

TÁC GIẢ: 8 SYNC

CỘNG ĐỒNG



[Kevin Nguyễn](#)



[Nhóm Chia Sẻ Công Nghệ](#)



[Nhóm BlockChain](#)



[Tiktok: 8 Sync](#)



[Youtube: 8 Sync Dev](#)



[Zalo](#)

KHÓA HỌC:



[Fullstack Python](#)



[Fullstack Nextjs](#)



[Fullstack Android-IOs](#)

Tài liệu sẽ được cập nhật định kì và thông báo trong group nên các bạn chú ý nhen .



HƯỚNG DẪN SỬ DỤNG TOÁN TỬ GÁN TRONG PYTHON

Toán tử = (bằng) được định nghĩa là toán tử gán trong Python. Giá trị của biểu thức Python ở bên phải của nó được gán cho một biến duy nhất ở bên trái. Toán tử = như trong lập trình nói chung (và Python cụ thể) không nên bị nhầm lẫn với việc sử

dụng trong Toán học, nơi nó chỉ ra rằng các biểu thức ở cả hai bên của ký hiệu bằng nhau.

VÍ DỤ VỀ TOÁN TỬ GÁN TRONG PYTHON

Xem xét các câu lệnh Python sau –



< 8 Sync Dev />

```
a = 10
b = 5
a = a + b
print(a)
```

Ở lần gọi đầu tiên, ít nhất đối với một người mới vào lập trình nhưng biết toán học, câu lệnh "a=a+b" trông lạ lùng. Làm sao a có thể bằng "a+b"? Tuy nhiên, cần nhấn mạnh rằng ký hiệu = là một toán tử gán ở đây và không được sử dụng để chỉ sự bằng nhau của LHS và RHS.

Bởi vì nó là một phép gán, biểu thức bên phải được tính toán thành 15, giá trị được gán cho a.

Trong câu lệnh "a += b", hai toán tử "+" và "=" có thể được kết hợp trong một toán tử "+=". Nó được gọi là toán tử cộng và gán. Trong một câu lệnh đơn, nó thực hiện phép cộng của hai toán hạng "a" và "b", và kết quả được gán cho toán hạng bên trái, tức là "a".

TOÁN TỬ GÁN TĂNG CƯỜNG TRONG PYTHON

Ngoài toán tử gán đơn giản, Python cung cấp thêm một số toán tử gán khác cho việc sử dụng nâng cao. Chúng được gọi là toán tử tích lũy hoặc tăng cường. Trong chương này, chúng ta sẽ tìm hiểu cách sử dụng các toán tử gán tăng cường được định nghĩa trong Python.

Python có các toán tử gán tăng cường cho tất cả các toán tử số học và so sánh.

Các toán tử gán tăng cường của Python kết hợp phép cộng và gán trong một câu lệnh. Vì Python hỗ trợ phép tính hỗn hợp, hai toán tử hạng có thể thuộc các loại khác nhau. Tuy nhiên, loại của toán tử trái sẽ thay đổi thành toán tử bên phải, nếu nó rộng hơn.

VÍ DỤ

Toán tử += là một toán tử tăng cường. Nó cũng được gọi là toán tử cộng tích lũy, vì nó thêm "b" vào "a" và gán kết quả trở lại một biến.

Dưới đây là các toán tử gán tăng cường trong Python:

- Toán tử Tăng cường Cộng
- Toán tử Tăng cường Trừ
- Toán tử Tăng cường Nhân
- Toán tử Tăng cường Chia
- Toán tử Tăng cường Phần dư
- Toán tử Tăng cường Số mũ
- Toán tử Tăng cường Chia lấy phần nguyên

TOÁN TỬ TĂNG CƯỜNG CỘNG (+=)

Các ví dụ sau sẽ giúp bạn hiểu cách hoạt động của toán tử "+=" –



< 8 Sync Dev />

```
a = 10
b = 5
print("Tăng cường cộng của số nguyên và số nguyên")
a += b # tương đương với a = a + b
print("a =", a, "type(a):", type(a))

a = 10
b = 5.5
print("Tăng cường cộng của số nguyên và số thực")
```

```
a += b # tương đương với a = a + b
print("a =", a, "type(a):", type(a))

a = 10.50
b = 5 + 6j
print("Tăng cường cộng của số thực và số phức")
a += b # tương đương với a = a + b
print("a =", a, "type(a):", type(a))
```

Nó sẽ tạo ra đầu ra sau:



< 8 Sync Dev />

```
Tăng cường cộng của số nguyên và số nguyên
a= 15 type(a): <class 'int'>
Tăng cường cộng của số nguyên và số thực
a= 15.5 type(a): <class 'float'>
Tăng cường cộng của số thực và số phức
a= (15.5+6j) type(a): <class 'complex'>
```

TOÁN TỬ TĂNG CƯỜNG TRỪ' (-=)

Sử dụng ký hiệu -= để thực hiện phép trừ và gán trong một câu lệnh. Câu lệnh "a -= b" thực hiện phép gán "a = a - b". Các toán hạng có thể là bất kỳ kiểu số nào. Python thực hiện chuyển đổi kiểu ngầm định trên đối tượng có kích thước nhỏ hơn.



