unacademy

Operators and Control Statements

Course on Data Structure and Algorithms Using Python

Ankush Saklecha • Lesson 2 • Dec 9, 2023

Python Programming

Operators

Operators

- Operators are used to perform several operations on variables values and constants.
- Operators are defined between operands.

Examples:
a=b
a+b
a>=b
a and b

Types of Operator

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Arithmetic operators

+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

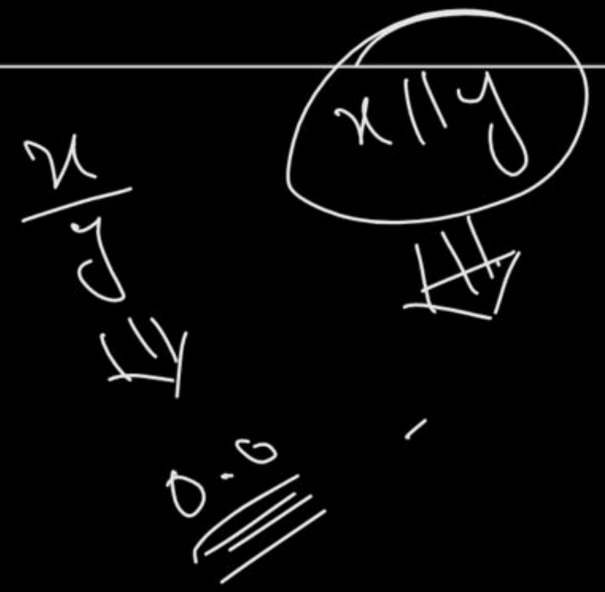
unacademy

Guess Output?

x=30.5
y=4
print(x/y) → 7.625
print(x%y) → 2.5
print(x//y) → 7

x=0
y=4
print(x/y) → 0/4 ⇒ 0.0
print(x%y) → 0
print(x//y) → 0

x=30
y=0
print(x/y) → Error
print(x%y) → Error



Let's Try…

Guess Output?

```
x=30.5
y=4
print(x/y)
print(x%y)
print(x//y)
```

7.625

2.5

7

```
x=0
y=4
print(x/y)
print(x%y)
print(x//y)
```

0.0

0

0

```
x=30
y=0
print(x/y)
print(x%y)
```

Error

Error

Assignment Operators

	operation → meaning
=	x = 5
+=	x += 3
-=	x -= 3
*=	x *= 3
/=	x /= 3
%=	x %= 3
//=	x //= 3
**=	x **= 3
&=	x &= 3
|=	x |= 3
^=	x ^= 3

meaning
x = 5
x = x + 3
x = x - 3
x = x * 3
x = x / 3
x = x % 3
x = x // 3
x = x ** 3
x = x & 3
x = x | 3
x = x ^ 3

bitwise

and

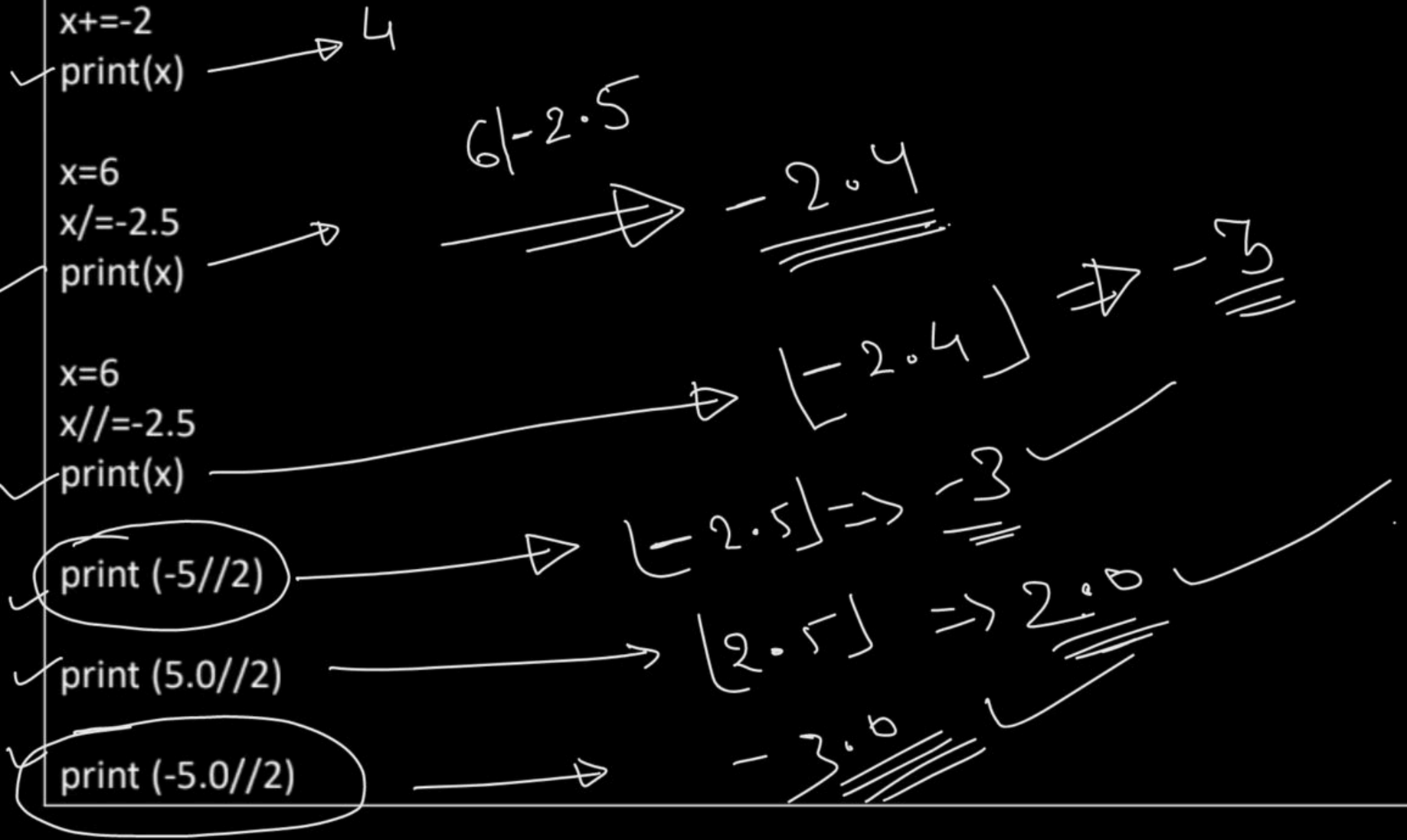
or

or

xor

Guess Output?

```
x=6
x+=-2
print(x)

x=6
x/=-2.5
print(x)

x=6
x//=-2.5
print(x)

print (-5//2)

print (5.0//2)

print (-5.0//2)
```



Let's Try…

Guess Output?

```
x=6
x+=-2
print(x)
```

4

```
x=6
x/=-2.5
print(x)
```

-2.4

```
x=6
x//=-2.5
print(x)
```

-3.0

```
print (-5//2)
```

-3

```
print (5.0//2)
```

2.0

```
print (-5.0//2)
```

-3.0

Comparison Operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical Operators

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

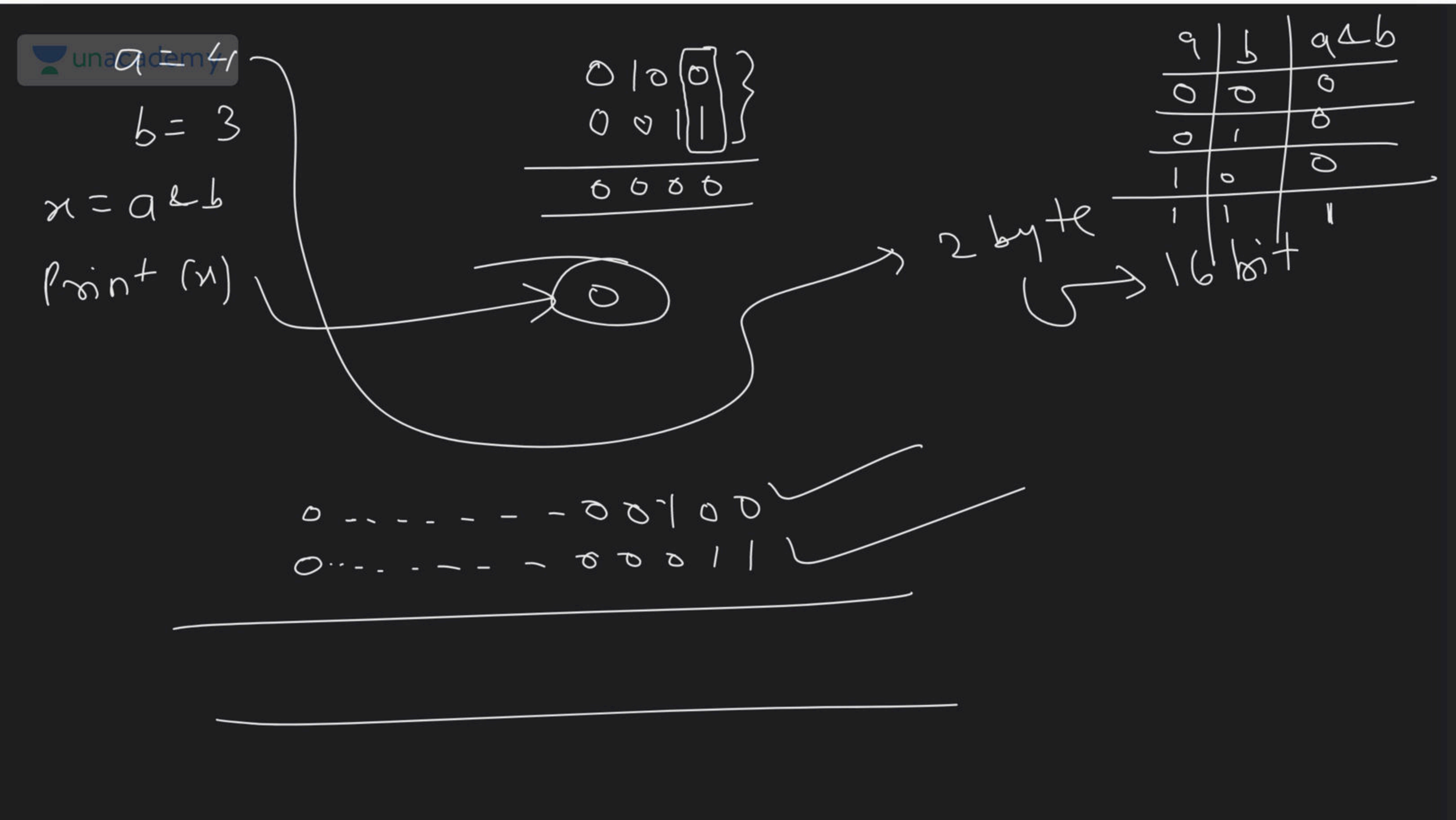
T and T

not (True)

→ False

Bitwise Operators

&	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


a = 4
b = 3
x = a & b
Print (x)
0 1 0 0
0 0 1 1
0 0 0 0
0
2 byte
16 bit
a b a&b
0 0 0
0 1 0
1 0 0
1 1 1
0 ..... 0 0 0 0 0
0 ..... 0 0 0 1 1

unacademy

OR

a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

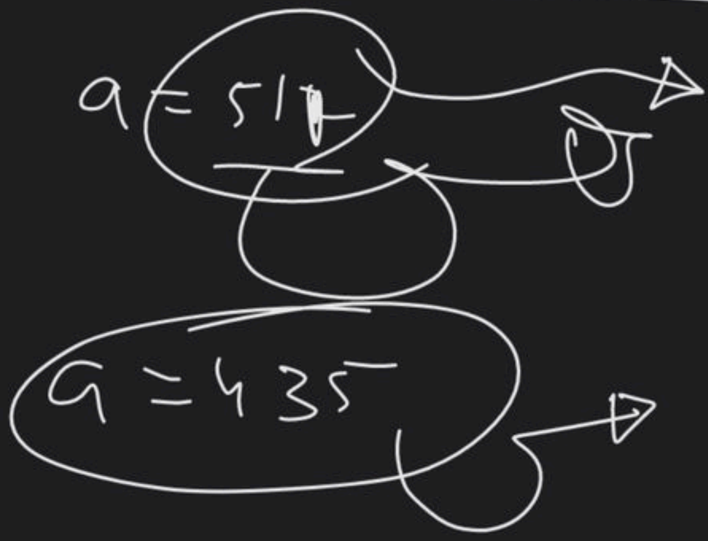
$a = 4$.

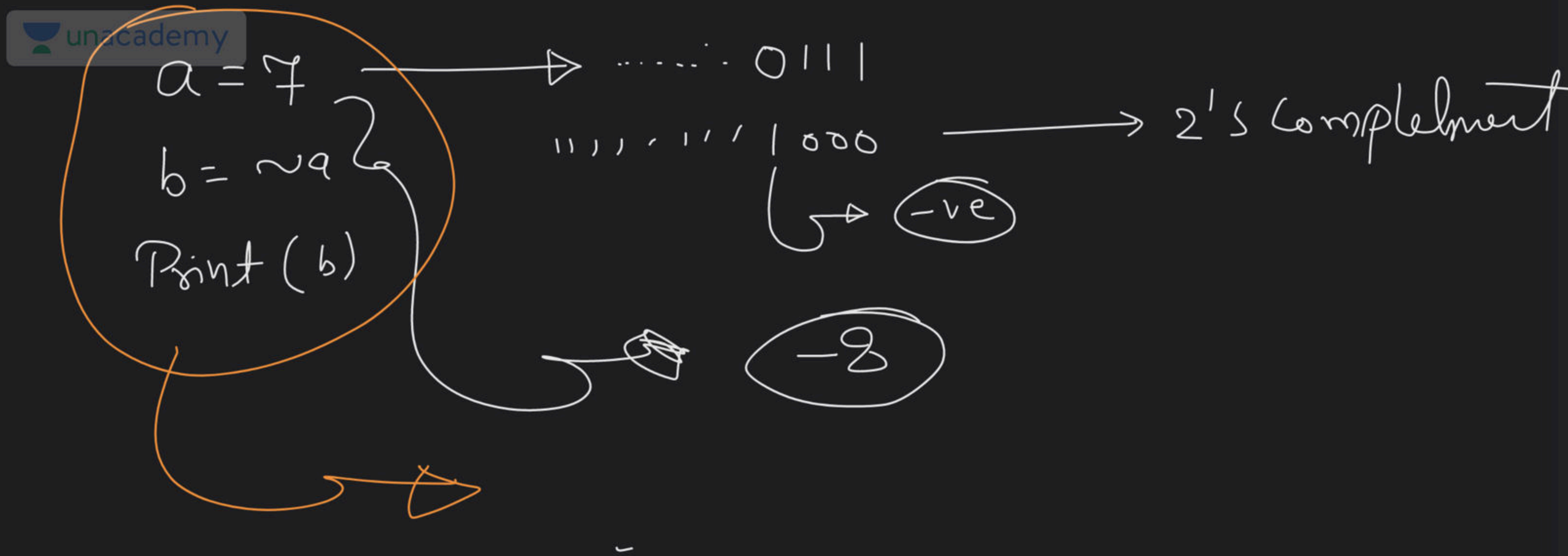
$b = 3$.

