

logical operators with example

Here is a complete explanation of Logical Operators in Python along with examples and truth table 

 **Logical Operators in Python**

Logical operators are used to combine **conditional statements** (conditions that return True or False).

Python has **three logical operators**:

Operator	Description	Example
and	True only if both conditions are True	<code>a > 10 and a < 20</code>
or	True if at least one condition is True	<code>a > 10 or a < 5</code>
not	Reverses the result (True → False, False → True)	<code>not(a > 10)</code>

 **Example in Python**

python  Copy code

```

a = 15
b = 5

# AND operator
print(a > 10 and b < 10)    # True (both conditions True)

# OR operator
print(a < 10 or b < 10)     # True (one condition True)

# NOT operator
print(not(a > 10))         # False (because a > 10 is True)

```

 **Truth Table of Logical Operators** **AND Operator (and)**

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

 **OR Operator (or)**

A	B	A or B
True	True	True

A	B	A or B
True	False	True
False	True	True
False	False	False

✓ NOT Operator (not)

A	not A
True	False
False	True

Real Life Example

python

 Copy code

```
age = 22
have_id = True

if age >= 18 and have_id:
    print("You are allowed to vote.")
else:
    print("You cannot vote.")
```

Interview Tips

- ◆ and → Both conditions must be true
 - ◆ or → At least one condition must be true
 - ◆ not → Reverses the boolean value

Would you like **practice questions** with solutions? 😊 🤞



Bitwise example with all concept

Here is a complete explanation of Bitwise Operators in Python with all key concepts + examples + interview notes 

⚡ Bitwise Operators in Python

Bitwise operators work on bits (0 and 1) of integers.

They perform operations at the binary (bit) level.

Operator	Symbol	Meaning
AND	&	Returns 1 if both bits are 1

Operator	Symbol	Meaning
OR		Returns 1 if any bit is 1
XOR	^	Returns 1 if bits are different
NOT	~	Flips bits (gives negative result in Python)
Left Shift	<<	Shifts bits left (multiply by 2)
Right Shift	>>	Shifts bits right (divide by 2)

BitFields AND & Example

python

Copy code

```
a = 5      # 5 = 0101
b = 3      # 3 = 0011

print(a & b)  # 0101 & 0011 = 0001 → 1
```

Output: 1

BitFields OR | Example

python

Copy code

```
a = 5    # 0101
b = 3    # 0011

print(a | b)  # 0111 → 7
```

Output: 7

BitFields XOR ^ Example

python

Copy code

```
a = 5    # 0101
b = 3    # 0011

print(a ^ b)  # 0110 → 6
```

Output: 6

XOR gives 1 when **bits are unequal**.

BitFields NOT ~ Example

python

Copy code

```
a = 5    # 0101
print(~a)
```

Output: -6

📌 NOT formula:

Copy code

```
~x = -(x + 1)
```

So: $\sim 5 = -(5 + 1) = -6$

▶ Left Shift << Example

python

Copy code

```
a = 5          # 5 = 0101
print(a << 1) # 1010 = 10
print(a << 2) # 10100 = 20
```

📌 Left shift = Multiply by 2^n

So:

- $5 \ll 1 = 5 * 2 = 10$
 - $5 \ll 2 = 5 * 4 = 20$
-

◀ Right Shift >> Example

python

Copy code

```
a = 20          # 20 = 10100
print(a >> 1) # 1010 = 10
print(a >> 2) # 101 = 5
```

📌 Right shift = Divide by 2^n

So:

- $20 \gg 1 = 20 / 2 = 10$
 - $20 \gg 2 = 20 / 4 = 5$
-

☰ Bitwise Operator Truth Table

A	B	A & B	A B	A ^ B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

💡 Interview Questions Based on Bitwise Operators

📌 Q1: Swap two numbers without using third variable

python

Copy code

```
a = 10
b = 20

a = a ^ b
b = a ^ b
a = a ^ b

print(a, b) # 20 10
```

❖ Q2: Check if a number is odd or even using bitwise

python

[Copy code](#)

num = 7

```
if num & 1:
    print("Odd")
else:
    print("Even")
```

❖ Q3: Multiply a number by 8 using bitwise

bash

[Copy code](#)

num << 3

👉 Want practice questions with answers?

