Operators

🎯 Session Objectives

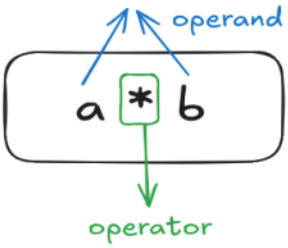
- 🔧 Understand what operators are and why they are used
- 🧮 Explore different types of operators in Python
- 🔢 Learn about operator precedence and order of execution
- ⚠️ Understand constraints in programming

What are Operators?

Operators are tools in programming used to perform actions like arithmetic, comparisons, assignments, logical evaluations, etc.

Types of Operators in Python:

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Membership Operators
7. Identity Operators



Arithmetic Operators:

- '+' Addition
- '-' Subtraction
- '*' Multiplication
- '/' Division (floating)
- '%' Modulus (remainder)
- '**' Exponentiation
- '//' Floor Division

```
x = 11
y = 7
z = 15
print(x+y) # 18 'One 8' [Virat Kohli]
print(x-y) # 4
print(x*y) # 77 ['Shubman Gill']
print(x/y) # 1.57....
print(x%y) # 4
print(z%y) # 1
print(z%x) # 4
print(x ** 2) # 121
print(x // 2) # 5
```

```
18
4
77
1.5714285714285714
4
1
4
121
5
```

```
7/11\1
   7
-------
   4 -> remainder [11 % 7] -> 4
```

11 ^ 2 != 11 ** 2 => 121

XOR

```
print(x ** y)
```

```
19487171
```

```
# You can't do addition with 2 different data type
x = '10' + 11 # TypeError: can only concatenate str (not "int") to str
print(x)
```

Comparison Operators (Return Boolean Result):

- '==' Equal to
- '!=' Not Equal to
- '>' Greater Than
- '>=' Greater Than or Equal to
- '<' Less Than
- '<=' Less Than or Equal to

```
x = 11
y = 7
z = 15
print(x==y) # False
print(x!=y) # True
print(z!=y) # True
print(z!=15) # False
print(x>y) # True
print(x>=y) # True
print(z<=y) # False
print(z<x)  # False
print(x != 2) # True
print(x >= 2) # True
```

```
False
True
True
False
True
True
False
False
True
True
```

Logical Operators:

'and'

Cond1	Cond2	Result
T	T	T
T	F	F
F	T	F
F	F	F

'or'

Cond1	Cond2	Result
T	T	T
T	F	T
F	T	T
F	F	F

'not'

Cond1	Result
T	F
F	T

Logical Operators:

- and : 'Return True' if both the conditions are True
- or : 'Return False' if both the conditions are False
- not : Return the opposite.

'not' > 'and' > 'or'

```
x = 11
y = 7
z = 15
print((x==y)and(z!=y)or(y>x)) # False
print((x!=y)or(z!=y)or(y>x)) # True
print((x>=y)and(z==y)and(z>x)) # False
print(not(z!=15)) # True
print(not((x<=y)and(z>=y)or(y!=x))) # False
```

```
False
True
False
True
False
```

```
print(not(x<=y)and(z>=y)or(y!=x)) # True
```

```
True
```

```
print(not(x<=y)or(z<=y)and(y!=x)) # (T or F and T) # (T or F) # T
```

```
True
```

Bitwise Operators:

- 'AND' = '&' -> 'Both bit with value 1 return 1 else 0'
- 'OR' = '|' -> 'Any bit with value 1 return 1 else 0'
- 'XOR' = '^' -> 'Alternative Bits return 1 else same bits returns 0'
- 'NOT' = '~' -> '2s Complement to check the sign and evaluate'
- 'Left Shift' = '<<' -> 'Shift bits to the left'
- 'Right Shift' = '>>' -> 'Shift bits to the right'

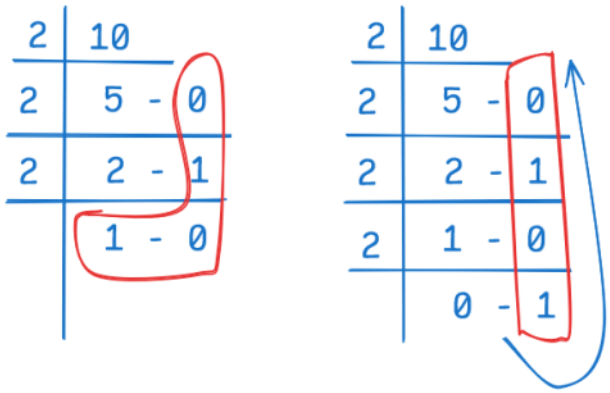
'and'

bit1	bit2	Result
1	1	1
1	0	0
0	1	0
0	0	0

'or'

bit1	bit2	Result
1	1	1
1	0	1
0	1	1
0	0	0

10 decimal -> 1010



1010

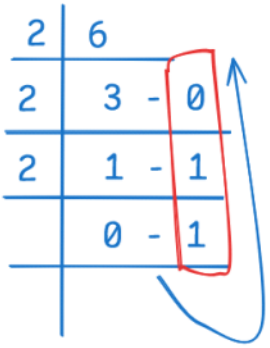
= (1 * 2^3) + (0 * 2^2) + (1 * 2^1) + (0 * 2^0)
= 8 + 0 + 2 + 0 = 10

2^3 2^2 2^1 2^0

AND - '&'

```
  1010
& 0110
------
  0010
```

10 & 6 → 2




```
x = 10
y = 6
print(x & y) # 2
```

```
2
```

OR - '|'

```
  1010
| 0110
------
  1110
```

10 | 6 → 14

```
x = 10
y = 6
print(x | y) # 14
```

```
14
```

= (1 * 2^3) + (1 * 2^2) + (1 * 2^1) + (0 * 2^0)
= 8 + 4 + 2 + 0 = 14

'XOR'

bit1	bit2	Result
1	1	0
1	0	1
0	1	1
0	0	0

XOR - '^'

```
  1010
^ 0110
------
  1100
```

10 ^ 6

12

```
x = 10
y = 6
print(x ^ y) # 12
```

```
12
```

= (1 * 2^3) + (1 * 2^2) + (0 * 2^1) + (0 * 2^0)
= 8 + 4 + 0 + 0 = 12

NOT '~' ~10

10 -> 00001010 (magnitude)

1's complement -> 11110101 (Flip the bits)

+1

2's complement

leftmost bit -> Direction
1 -> negative
0 -> positive

dir -ve

```
  00001010
+         1
----------
  00001011
```

-11

```
x = 10
print(~x)
```

```
-11
```

```
x = 7
print(~x)
```

```
-8
```

```
x = 17
print(~x)
```

```
-18
```

~X = -(X+1)

```
x = -17
print(~x)
```

```
16
```

= -(-17+1)
= -(-16)
= 16

Left Shift <<

```
x = 10
print(x<<2)
```

```
40
```

x = 10 -> 1010 ← x << 2

00

x * 2^shift

10 * 2^2 = 10 * 4 = 40

101000 →

=(1 * 2^5) + (0 * 2^4)+(1 * 2^3) + (0 * 2^2) + (0 * 2^1) + (0 * 2^0)
= 32 + 0 + 8 + 0 + 0 + 0 -> 40

Right Shift >>

```
x = 10
print(x>>2)
```

```
2
```

x = 10 -> 1010 x >> 2

out

00

x // 2^shift

10 // 2^2 = 10 * 4 = 2

0010 →

=(0 * 2^3) + (0 * 2^2) + (1 * 2^1) + (0 * 2^0)
= 0 + 0 + 2 + 0 = 2