- Add r4,r2,r3
- Pc-→MAR
- Issue read
- Wait for the requested word to MDR
- Mdr→IR
- R4,R2→ALU
- ADDDITION OPERATION
- ALU→R3
- RESULT IN MDR
- MAR
- PC

You

- There are many ways to connect different parts inside a computer together.
- A group of lines that serves as a connecting path for several devices is called a *bus*.
- System bus-connects major computer components.

  Address Bus-unidirectional

  Data Bus-bidirectional

  Control Bus-regulates the activity sending control signals like memory read,memory write,I/O read I/O write,interrupt request,reset so on.

Keywords

- Keywords are a list of reserved words that have predefined meaning.
- Keywords are special vocabulary and cannot be used by programmers as identifiers for variables, functions, constants or with any identifier name.
- Attempting to use a keyword as an identifier name will cause an error.
- There 33 Keywords -
  - contains only alphabets
  - Except True, False, None- all contains only lowercase alphabet
  - Switch concept is not there in python
  - do-while is not there
  - Int, float, complex .. keywords are not there in python

REC

You

REC

List of Keywords in Python

and	as	not
assert	finally	or
break	for	pass
class	from	nonlocal
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield
False	True	None

You

REC

Statements and Expressions

- A statement is an instruction that the Python interpreter can execute. Python program consists of a sequence of statements.

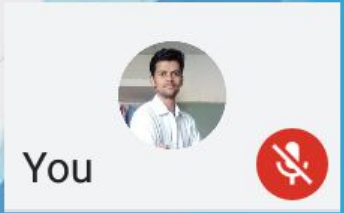
Ex: z = 1 is an assignment statement.

- Expression is an arrangement of values and operators which are evaluated to make a new value. Expressions are statements as well.

```
EX:  >>> 20  or   >>> z   or   >>> z + 20
```

- A value is the representation of some entity like a letter or a number that can be manipulated by a program.

```
>>> 8 + 2

      10
```



REC

Variables

- Variable is a named placeholder to hold any type of data which the program can use to assign and modify during the course of execution.
- In Python, there is no need to declare a variable explicitly by specifying whether the variable is an integer or a float or any other type.
- *To define a new variable in Python, we simply assign a value to a name.*
- If a need for variable arises you need to think of a variable name based on the rules mentioned in the following subsection and use it in the program.

REC

gal Variable Names

Follow the below-mentioned rules for creating legal variable names in Python.