$$\begin{array}{r} 0100 \\ 0011 \\ \hline 0111 \end{array}$$

XOR (^)

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0




a = 7
b = ~a
Print (b)
0111
1000
2's Complement
-ve
-8

Our system always works in 2's complement form.

int ⇒ 2 byte ⇒ -32768 to 32767

2 byte ⇒ 16 bit

$\{-(2^{n-1})$ to $+(2^{n-1}-1)\}$

-2^{16-1} to $+(2^{16}-1)$

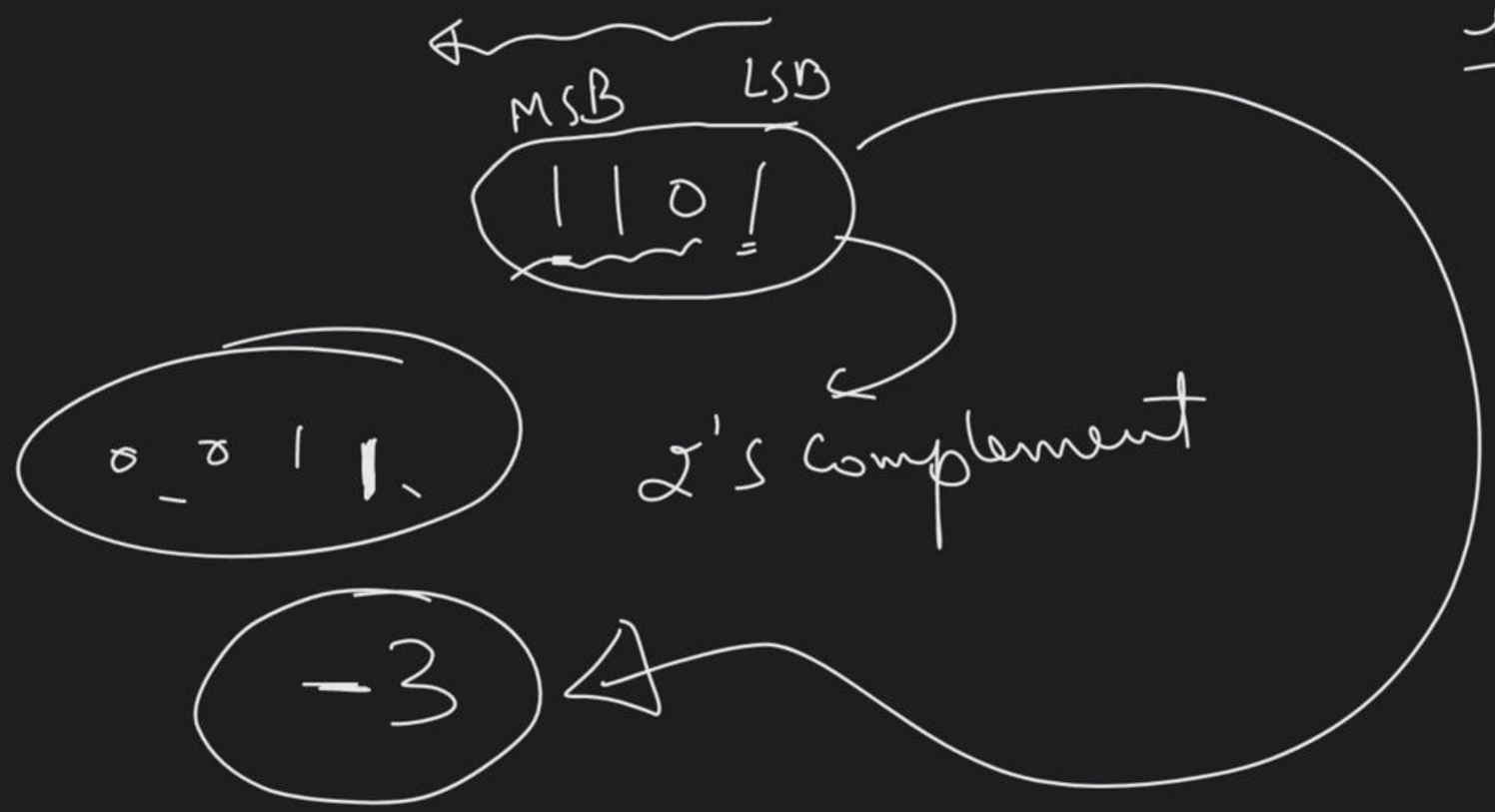
4 bit

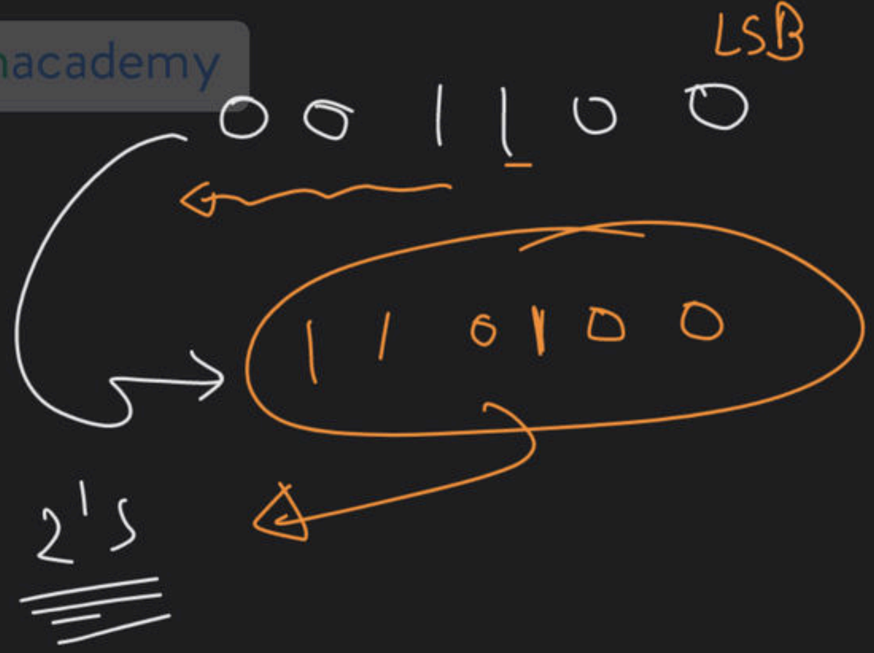
5 ⇒ 0101

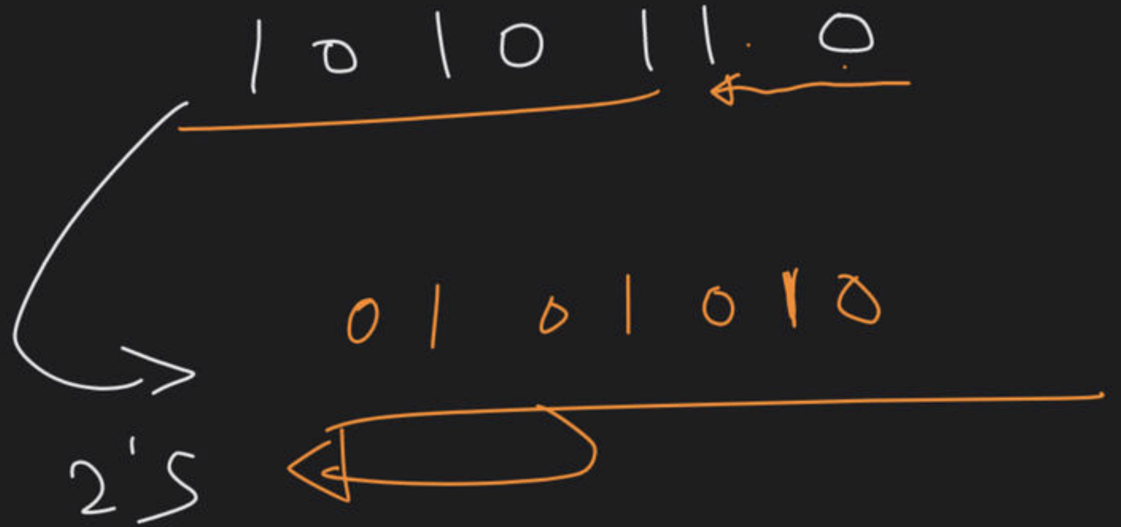
MSB

0 ⇒ +ve

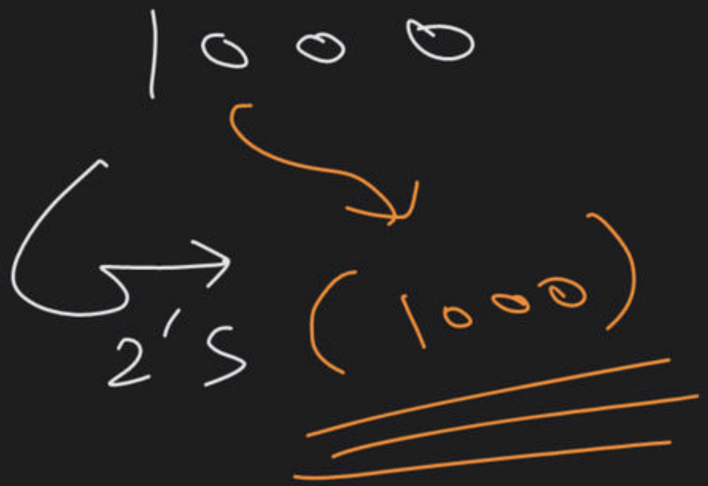
1 ⇒ -ve.



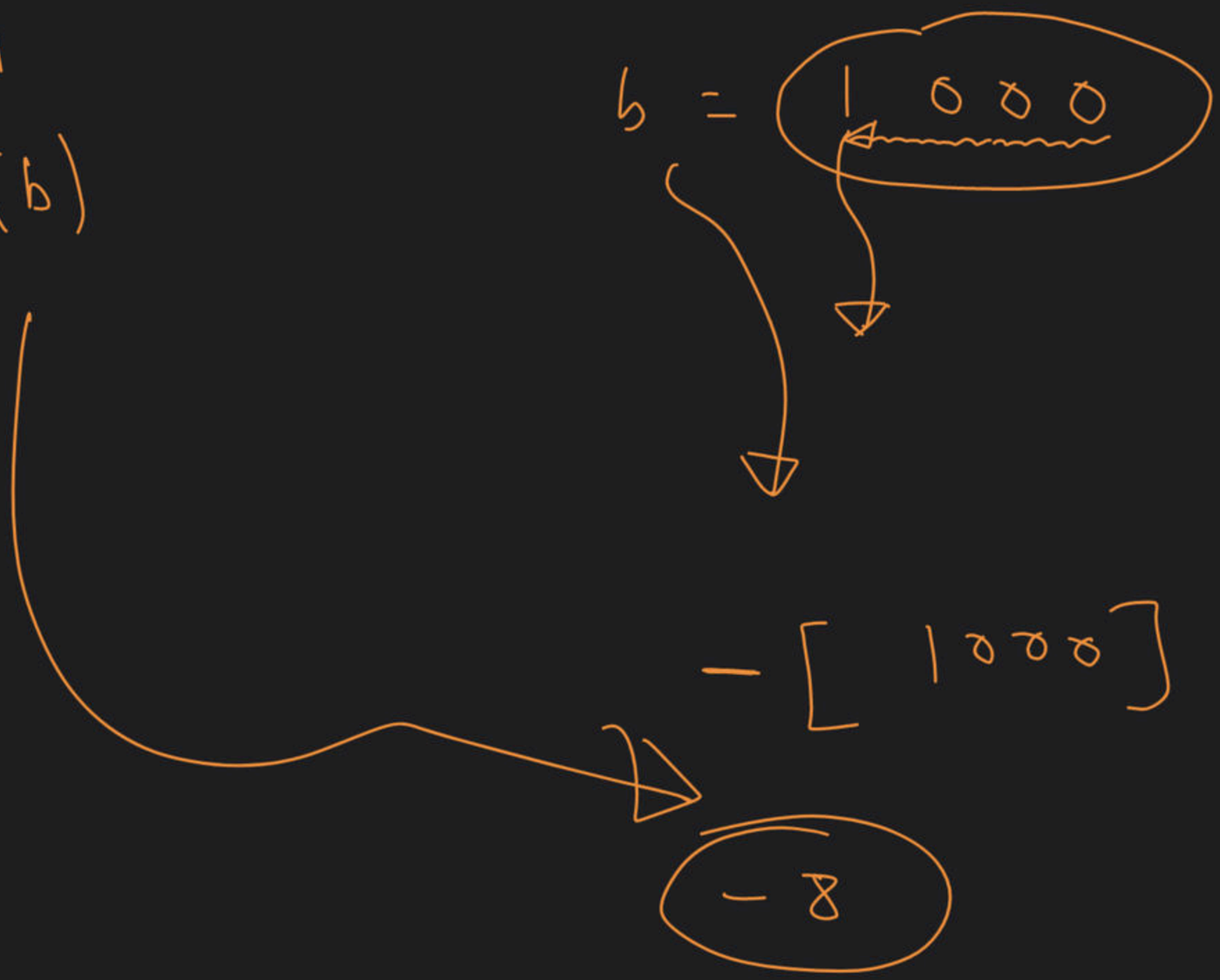

LSB
0 0 1 1 0 0
1 1 0 1 0 0
2's


1 0 1 0 1 1 0
0 1 0 1 0 1 0
2's


1 0 0 0 0
2's (10000)


1 0 0 0
2's (1000)

