

# Variables, operators and conditional statements:

You don't have to *declare* variables, as in many other programming languages. You can create a new variable in Python by assigning it a value.

```
>>> x = 3          # creates x, assigns int
>>> print(x)

>>> x = "abc"      # re-assigns x a string
>>> print(x)

>>> x = 3.14       # re-assigns x a float
>>> print(x)

>>> y = 6          # creates y, assigns int
>>> x * y          # uses x and y
```

www.goclasses.in

Question:

```
var_A = 11
var_B = var_A
var_A = 42
```

After this code is executed, the value of var\_B is:

42       Var-A       Var-B

↳ 11 is answer

<https://www.cs.toronto.edu/~guerzhoy/180/midterm/mt2010/solution.pdf>

27:53 -35:20

## Variable Names



- Variable names must:
  - Start with a letter or an underscore (`_`)
  - Contain only letters, digits, or underscores
  - Cannot be a "built in" command in Python (e.g., `for`)
- Variable names are case sensitive
  - `Hello` is not the name as `hello`

www.goclasses.in

28:17 -34:56

## Arithmetic Operators

- Operators

<code>+</code>	"addition"	Ex.: <code>num3 = num1 + num2</code>	<u>5</u>	<u>2</u>	<u>num3</u>	7
<code>-</code>	"subtraction"	Ex.: <code>num3 = num1 - num2</code>				3
<code>*</code>	"multiplication"	Ex.: <code>num3 = num1 * num2</code>				10
<code>/</code>	"division"	Ex.: <code>num3 = num1 / num2</code>				2.5
<code>//</code>	"integer division"	Ex.: <code>num3 = num1 // num2</code>				2
<code>%</code>	"remainder"	Ex.: <code>num3 = num1 % num2</code>				1
<code>**</code>	"exponentiation"	Ex.: <code>num3 = num1 ** num2</code>				25
<code>-</code>	"negation" (unary)	Ex.: <code>num3 = -num1</code>				-5



www.goclasses.in

**GO CLASSES**

In [9]: 5/2  
Out[9]: 2.5  
In [10]: 5//2  
Out[10]: 2

2)  $\frac{5}{2}$



- Precedence of operator (in order)
  - ( ) "parentheses" highest
  - \*\* "exponentiation"
  - "negation" (unary)
  - \*, /, //, %
  - +, - lowest
- Operators in same precedence category are evaluated left to right
  - Similar to rules of evaluating expressions in algebra

[www.goclasses.in](http://www.goclasses.in)



## Precedence Example

$$x = 1 + 3 * 5 / 2$$

15  
7.5  
8.5



Host

SES

x 8.5

www.goclasses.in

48:22 - 48:51

## Precedence of Arithmetic Operators:

- High priority \* , /, //, %
- Low priority +, -
- The basic evaluation procedure includes two left-to-right passes through the expression.
- During the first pass, the high priority operators are applied as they are encountered.
- During the second pass, the low priority operators are applied as they are encountered.



Host

www.goclasses.in

A screenshot of a video player interface for 'GO CLASSES'. In the top right corner, there is a video feed of a man wearing glasses and a dark t-shirt, labeled 'Host'. The main area shows a handwritten mathematical calculation:

$$J = \underline{2 * 3} / 4 + 2 / 5 + 8 / 5$$

↓

$$= 6 + 0.4 + 1.6$$

+ -

$$= 3.5$$

The video player has a progress bar at the bottom with '52:34' on the left and '-10:39' on the right. The URL 'www.goclasses.in' is visible above the progress bar.

A screenshot of a video player interface for 'GO CLASSES'. In the top right corner, there is a video feed of a man wearing glasses and a dark t-shirt, labeled 'Host'. The main area shows a slide titled 'OPERATORS ON ints and floats' with the following content:

- $i+j$  → the **sum**
- $i-j$  → the **difference**
- $i*j$  → the **product**
- $i/j$  → **division**
- $i \% j$  → the **remainder** when  $i$  is divided by  $j$
- $i^{**}j$  →  $i$  to the **power** of  $j$

Below the list, there is a link: [https://ocw.mit.edu/courses/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/e921a690079369751bcce3e34da6c6ee/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/e921a690079369751bcce3e34da6c6ee/MIT6_0001F16_Lec1.pdf)

The video player has a progress bar at the bottom with '56:14' on the left and '-06:59' on the right. The URL 'www.goclasses.in' is visible above the progress bar.

**Quiz:**

- You want to calculate:  
$$\frac{20}{5 \times 2} = \frac{20}{10} = 2$$
- Which one can you **not** use?
  - a)  $20 / 5 / 2$  Not use
  - b)  $20 / 5 * 2$  Not use
  - c)  $20 / (5 * 2)$  use this

www.goclasses.in  
01:40 -> -1:51:05

**Quiz**

- What is  $-2^{**2}$  in Python?
  - a) 4 i.e.  $(-2)^{**2}$
  - b) -4 i.e.  $-(2^{**2})$

**Operator Precedence Diagram:**

- Higher precedence:**  $( )$ ,  $**$
- Lower precedence:**  $+ -$  (unary: sign),  $* / \% //$ ,  $+ -$  (binary)

$- (2^{**2})$

www.goclasses.in

Unary priority <  $**$

GO CLASSES

Quiz:

Python Expression	Result	Type of Result
$9 / 3$	3.0	float
$9 // 3$	3	int

www.goclasses.in

Host

03:22 - 1:49:23

GO CLASSES

$3 + 5 \equiv 8$  or  $8.0$

$10 + 2 \equiv 12$

$10 / 2 \equiv 5$  or  $5.0$

Host

05:05 - 1:47:40

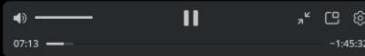
## Implicit Type Conversion

```
num1 = 5
num2 = 2
num3 = 1.9
```

5 + 1.9 → 6.9 float  
num1 + num3 → 7 int ≡

0000101  
float 6.0

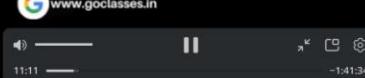
<https://web.stanford.edu/class/archive/cs/cs106a/cs106a.1226/lectures/05-arithmetic-expressions/5-Expressions.pdf>



## Explicit Type Conversion

```
num1 = 5
num2 = 2
num3 = 1.9
```

- Use **float(value)** to create new real-valued number
  - `float(num1) = 5.0 (float)`  
Note that num1 is not changed. We created a new value.  
`num1 + float(num2) = 7.0 (float)`  
`num1 + num2 = 7 (int)`
- Use **int(value)** to create a new integer-valued number (truncating anything after decimal)
  - `int(num3) = 1 (int)`  
`int(-2.7) = -2 (int)`



**GO CLASSES**

```
In [7]: int(True)  
Out[7]: 1  
  
In [8]: bool(1)  
Out[8]: True  
  
In [9]: int(False)  
Out[9]: 0  
  
In [10]: bool(0)  
Out[10]: False  
  
bool(100) : True  
bool(-1) : True  
bool(0.0) : False
```

**GO CLASSES**

bool(0) → for integer 0, bool value is false  
False

bool(0.0) → for float 0.0, bool value is false  
False

other than these 2 any int/float is TRUE

Host

23:08 - 1:29:37

GO CLASSES

Internally, Python uses 0 to represent False and 1 to represent True. You can convert from Boolean to int using the int function and from int to Boolean using the bool function.



Host

```
>>> b1 = (-3 < 3)
>>> print(b1)           → True
>>> int(b1)             → 1
>>> bool(1)              → True
>>> bool(0)              → False
>>> bool(4)               # what happened here?
>>>                      → True
```

24:04 -1:28:41

GO CLASSES

In a **Boolean context**—one that expects a Boolean value—False, 0, "", (the empty string), and None all stand for False and *any other value stands for True*.



Host

```
>>> bool("xyz")
True
>>> bool(0.0)
False
>>> bool("")
False
>>> if 4: print("xyz")    # 4 == True, in this context
xyz
>>> if "ab": print("xyz") # "ab" == True
xyz
>>> if "": print("xyz")   # "" == False
>>>
```

This is very useful in many programming situations.

## Logical Operators

- There are three logical operators:
  - **and**
  - **or**
  - **not**
- They allow us to build more complex Boolean expressions
  - By combining simpler Boolean expressions

---

www.goclasses.in

### and Operation Checks Both

The **and** operation takes two Boolean values and evaluates to **True** if **both** values are **True**. In other words, it evaluates to **False** if **either** value is **False**.

We use **and** when we want to require that both conditions be met at the same time.

Example:  
`(x >= 0) and (x < 10)`

a	b	a and b
True	True	True
True	False	False
False	True	False
False	False	False

false if both are not ↗

---

www.goclasses.in

**or Operation Checks Either**

The **or** operation takes two Boolean values and evaluates to **True** if **either** value is **True**. In other words, it only evaluates to **False** if **both** values are **False**.

We use **or** when there are multiple valid conditions to choose from.

Example:

```
(day == "Saturday") or (day == "Sunday")
```

*true if any one of the operands are true*

**Give examples of associativity in Python**

- For example, the product (\*) and the modulus (%) have the same precedence. So, if both appear in an expression, then the left one will get evaluated first.
- # Testing Left-right associativity  
# Result: 1  
print(4 \* 7 % 3)
- # Testing left-right associativity  
# Result: 0  
print(2 \* (10 % 5))
- As said earlier, the only operator which has right-to-left associativity in Python is the exponent (\*\*) operator.

• See the examples below.

- # Checking right-left associativity of \*\* exponent operator
- # Output: 256

```
print(4 ** 2 ** 2)
```

- # Checking the right-left associativity
- # of \*\*
- # Output: 256

```
print((4 ** 2) ** 2)
```

- You might have observed that the 'print(4 \*\* 2 \*\* 2)' is similar to '(4 \*\* 2 \*\* 2)'.

UPDATE Read the [migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - please note that updating to Notebook 7 might break some of your extensions.

Don't show anymore Host

jupyter Python Programming GO Classes (unsaved changes)

In [24]: `4**2**3`  
Out[24]: `65536`

In [25]: `4**(2**3)`  
Out[25]: `65536`

In [26]: `(4**2)**3`  
Out[26]: `4096`

In [ ]:

Right to left

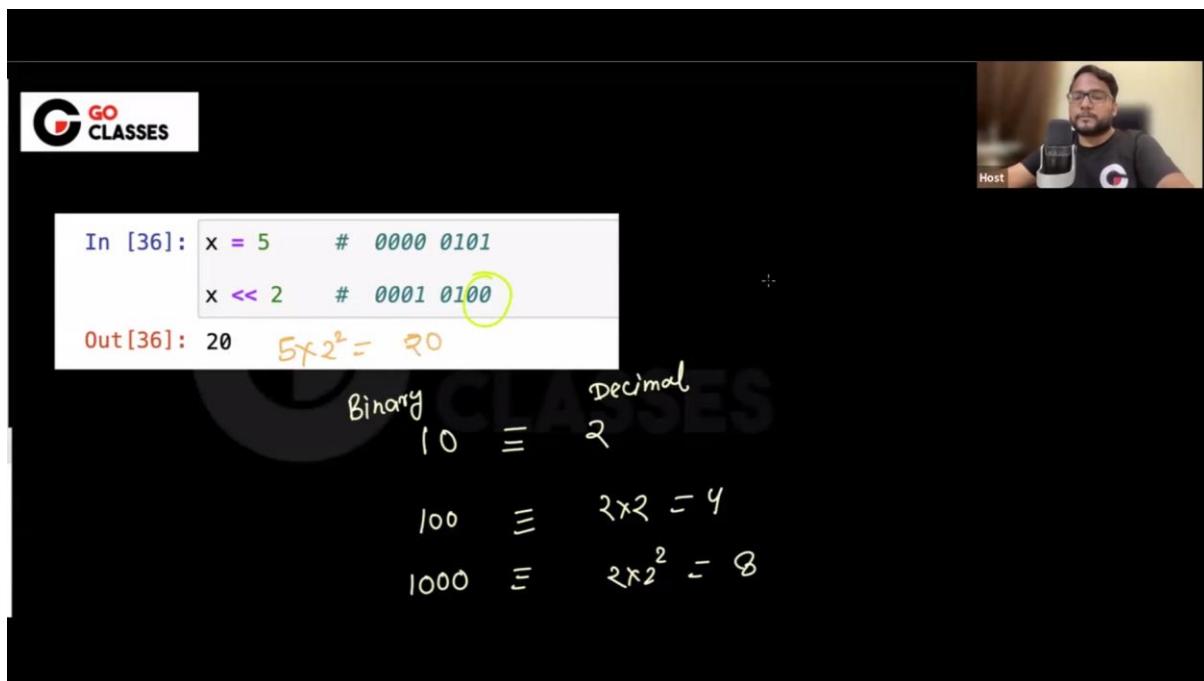
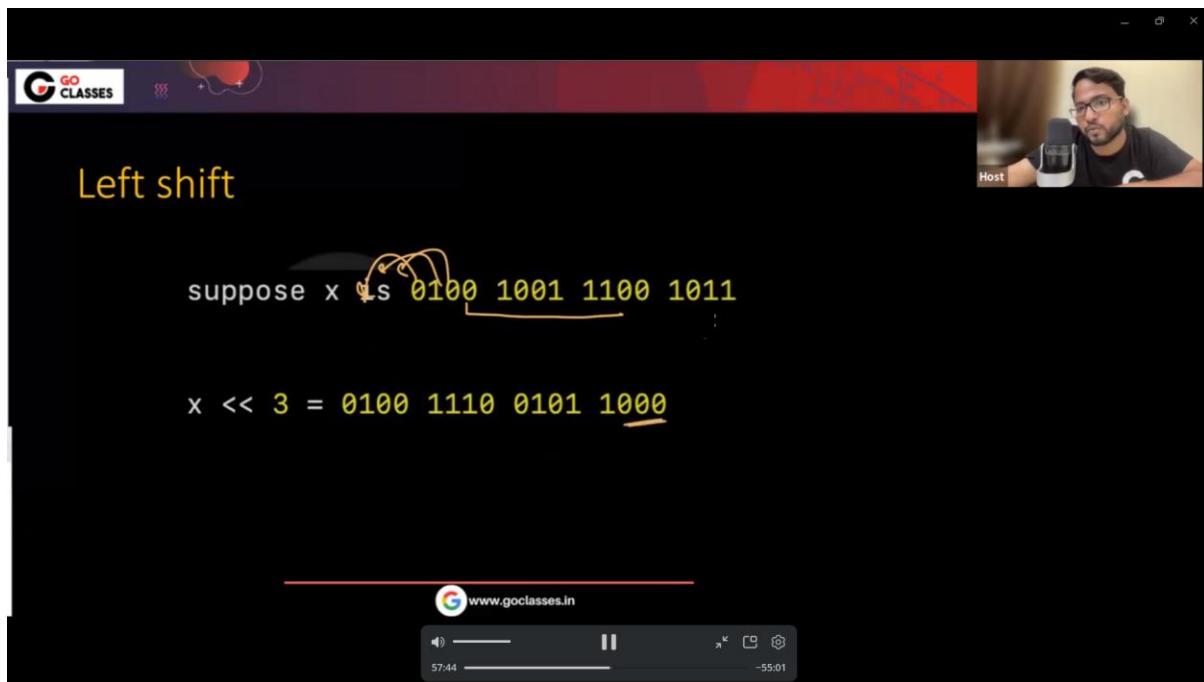
**Bitwise Operators in Python**

Operator	Operation	Sample Expression	Result
<code>&amp;</code>	Bitwise AND	<code>a &amp; b</code>	<ul style="list-style-type: none"> <li>Each bit position in the result is the logical AND of the bits in the corresponding position of the operands.</li> <li>• 1 if both bits are 1, otherwise 0.</li> </ul>
<code> </code>	Bitwise OR	<code>a   b</code>	<ul style="list-style-type: none"> <li>Each bit position in the result is the logical OR of the bits in the corresponding position of the operands.</li> <li>• 1 if either bit is 1, otherwise, 0.</li> </ul>
<code>~</code>	Bitwise NOT	<code>~a</code>	<ul style="list-style-type: none"> <li>Each bit position in the result is the logical negation of the bit in the corresponding position of the operand.</li> <li>• 1 if the bit is 0 and 0 if the bit is 1.</li> </ul>
<code>^</code>	Bitwise XOR (exclusive OR)	<code>a ^ b</code>	<ul style="list-style-type: none"> <li>Each bit position in the result is the logical XOR of the bits in the corresponding position of the operands.</li> <li>• 1 if the bits in the operands are different, 0 if they're equal.</li> </ul>
<code>&gt;&gt;</code>	Bitwise right shift	<code>a &gt;&gt; n</code>	Each bit is shifted right <code>n</code> places.
<code>&lt;&lt;</code>	Bitwise left shift	<code>a &lt;&lt; n</code>	Each bit is shifted left <code>n</code> places.

<https://static.realpython.com/guides/python-operators.pdf>

<i>op1</i>	<i>op2</i>	<i>op1 &amp; op2</i>	<i>op1   op2</i>	<i>op1 ^ op2</i>
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

[www.goclasses.in](http://www.goclasses.in)



Right Shift

Host

```
suppose x is 0100 1001 1100 1001  
x >> 3 = 0000 1001 0011 1001
```

www.goclasses.in

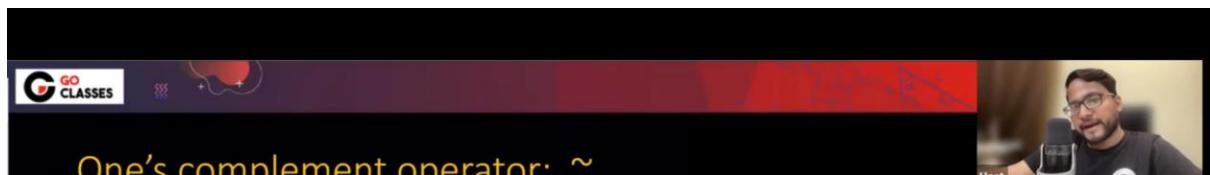
1:01:03 -51:42

Host

```
0000 1010  
y = 10  
x = y << 1  
print(x)  
↳ 20
```

0000 1010  
y = 10  
x = y >> 1  
print(x)  
↳ 5

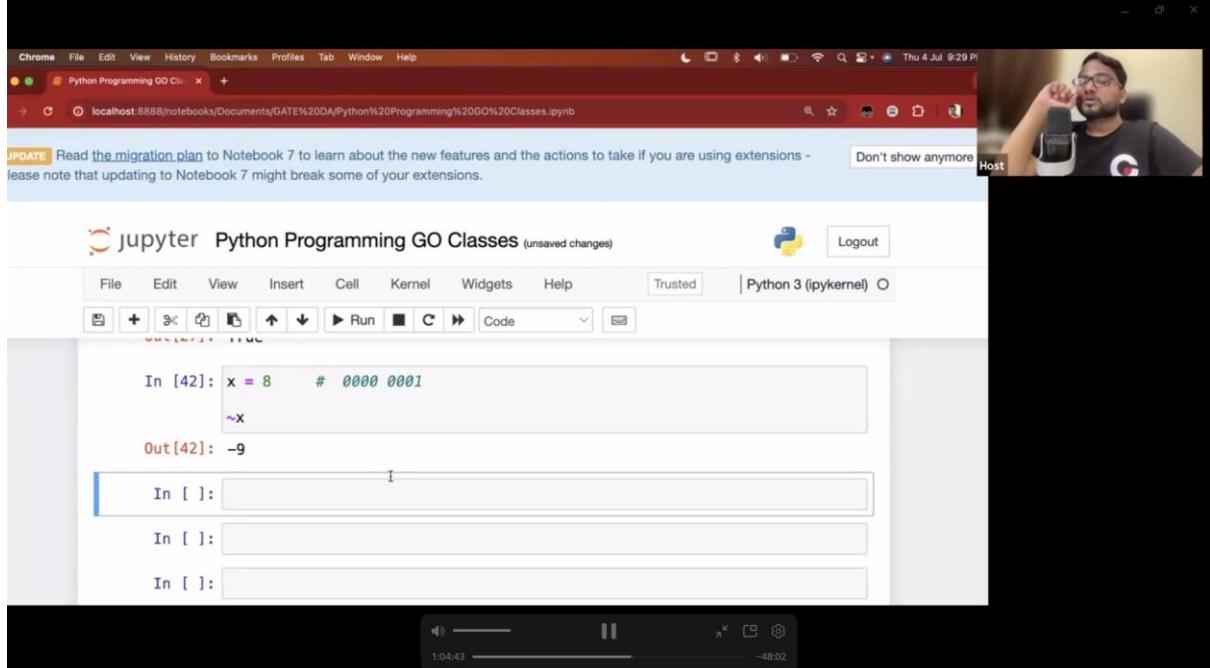
www.goclasses.in



One's complement operator:  $\sim$

```
x = 1001 0110 1100 1011
~x = 0110 1001 0011 0100
```

www.goclasses.in



Chrome File Edit View History Bookmarks Profiles Tab Window Help

localhost:8888/notebooks/Documents/GATE%20DA/Python%20Programming%20GO%20Classes.ipynb

UPDATE Read the [migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.

Don't show anymore Host

jupyter Python Programming GO Classes (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel) Logout

In [42]: `x = 8 # 0000 0001`  
~x  
Out[42]: -9

In [ ]:

In [ ]:

In [ ]:

1:04:43 -48:02

GO CLASSES

# 1111 0111

Host

$1111\ 0111$

$-32 + 16$

$$= -16 + 7 = -9$$

$0111$

$2^0$

$2^1$

$2^2$

$2^3$

$2^4$

$2^5$

$2^6$

$-64 + 32 + 16$

$$= -16 + 7 = -9$$

1:12:02 - 40:43

GO CLASSES

Host

Arithmetic operators	$+, -, /, //, \cdot$
Relational operators	$>, <=, >=, !=$
Logical operators	and, or, not
Bitwise operators	$\&,  , \wedge, \ll, \gg$

1:16:18 - 36:27

A screenshot of a video conference interface. In the top right corner, there is a video feed of a man wearing glasses and a black t-shirt with a white logo. Below his video, the word "Host" is written. The top left corner features the "GO CLASSES" logo. The main area of the screen shows a code editor with the following Python code:

```
if condition:  
    code 1  
else:  
    code 2
```

The code editor has a dark theme with syntax highlighting. Below the code editor, there is a navigation bar with icons for back, forward, and search, along with the URL "www.goclasses.in". The time "1:16:45" is shown on the left, and "-36:00" is on the right.

A screenshot of a video conference interface, similar to the one above. It shows a host in the top right and the "GO CLASSES" logo in the top left. The main area displays a code editor with the following Python code:

```
a = 33  
b = 200  
if b > a:  
    print("b is greater than a")
```

The code editor has a dark theme with syntax highlighting. Below the code editor, there is a navigation bar with icons for back, forward, and search, along with the URL "www.goclasses.in". The time "1:17:13" is shown on the left, and "-35:32" is on the right.

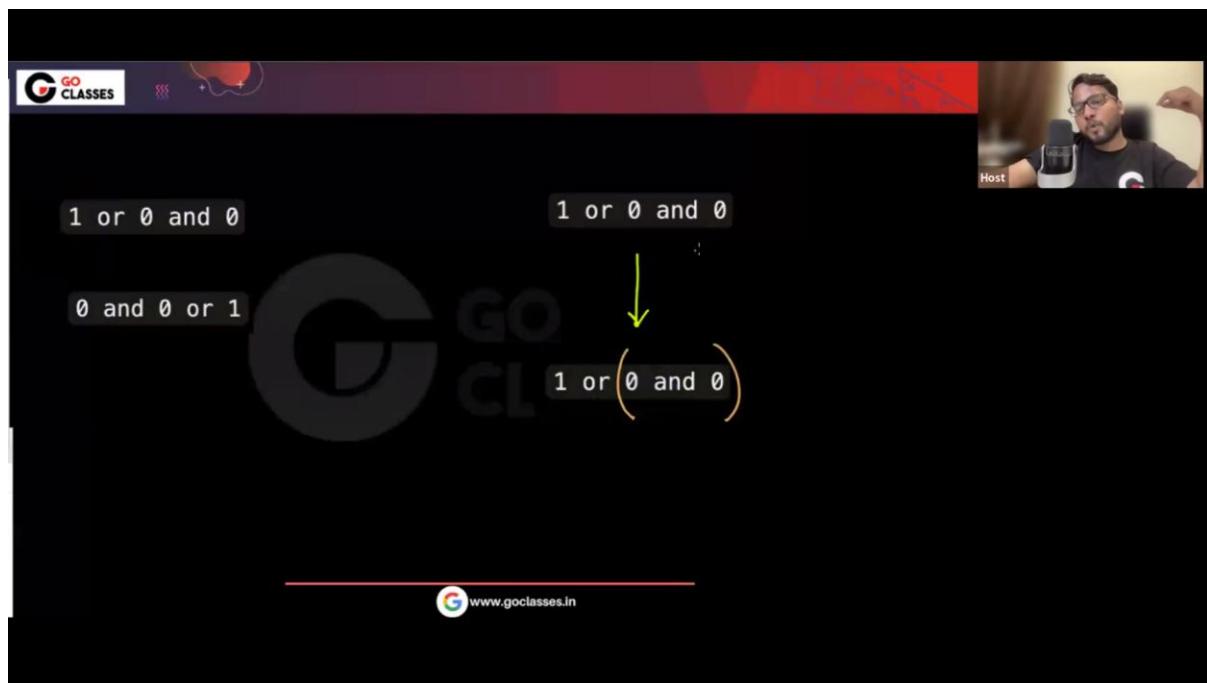
A screenshot of a Go Classes video player. On the left, there is a toolbar with icons for search, refresh, and other controls. In the center, a video frame shows a male host wearing glasses and a black t-shirt with a red logo, sitting in front of a microphone. The video content itself is a Python script:

```
print("hello")  
if x < 10:  
    print("wahoo!")  
elif x <= 99:  
    print("meh")  
else:  
    print("ruh roh")  
print("goodbye")
```

Handwritten annotations in yellow and green are overlaid on the code:

- $x = 50$  is written above the first line.
- The word "meh" is underlined below "print("meh")".
- The word "goodbye" is underlined below "print("goodbye")".
- A double-headed arrow symbol ( $\rightleftharpoons$ ) is placed between the two underlined words.

At the bottom of the video frame, there is a watermark for [www.goclasses.in](http://www.goclasses.in).



And > Or

The screenshot shows a video player interface for 'GO CLASSES'. At the top, there are two boxes: '1 or 0 and 0' and '0 and 0 or 1'. Below them is a large 'G' logo. To the right of the 'G' is a diagram illustrating the evaluation of the expression '1 or (0 and 0)'. A yellow arrow points down to the inner expression '0 and 0', and another yellow arrow points from the outer expression '1 or (0 and 0)' to the result '1 and 0', which is then simplified to '0'. The word 'WRONG' is written next to the final result '0'. The video player controls at the bottom show a progress bar from 1:35:45 to -17:00.

The screenshot shows a video player interface for 'GO CLASSES'. At the top, there is a box containing 'expr1 or expr2 and expr3'. Below it, a handwritten note says 'first step: put brackets' with an arrow pointing to a box around 'expr1'. Next to it is the text 'expr1 or (expr2 and expr3)'. Below this, another handwritten note says 'Second step: order of evaluation: left to right' with an arrow pointing to the text 'expr1 will get evaluated first'.

Precedence order of all operators in Python.

<code>**</code>	Exponentiation [5]
<code>+x, -x, ~x</code>	Positive, negative, bitwise NOT
<code>*, @, /, //, %</code>	Multiplication, matrix multiplication, division, floor division, remainder [6]
<code>+, -</code>	Addition and subtraction
<code>&lt;&lt;, &gt;&gt;</code>	Shifts
<code>&amp;</code>	Bitwise AND
<code>^</code>	Bitwise XOR
<code> </code>	Bitwise OR
<code>in, not in, is, is not, &lt;, &lt;=, &gt;, &gt;=, !=, ==</code>	Comparisons, including membership tests and identity tests
<code>not x</code>	Boolean NOT
<code>and</code>	Boolean AND
<code>or</code>	Boolean OR

Arithmetic      Bitwise      Logical Operators      Relational

Consider the following program fragment.  
Which of the following `if` condition(s) prints GO Classes?

```

b = 1
c = 1
d = 0

if (0 and 0 == 0):
    print("1 GO Classes")

if b or b - 1 == 0:
    print("2 GO Classes")

if c or c - 1 == 0:
    print("3 GO Classes")

if d or d + 1 == 0:
    print("4 GO Classes")

```

Handwritten annotations:

- Condition 1:  $0 \text{ and } (0 == 0)$  is marked as false.
- Condition 2:  $1 \text{ or } 1 - 1 == 0$  is marked as anything.
- Condition 3:  $1 \text{ or } (0 == 0)$  is marked as anything.
- Condition 4:  $d + 1 == 0$  has a large X next to it.

www.goclasses.in