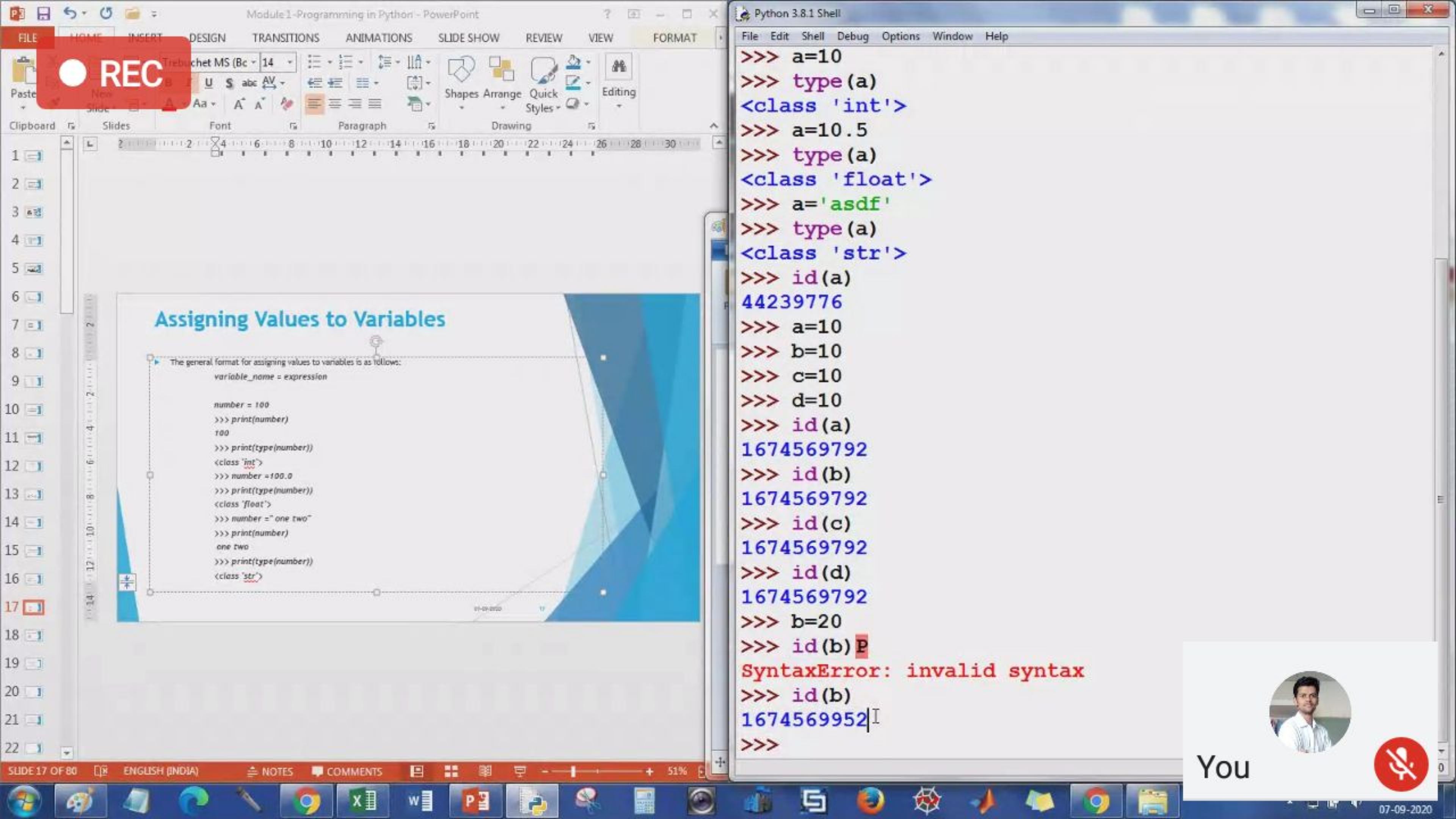
- • Variable names can consist of any number of letters, underscores and digits.
- • Variable should not start with a number.
- • Python Keywords are not allowed as variable names.
- • Variable names are case-sensitive. For example, computer and Computer are different variables.
- Ensure variable names are descriptive and clear enough. This allows other programmers to have an idea about what the variable is representing.

You

Assigning Values to Variables

- The general format for assigning values to variables is as follows:

```
variable_name = expression

number = 100
>>> print(number)
100
>>> print(type(number))
<class 'int'>
>>> number =100.0
>>> print(type(number))
<class 'float'>
>>> number =" one two"
>>> print(number)
 one two
>>> print(type(number))
<class 'str'>
```


REC
Module1-Programming in Python - PowerPoint
FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW FORMAT
SLIDE 17 OF 80 ENGLISH (INDIA) NOTES COMMENTS 51%
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
You
07-09-2020

Assigning Values to Variables

- The general format for assigning values to variables is as follows:

```
variable_name = expression

number = 100
>>> print(number)
100
>>> print(type(number))
<class 'int'>
>>> number =100.0
>>> print(type(number))
<class 'float'>
>>> number =" one two"
>>> print(number)
 one two
>>> print(type(number))
<class 'str'>
```

```
>>> a=10
>>> type(a)
<class 'int'>
>>> a=10.5
>>> type(a)
<class 'float'>
>>> a='asdf'
>>> type(a)
<class 'str'>
>>> id(a)
44239776
>>> a=10
>>> b=10
>>> c=10
>>> d=10
>>> id(a)
1674569792
>>> id(b)
1674569792
>>> id(c)
1674569792
>>> id(d)
1674569792
>>> b=20
>>> id(b)P
SyntaxError: invalid syntax
>>> id(b)
1674569952
>>> 
```