Just reply "Yes, send practice questions" 😊

[🔗](#) [✍](#) [💡](#) [↑](#) [🕒](#) [...](#)

m = 10 1010 010100 101000
n = 4 0100
128 64 32 16 8 4 2 1 explain

Here is the full explanation of your example with **binary conversion + shifting + bit positions** 👇

❖ Given:

ini

[Copy code](#)

```
m = 10 → 1010
n = 4 → 0100
```

Before understanding this, we must know **how binary bits represent numbers**.

👉 **Binary Position Table**

Bit Position	128	64	32	16	8	4	2
Power of 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1

📌 Every binary number is made using these values.

12 34 Convert m = 10 to Binary

Find numbers that add up to 10 using the above table:

ini

Copy code

$$10 = 8 + 2$$

8	4	2	1
1	0	1	0

So, 10 = 1010 in binary ✓

ini

Copy code

$$m = 10 \rightarrow 1010$$

12 34 Convert n = 4 to Binary

ini

Copy code

$$4 = 4$$

8	4	2	1
0	1	0	0

So, 4 = 0100 in binary ✓

ini

Copy code

$$n = 4 \rightarrow 0100$$

📌 Understanding Your Question:

You wrote:

ini

Copy code

$$m = 10 \quad 1010 \quad 010100 \quad 101000$$

This is showing left shift operations:

Expression	Meaning	Result Binary	Value
1010	original	1010	10
1010 << 1	shift left by 1 bit	010100	20

Expression	Meaning	Result Binary	Value
1010 << 2	shift left by 2 bits	101000	40

⌚ Left Shift Rule

bash

Copy code

```
x << n = x × (2^n)
```

So:

- $10 \ll 1 = 10 \times 2 = 20$
- $10 \ll 2 = 10 \times 4 = 40$

🌐 Example in Python

python

Copy code

```
m = 10

print(m << 1) # 20
print(m << 2) # 40
```

✓ Final Explanation Summary

Concept	Example
Binary values	128 64 32 16 8 4 2 1
10 in binary	1010
4 in binary	0100
Left shift 1	$1010 \ll 1 = 010100 = 20$
Left shift 2	$1010 \ll 2 = 101000 = 40$

🎯 Want more?

Do you want right shift examples + negative number bit representation also?

Reply Yes 😊

🔗 🤝 🔍 ⬆️ ⬇️ ⏪ ...

print(m & n) 0000

You wrote:

SCSS

Copy code

```
print(m & n) → 0000
```

Let's explain why the output becomes 0000.

★ Given:

ini

 Copy code

```
m = 10 → 1010
n = 4  → 0100
```

💡 Bitwise AND &

The AND operator compares each bit:

Bit	Rule
1 & 1 → 1	
1 & 0 → 0	
0 & 1 → 0	
0 & 0 → 0	

🔍 Calculate m & n

yaml

 Copy code

```
1010  (10)
& 0100  (4)
-----
0000
```

➡ Why?

There is no position where both bits are 1 at the same time.

💡 Final Output

python

 Copy code

```
print(m & n) # 0
```

✓ Binary result = 0000

✓ Decimal value = 0

🎁 Small Tip (Interview)

If two numbers do not share any common bit of 1, their AND will always be 0.

Would you like OR | and XOR ^ explanation with the same numbers (10 & 4)? 😊

...

print(m | n) (01,10,11)=1...00=0 1110

Let's clearly explain your statement:

SCSS

Copy code

`print(m | n)`

We know:

ini

Copy code

`m = 10 → 1010`
`n = 4 → 0100`

💡 Bitwise OR | Rules

Bits Compared

Result

1 OR 1 → 1

1 OR 0 → 1

0 OR 1 → 1

0 OR 0 → 0

📌 If any one bit is 1, result becomes 1

🔍 Operation

yaml

Copy code

```
1010 (10)
| 0100 (4)
-----
1110
```

🧠 Explanation of Each Bit:

m	n	$m n$
1	0	1
0	1	1
1	0	1
0	0	0

👉 Final = 1110

🎯 Convert 1110 to Decimal

ini

 Copy code

```
1110 = 8 + 4 + 2 + 0 = 14
```

✓ Final Output

python

 Copy code

```
print(m | n) # 14
```

✳️ Binary = 1110

✳️ Decimal = 14

⭐ Tip (Very Important)

Bitwise OR produces 1 if at least one bit is 1.

