: -6, -7
  - Chuỗi: "Hello"
  - Ký tự: 'A'
  - Logic: True, False

# Kiểu dữ liệu (Data Type)

- Một kiểu dữ liệu cung cấp một bộ các giá trị mà từ đó một biểu thức (như biến, hàm ...) có thể lấy giá trị của nó
- Trong python khi khai báo biến và gán cho biến một giá trị đồng nghĩa xác định kiểu dữ liệu cho biến đó.
- Ví dụ:
  - `x = 10`  
# x mang giá trị 10, kiểu dữ liệu của x là số nguyên
  - `gender = true`  
# gender mang giá trị true, kiểu dữ liệu của gender là kiểu boolean

# Chuỗi

- Chuỗi bao gồm các ký tự liên tiếp nhau
- Có thể khai báo chuỗi sử dụng dấu nháy đơn hoặc nháy kép
- Ví dụ:

```
year = "2017"
```

```
myString = ""
```

- Nối chuỗi với toán tử +

```
year = "2017"  
print("Năm nay là " + year)
```

```
year = 2017  
print("Năm nay là " + year)
```

→ Error



# Số

- Có thể sử dụng số nguyên hoặc số thập phân
- Ví dụ:

```
# số nguyên dương  
year = 2017
```

```
# số thập phân  
dollarExchangeRate = 22727.50
```

# Boolean

- Kiểu dữ liệu boolean chỉ có hai giá trị là true và false

```
myBool = True  
print(myBool) # True
```

```
myBool = (5 < 2)  
print(myBool) # False
```

```
myBool = (5 >= 5)  
print(myBool) # True  
print(10 + myBool) # 11
```

```
myBool = (5 > 5)  
print(myBool) # False  
print(10 + myBool) # 10
```

# Thảo luận

Toán tử gán

Toán tử số học

Toán tử so sánh

Toán tử logic

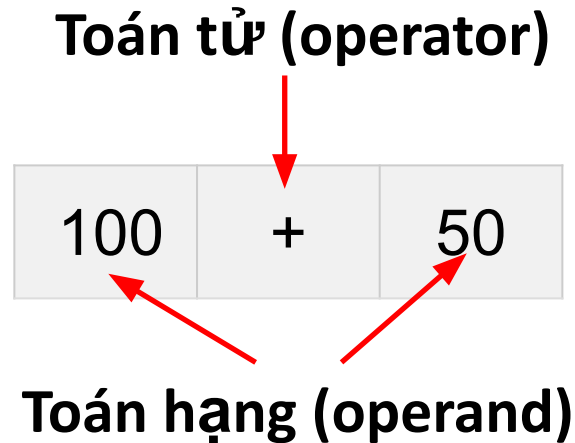
# Toán tử (Operator)

---

- Toán tử là các ký hiệu được sử dụng để thực hiện các thao tác trong các biểu thức và sinh ra kết quả cuối
- Có nhiều loại toán tử khác nhau:
  - Toán tử toán học
  - Toán tử gán
  - Toán tử cộng chuỗi
  - Toán tử so sánh
  - Toán tử logic

# Toán tử toán học (arithmetic)

- Toán tử toán học được sử dụng trong các biểu thức toán học
- Toán tử toán học được sử dụng trên các giá trị số (hoặc là các biến kiểu số)
- Toán tử toán học thông thường có 2 toán hạng



Toán tử	Mô tả
+	Cộng
-	Trừ
*	Nhân
/	Chia
%	Chia lấy phần dư (modulus)

# Toán tử toán học: Ví dụ


- Sử dụng với các giá trị:

**Giá trị số**


$$x = 100 + 500$$

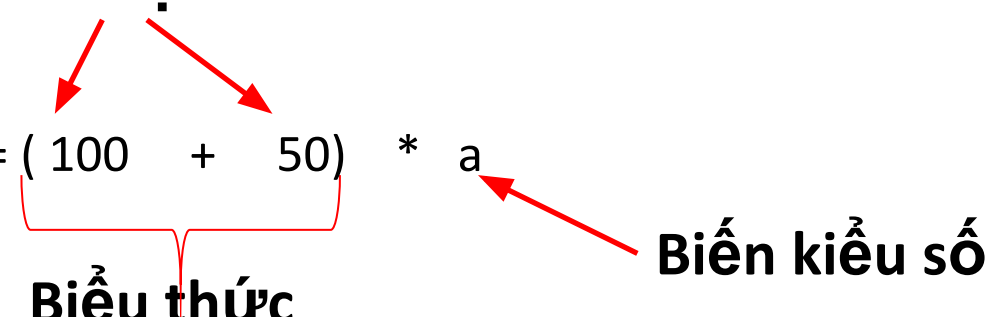
- Sử dụng với các biến:

**Biến kiểu số**


$$x = a + b$$

- Sử dụng với biểu thức:

**Giá trị số**


$$x = (100 + 50) * a$$

**Biểu thức**

**Biến kiểu số**

# Toán tử cộng: Ví dụ

---

$$x = 5$$

$$y = 2$$

$$z = x + y$$

7



# Toán tử trừ: Ví dụ

---

$$x = 5$$

$$y = 2$$

$$z = x - y$$

3





# Toán tử nhân: Ví dụ

---

$$x = 5$$

$$y = 2$$

$$z = x * y$$

10



# Toán tử chia: Ví dụ

---

$$x = 5$$

$$y = 2$$

$$z = x / y$$

2.5



# Toán tử chia lấy số dư: Ví dụ

---

$x = 5$

$y = 2$

$z = x \% y$

1



# Toán tử gán (assignment)

- Toán tử gán được sử dụng để gán giá trị cho một biến
- Toán tử gán có thể sử dụng với tất cả các kiểu dữ liệu
- Ví dụ:

`x = 10`

Operator	Example	Same As
=	<code>x = y</code>	<code>x = y</code>
+=	<code>x += y</code>	<code>x = x + y</code>
-=	<code>x -= y</code>	<code>x = x - y</code>
*=	<code>x *= y</code>	<code>x = x * y</code>
/=	<code>x /= y</code>	<code>x = x / y</code>
%=	<code>x %= y</code>	<code>x = x % y</code>

# Toán tử Cộng bằng: Ví dụ

---

$x = 10$

$x += 5$

# Tương đương với  $x = x + 5$

15



# Toán tử cộng chuỗi (string concatenate)

- Toán tử cộng chuỗi được sử dụng để nối hai chuỗi
- Có thể nối chuỗi với số

txt1 = "John"

txt2 = "Doe"

txt3 = txt1 + " " + txt2

"John Doe"



10 → x = 5 + 5

Error → y = "5" + 5

Error → z = "Hello" + 5

# Toán tử so sánh (comparision)

- Toán tử so sánh được dùng để đánh giá mức độ tương quan giữa các giá trị

#	Mô tả	Ví Dụ
==	So sánh giá trị của các đối số xem có bằng nhau hay không. Nếu bằng nhau thì kết quả trả về sẽ là <b>True</b> và ngược lại sẽ là <b>False</b> .	a == b # False
!=	So sánh giá trị của các đối số xem có khác nhau hay không. Nếu khác nhau thì kết quả trả về sẽ là <b>True</b> và ngược lại sẽ là <b>False</b> .	a != b # True
<	Dấu < đại diện cho phép toán nhỏ hơn, nếu đối số 1 nhỏ hơn đối số 2 thì kết quả sẽ trả về là True và ngược lại sẽ là False.	a < b # True
>	Dấu > đại diện cho phép toán lớn hơn, nếu đối số 1 lớn hơn đối số 2 thì kết quả sẽ trả về là True và ngược lại sẽ là False.	a > b # False
<=	Dấu > đại diện cho phép toán nhỏ hơn hoặc bằng, nếu đối số 1 nhỏ hơn hoặc bằng đối số 2 thì kết quả sẽ trả về là True và ngược lại sẽ là False.	a <= b # True
>=	Dấu > đại diện cho phép toán lớn hơn hoặc bằng, nếu đối số 1 lớn hơn hoặc bằng đối số 2 thì kết quả sẽ trả về là True và ngược lại sẽ là False.	a >= b # False

# Toán tử logic (logical)

- Toán tử logic được dùng trong các biểu thức logic (true/false)

#	Mô tả	Ví dụ
<code>and</code>	Nếu 2 vế của toán tử này đều là True thì kết quả sẽ là True và ngược lại nếu 1 trong 2 vế là False thì kết quả trả về sẽ là False.	<pre>A = True B = False  print (A and B) # False</pre>
<code>or</code>	Nếu 1 trong 2 vế là True thì kết quả trả về sẽ là True và ngược lại nếu cả 2 vế là False thì kết quả trả về sẽ là False.	<pre>print (A or B) # True</pre>
<code>not</code>	Đây là dạng phủ định, nếu biểu thức là True thì nó sẽ trả về là False và ngược lại.	<pre>print (not A) # False</pre>



# Toán tử **and**

---

**Giá trị biến  
a**

**Giá trị biến  
b**

**Kết quả  
(a and b)**

True

True

True

True

False

False

False

True

False

False

False

False

# Toán tử **or**

---

**Giá trị biến  
a**

True

True

False

False

**Giá trị biến  
b**

True

False

True

False

**Kết quả  
(a or b)**

True

True

True

False

# Toán tử **not**

---

**Giá trị biến  
a**

True

False

**Kết quả  
not a**

False

True

# Độ ưu tiên của các toán tử

- Trong một biểu thức có nhiều phép toán thì chúng sẽ lần lượt được đánh giá dựa vào độ ưu tiên
- Có thể sử dụng dấu ngoặc “()” để thay đổi độ ưu tiên của các toán tử
- Các toán tử có cùng độ ưu tiên thì sẽ thực hiện từ trái sang phải

## Operators

()

\*\*

+x, -x, ~x

\*, /, //, %

+, -

<<, >>

&

^

|

==, !=, >, >=, <, <=, is, is not, in, not in

not

and

or

## Meaning

Parentheses

Exponent

Unary plus, Unary minus, Bitwise NOT

Multiplication, Division, Floor division, Modulus

Addition, Subtraction

Bitwise shift operators

Bitwise AND

Bitwise XOR

Bitwise OR

Comparisons, Identity, Membership operators

Logical NOT

Logical AND

Logical OR

# Độ ưu tiên của các toán tử: Ví dụ

x = 5

y = 10

z = (x \* y) < 5 \* 10 and 6 > 3

( 50 ) < 50 and 6 > 3

False and True

False

# Demo

Toán tử gán

Toán tử số học

Toán tử so sánh

Toán tử logic

# Tóm tắt bài học

---

- Biến được sử dụng để đại diện cho vùng nhớ chứa dữ liệu
- Dữ liệu trong vùng nhớ được gọi là giá trị
- Có thể thay đổi giá trị của biến thông qua phép gán
- Hằng số đại diện cho một giá trị cố định
- Kiểu dữ liệu được sử dụng để phân loại dữ liệu
- Các kiểu dữ liệu thông dụng: chuỗi, số, boolean
- Toán tử được sử dụng để thực hiện các thao tác trong biểu thức
- Các loại toán tử thông dụng: Toán học, gán, cộng chuỗi, so sánh, logic
- Các toán tử được thực hiện lần lượt theo độ ưu tiên

# Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo