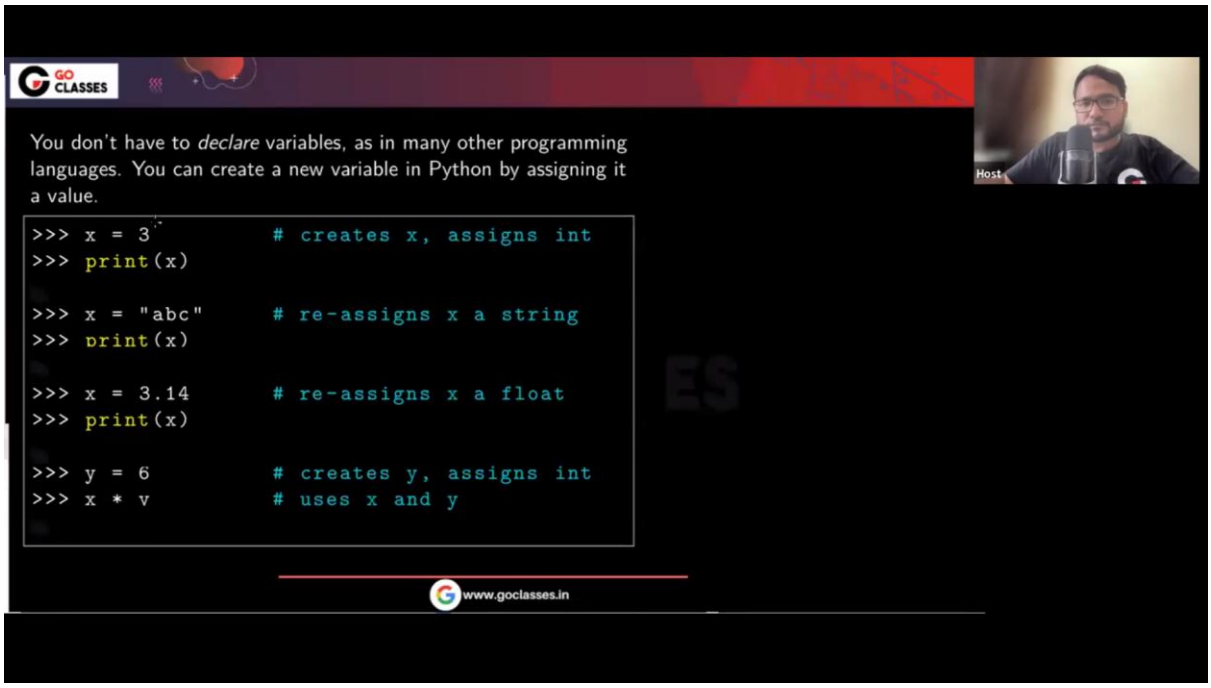
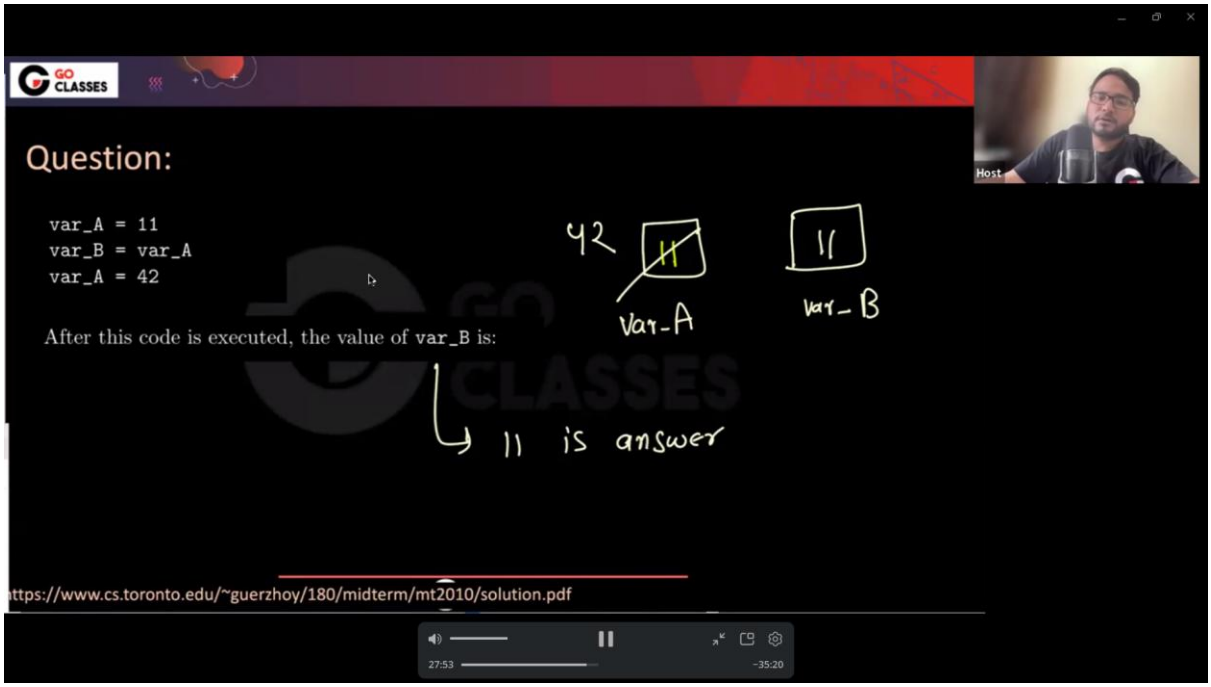
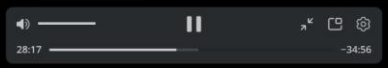
Variables,operators and conditional statements:





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`



Arithmetic Operators

- Operators

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

GO CLASSES


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

In [10]: 5//2
Out[10]: 2
```

2 ⟌ 5 ⌊2

GO CLASSES

- 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

Precedence Example


```
x = 1 + 3 * 5 / 2
        15
          7.5
    8.5
```

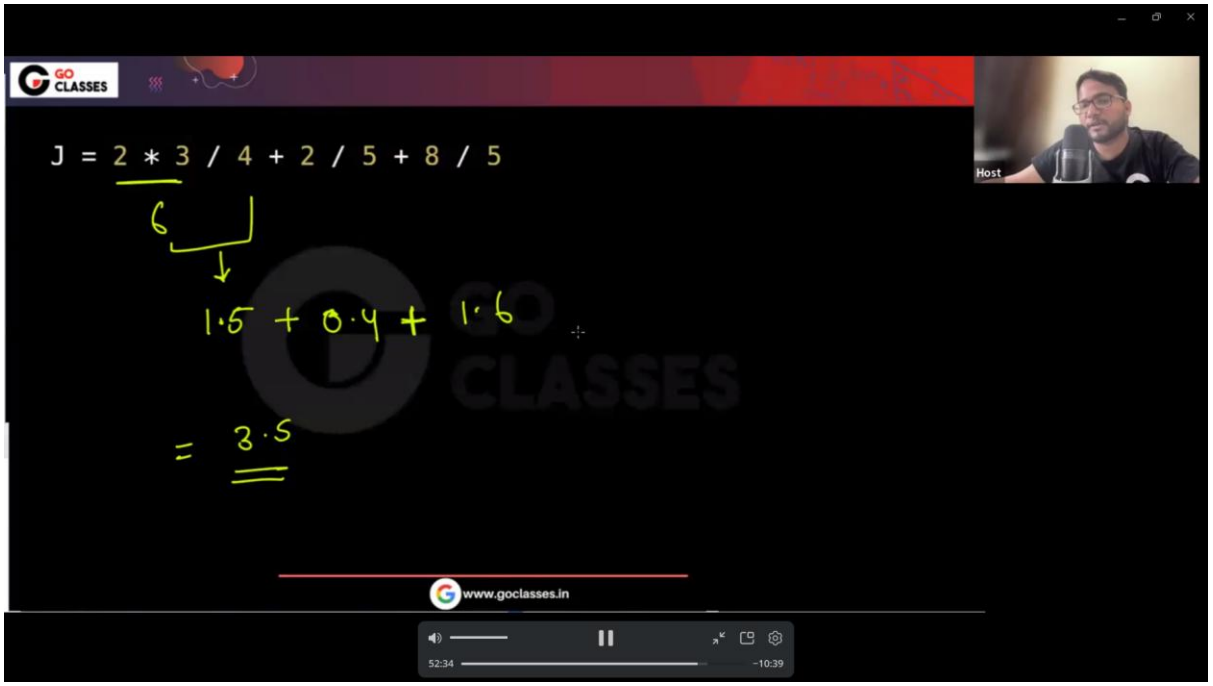
x 8.5

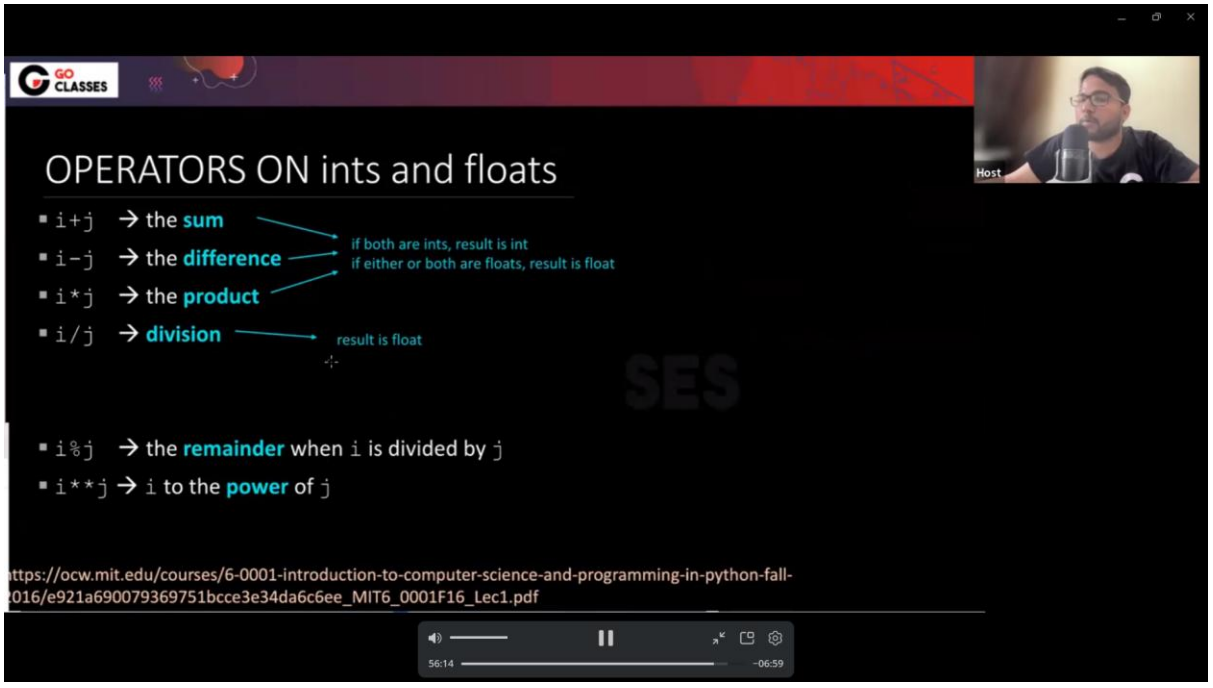
GO CLASSES

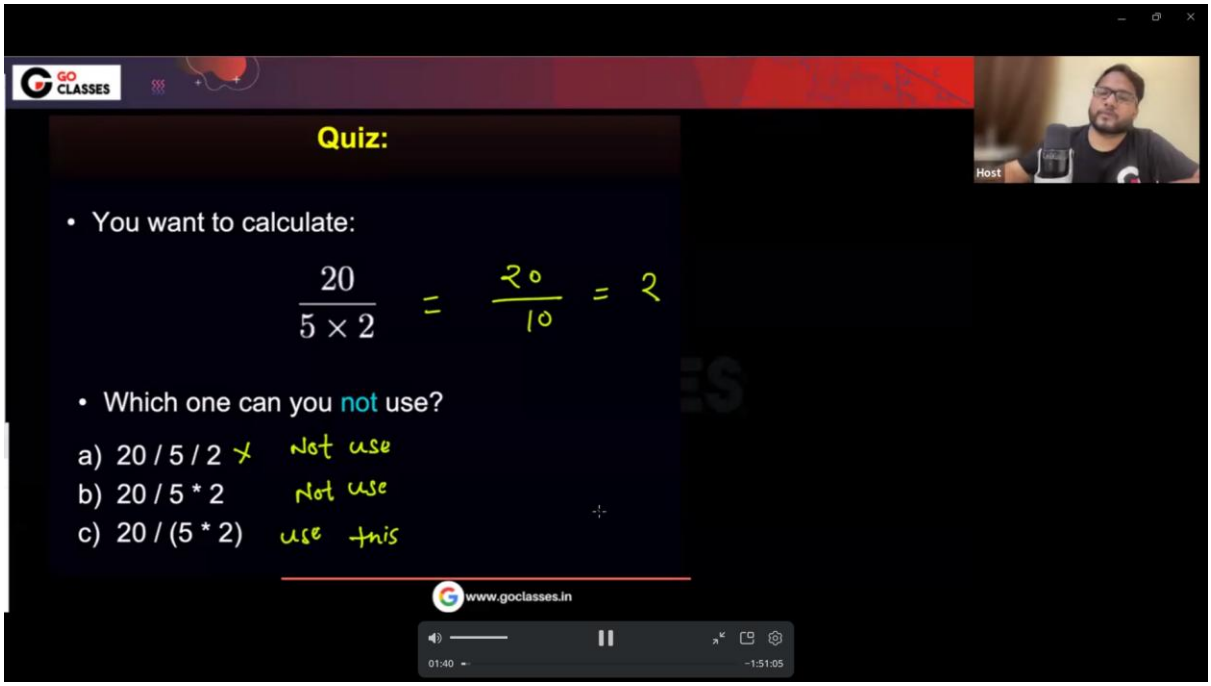
Precedence of Arithmetic Operators:

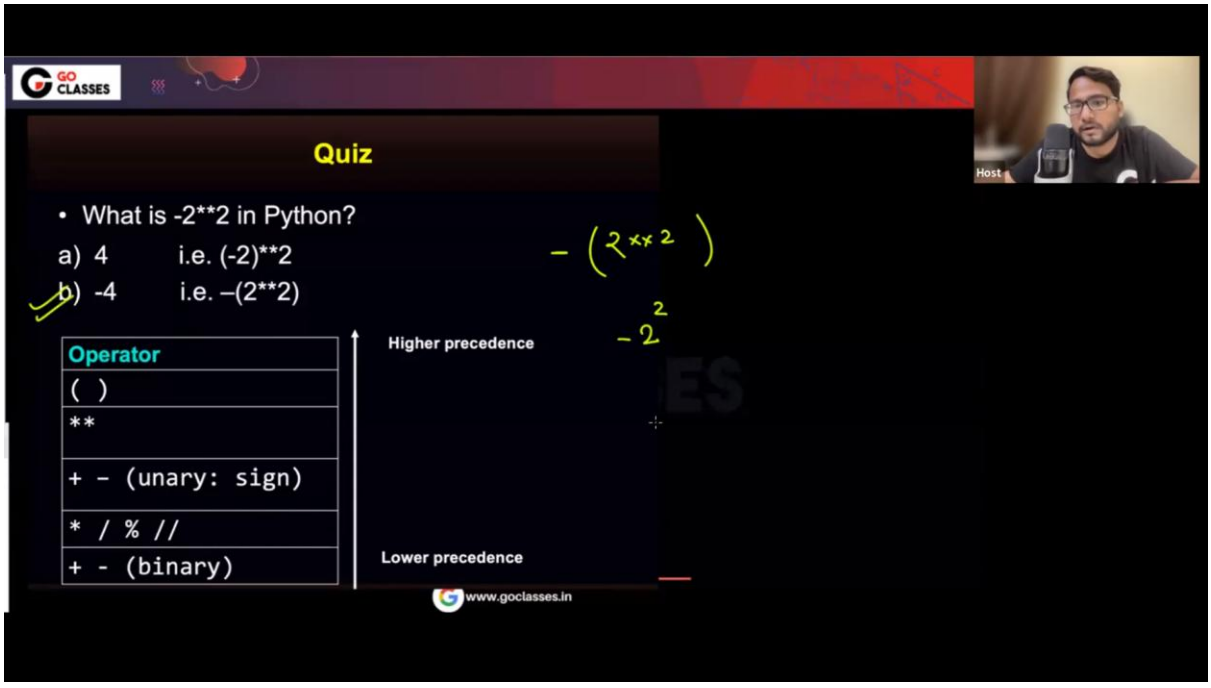
- High priority $*, /, //, \%$
- Low priority $+, -$

- 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.

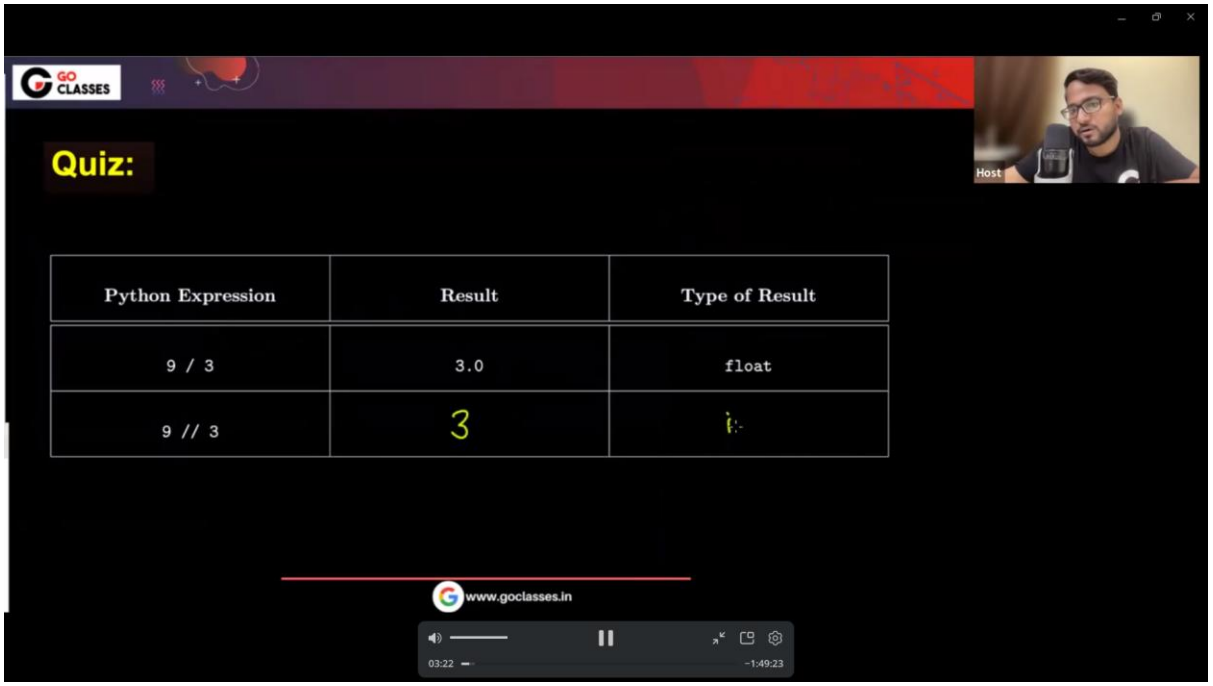

GO CLASSES
J = 2 * 3 / 4 + 2 / 5 + 8 / 5
6
1.5 + 0.4 + 1.6
= 3.5
Host
www.goclasses.in
52:34
-10:39


GO CLASSES
OPERATORS ON ints and floats
i+j → the sum
i-j → the difference
i*j → the product
if both are ints, result is int
if either or both are floats, result is float
i/j → division
result is float
i%j → the remainder when i is divided by j
i**j → i to the power of j
https://ocw.mit.edu/courses/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/e921a690079369751bcce3e34da6c6ee_MIT6_0001F16_Lec1.pdf
Host
56:14
-06:59





Unary priority < **



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

Implicit Type Conversion

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

5 + 1.9

num1 + num3 → 6.9 float

num1 + num2 → 7 int

0000101

float 5.0

=

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

07:13 -1:45:32

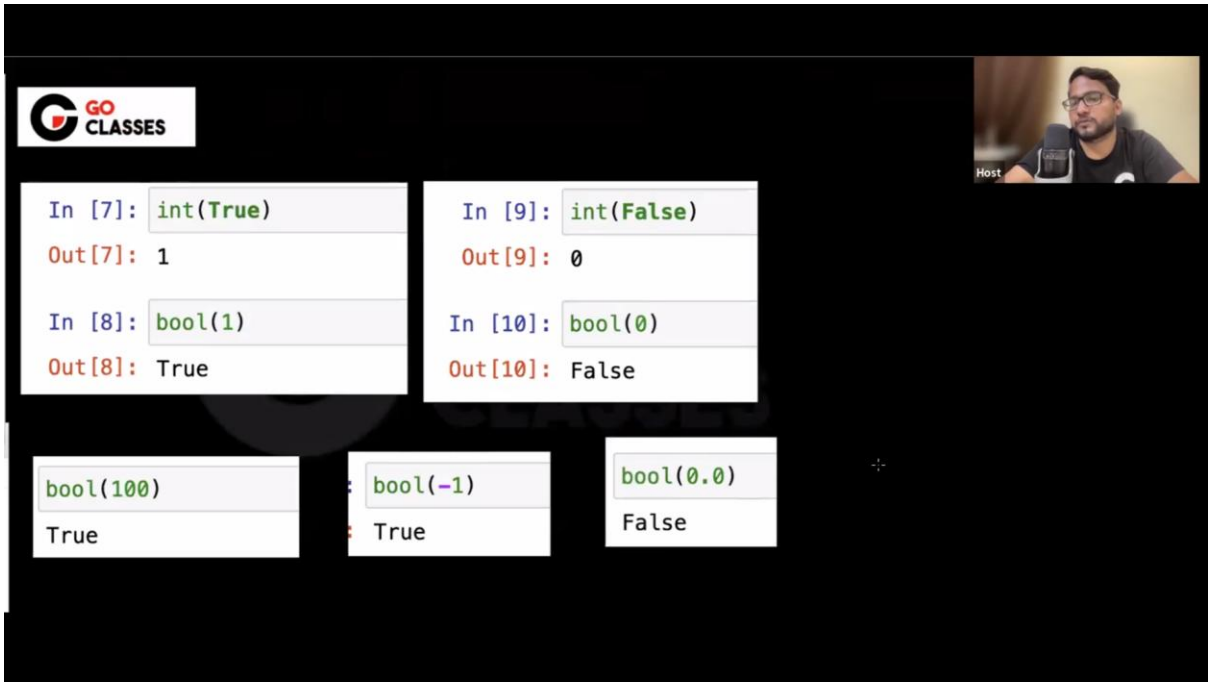
Explicit Type Conversion

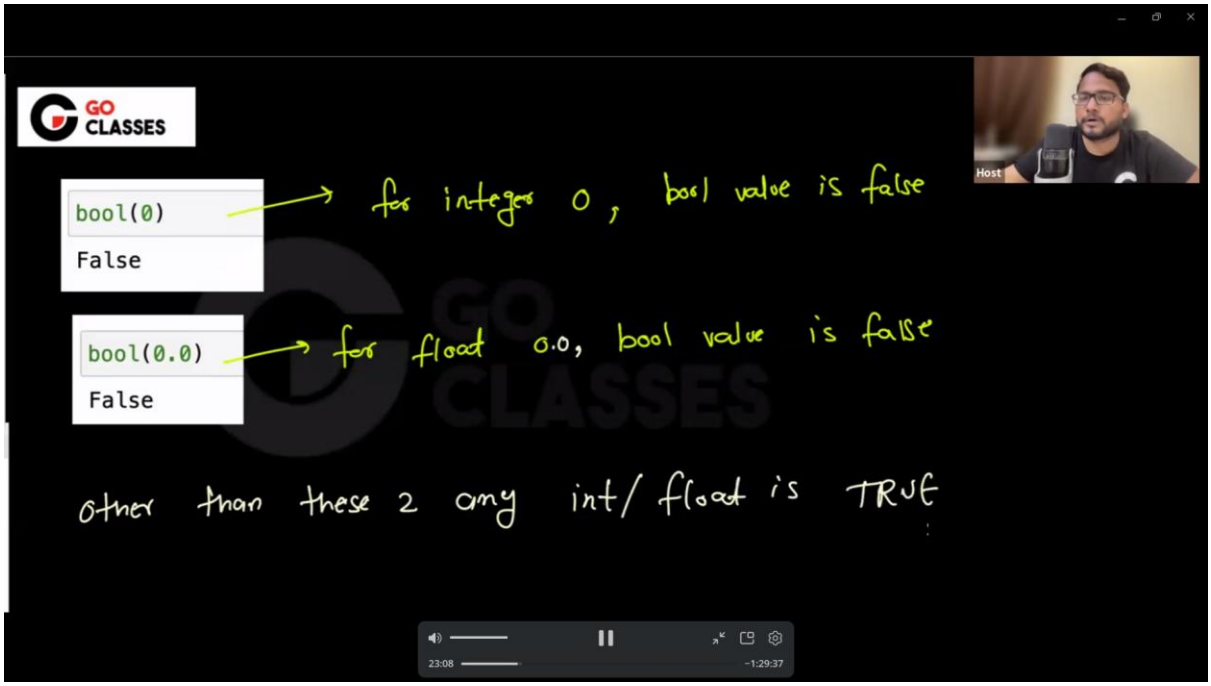
```
num1 = 5
num2 = 2
num3 = 1.9
```

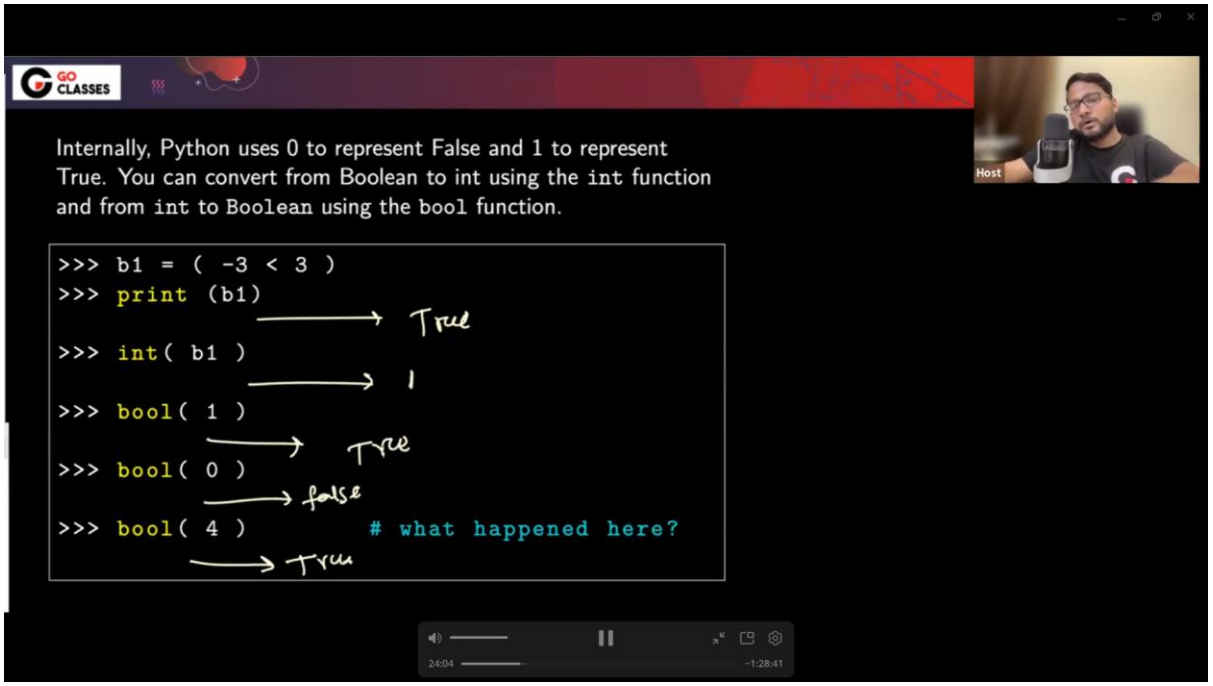
- Use **`float(value)`** to create new real-valued number
  - **`float(num1)`** = 5.0 **(float)**
  - Note that **num1** is not changed. We created a new value.
  - **`num1 + float(num2)`** = 7.0 **(float)**
  - **`num1 + num2`** = 7 **(int)**
- Use **`int(value)`** to create a new integer-valued number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

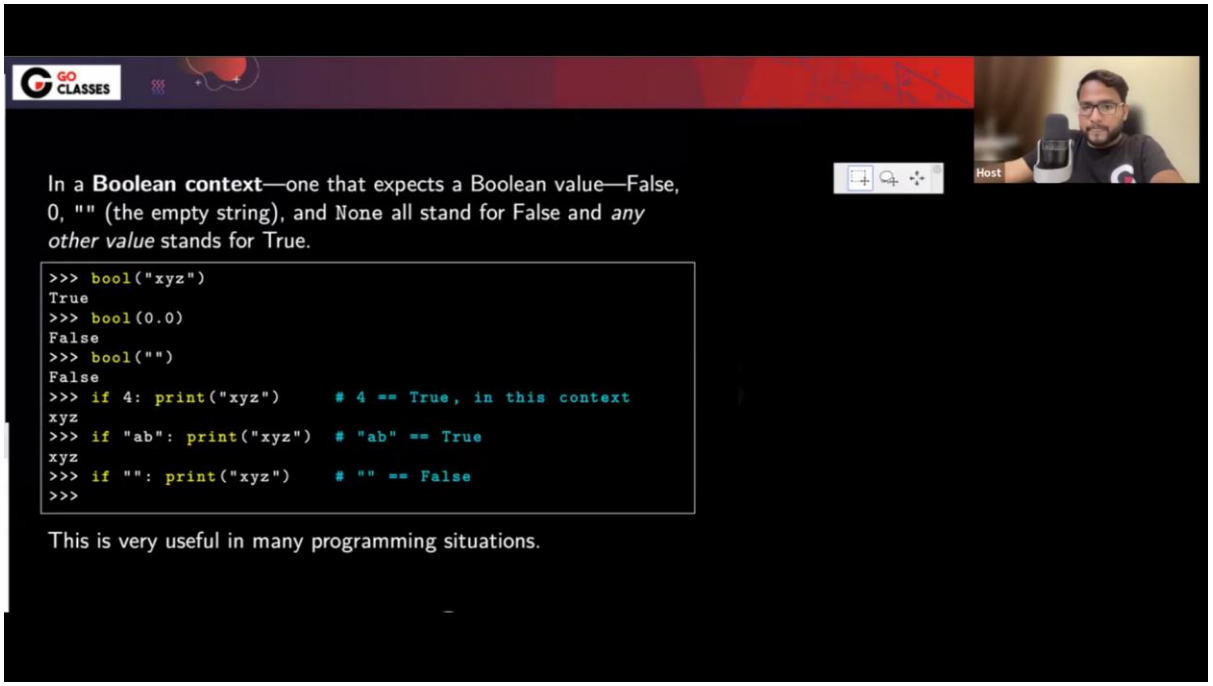
www.goclasses.in

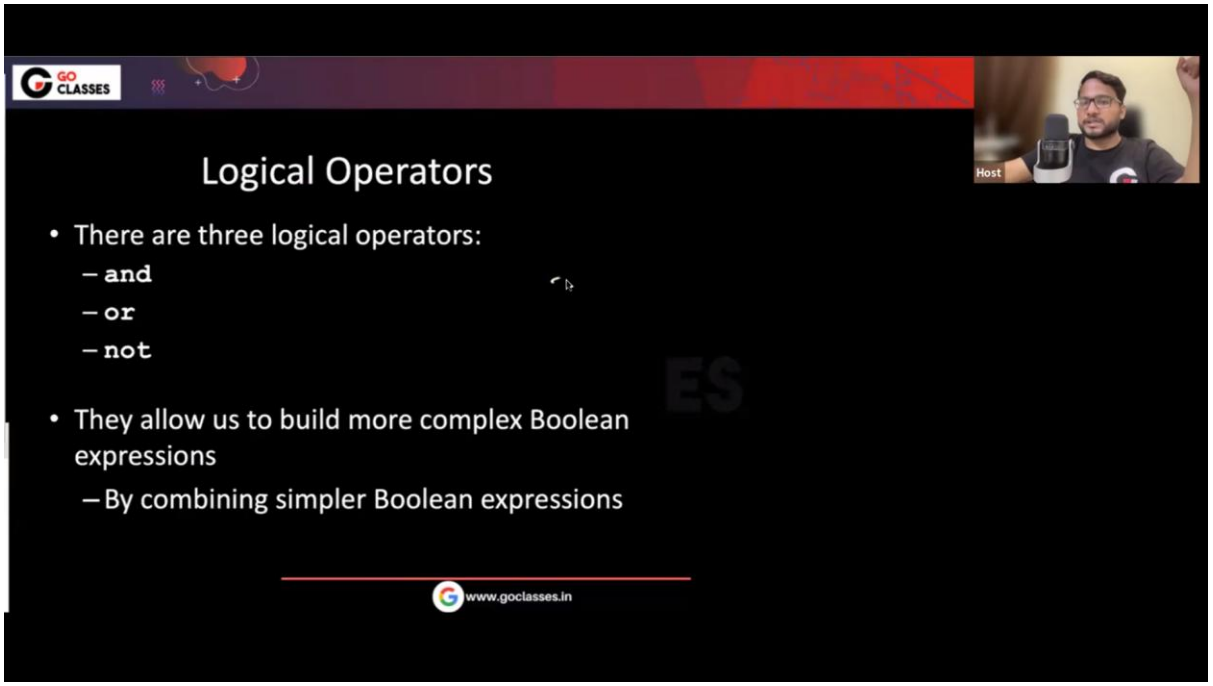
11:11 -1:41:34

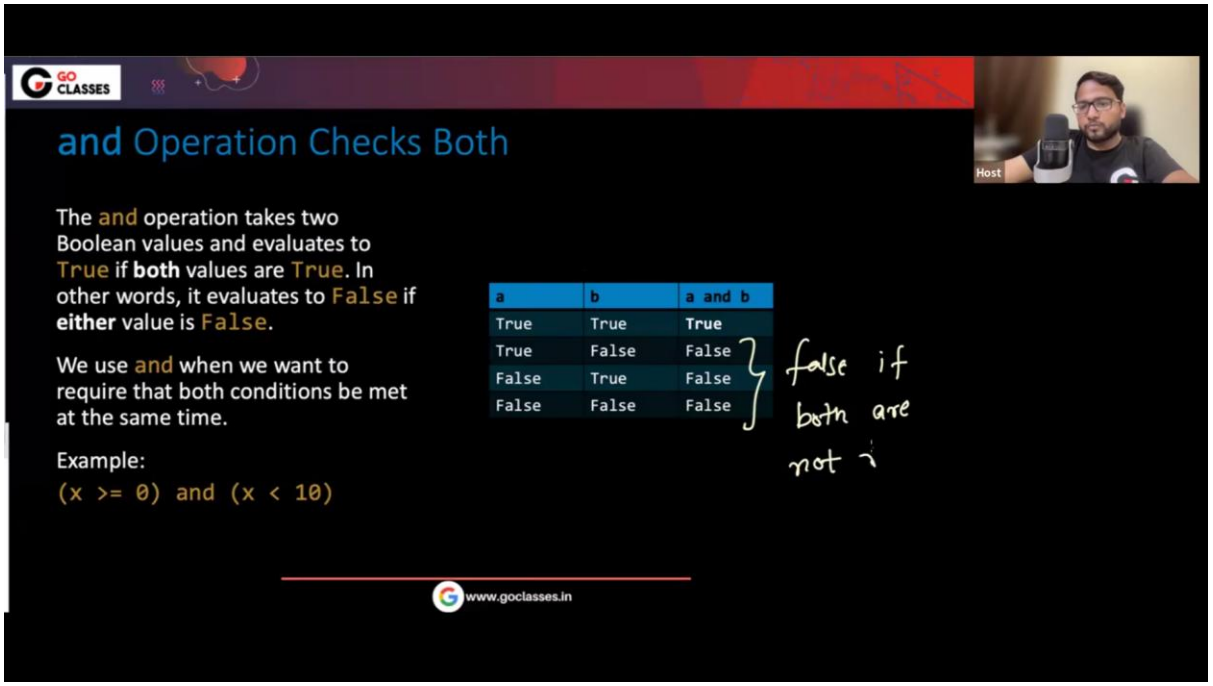

GO CLASSES
Host
In [7]: int(True)
Out[7]: 1
In [8]: bool(1)
Out[8]: True
In [9]: int(False)
Out[9]: 0
In [10]: bool(0)
Out[10]: False
bool(100)
True
bool(-1)
True
bool(0.0)
False


GO CLASSES
Host
bool(0)
False
for integer 0, bool value is false
bool(0.0)
False
for float 0.0, bool value is false
other than these 2 any int/float is TRUE
23:08
-1:29:37


GO CLASSES
Host
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.
>>> 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
Host
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.
>>> 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.


GO CLASSES
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
Host


GO CLASSES
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
Host

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.

a	b	a or b
True	True	True
True	False	True
False	True	True
False	False	False

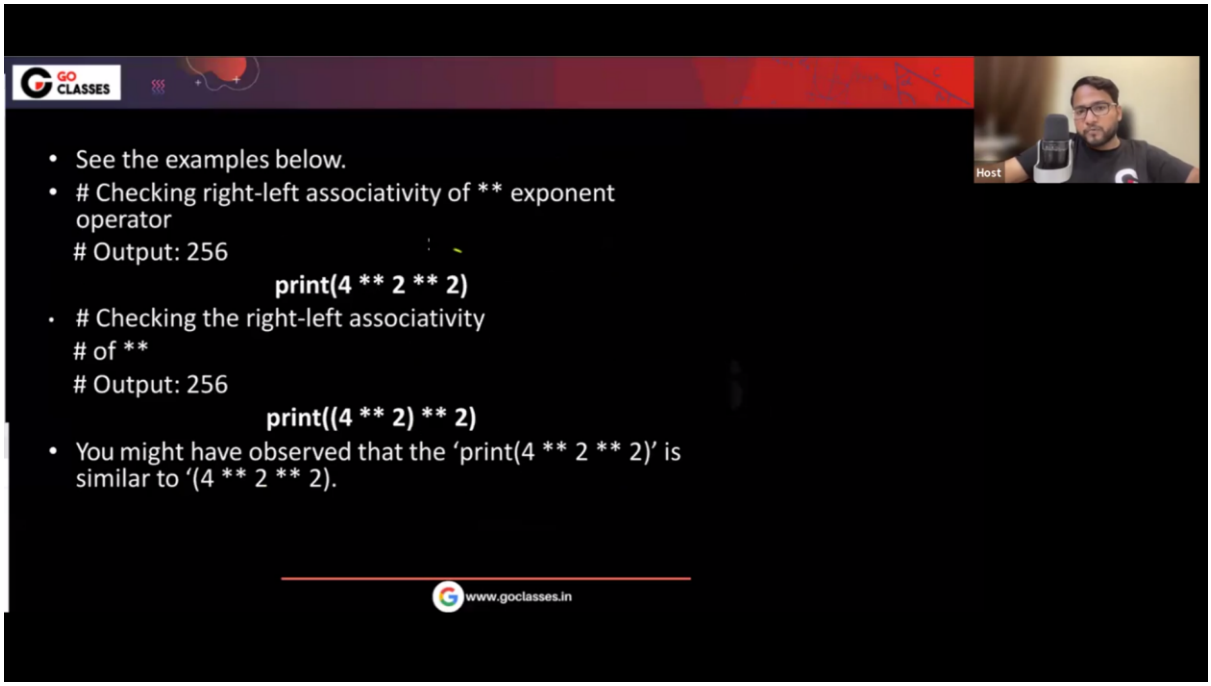
true if any one of the operands are true

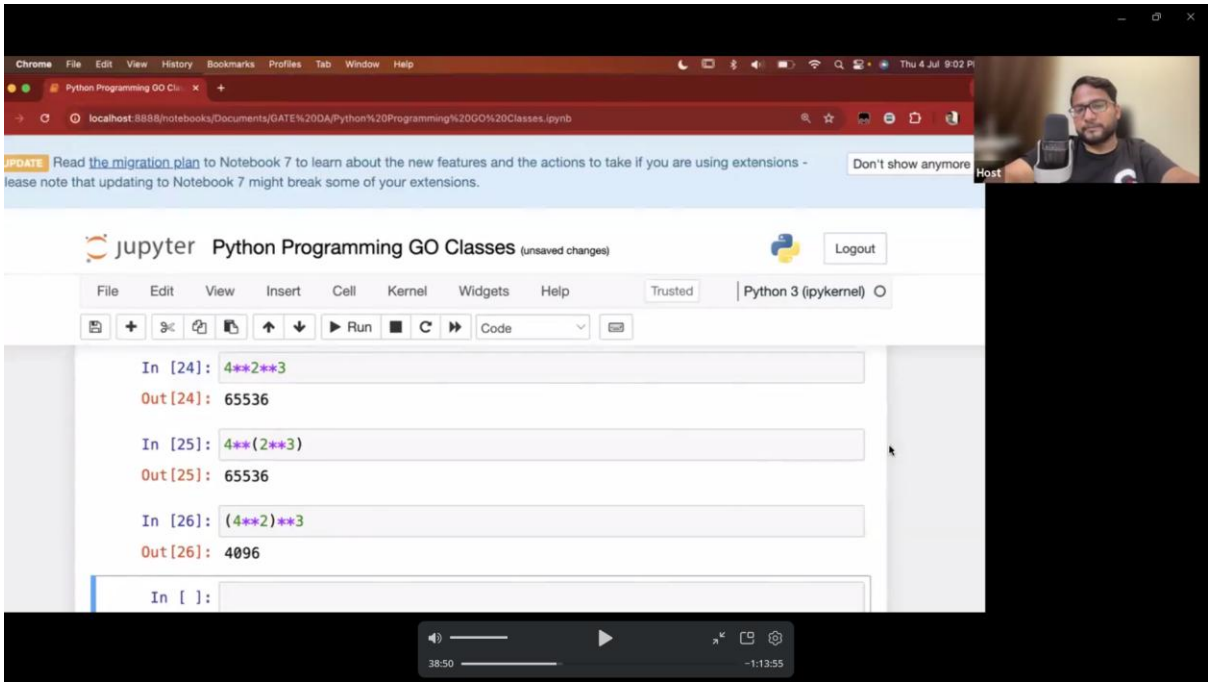
Example:

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

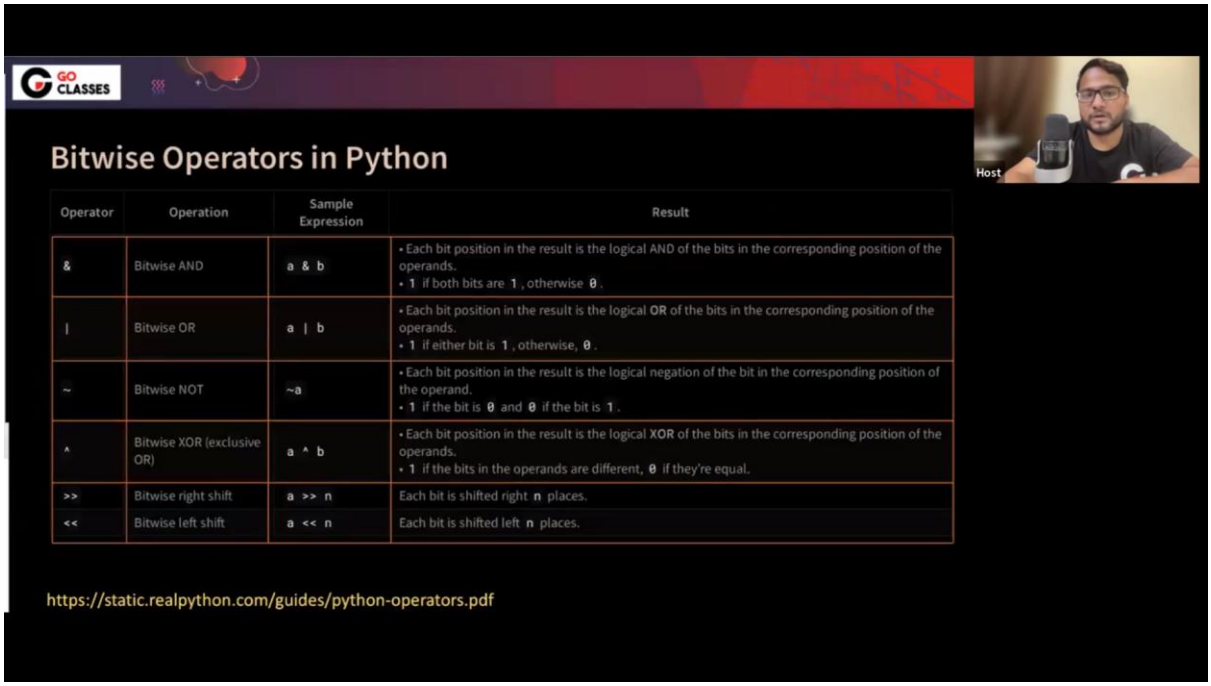

Give examples of associativity in Python

- 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.

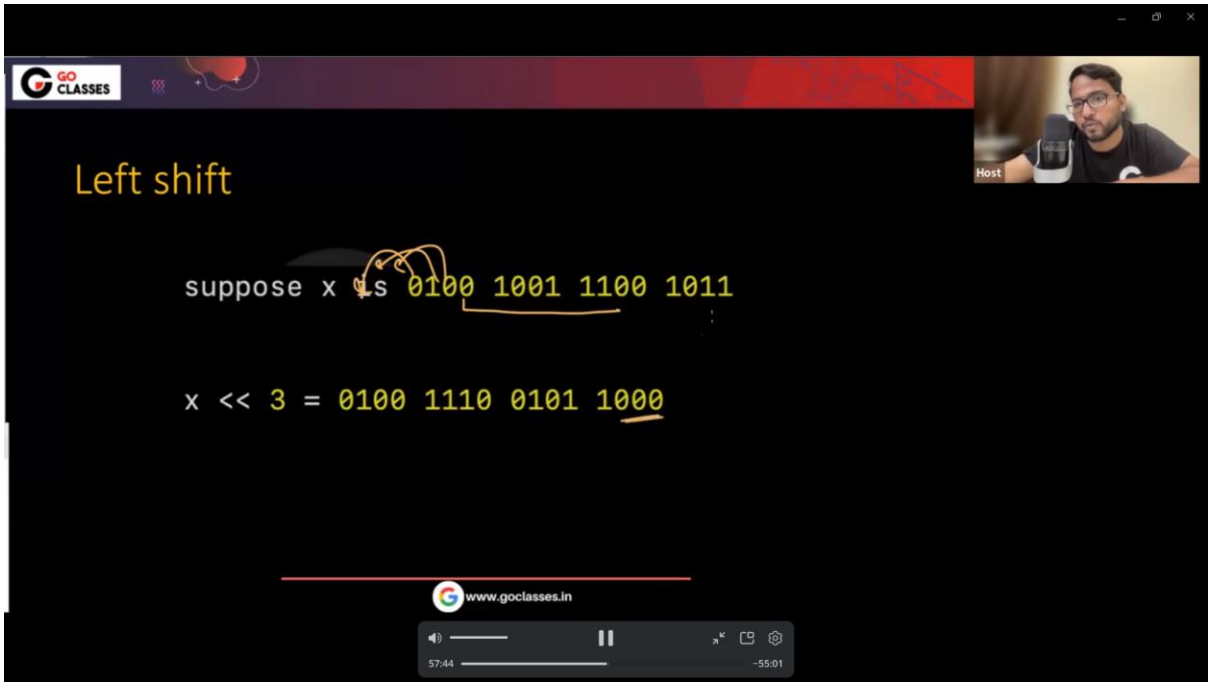


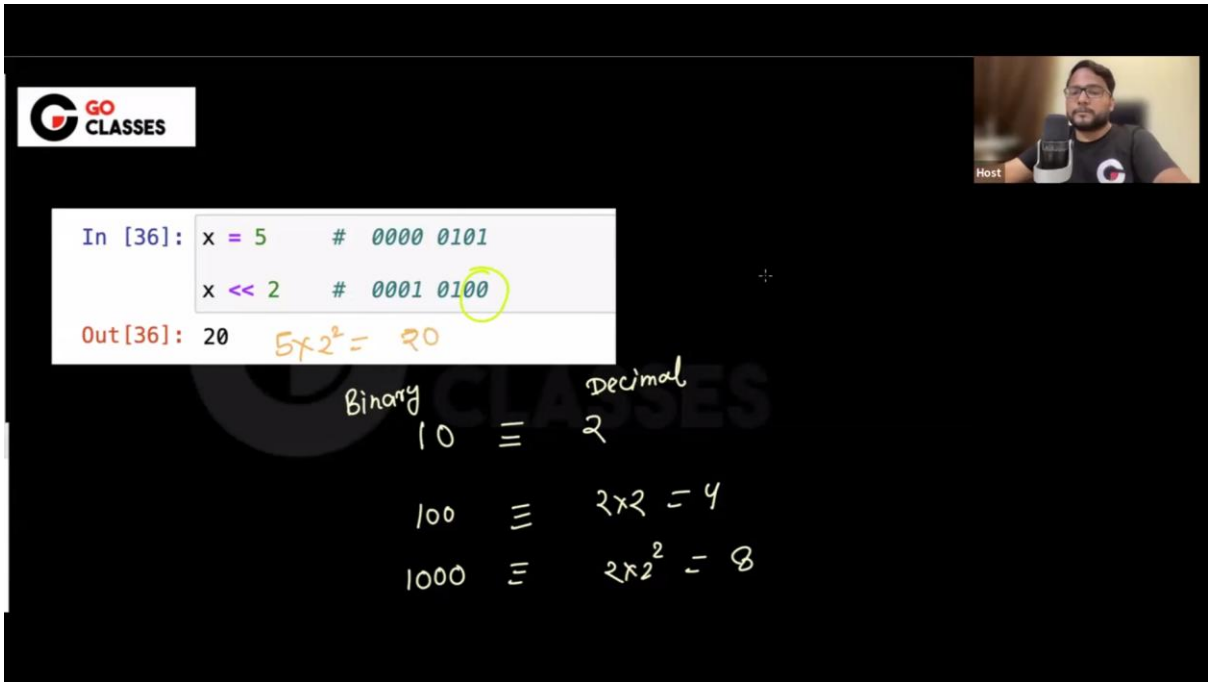


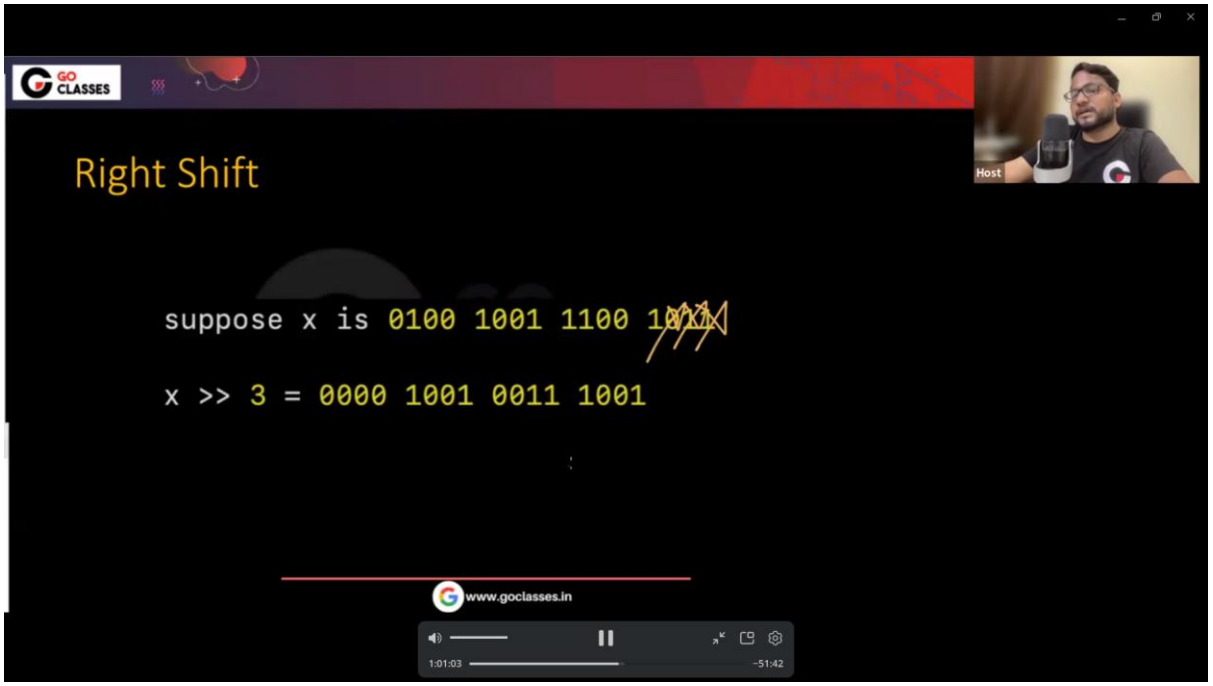
Right to left

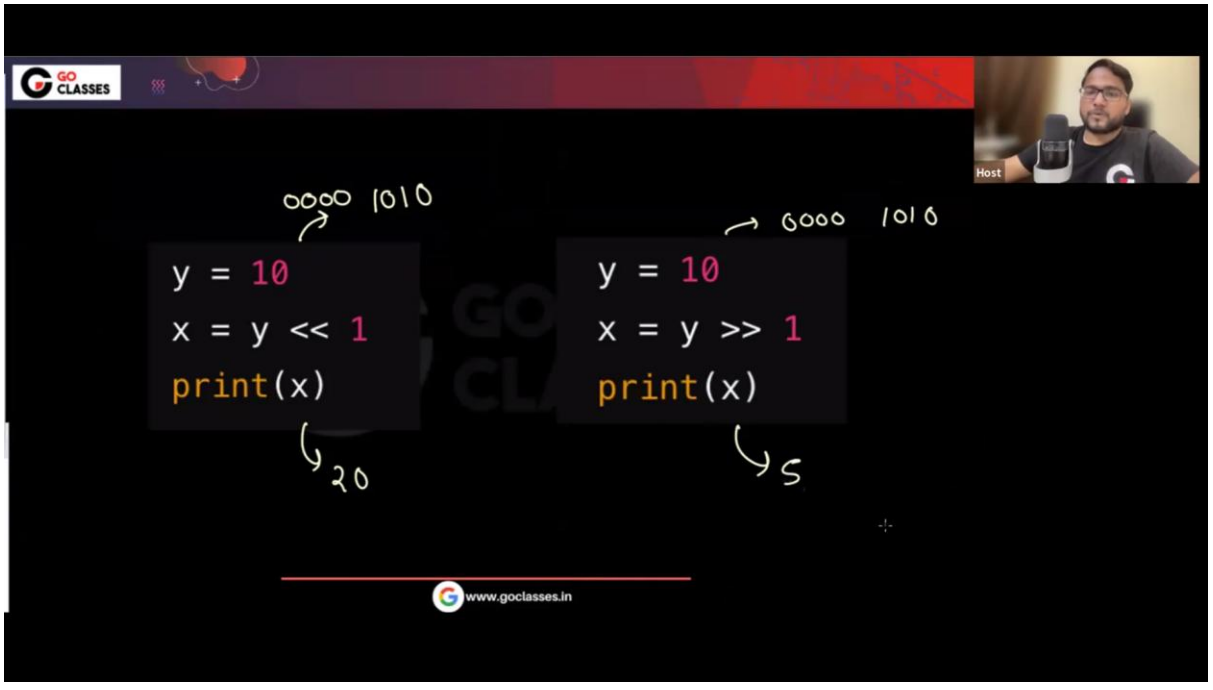

GO CLASSES
Host
Bitwise Operators in Python
Operator | Operation | Sample Expression | Result
& | Bitwise AND | a & b | • Each bit position in the result is the logical AND of the bits in the corresponding position of the operands. • 1 if both bits are 1, otherwise 0.
| | Bitwise OR | a | b | • Each bit position in the result is the logical OR of the bits in the corresponding position of the operands. • 1 if either bit is 1, otherwise, 0.