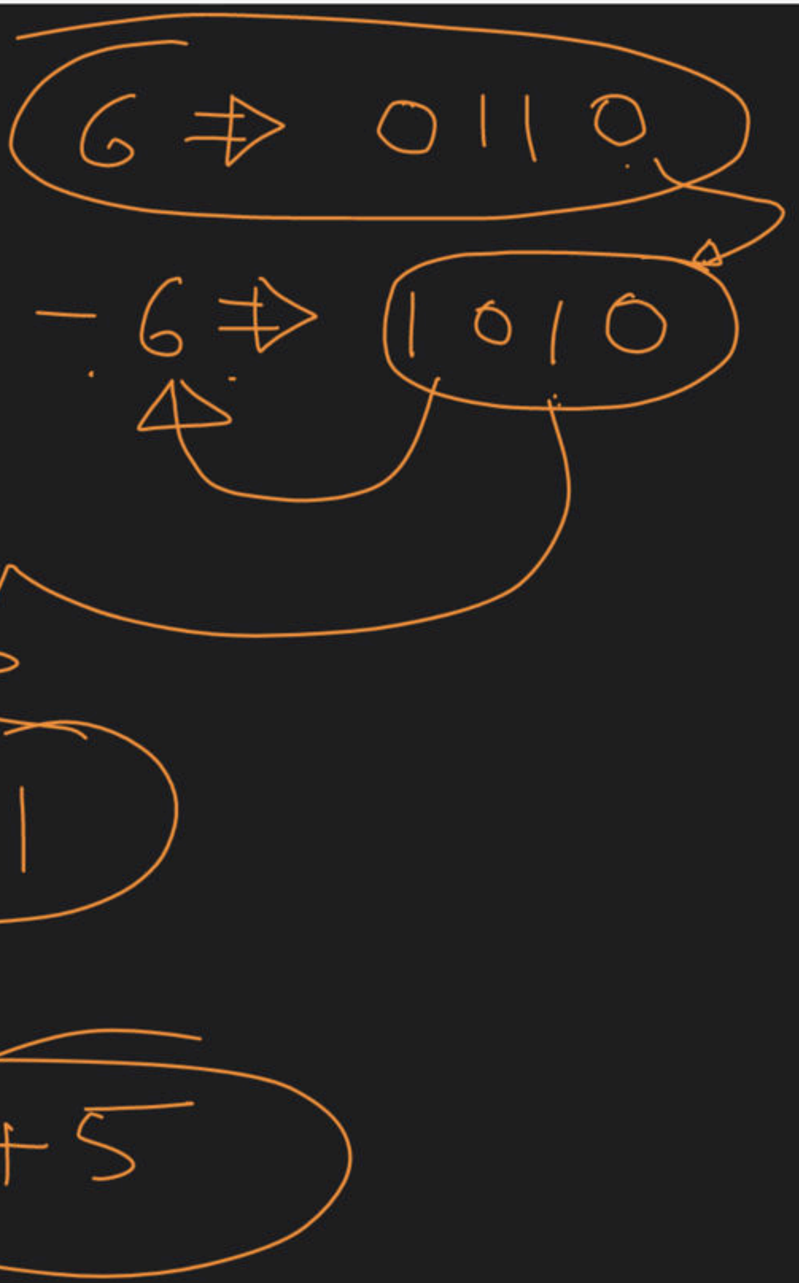
```
a=7
b=~a
Print(b)
```



a = -6

b = ~a

Print(b)



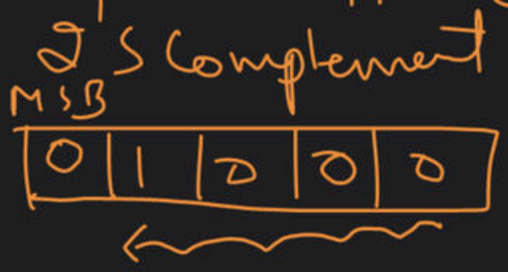
0
MSB
↳
+ve

1 ↳ -ve ≡ 2's Comple

unacademy

n = 5 bit

a = 8 ⇒ 2's Complement



a = -8 ⇒

⇒ − (01000) ⇒ -8

b = 7 ⇒

b = -7 ⇒

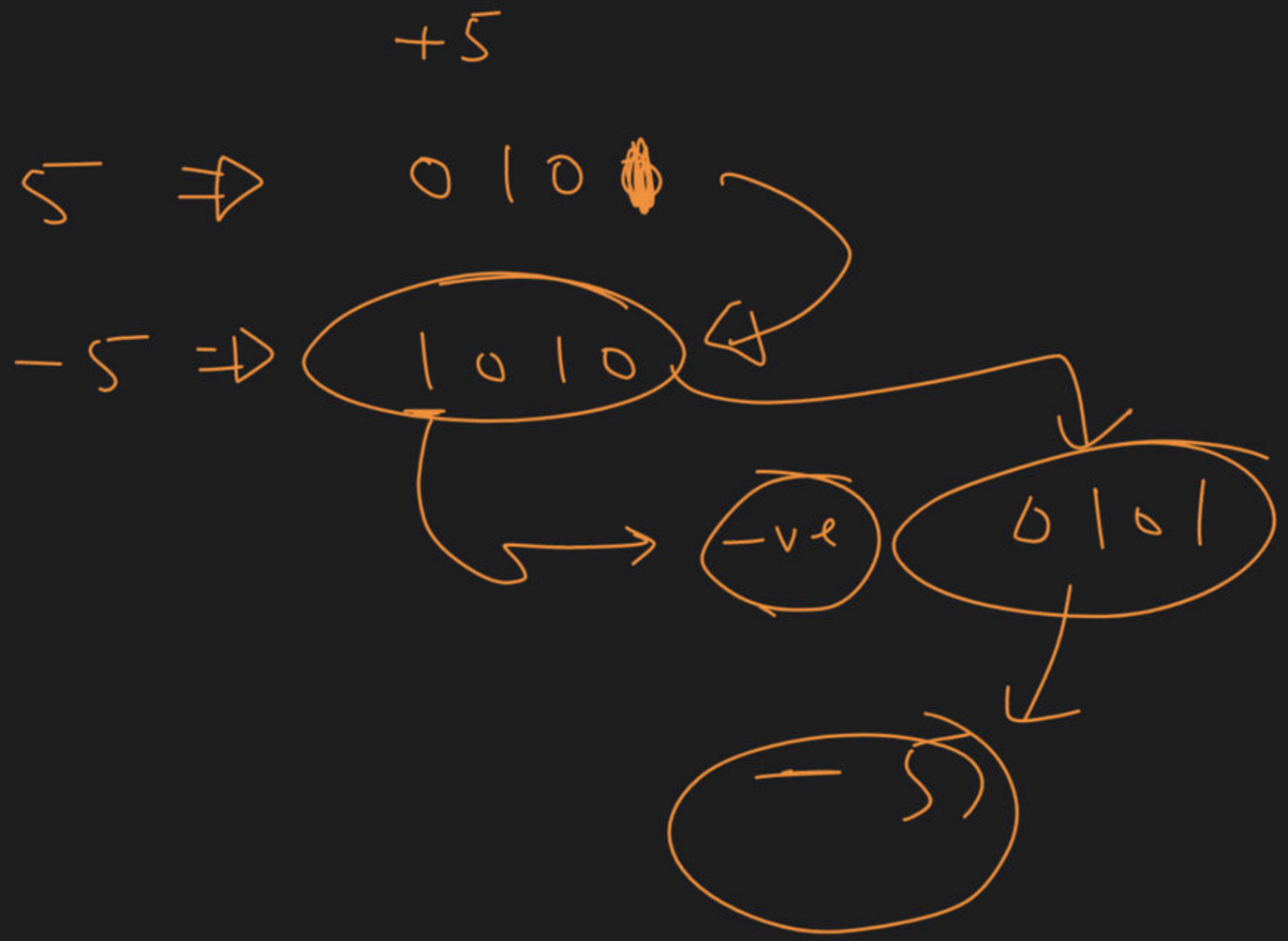
⇒ − (00111) ⇒ -7

unacademy

1's Complement

1100

↳ 0011



133 ⟹ (01000101)

0 1 0 0 0 0 1 0 1

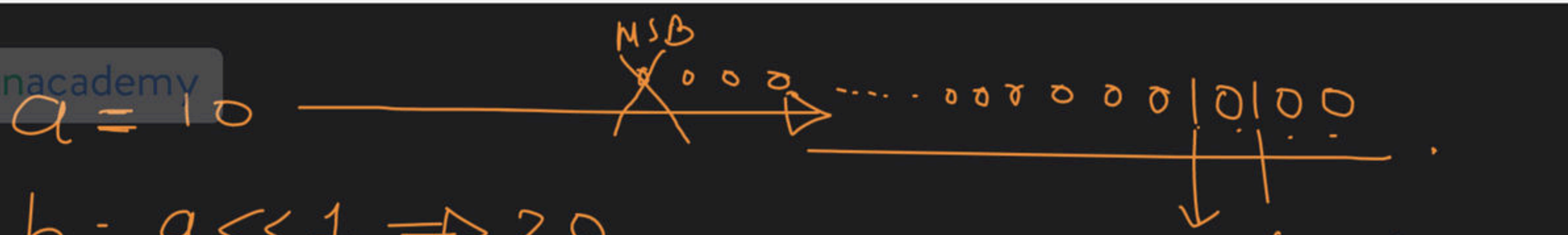
2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

128 + 5

128 + 4 + 1

& , | , ^ , << left shift.

~ ~



b = a<<1 ⇒ 20

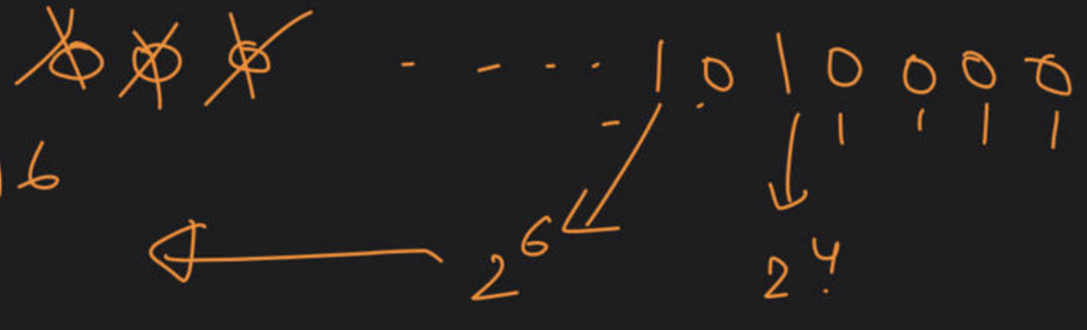
c = a<<2 ⇒ 40

d = a<<3 ⇒ 80

e = a << n

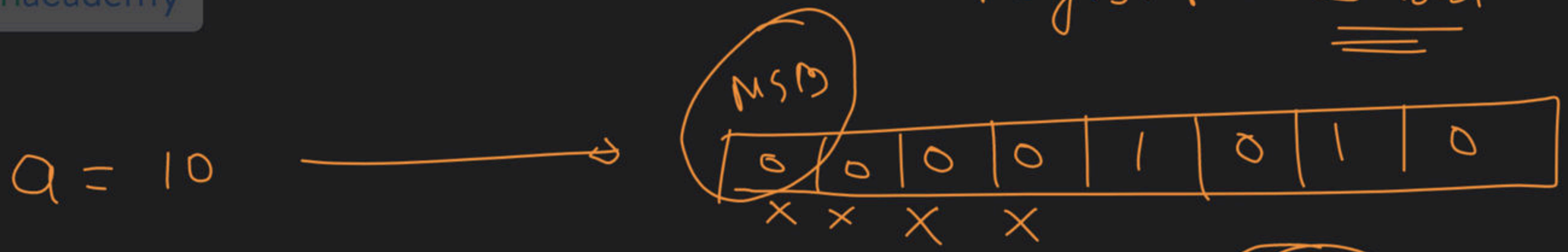
$a * 2^n$

80



unacademy

@ Register = 8 bit



b = a<<3 ⇒ 0 1 0 1 0 0 0 0 ⇒ 80 ⇒ $10 * 2^3$ ⇒ 80

c = a<<4 ⇒ 1 0 1 0 0 0 0 0 ⇒ ~~160~~

$-[2^6 + 2^5]$

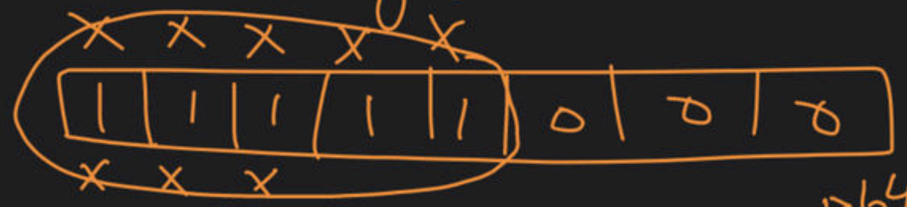
64 + 32

−96

16 bit

$a = -8$

Register = 8 bit 00001000



$b = a << 3$ ⇒ 11000000 ⇒ −[01000000] ⇒ −64

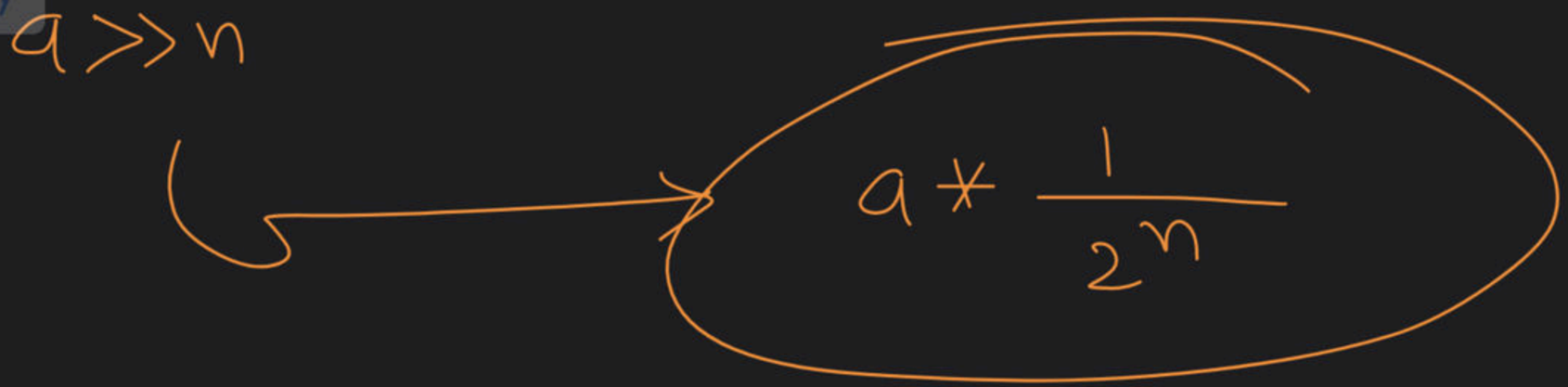
64

$c = a << 4$ ⇒ 1 0 0 0 0 0 0 0 ⇒ −[128]

$d = a << 5$ ⇒ 00000000 ⇒

$-8 * 2^3 \Rightarrow -8 * 8 \Rightarrow 64$

$-8 * 2^4 \Rightarrow -8 * 16 \Rightarrow -128$



$a = -8$

$b = a >> 1$

$a = -8$

1 1 1 1 1 1 1 0 0 0

$a >> 2$

1 1 1 1 1 1 1 0

$= [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0]$

$= -2$

unacademy

a = -7

8 bit

0 0 0 0 0 1 1 1

b = a >> 1

1 1 1 1 1 1 0 0

- (0 0 0 0 0 1 0 0)

