5️⃣

Day 5: Operators (Membership & Bitwise Operators)

Operators

Types of operators:

- Arithmetic operators (+, -, *, /, %, **, //)
- Assignment operators(=, +=, -=, *=, /=)
- Comparison operators(==, ! =, >, <, > =, < =)
- Logical operators(and, or, not)
- Identity operators
- Membership operators
- Bitwise operators

Membership Operators

Membership operators are used to test if a sequence is presented in an object.

```
# in (Returns True if a sequence with the specified value is pre
x = [1, 2]
print(2 in x)

# not in (Returns True if a sequence with the specified value is
x = ["a", "b"]
print("c" not in x)
```

Binary Numbers

Binary numbers are numbers expressed in base-2, using only two digits: **0** and **1**. Binary numbers are the numbers which computer understands.

0 - 0000

1 - 0001

2 - 0010

3 - 0011

4 - 0100

5 - 0101

6 - 0110

7 - 0111

8 - 1000

9 - 1001

10 - 1010

11 - 1011

12 - 1100

Bitwise Operators

Bitwise operators are used to compare (binary) numbers.

Bitwise operators in Python operate directly on the binary representation of integers and are commonly used in scenarios requiring low-level, performance-sensitive operations.

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off

>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

To divide any number by 2, convert to binary and use right shift.

To multiply any number by 2, convert to binary and use left shift.

```
# &
print(6 & 3)
print(bin(6)) # 110
print(bin(3)) # 011
# output: 2 # 010

# |
print (6 | 3)
print(bin(6)) # 110
print(bin(3)) # 011
# output: 7 # 111

# ^
print (6 ^ 3)
print(bin(6)) # 110
print(bin(3)) # 011
# output: 5 # 101

# ~
print(~3) # 0000000000000011
# output: -4 # 1111111111111100
# By default it is 16 bits

# << (also used to multiply by 2, 4, 8 etc.)
x = 4
print(x<<1) # 100
# output: 8 # 1000


# >> (also used to divide by 2, 4, 8 etc.)
```

```
x = 4
print(x>>1) # 100
# output: 2 # 010
```