< 8 Sync Dev />

```
a = 10
b = 5
print("Tăng cường trừ của số nguyên và số nguyên")
a -= b # tương đương với a = a - b
print("a =", a, "type(a):", type(a))
```



```

a = 10
b = 5.5
print("Tăng cường trừ của số nguyên và số thực")
a -= b # tương đương với a = a - b
print("a =", a, "type(a):", type(a))

a = 10.50
b = 5 + 6j
print("Tăng cường trừ của số thực và số phức")
a -= b # tương đương với a = a - b
print("a =", a, "type(a):", type(a))

```

Kết quả sẽ là:

```

< 8 Sync Dev />

Tăng cường trừ của số nguyên và số nguyên
a= 5 type(a): <class 'int'>
Tăng cường trừ của số nguyên và số thực
a= 4.5 type(a): <class 'float'>
Tăng cường trừ của số thực và số phức
a= (5.5-6j) type(a): <class 'complex'>

```

TOÁN TỬ TĂNG CƯỜNG NHÂN (*=)

Toán tử "*" hoạt động theo cùng một nguyên tắc. "a *= b" thực hiện phép nhân và gán, và tương đương với "a = a * b". Trong trường hợp nhân tăng cường của hai số phức, quy tắc nhân như đã thảo luận trong chương trước được áp dụng.

```

< 8 Sync Dev />

a = 10
b = 5
print("Tăng cường nhân của số nguyên và số nguyên")

```

```

a *= b # tương đương với a = a * b
print("a =", a, "type(a):", type(a))

a = 10
b = 5.5
print("Tăng cường nhân của số nguyên và số thực")
a *= b # tương đương với a = a * b
print("a =", a, "type(a):", type(a))

a = 6 + 4j
b = 3 + 2j
print("Tăng cường nhân của số phức và số phức")
a *= b # tương đương với a = a * b
print("a =", a, "type(a):", type(a))

```

Kết quả sẽ là:



< 8 Sync Dev />

```

Tăng cường nhân của số nguyên và số nguyên
a= 50 type(a): <class 'int'>
Tăng cường nhân của số nguyên và số thực
a= 55.0 type(a): <class 'float'>
Tăng cường nhân của số phức và số phức
a= (10+24j) type(a): <class 'complex'>

```

TOÁN TỬ TĂNG CƯỜNG CHIA (/=)

Ký hiệu kết hợp `/=` hoạt động như một toán tử chia và gán, vì vậy `a /= b` tương đương với `a = a / b`. Phép chia của toán hạng `int` hoặc `float` là `float`. Chia hai số phức trả về một số phức.



< 8 Sync Dev />

```
a = 10
b = 5
print("Tăng cường chia của số nguyên và số nguyên")
a /= b # tương đương với a = a / b
print("a =", a, "type(a):", type(a))
```

```
a = 10
b = 5.5
print("Tăng cường chia của số nguyên và số thực")
a /= b # tương đương với a = a / b
print("a =", a, "type(a):", type(a))
```

```
a = 6 + 4j
b = 3 + 2j
print("Tăng cường chia của số phức và số phức")
a /= b # tương đương với a = a / b
print("a =", a, "type(a):", type(a))
```

Kết quả sẽ là:



< 8 Sync Dev />

```
Tăng cường chia của số nguyên và số nguyên
a= 2.0 type(a): <class 'float'>
Tăng cường chia của số nguyên và số thực
a= 1.8181818181818181 type(a): <class 'float'>
Tăng cường chia của số phức và số phức
a= (2+0j) type(a): <class 'complex'>
```

TOÁN TỬ TĂNG CƯỜNG PHẦN DƯ (%=)

Để thực hiện phép chia lấy phần dư và gán trong một câu lệnh, sử dụng toán tử "%=". Như toán tử phần dư, phiên bản tăng cường của nó cũng không được hỗ trợ cho số phức.

< 8 Sync Dev />

```
a = 10
b = 5
print("Tăng cường phần dư của số nguyên và số nguyên")
a %= b # tương đương với a = a % b
print("a =", a, "type(a):", type(a))

a = 10
b = 5.5
print("Tăng cường phần dư của số nguyên và số thực")
a %= b # tương đương với a = a % b
print("a =", a, "type(a):", type(a))
```

Kết quả sẽ là:

< 8 Sync Dev />

```
Tăng cường phần dư của số nguyên và số nguyên
a= 0 type(a): <class 'int'>
Tăng cường phần dư của số nguyên và số thực
a= 4.5 type(a): <class 'float'>
```

TOÁN TỬ TĂNG CƯỜNG SỐ MŨ (**=)

Toán tử "**=" dẫn đến việc tính toán "a" lũy thừa "b", và gán giá trị trở lại "a". Dưới đây là một số ví dụ –