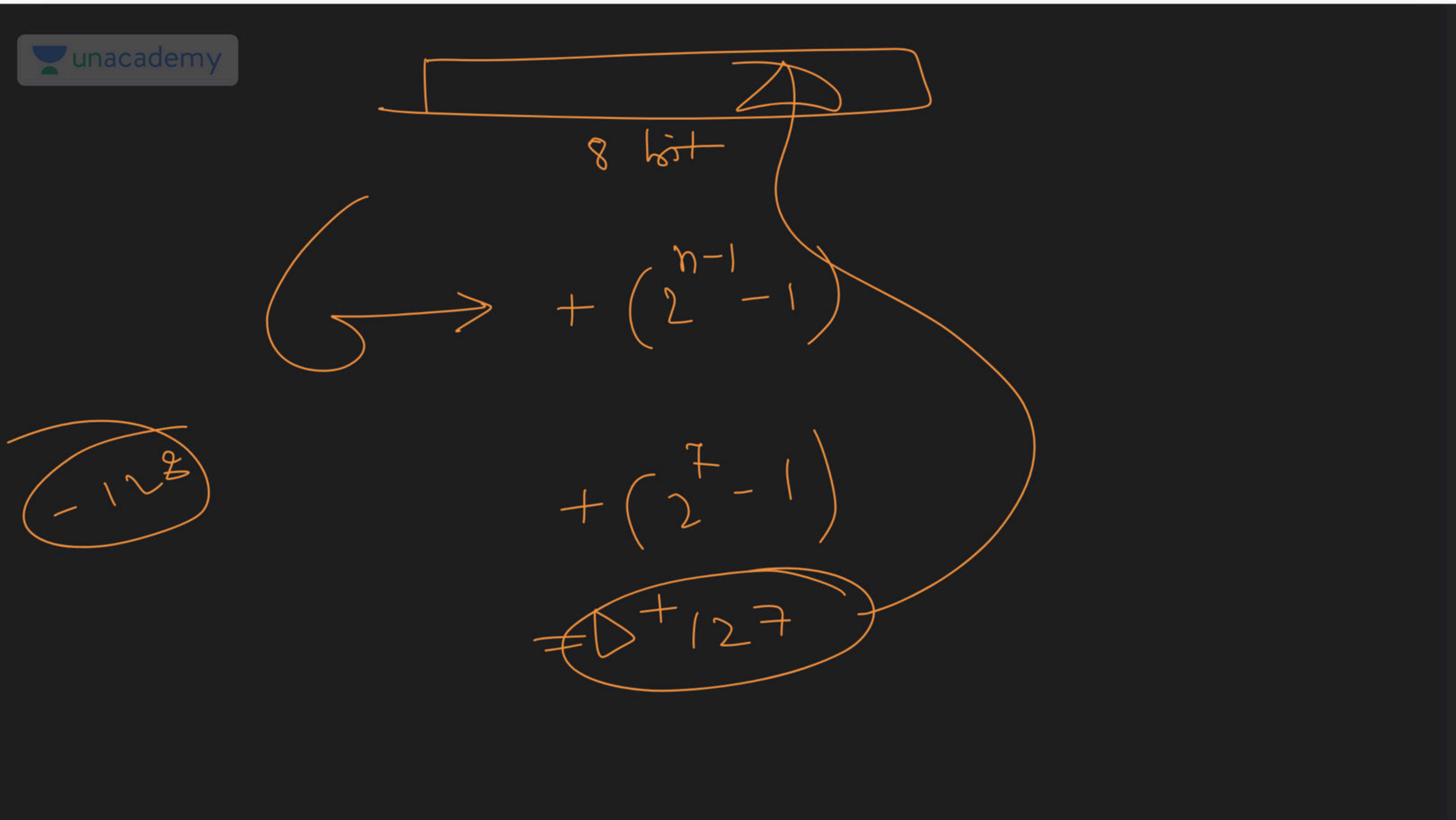
- 4

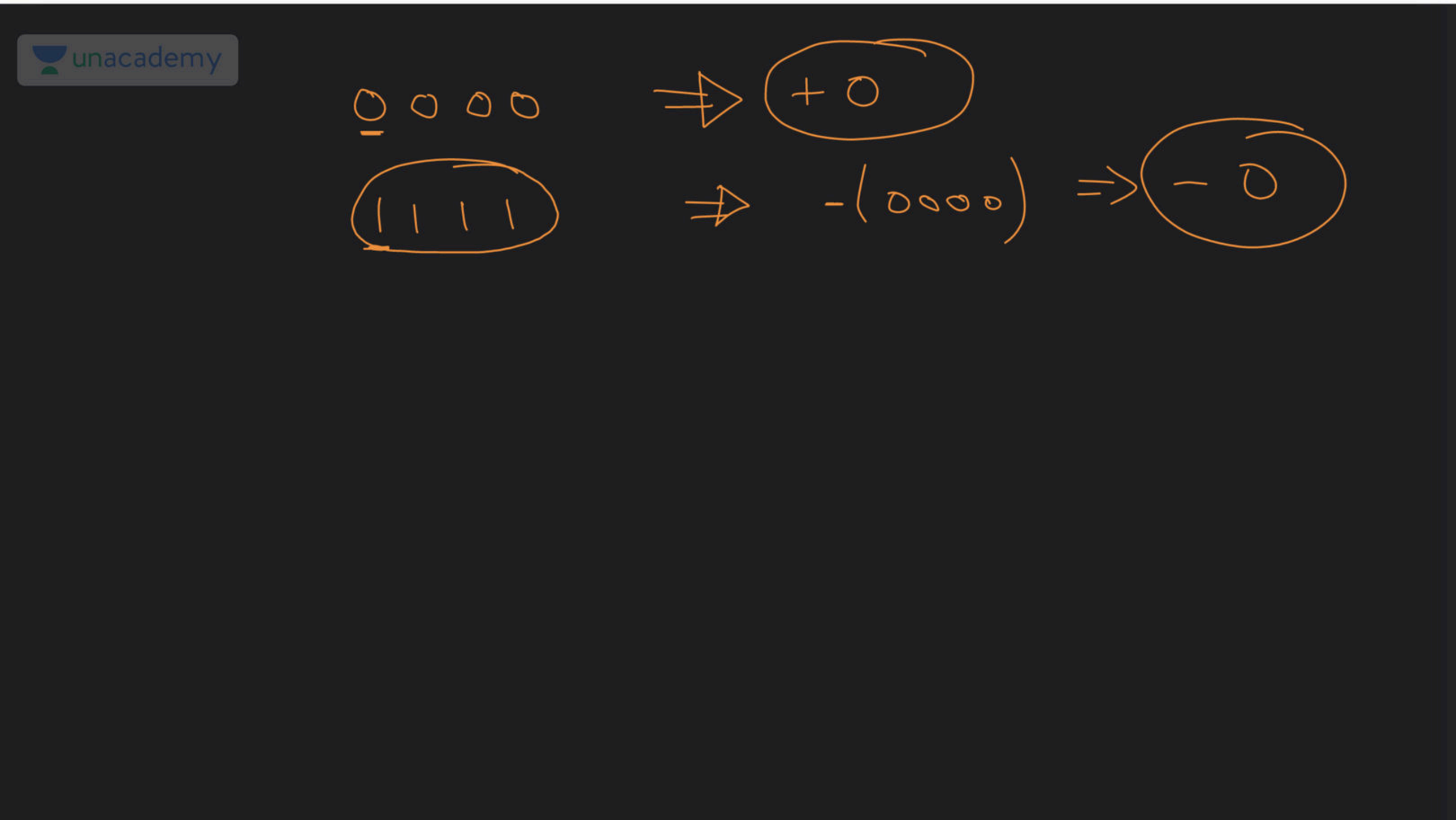
$$-7 * \frac{1}{2} \Rightarrow \lfloor -3.5 \rfloor$$

$\hookrightarrow -4$

$$7 * \frac{1}{2} \Rightarrow \lfloor 3.5 \rfloor$$

$\hookrightarrow 3$


unacademy
8 bit
$+(2^{n-1}-1)$
$+(2^{7}-1)$
+127
-128


unacademy
0 0 0 0
+ 0
1 1 1 1
- (0 0 0 0)
- 0

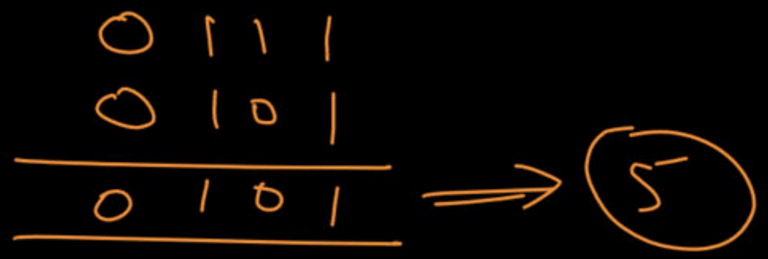
unacademy

x = n | 2

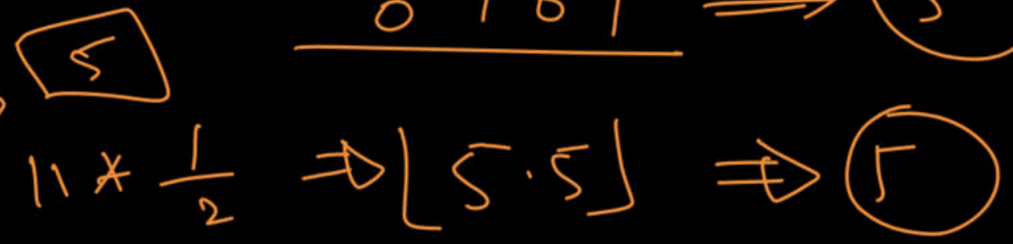
Guess Output? .

1. x=6
   x|=2
   print(x)

2. x=7
   x&=5
   print(x)



3. x=11
   x>>=1
   print(x)



4. x=28
   x<<=2
   print(x)

28 * 2^2 => 28 * 4 ⇒ 112

5. x = 5
   x ^= 3
   print(x)

Let's Try…

Guess Output? .

```
x=6
x|=2
print(x)
```

6

```
x=7
x&=5
print(x)
```

5

```
x=11
x>>=1
print(x)
```

5

```
x=28
x<<=2
print(x)
```

112

```
x = 5
x ^= 3
print(x)
```

6

Identity Operators

Operator	Description	Example
Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Membership Operators

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

'A' in Letter

Guess Output?

```
Letter=['a','b','c','e','g']
print('A' in Letter)
```
→ False

```
Letter=['a','b','c','e','g']
print('hi' in Letter)
```
→ False

```
Letter=['a','b','c','e','g']
print('HELLO' not in Letter)
```
→ True

```
Letter=['A','b','c','e','g']
print('%c' %65 in Letter)
```
→ True

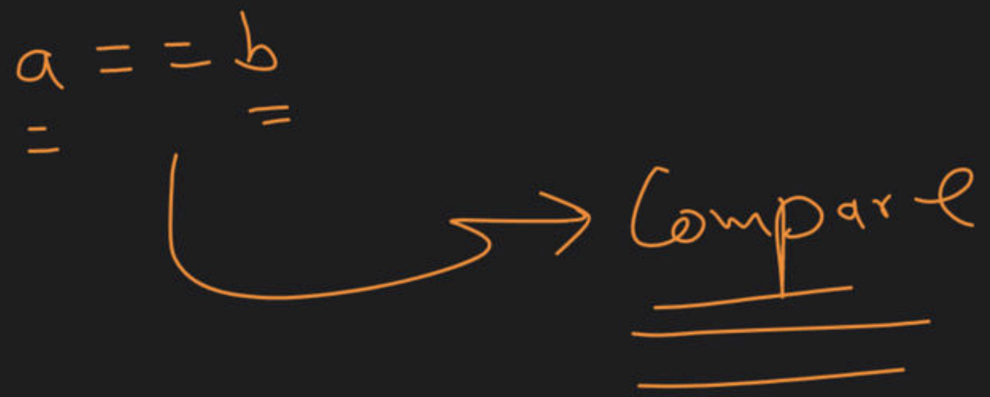
```
a=[2,4,6,8,10]
b=[2,4,6,8,10]
c=a
print(a is c)
print(a is b)
print(a==b)
```

a, b → different list objects

print(a is c) → True

print(a is b) → False

print(a==b) → True



Let's Try…

Guess Output?

```
Letter=['a','b','c','e','g']
print('A' in Letter)
```

False

```
Letter=['a','b','c','e','g']
print('hi' in Letter)
```

False

```
Letter=['a','b','c','e','g']
print('HELLO' not in Letter)
```

True

```
Letter=['A','b','c','e','g']
print('%c' %65 in Letter)
```

True

```
a=[2,4,6,8,10]
b=[2,4,6,8,10]
c=a
print(a is c)
print(a is b)
print(a==b)
```