REC

List of Arithmetic Operators

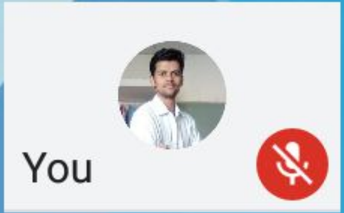
Operator	Operator Name	Description	Example
+	Addition operator	Adds two operands, producing their sum.	p + q = 5
–	Subtraction operator	Subtracts the two operands, producing their difference.	p – q = –1
*	Multiplication operator	Produces the product of the operands.	p * q = 6
/	Division operator	Produces the quotient of its operands where the left operand is the dividend and the right operand is the divisor.	q / p = 1.5
%	Modulus operator	Divides left hand operand by right hand operand and returns a remainder.	q % p = 1
**	Exponent operator	Performs exponential (power) calculation on operators.	p**q = 8
//	Floor division operator	Returns the integral part of the quotient.	9//2 = 4 and

Note: The value of p is 2 and q is 3.

Assignment Operators

- Assignment operators are used for assigning the values generated after evaluating the right operand to the left operand. Assignment operation always works from right to left.
- Assignment operators are either simple assignment operator or compound assignment operators. Simple assignment is done with the equal sign (=) and simply assigns the value of its right operand to the variable on the left.

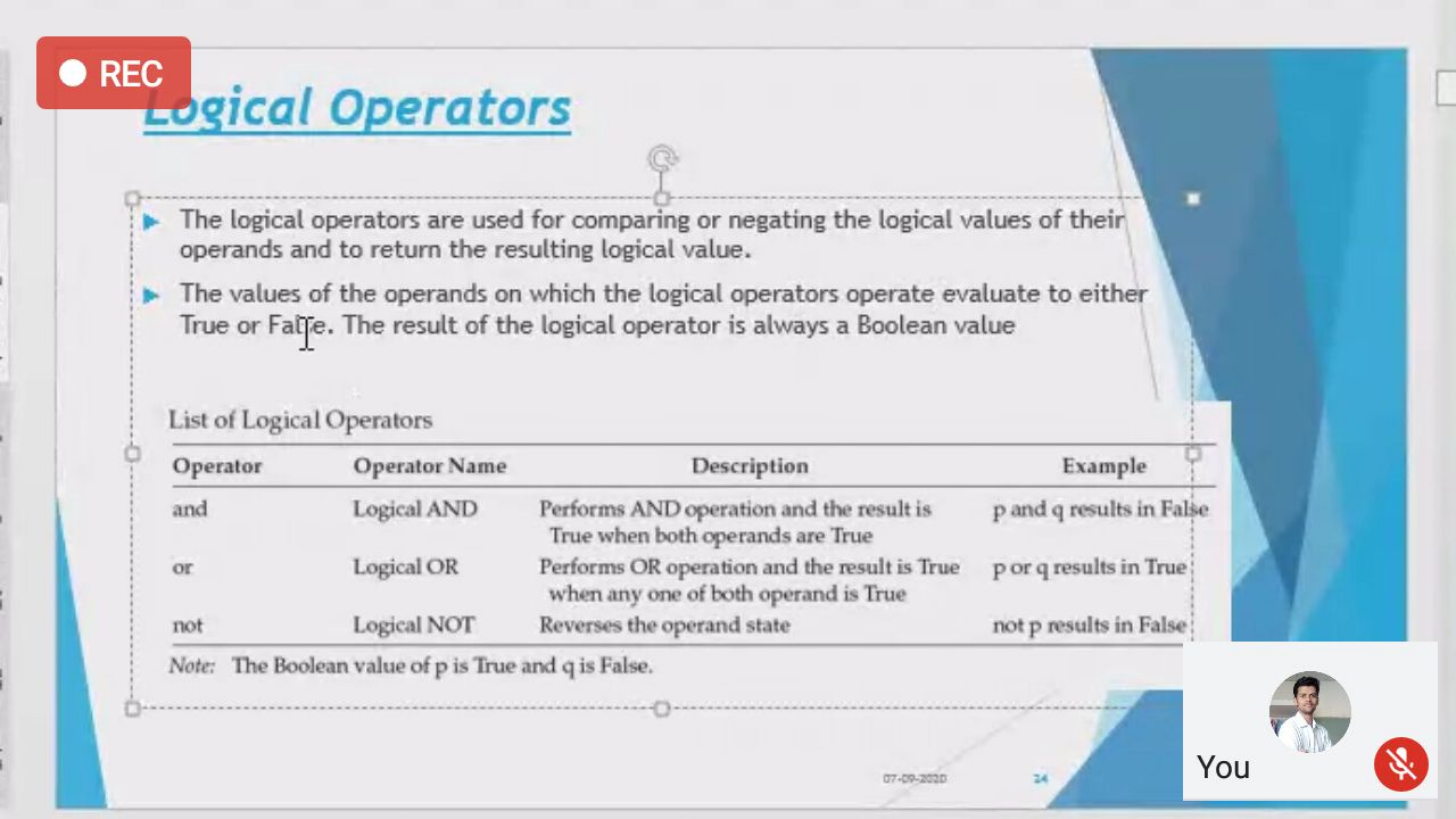
```
>>> x=5

>>> x=x+1

>>> x

6

>>> x+=1

>>> x

7
```



List of Assignment Operators

Operator	Operator Name	Description	Example
=	Assignment	Assigns values from right side operands to left side operand.	z = p + q assigns value of p + q to z
+=	Addition Assignment	Adds the value of right operand to the left operand and assigns the result to left operand.	z += p is equivalent to z = z + p
−=	Subtraction Assignment	Subtracts the value of right operand from the left operand and assigns the result to left operand.	z −= p is equivalent to z = z − p
*=	Multiplication Assignment	Multiplies the value of right operand with the left operand and assigns the result to left operand.	z *= p is equivalent to z = z * p
/=	Division Assignment	Divides the value of right operand with the left operand and assigns the result to left operand.	z /= p is equivalent to z = z / p
**=	Exponentiation Assignment	Evaluates to the result of raising the first operand to the power of the second operand.	z**= p is equivalent to z = z ** p
//=	Floor Division Assignment	Produces the integral part of the quotient of its operands where the left operand is the dividend and the right operand is the divisor.	z //= p is equivalent to z = z // p
%=	Remainder Assignment	Computes the remainder after division and assigns the value to the left operand.	z %= p is equ z = z % p

ython programming language doesn't support Auto increment (+ +) and Auto decrement (-


REC

Logical Operators

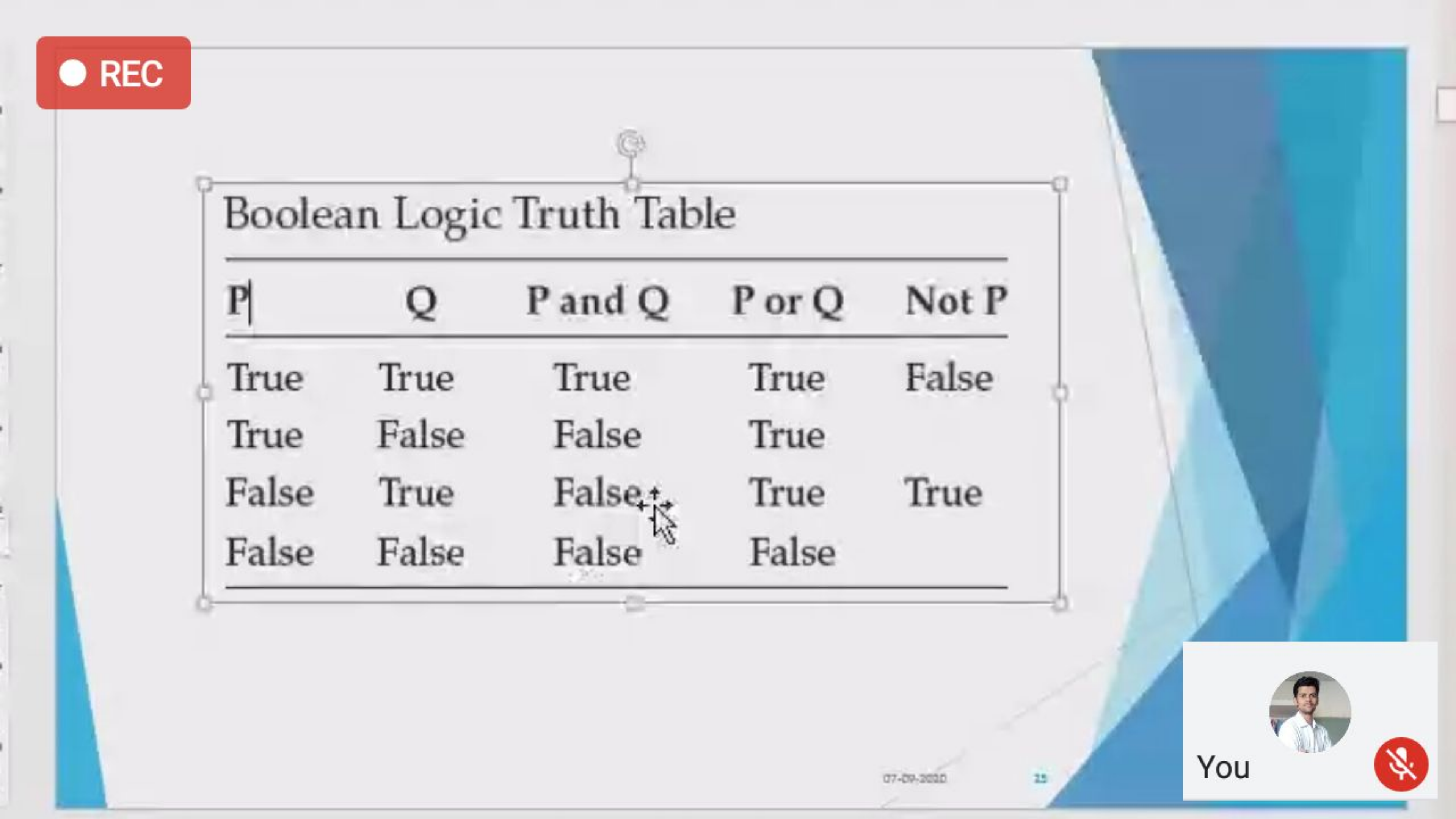
- The logical operators are used for comparing or negating the logical values of their operands and to return the resulting logical value.
- The values of the operands on which the logical operators operate evaluate to either True or False. The result of the logical operator is always a Boolean value

List of Logical Operators

Operator	Operator Name	Description	Example
and	Logical AND	Performs AND operation and the result is True when both operands are True	p and q results in False
or	Logical OR	Performs OR operation and the result is True when any one of both operand is True	p or q results in True
not	Logical NOT	Reverses the operand state	not p results in False

Note: The Boolean value of p is True and q is False.

You


REC

Boolean Logic Truth Table

P	Q	P and Q	P or Q	Not P
True	True	True	True	False
True	False	False	True	
False	True	False	True	True
False	False	False	False	

You

REC

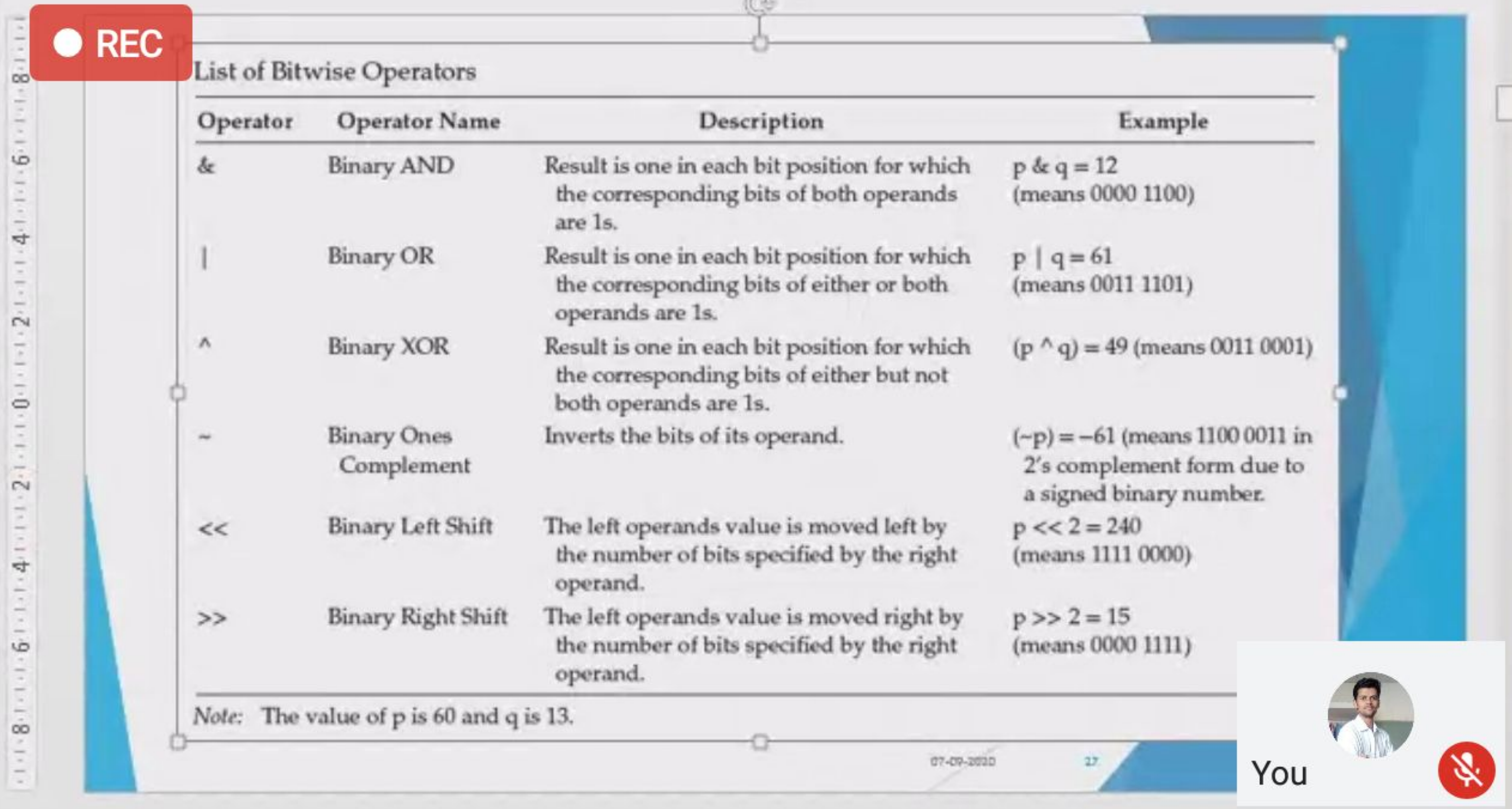
Bitwise Operators

- Bitwise operators treat their operands as a sequence of bits (zeroes and ones) and perform bit by bit operation. For example, the decimal number ten has a binary representation of 1010. Bitwise operators perform their operations on such binary representations, but they return standard Python numerical values.

Bitwise Truth Table

P	Q	P & Q	P | Q	P ^ Q	~ P
0	0	0	0	0	1
0	1	0	1	1	
1	0	0	1	1	0
1	1	1	1	0	

You



List of Bitwise Operators

Operator	Operator Name	Description	Example
&	Binary AND	Result is one in each bit position for which the corresponding bits of both operands are 1s.	p & q = 12 (means 0000 1100)
|	Binary OR	Result is one in each bit position for which the corresponding bits of either or both operands are 1s.	p | q = 61 (means 0011 1101)
^	Binary XOR	Result is one in each bit position for which the corresponding bits of either but not both operands are 1s.	(p ^ q) = 49 (means 0011 0001)
~	Binary Ones Complement	Inverts the bits of its operand.	(~p) = −61 (means 1100 0011 in 2's complement form due to a signed binary number.
<<	Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	p << 2 = 240 (means 1111 0000)
>>	Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	p >> 2 = 15 (means 0000 1111)

Note: The value of p is 60 and q is 13.

REC

Precedence and Associativity

- Operator precedence determines the way in which operators are parsed with respect to each other. Operators with higher precedence become the operands of operators with lower precedence.
- Associativity determines the way in which operators of the same precedence are parsed. Almost all the operators have left-to-right associativity.
- `>>> 2 + 3 * 6`
- `>>> (2 + 3) * 6`
- `>>> 6 * 4 / 2`

You