Would you like to see XOR ^ example for 10 and 4 next? 😊

...

```
print(m ^ n) (01,10)=1...(00,11)=0 1110
```

Let's explain your line:

bash

 Copy code

```
print(m ^ n)
(01,10)=1 ... (00,11)=0      → 1110
```

⌚ Bitwise XOR ^ Rules

Bits Compared	Result
0 ^ 0 → 0	
1 ^ 1 → 0	
0 ^ 1 → 1	
1 ^ 0 → 1	

⭐ XOR gives 1 only when bits are different.

⭐ Given:

ini

 Copy code

$m = 10 \rightarrow 1010$
 $n = 4 \rightarrow 0100$

⌚ Operation: $m \wedge n$

yaml

Copy code

$$\begin{array}{r} 1010 \quad (10) \\ \wedge \quad 0100 \quad (4) \\ \hline 1110 \end{array}$$

🧠 Bit-by-bit Explanation:

m	n	$m \wedge n$	Explanation
1	0	1	Bits different
0	1	1	Bits different
1	0	1	Bits different
0	0	0	Bits same

👉 Final result = 1110

🎯 Convert 1110 to Decimal

ini

Copy code

$$1110 = 8 + 4 + 2 + 0 = 14$$

✓ Final Output

python

Copy code

```
print(m ^ n) # 14
```

- 📌 Binary = 1110
- 📌 Decimal = 14

💡 Quick Tip (Interview)

- ◆ XOR (^) is used when you want to detect differences between bits.
- ◆ Used in encryption, swapping numbers, error detection.

🎁 Want an example of XOR used to swap numbers without third variable?

Reply Yes 😊



```
print(m >> 2) 1010->>> 0101 ->>> 0010
```

Let's explain your line:

yaml

 Copy code

```
print(m >> 2)
1010 → 0101 → 0010
```

⭐ Given:

ini

 Copy code

```
m = 10
Binary = 1010
```

◀ Bitwise Right Shift >>

Right shift moves bits to the right.

Each shift divides the number by 2 (integer division).

⌚ Step-by-Step Shifting

◆ Step 1: m >> 1

SCSS

 Copy code

```
1010 (10)
→ 0101 (5)
```

⭐ 10 → 5 (divided by 2)

◆ Step 2: m >> 2

SCSS

 Copy code

```
0101 (5)
→ 0010 (2)
```

⭐ 5 → 2 (again divided by 2)

✓ Final Output

python

 Copy code

```
print(m >> 2) # 2
```

🧠 Binary Explanation

yaml

 Copy code

```
m = 10 = 1010
```

```
1010 >> 2 = 0010 (2)
```

💡 Mathematical Shortcut

nginx

Copy code

```
m >> n = m / (2^n)
```

So:

Copy code

```
10 >> 2 = 10 / (2^2) = 10 / 4 = 2
```

🎉 Final Summary Table

Operation	Binary Result	Decimal
10 >> 1	0101	5
10 >> 2	0010	2

Would you like to learn how **right shift works with negative numbers** (-10 >> 2)?

Reply Yes 😊

...

```
print(m << 2) 0000 1010
                0001 0100
                0010 1000
```

Let's explain your example:

yaml

Copy code

```
print(m << 2)
0000 1010
0001 0100
0010 1000
```

⭐ Given:

ini

Copy code

```
m = 10
Binary = 0000 1010
```

▶ Bitwise Left Shift \ll

Left shift moves bits to the left.
Each shift multiplies the number by 2.

⌚ Step-by-Step Shifting

- ◆ Step 1: $m \ll 1$

yaml

 Copy code

```
0000 1010 (10)
→ 0001 0100 (20)
```

✓ Shifting left once = multiply by 2

 $10 \times 2 = 20$

- ◆ Step 2: $m \ll 2$

yaml

 Copy code

```
0001 0100 (20)
→ 0010 1000 (40)
```

✓ Shifting left twice = multiply by 4

 $10 \times 4 = 40$

✓ Final Output

python

 Copy code

```
print(m << 2) # 40
```

🗣 Binary Final:

yaml

 Copy code
 $0010 1000 = 40$

💡 Mathematical Rule

bash

 Copy code
 $m \ll n = m \times (2^n)$

So:

bash

 Copy code
 $10 \ll 2 = 10 \times (2^2) = 10 \times 4 = 40$

⌚ Summary Table



Operation	Binary	Decimal
10	0000 1010	10