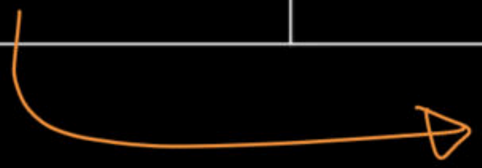
True

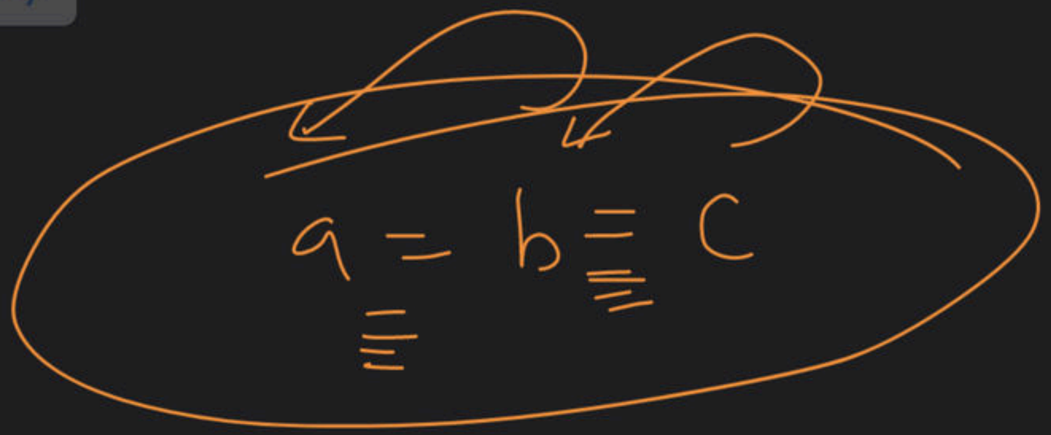
False

True

Operators Precedence

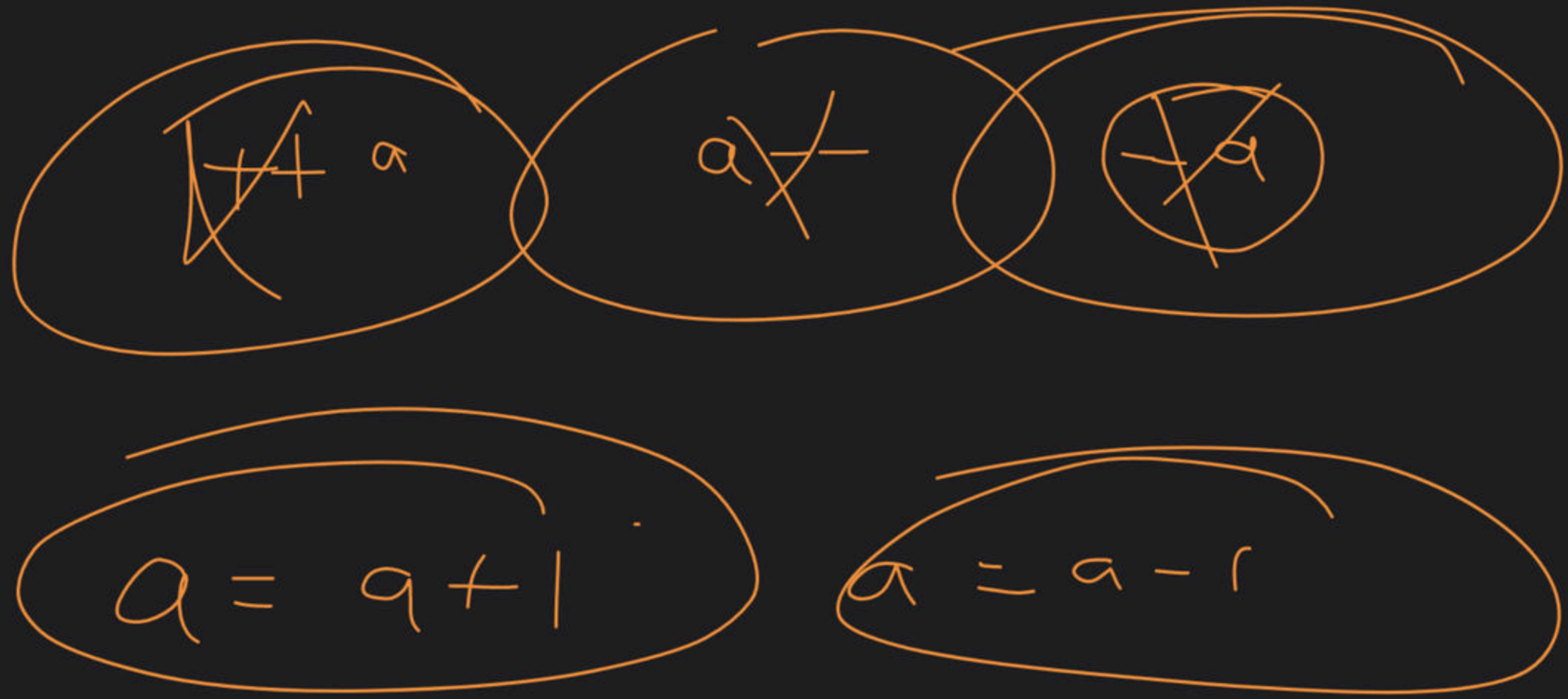
Operator	Description	Associativity
()	Parentheses	Left to Right
**	Exponentiation	Right to Left
+x -x ~x	Unary plus, unary minus, and bitwise NOT	
* / // %	Multiplication, division, floor division, and modulus	Left to Right
+ -	Addition and subtraction	Left to Right




a = b ≡ c

Operators Precedence

Operator	Description	Associativity
<< >>	Bitwise left and right shifts	Left to Right
&	Bitwise AND	Left to Right
^	Bitwise XOR	Left to Right
|	Bitwise OR	Left to Right
== != > >= < <= is is not in not in	Comparisons, identity, and membership operators	Left to Right
not	Logical NOT	Left to Right
and	AND	Left to Right
or	OR	Left to Right



Guess Output?

```
X=((4+12)*(15-10)*(12//3))
print(X)
```

(16 * 5 * 4) → 320

```
A=24+12*2-3**3
print(A)
```

24 + 12 * 2 − 27 ⇒ 24 + 24 − 27 → 21

```
print(24//2//4)
```

12 // 4 ⇒ 3

```
print(36*2%7/2)
```

72 % 7 / 2 ⇒ 2 | 2 ⇒ 1.0

```
print(20<<2>>2)
```

$20 * \frac{2^2}{1} * \frac{1}{2^2}$ ⇒ 20

```
print(80>>1&10)
```

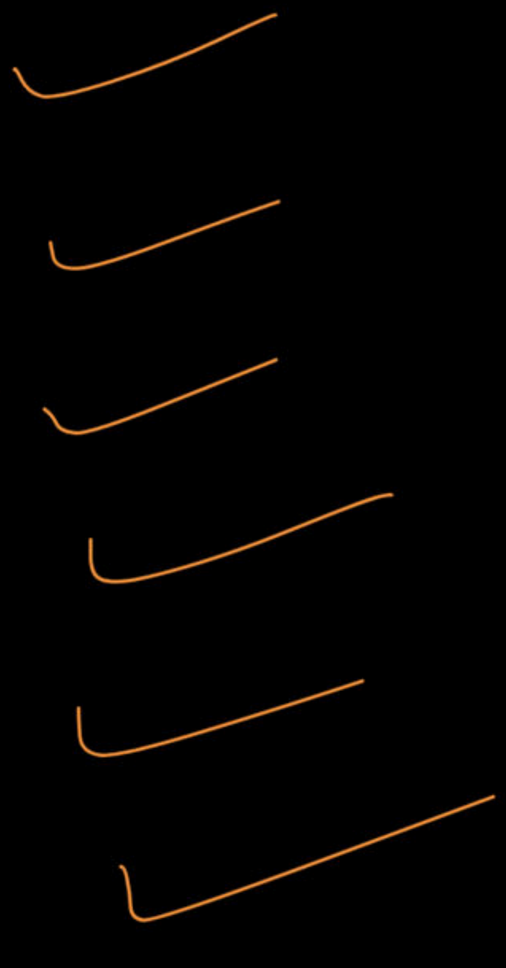
40 & 10 ⇒

```
1 0 1 0 0 0
0 0 1 0 1 0
-----------
0 0 1 0 0 0
```

→ 8

Let's Try…

Guess Output?	
X=((4+12)*(15-10)*(12//3)) print(X)	320
A=24+12*2-3**3 print(A)	21
print(24//2//4)	3
print(36*2%7/2)	1.0
print(20<<2>>2)	20
print(80>>1&10)	8



Left Shift

1. #print(12<<1) → 24
2. #print(11<<1) → 22
3. #print(-12<<1) → -24
4. #print(-11<<1) → -22
5. #print(-12<<2) → -48
6. #print(-12.5<<1) → Error

Right Shift

7. #print(12>>1) → 6
8. #print(11>>1) → 5
9. #print(-12>>1) → -6 → -3
10. #print(-11>>1)#print(-12.5>>1), #print(-11>>2)x=6x//=-2.5print(x)print(-5.0//2)

11. x = 6 ∴
    x //= -2.5 ⇒ x = 6 // -2.5
    Print(x) → -3.0

12. Print (-5.0 // 2) → -3.0

10 → -6

11 → Error

12

12.25

.25 * 2 = 0.5 ⇒ 0

0.5 * 2 = 1.0 ⇒ 1

0 * 2 ⇒ 0.0 ⇒ 0

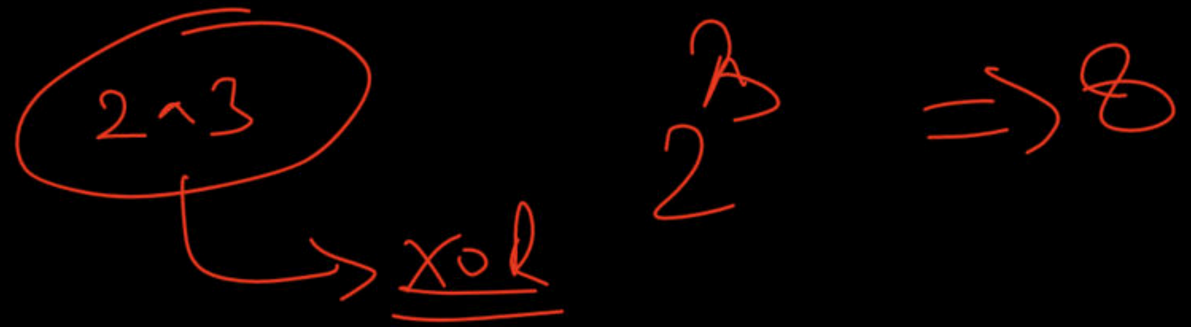
1100.010

$-5.5 \Rightarrow -6$

floor

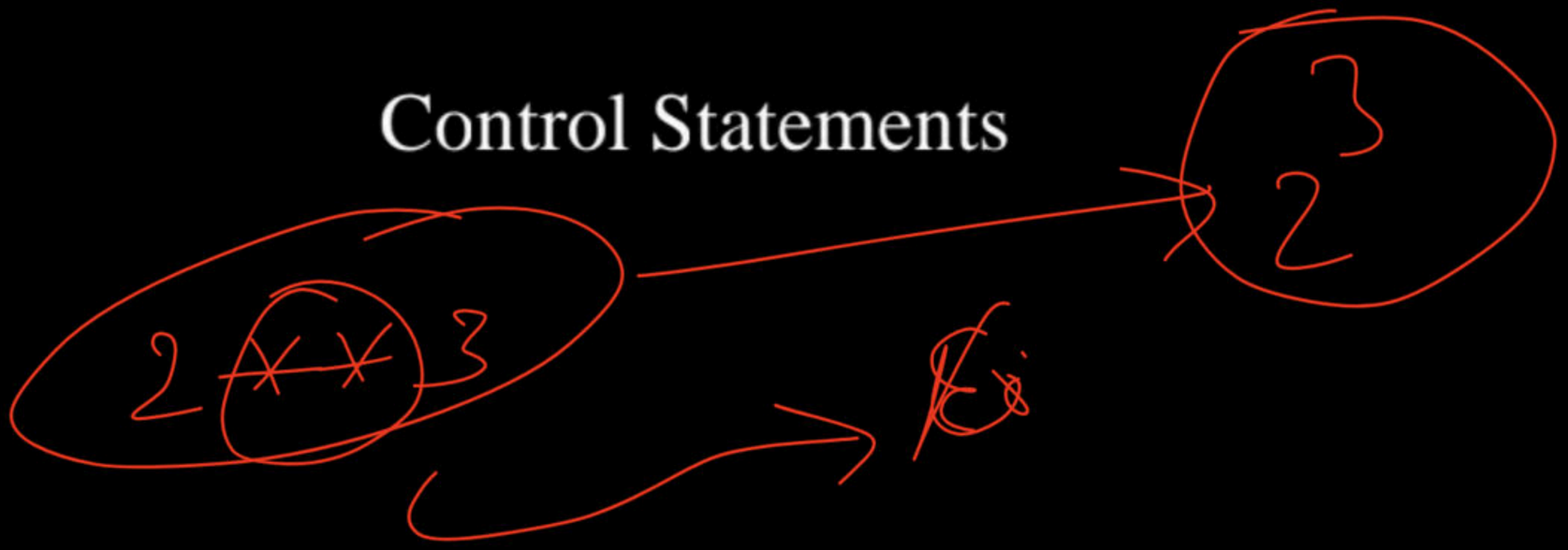
$5.5 \Rightarrow 5$

floor



Python Programming

Control Statements



Content

- Control Statements
- Types of Control Statements
- Conditional Statement
- Looping Statement
- Jumping Statement
- Sample Problems

Control Statements

- Control statements define the order of statement executions for the python program.

Sequential Statement Examples:

```
a=20
b=60
print(a+b)
```

Conditional Statement Examples:

```
If a>b:
   print('a is greater')
Else:
   print('b is greater')
```

Looping Statement Examples:

```
While True:
    print('Loop');
```

Jumping Statement Examples:

```
Name='Jhon'
for ch in Name:
    print(ch);
    if ch=='o':
        break;
```

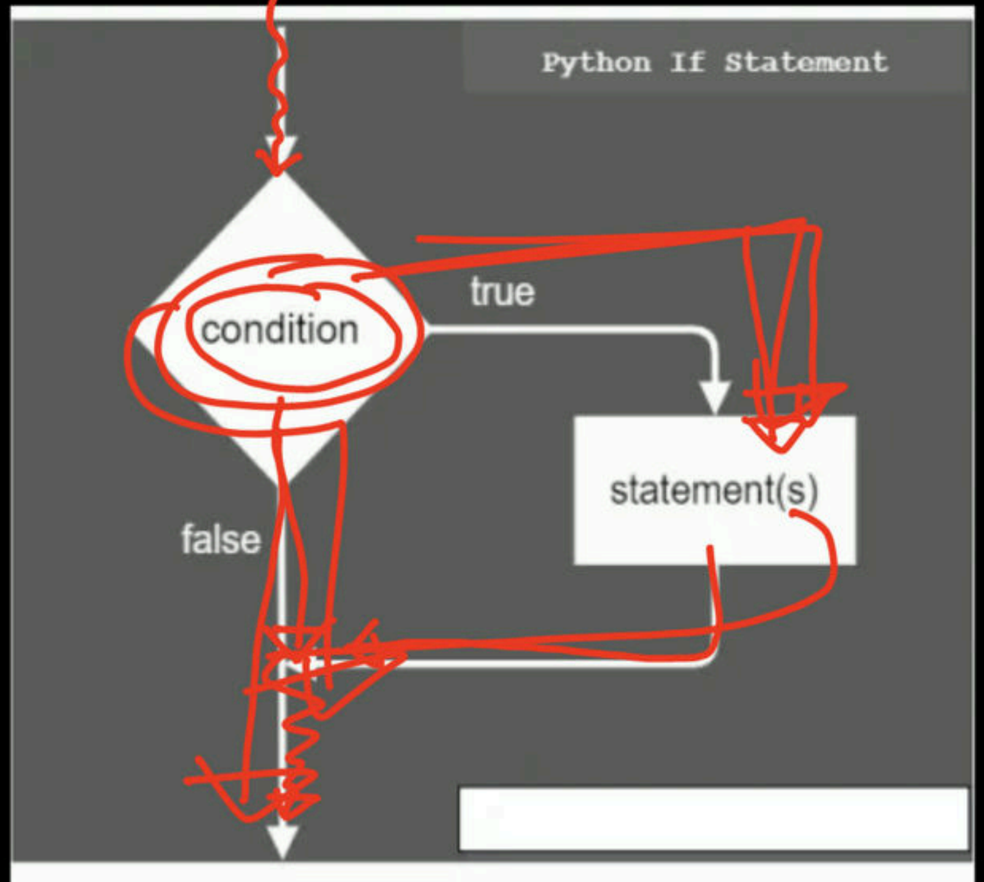
Conditional Statements

In python, the execution flow of codes may switch on the basis of certain decisions. There are four types of conditional statements:

- Simple If Statement
- If-Else Statement
- Nested If Statement
- Elif Ladders

Simple If Statement

- This statement allows execution of certain line of codes when condition is true or satisfied.



Simple If

Syntax:

```
if <condition>:
    statement1
    statement2
```

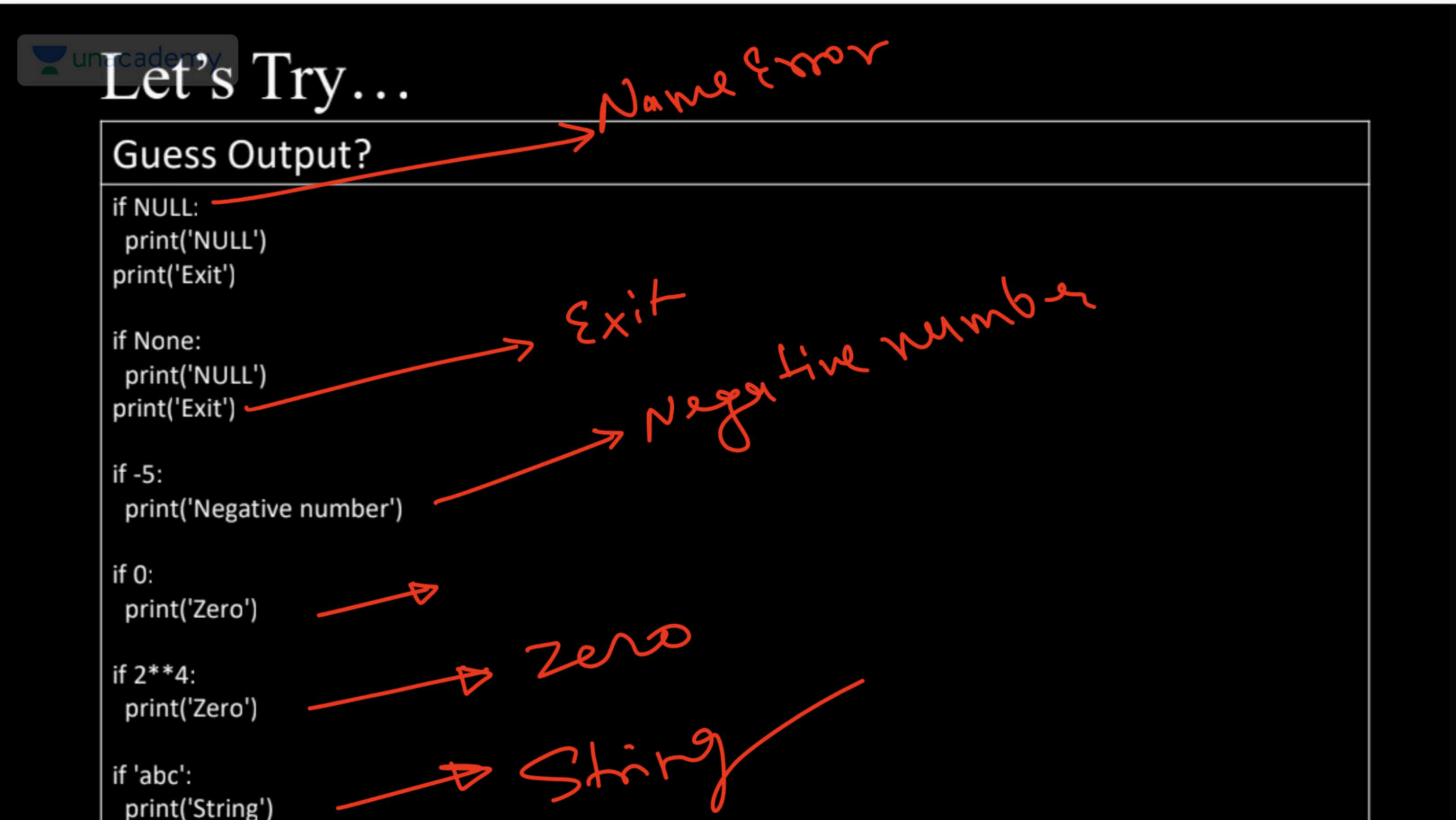
Example:

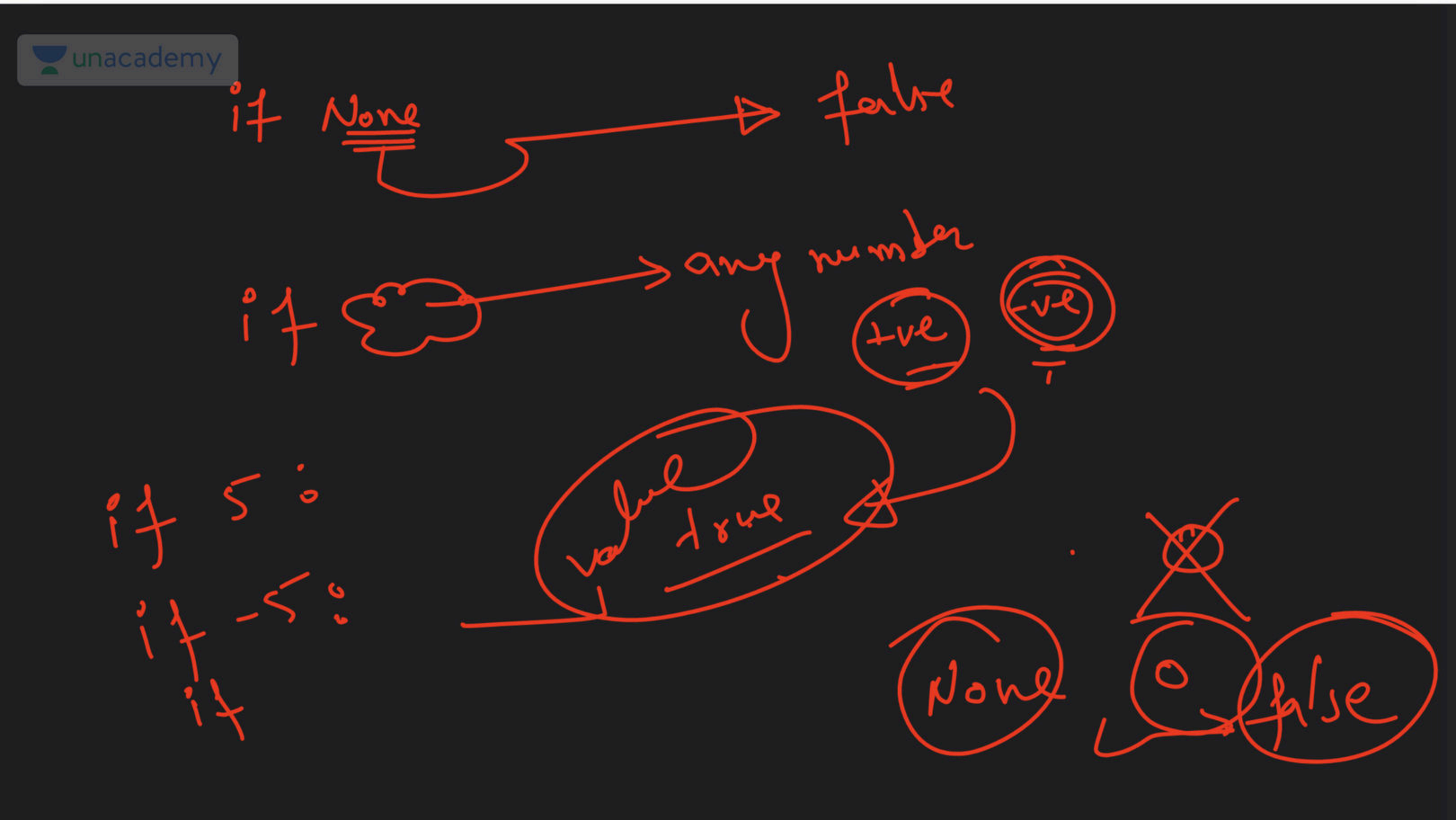
```
a=10
if a>18:
    print("You are eligible to vote")
    print("You are eligible for driving licenses ")
print("bye")
```

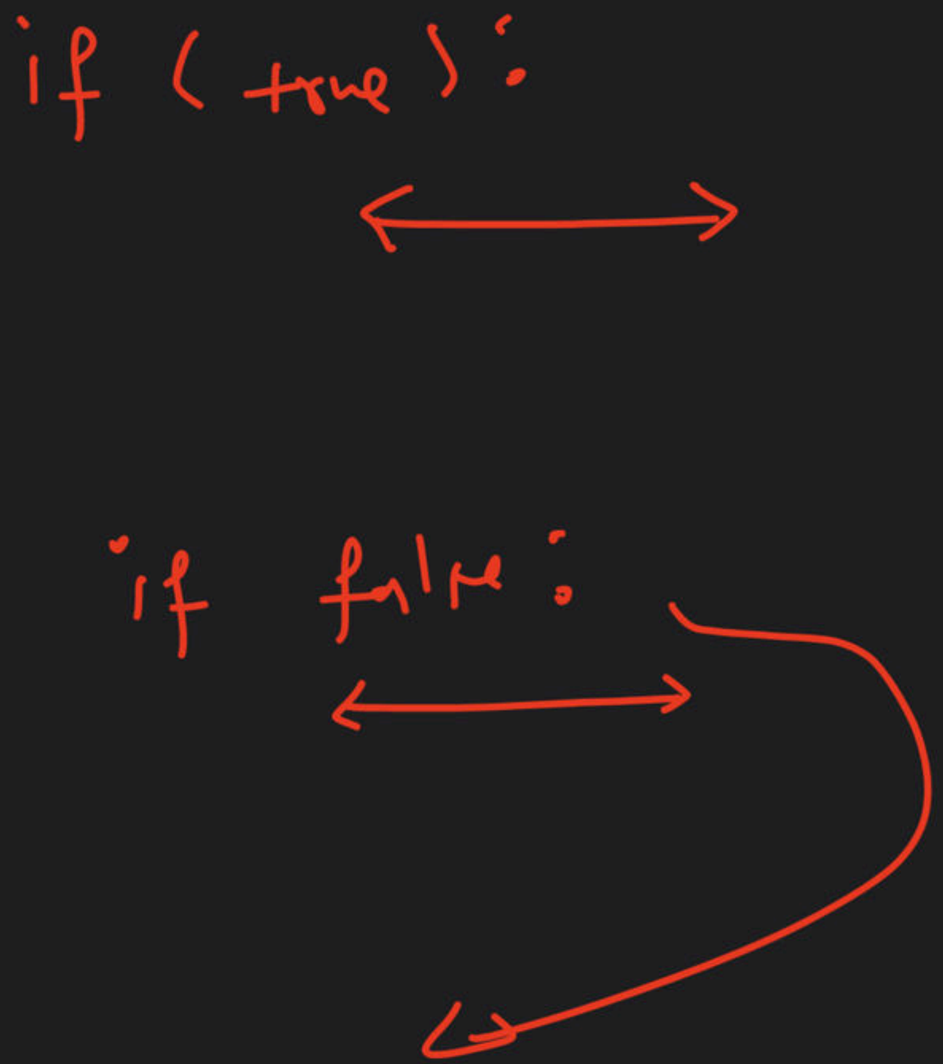
Let's Try…

Guess Output?

```
if NULL:
  print('NULL')
print('Exit')

if None:
  print('NULL')
print('Exit')

if -5:
  print('Negative number')

if 0:
  print('Zero')

if 2**4:
  print('Zero')

if 'abc':
  print('String')
```




if None
false
if
any number
+ve
-ve
if 5 :
if -5 :
if
value
true
None
0
false



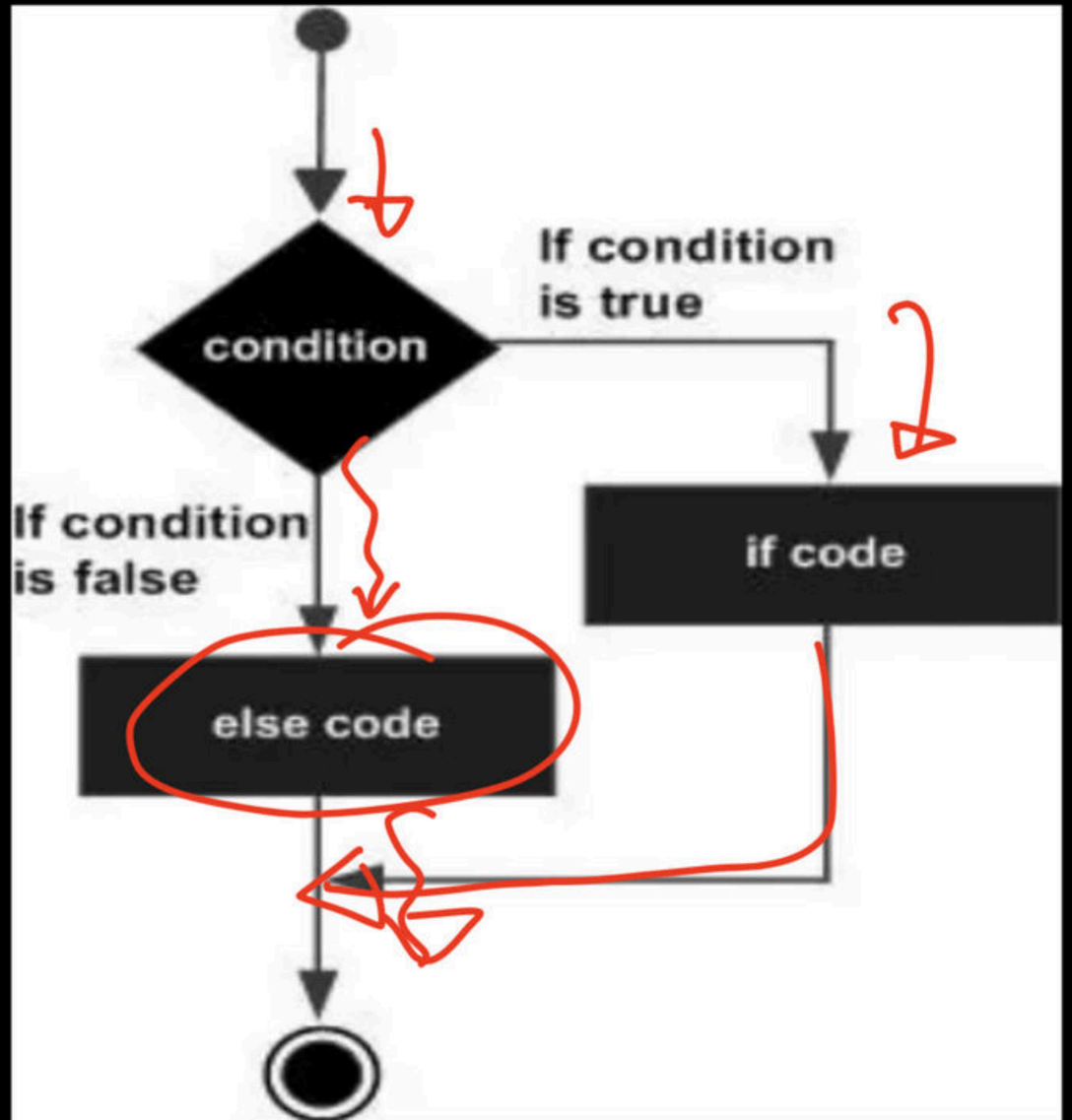
unacademy

if ____ :

if 0

if None

If Else Statement



If-Else

Syntax:

```
if <condition>:
    statement1
else:
    statement2
```

Example:

```
a=20
if a>=18:
    print("You are eligible to vote")
else:
    print("You are required to wait up to 18")
```

Let's Try…

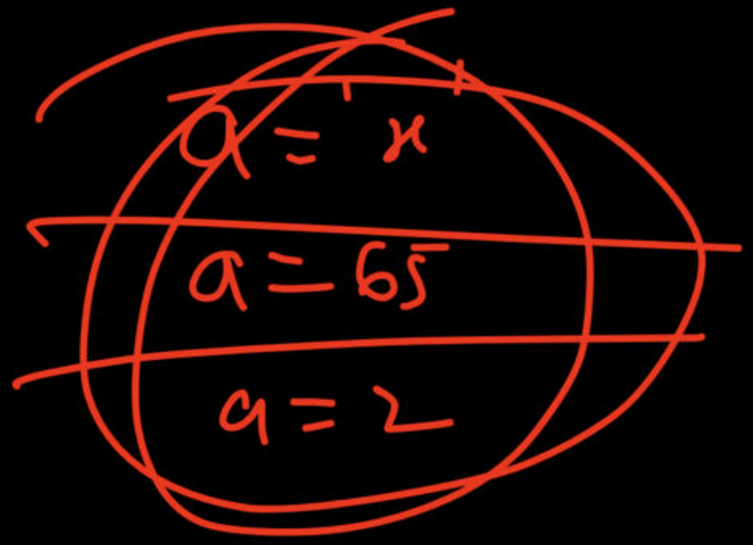
Guess Output?

1)
```
if True:
  print('True')
else:
  print('False')
```

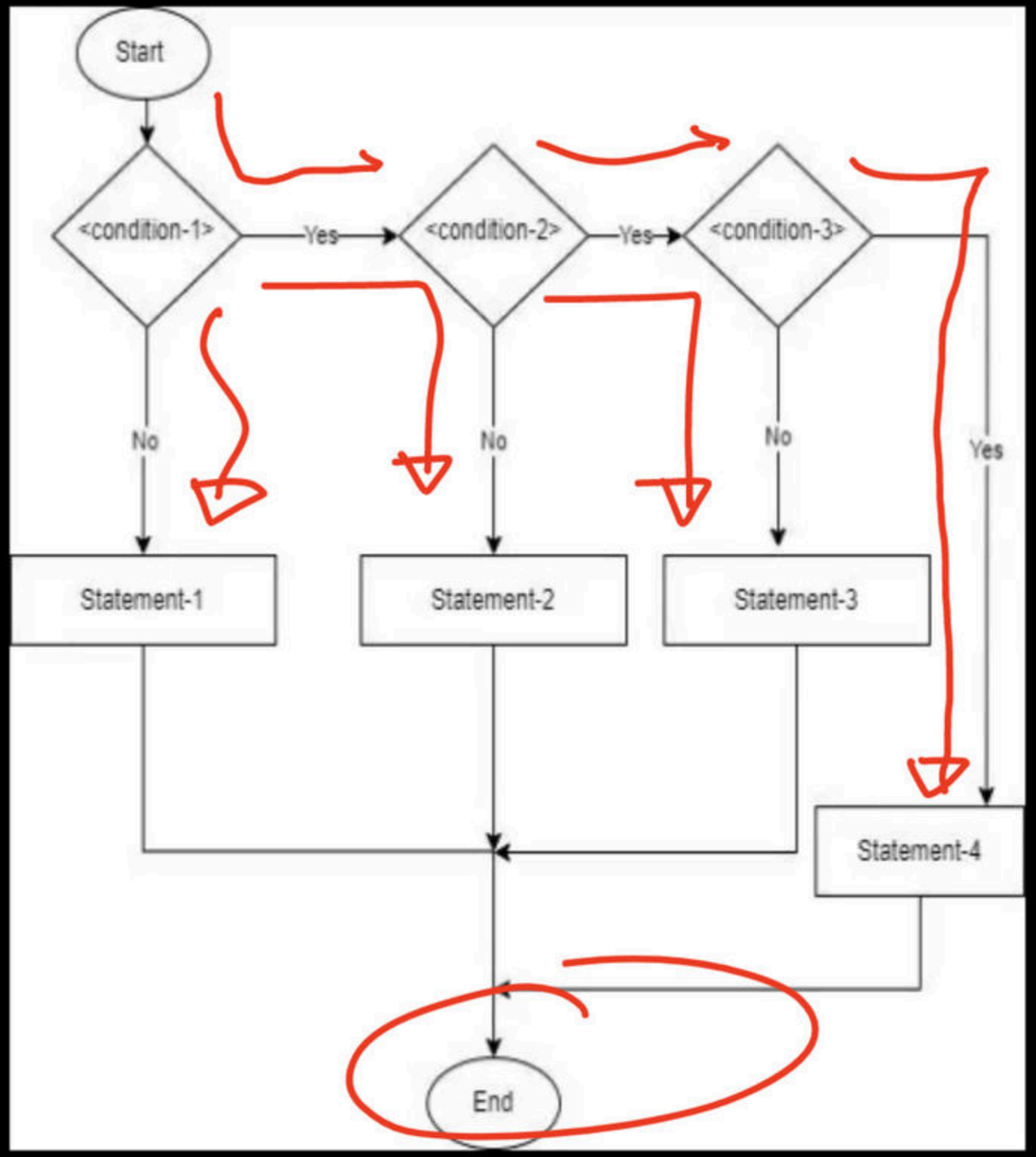
2)
```
if 6+6>=10:
  print('True')
else:
  print('False')
```

3)
```
A=5
if a>5 or a<=5:
  print('It works')
else:
  print('It does not work')
```

4)
```
if(a//b==2):
        print ("Yes")
else :
        print("No")
```



Nested If-Else Statement



Nested If-Else

Syntax:

```
if <condition>:
    statement1
    if<condition>:
        statement2
    else:
        statement3
else:
    statement4
```

Example:

```
a=20
if a>=18:
    print("You are eligible to vote")
    if a>60:
        print("You are senior citizen")
    else:
        print("You are not senior citizen")
else:
    print("You are child")
```

Let's Try…

Guess Output?

```
n=50
if n >20:
    if n>35:
        print("OK",)
        if n>45:
            print("GOOD")
        else:
            print("NO OUTPUT")
    else:
        print("NO OUTPUT")
else:
    print("NO OUTPUT")
```

OK
GOOD

```
a=10
b=5
if (a%b==0):
    print ( "Greater")
    if (a//b==0):
        print ( "Example")
    else:
        print ("Easy to learn with Learnpython4cbse")
else:
    print ("No Output")
```

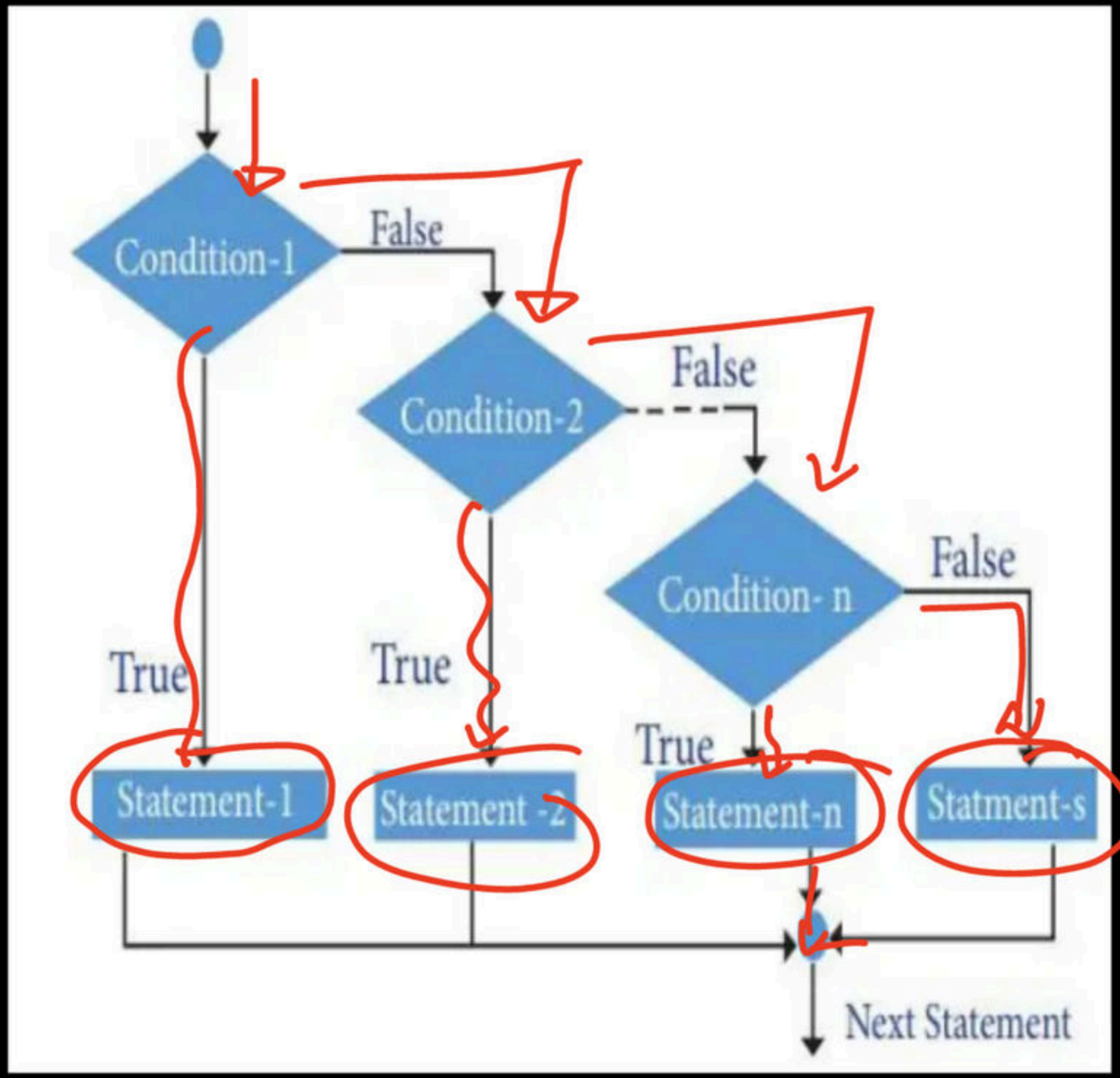
Greater

10%5 == 0 :

a % b

10//5 => 2 == 0

Elif Ladder



Elif Ladder

Syntax:

```
if <condition>:
    statement1
elif<condition>:
    statement2
elif<condition>:
    statement3
else:
    statement4
```

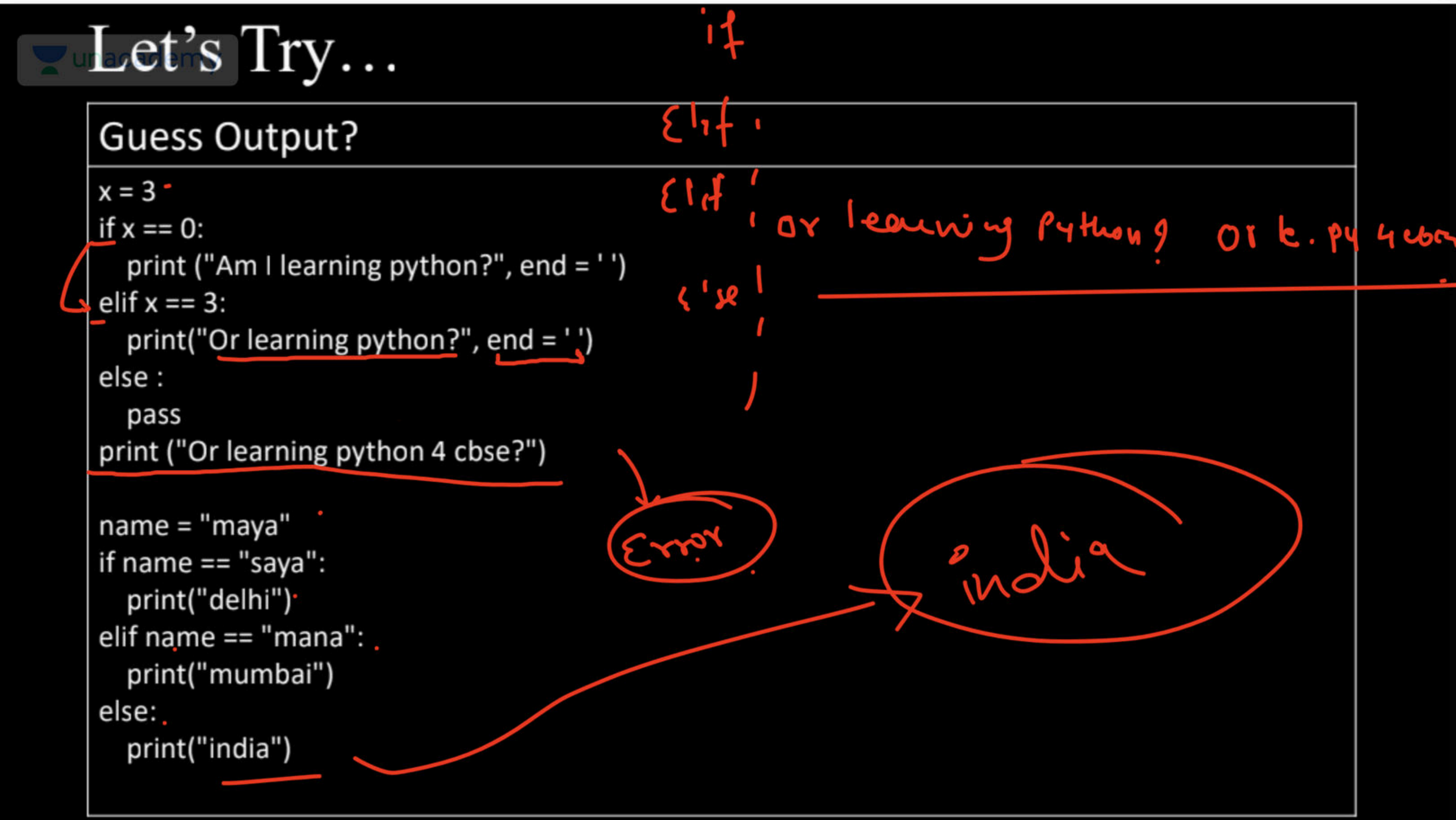
Example:

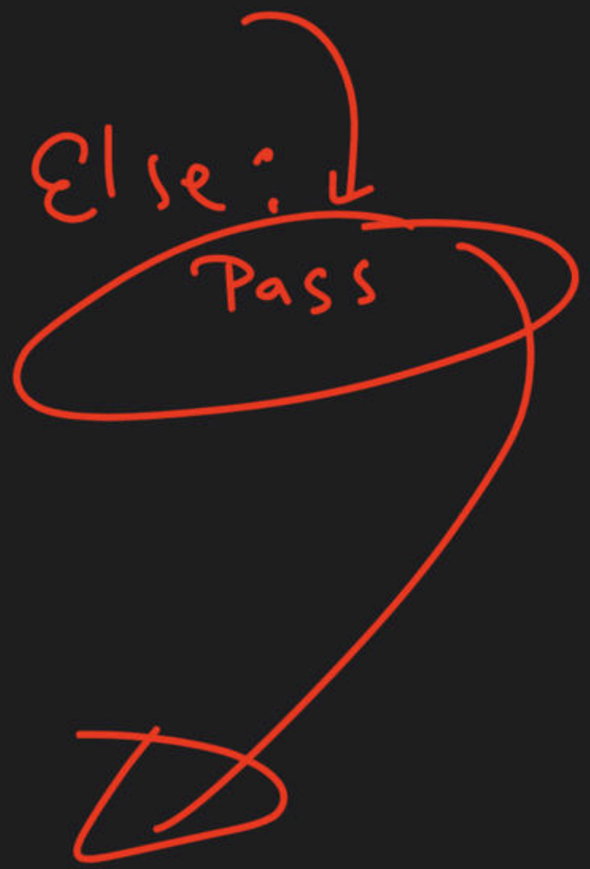
```
age=20
if age>=18:
  print("You are eligible to vote")
elif age>15:
  print("You are require to three year more for vote")
elif age>12:
  print("You are require to six year more for vote")
else:
  print("You are child")
```

Let's Try...