< 8 Sync Dev />

```
a = 10
b = 5
print("Tăng cường số mũ với số nguyên và số nguyên")
```



```
a **= b # tương đương với a = a ** b
print("a =", a, "type(a):", type(a))

a = 10
b = 5.5
print("Tăng cường số mũ với số nguyên và số thực")
a **= b # tương đương với a = a ** b
print("a =", a, "type(a):", type(a))

a = 6 + 4j
b = 3 + 2j
print("Tăng cường số mũ với số phức và số phức")
a **= b # tương đương với a = a ** b
print("a =", a, "type(a):", type(a))
```

Kết quả sẽ là:

```
< 8 Sync Dev />

Tăng cường số mũ với số nguyên và số nguyên
a= 100000 type(a): <class 'int'>
Tăng cường số mũ với số nguyên và số thực
a= 316227.7660168379 type(a): <class 'float'>
Tăng cường số mũ với số phức và số phức
a= (97.52306038414744-62.22529992036203j) type(a): <class
'complex'>
```

TOÁN TỬ TĂNG CƯỜNG CHIA LẤY PHẦN NGUYÊN (//=)

Đối với phép chia lấy phần nguyên và gán trong một câu lệnh, sử dụng toán tử "//=": "a //= b" tương đương với "a = a // b". Toán tử này không thể sử dụng với số phức.



< 8 Sync Dev />

```
a = 10
b = 5
print("Tăng cường chia lấy phần nguyên với số nguyên và số nguyên")
a //= b # tương đương với a = a // b
print("a =", a, "type(a):", type(a))

a = 10
b = 5.5
print("Tăng cường chia lấy phần nguyên với số nguyên và số thực")
a //= b # tương đương với a = a // b
print("a =", a, "type(a):", type(a))
```

Kết quả sẽ là:



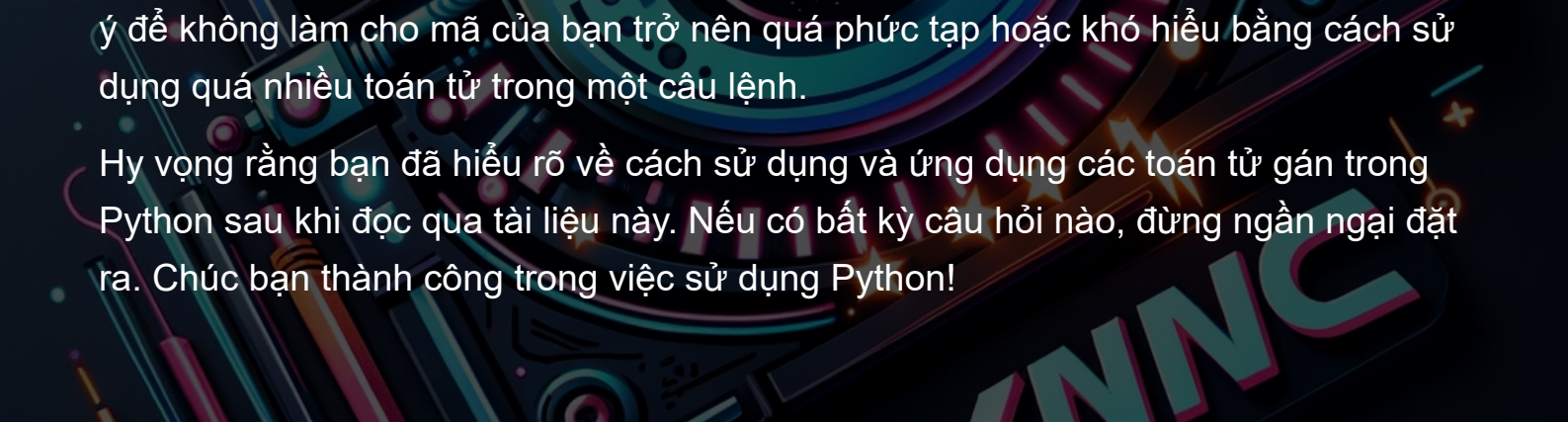
< 8 Sync Dev />

```
Tăng cường chia lấy phần nguyên với số nguyên và số nguyên
a= 2 type(a): <class 'int'>
Tăng cường chia lấy phần nguyên với số nguyên và số thực
a= 1.0 type(a): <class 'float'>
```

TỔNG KẾT NÈ

Trên đây là các toán tử gán trong Python cùng với các toán tử tăng cường. Chúng giúp rút ngắn và làm cho mã nguồn trở nên rõ ràng hơn trong các tình huống mà bạn cần thực hiện một phép tính và gán giá trị lại cho biến.

Khi sử dụng các toán tử này, hãy nhớ rằng chúng có thể giúp làm cho mã nguồn của bạn trở nên dễ đọc hơn và giảm thiểu việc lặp lại mã. Tuy nhiên, cũng cần chú



ý để không làm cho mã của bạn trở nên quá phức tạp hoặc khó hiểu bằng cách sử dụng quá nhiều toán tử trong một câu lệnh.

Hy vọng rằng bạn đã hiểu rõ về cách sử dụng và ứng dụng các toán tử gán trong Python sau khi đọc qua tài liệu này. Nếu có bất kỳ câu hỏi nào, đừng ngần ngại đặt ra. Chúc bạn thành công trong việc sử dụng Python!