~ | Bitwise NOT | ~a | • Each bit position in the result is the logical negation of the bit in the corresponding position of the operand. • 1 if the bit is 0 and 0 if the bit is 1.
^ | Bitwise XOR (exclusive OR) | a ^ b | • Each bit position in the result is the logical XOR of the bits in the corresponding position of the operands. • 1 if the bits in the operands are different, 0 if they're equal.
>> | Bitwise right shift | a >> n | Each bit is shifted right n places.
<< | Bitwise left shift | a << n | Each bit is shifted left n places.
https://static.realpython.com/guides/python-operators.pdf

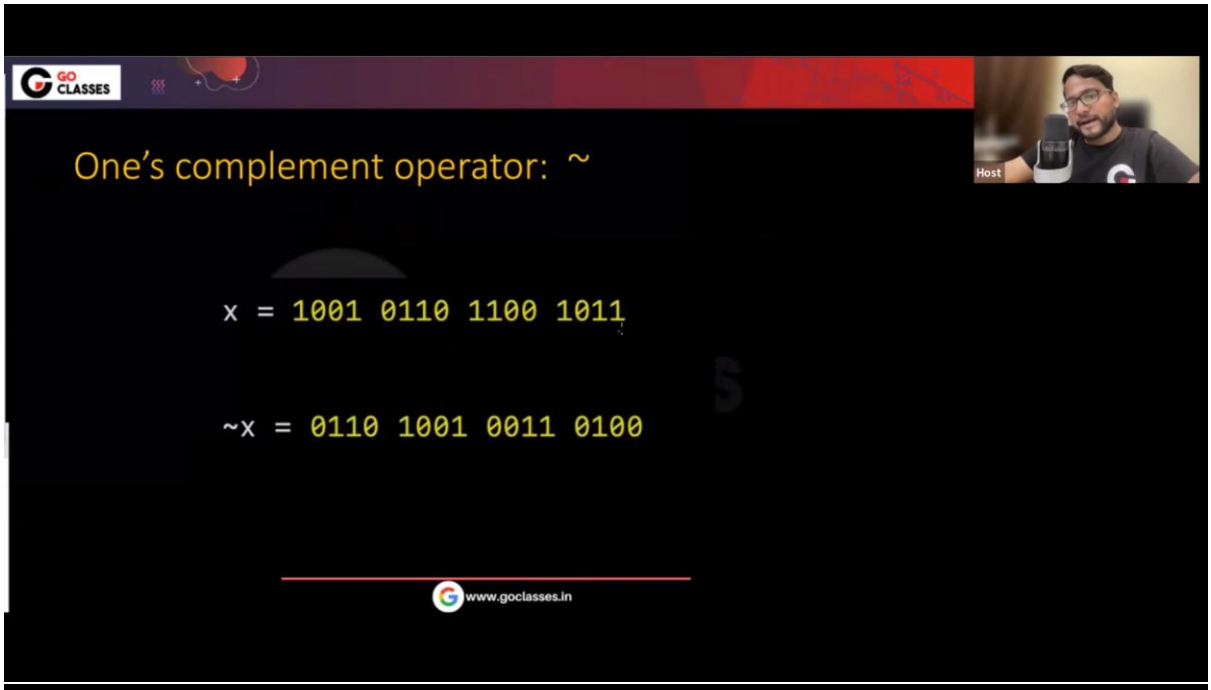

GO CLASSES
Host
op1 | op2 | op1 & op2 | op1 | op2 | op1 ^ op2
1 | 1 | 1 | 1 | 0
1 | 0 | 0 | 1 | 1
0 | 1 | 0 | 1 | 1
0 | 0 | 0 | 0 | 0
www.goclasses.in

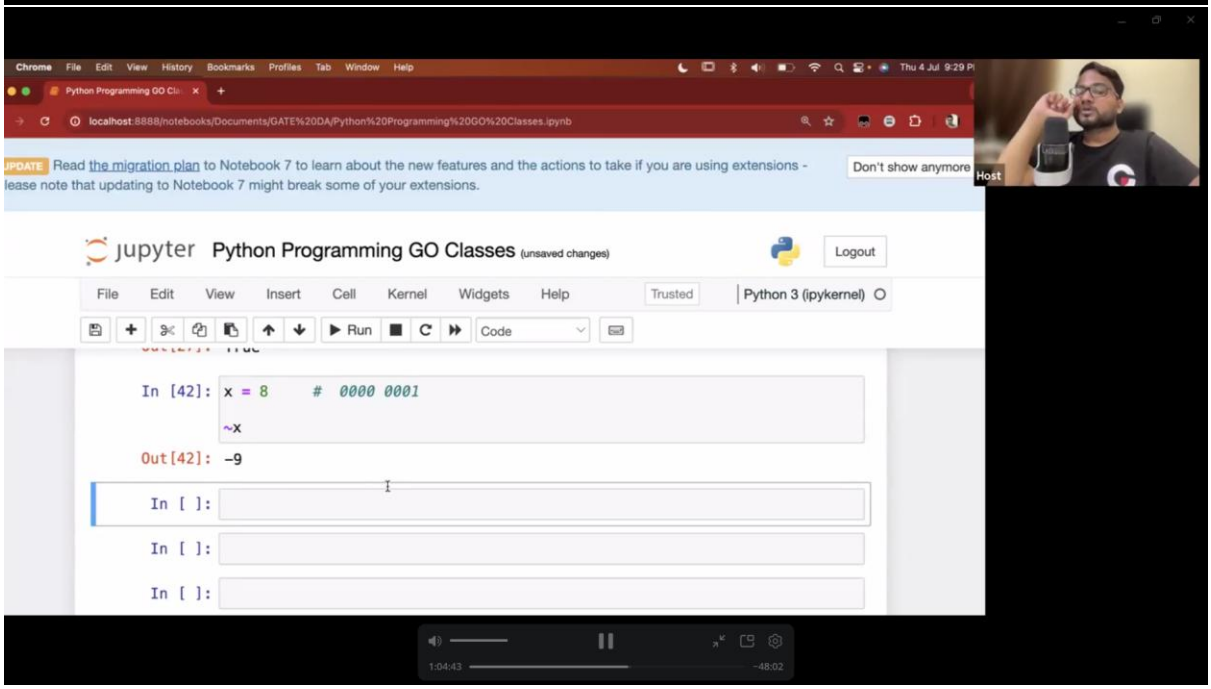

GO CLASSES
Left shift
suppose x is 0100 1001 1100 1011
x << 3 = 0100 1110 0101 1000
www.goclasses.in
57:44
-55:01
Host

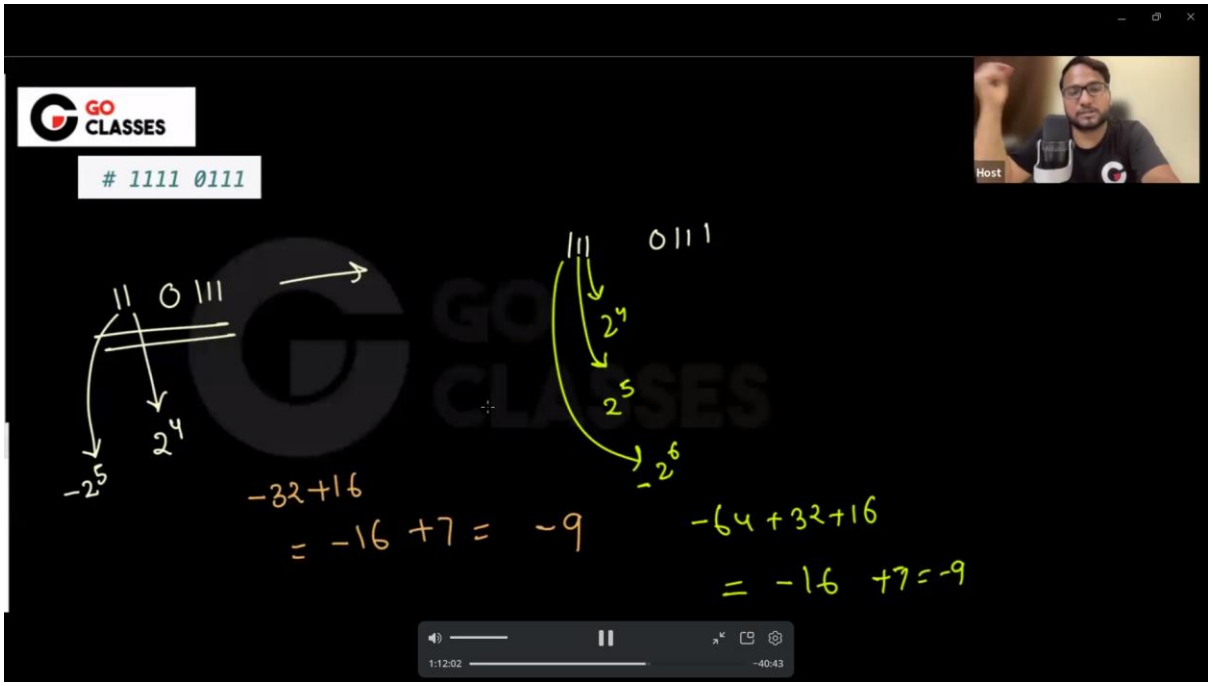

GO CLASSES
In [36]: x = 5 # 0000 0101
x << 2 # 0001 0100
Out[36]: 20
5x2² = 20
Binary
Decimal
10 ≡ 2
100 ≡ 2x2 = 4
1000 ≡ 2x2² = 8
Host

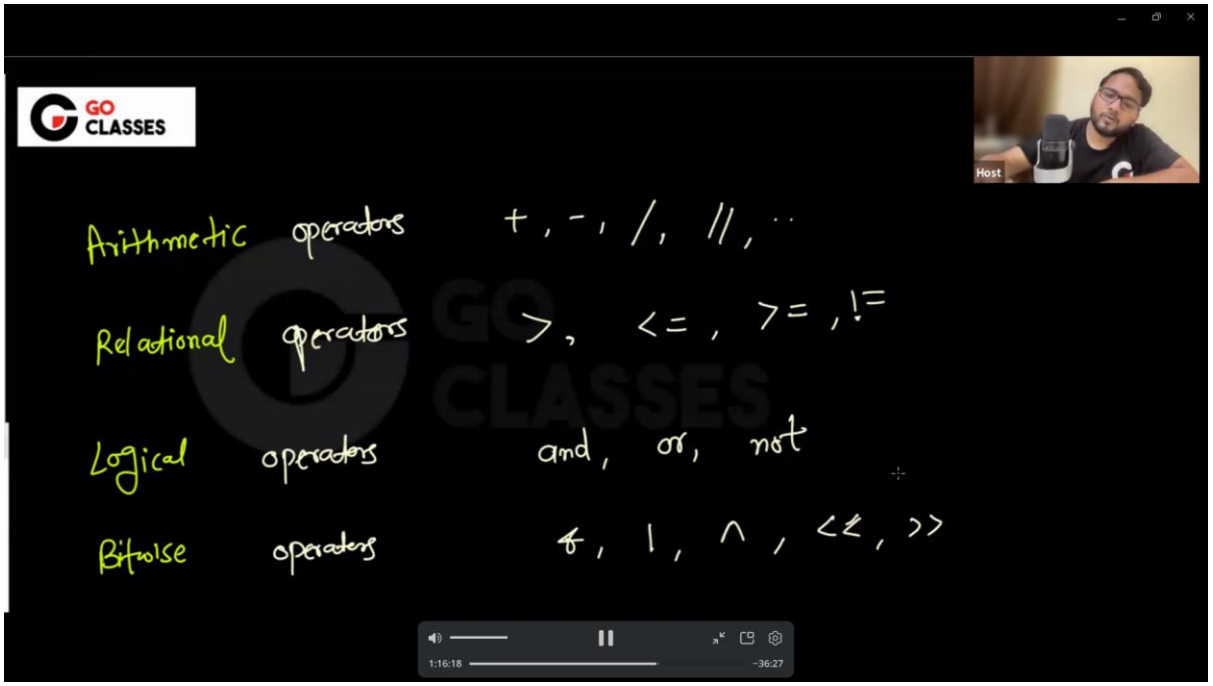

GO CLASSES
Host
Right Shift
suppose x is 0100 1001 1100 1011
x >> 3 = 0000 1001 0011 1001
www.goclasses.in
1:01:03
-51:42

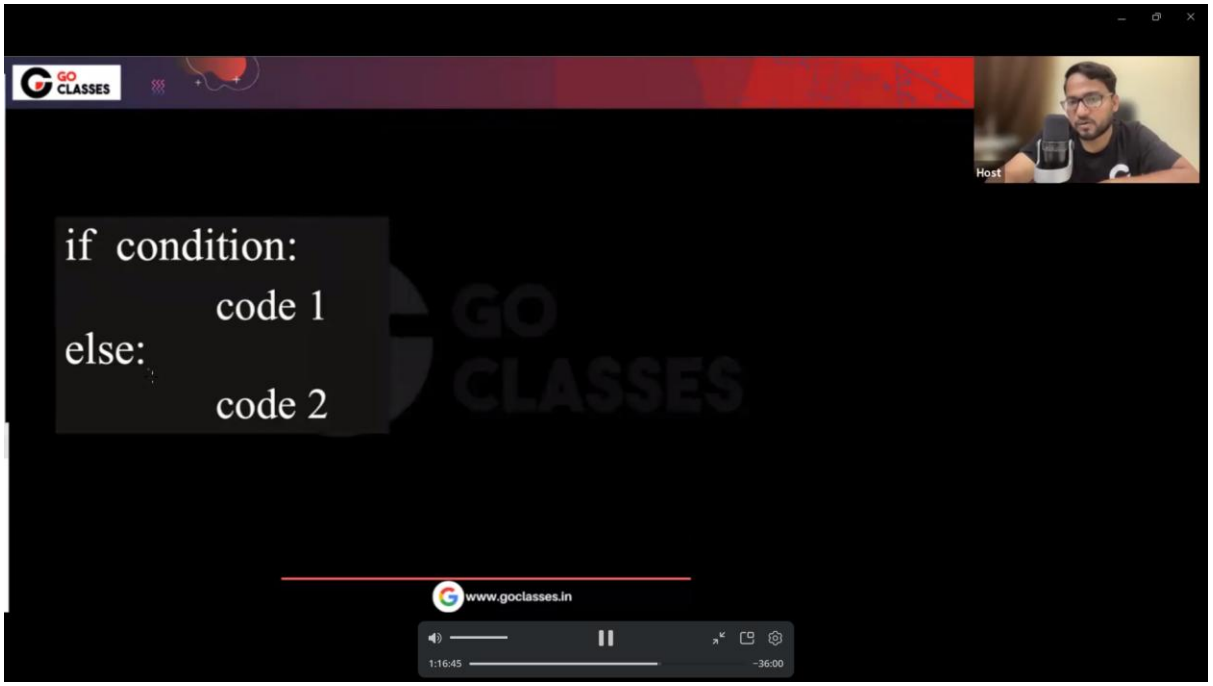

GO CLASSES
Host
0000 1010
y = 10
x = y << 1
print(x)
20
0000 1010
y = 10
x = y >> 1
print(x)
5
www.goclasses.in

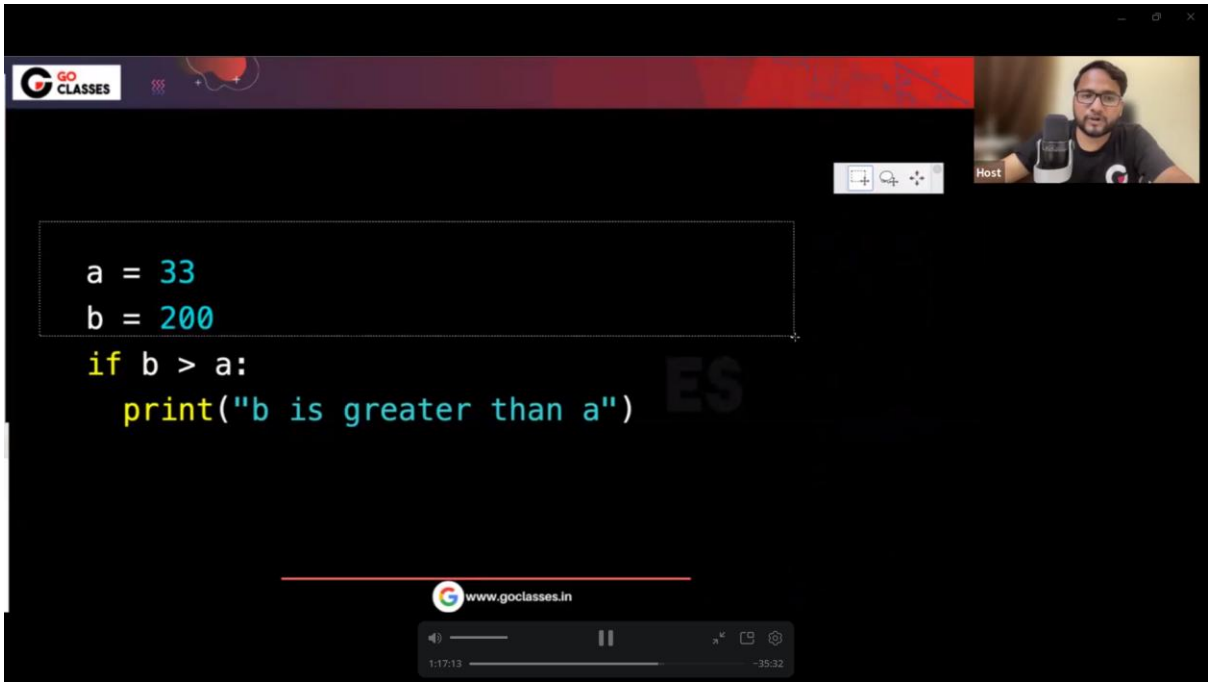

GO CLASSES
One's complement operator: ~
x = 1001 0110 1100 1011
~x = 0110 1001 0011 0100
www.goclasses.in
Host

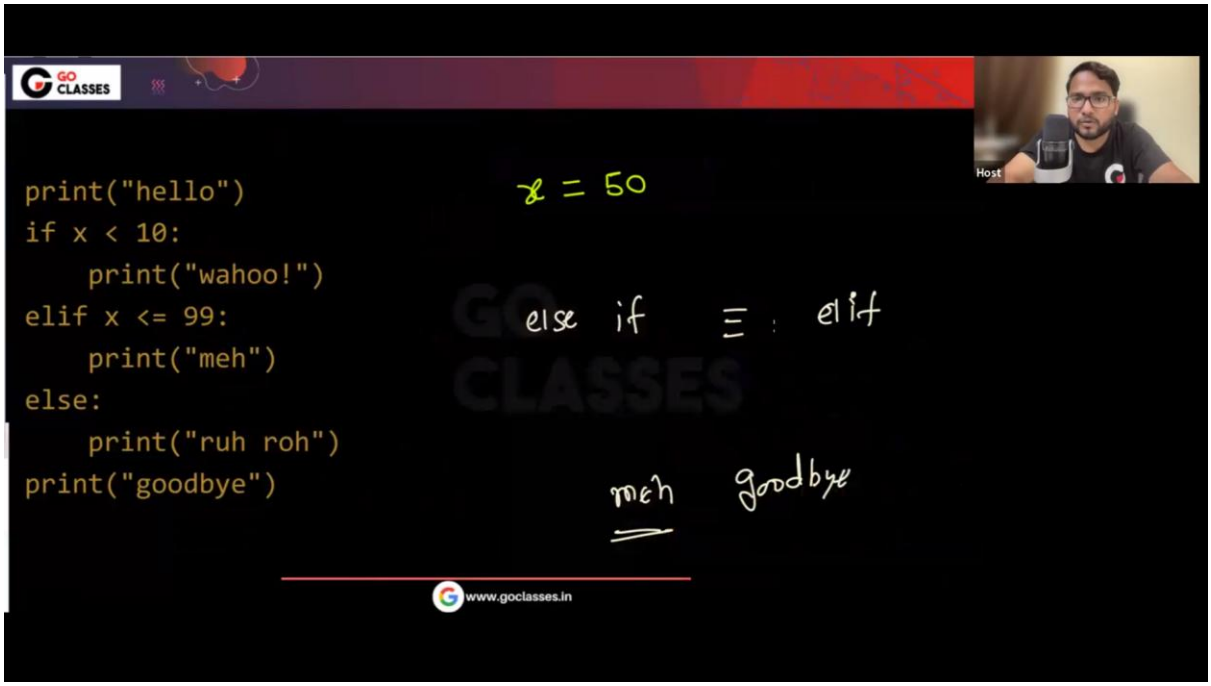

Chrome File Edit View History Bookmarks Profiles Tab Window Help
Python Programming GO Cla
localhost:8888/notebooks/Documents/GATE%20DA/Python%20Programming%20GO%20Classes.ipynb
Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - lease note that updating to Notebook 7 might break some of your extensions.
Don't show anymore
Host
jupyter Python Programming GO Classes (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [42]: x = 8 # 0000 0001
~x
Out[42]: -9
In [ ]:
In [ ]:
In [ ]:
1:04:43
-48:02

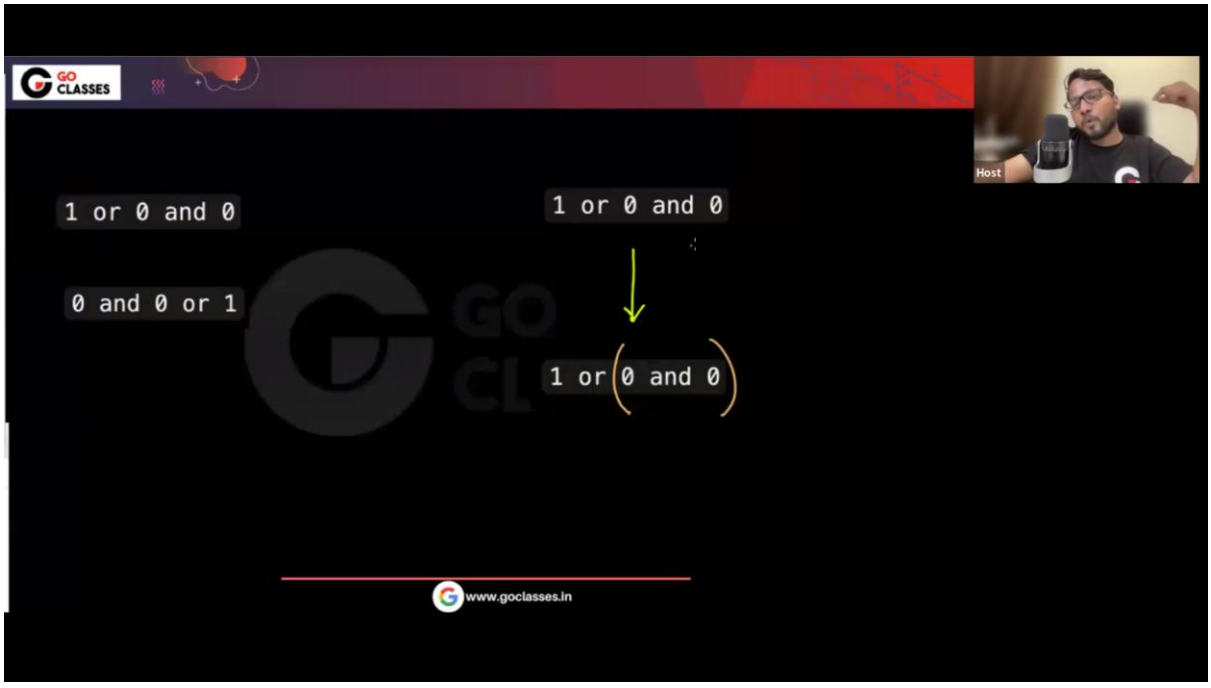

GO CLASSES
1111 0111
1111 0111
-2^5
2^4
2^4
2^5
-2^6
-32+16
= -16 +7 = -9
-64+32+16
= -16 +7 = -9
Host
1:12:02
-40:43


GO CLASSES
Arithmetic operators +, -, /, //, **
Relational operators >, <=, >=, !=
Logical operators and, or, not
Bitwise operators ~, |, ^, <<, >>
Host
1:16:18
-36:27

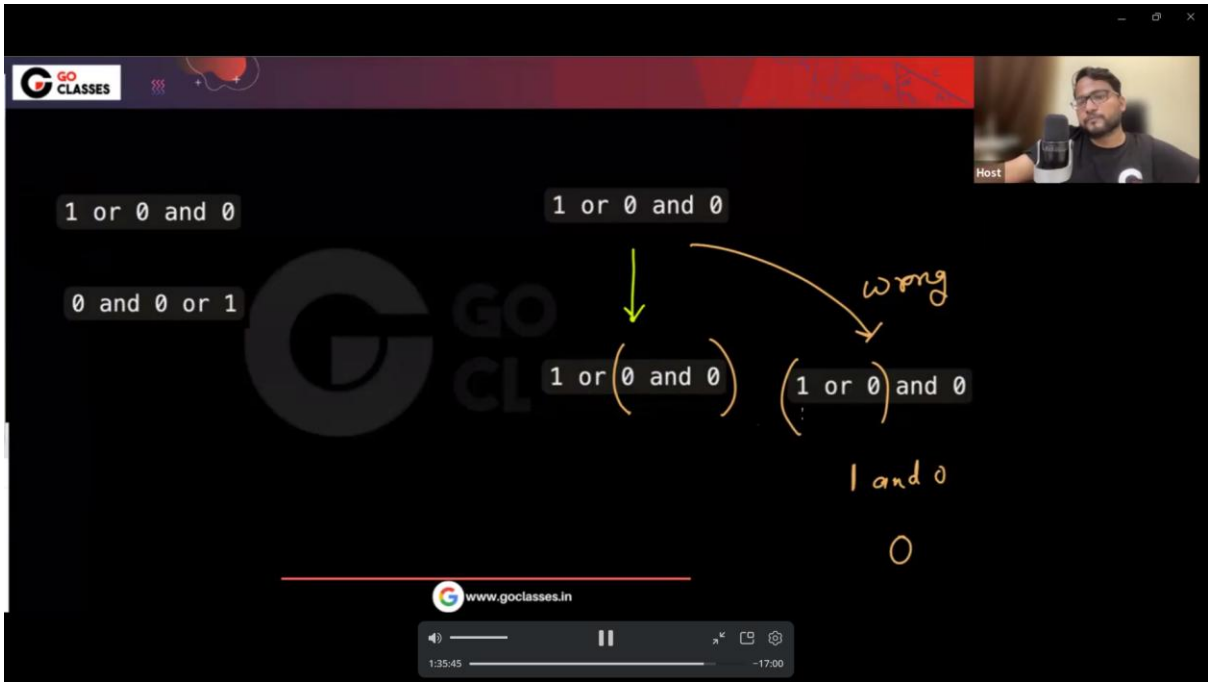

GO CLASSES
Host
if condition:
code 1
else:
code 2
GO CLASSES
www.goclasses.in
1:16:45
-36:00

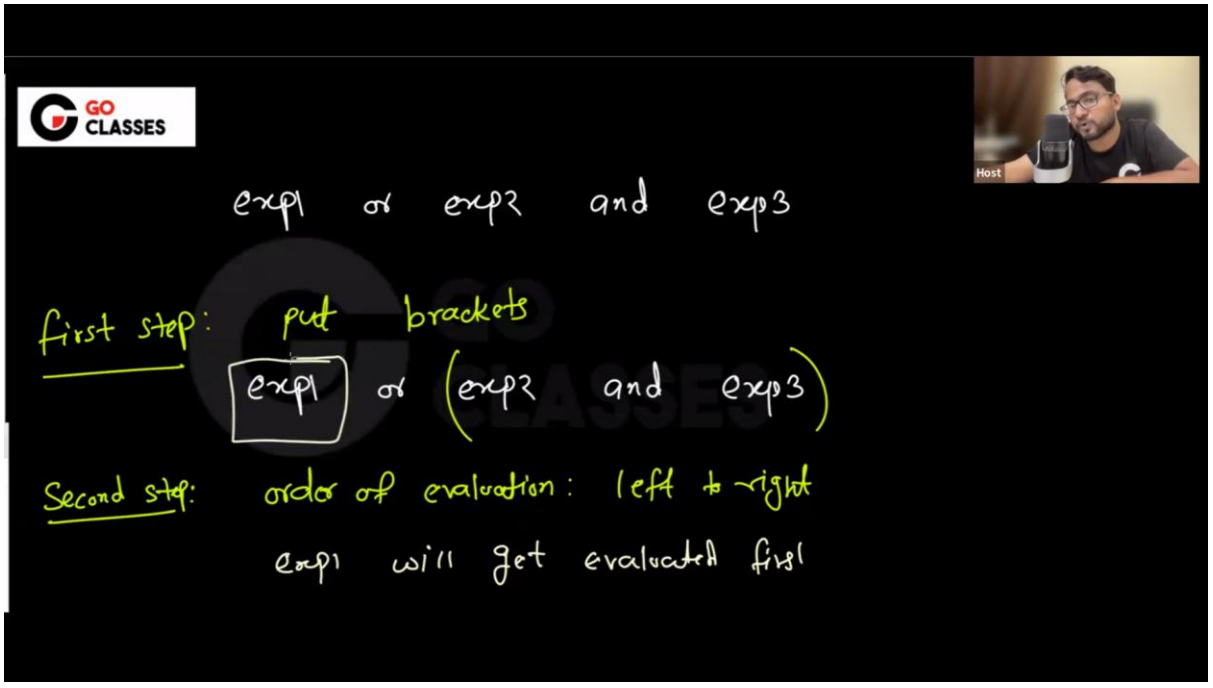

GO CLASSES
Host
a = 33
b = 200
if b > a:
print("b is greater than a")
www.goclasses.in
1:17:13
-35:32

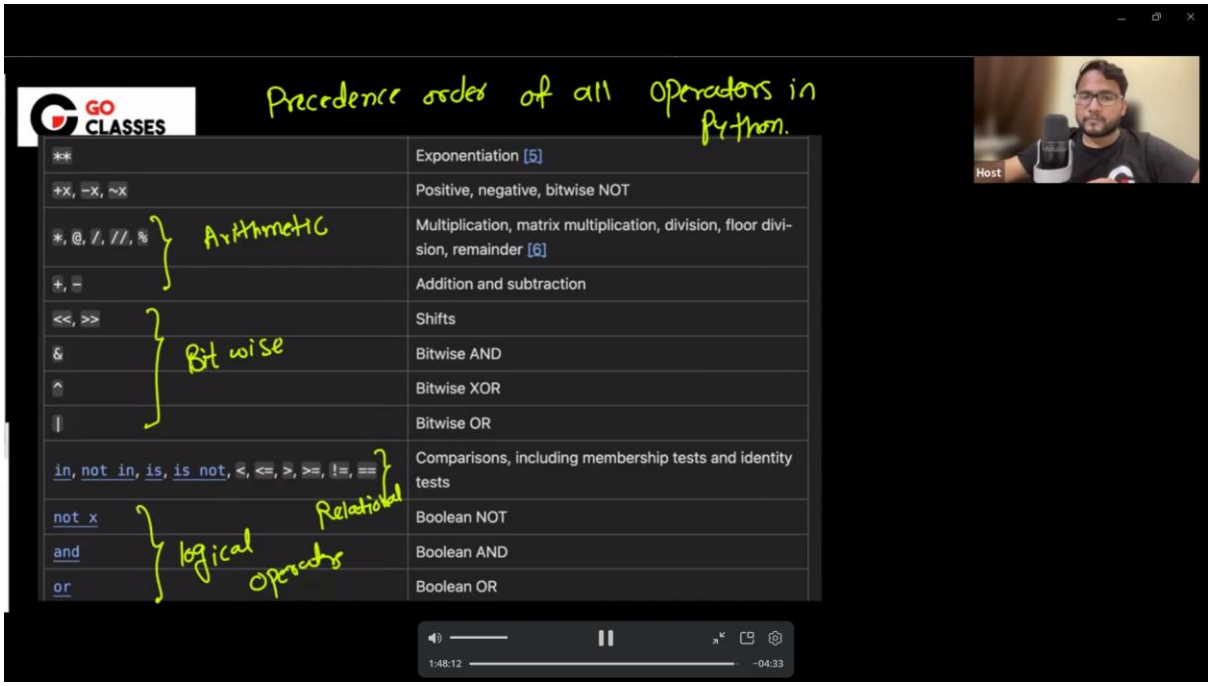

GO CLASSES
print("hello")
if x < 10:
print("wahoo!")
elif x <= 99:
print("meh")
else:
print("ruh roh")
print("goodbye")
x = 50
else if = elif
meh goodbye
www.goclasses.in
Host


GO CLASSES
1 or 0 and 0
0 and 0 or 1
1 or 0 and 0
1 or (0 and 0)
www.goclasses.in
Host

And > Or


GO CLASSES
Host
1 or 0 and 0
0 and 0 or 1
1 or 0 and 0
wrong
1 or 0 and 0
(1 or 0) and 0
1 and 0
0
www.goclasses.in
1:35:45
-17:00


GO CLASSES
Host
exp1 or exp2 and exp3
first step: put brackets
exp1 or (exp2 and exp3)
Second step: order of evaluation: left to right
exp1 will get evaluated first



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