Guess Output?

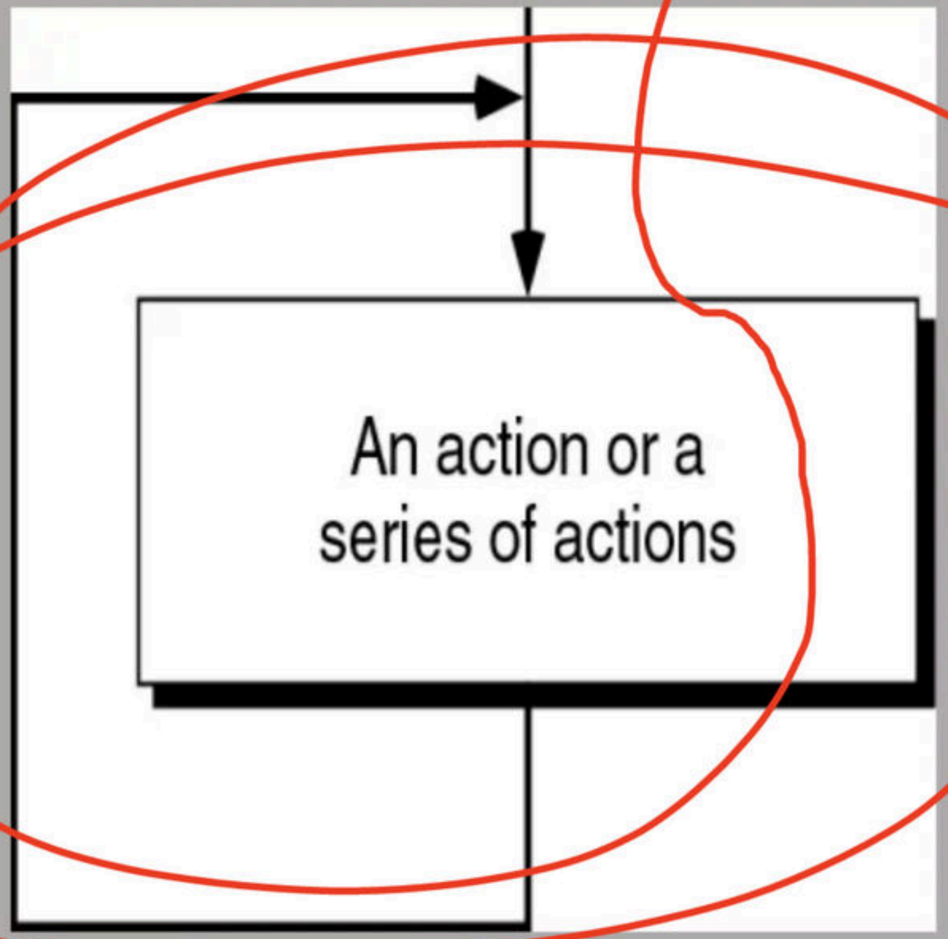
```
x = 3
if x == 0:
    print ("Am I learning python?", end = ' ')
elif x == 3:
    print("Or learning python?", end = ' ')
else :
    pass
print ("Or learning python 4 cbse?")

name = "maya"
if name == "saya":
    print("delhi")
elif name == "mana":
    print("mumbai")
else:
    print("india")
```





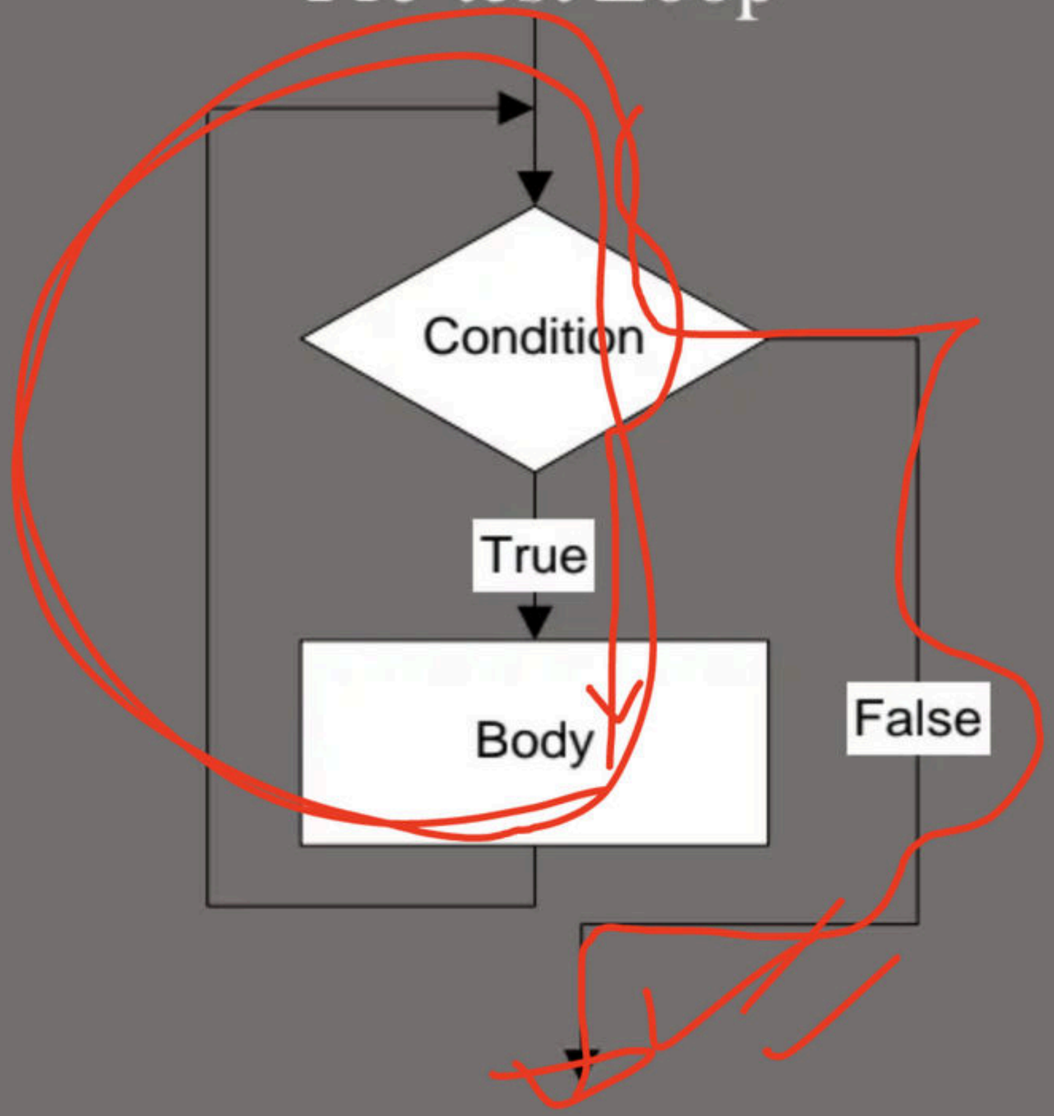
Looping Statement

- The main idea of a loop is to repeat an action or a series of actions.

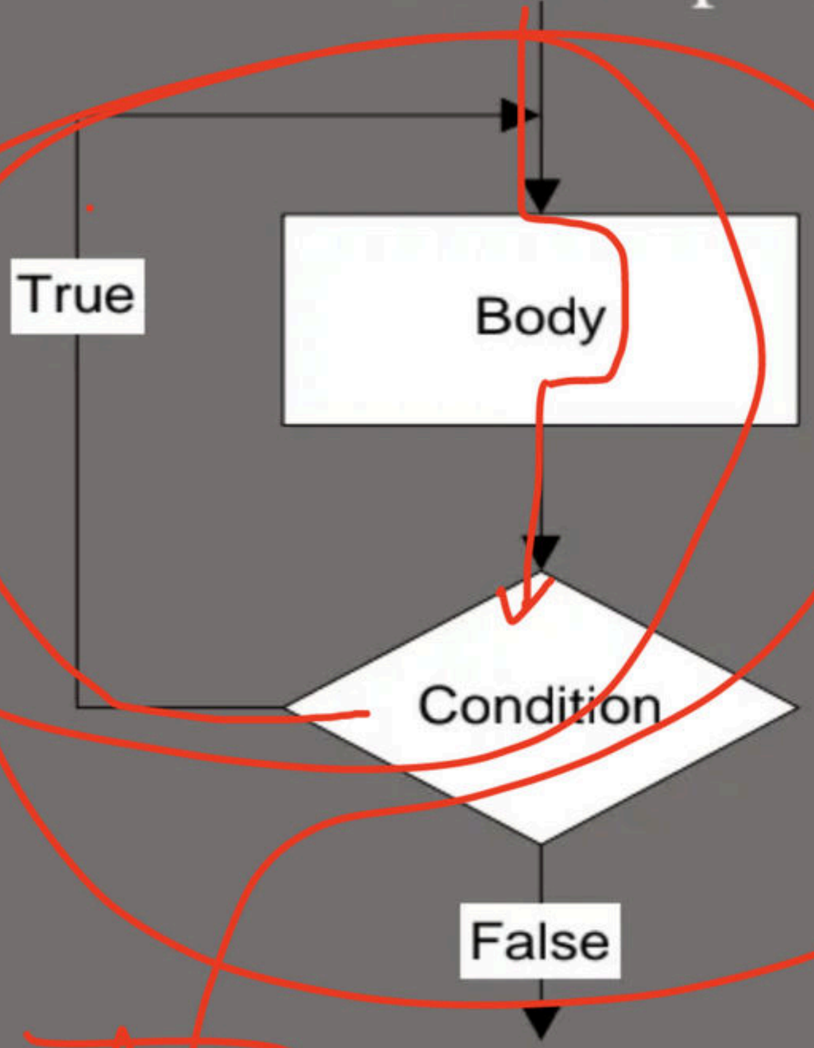


Types of Loop

Pre-test Loop



Post-test Loop



Switch-Case Statement

Switch Case

Syntax:

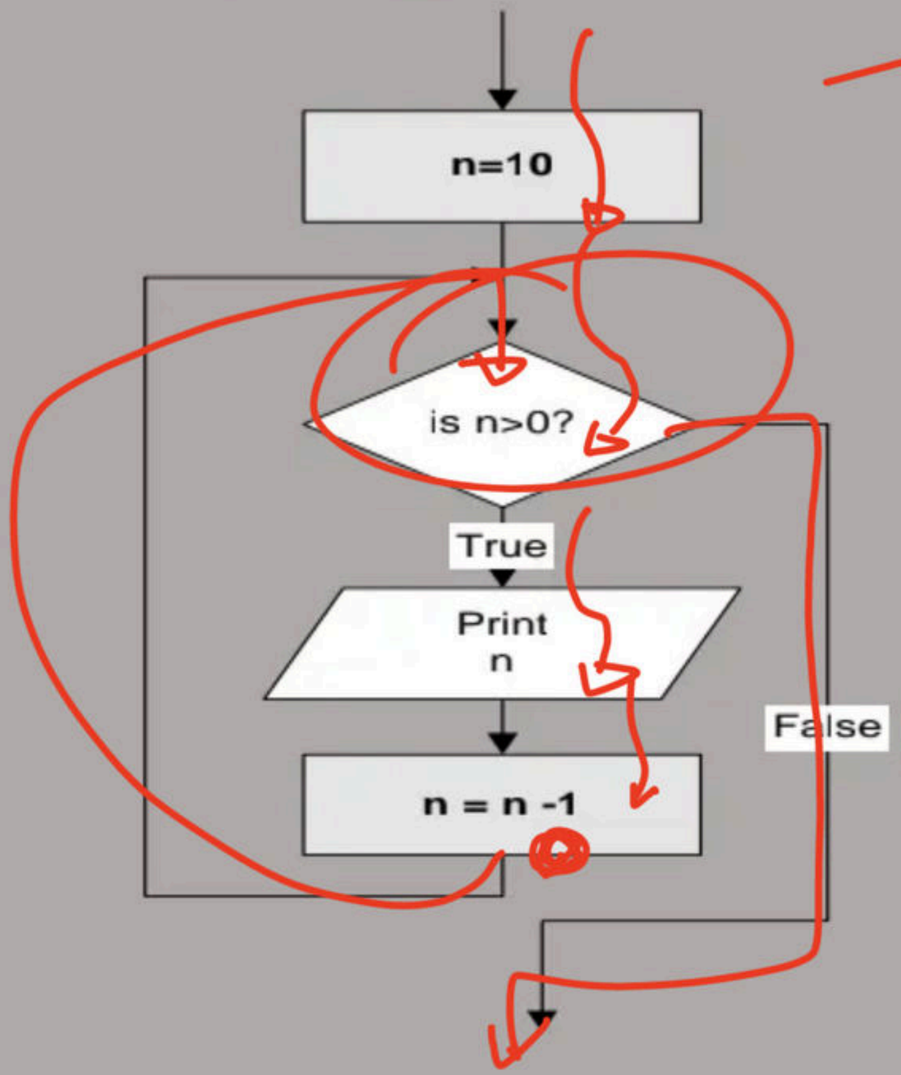
```
match <argument>:
    case 1:
        statement1
    case default:
        statement2
```

Example:

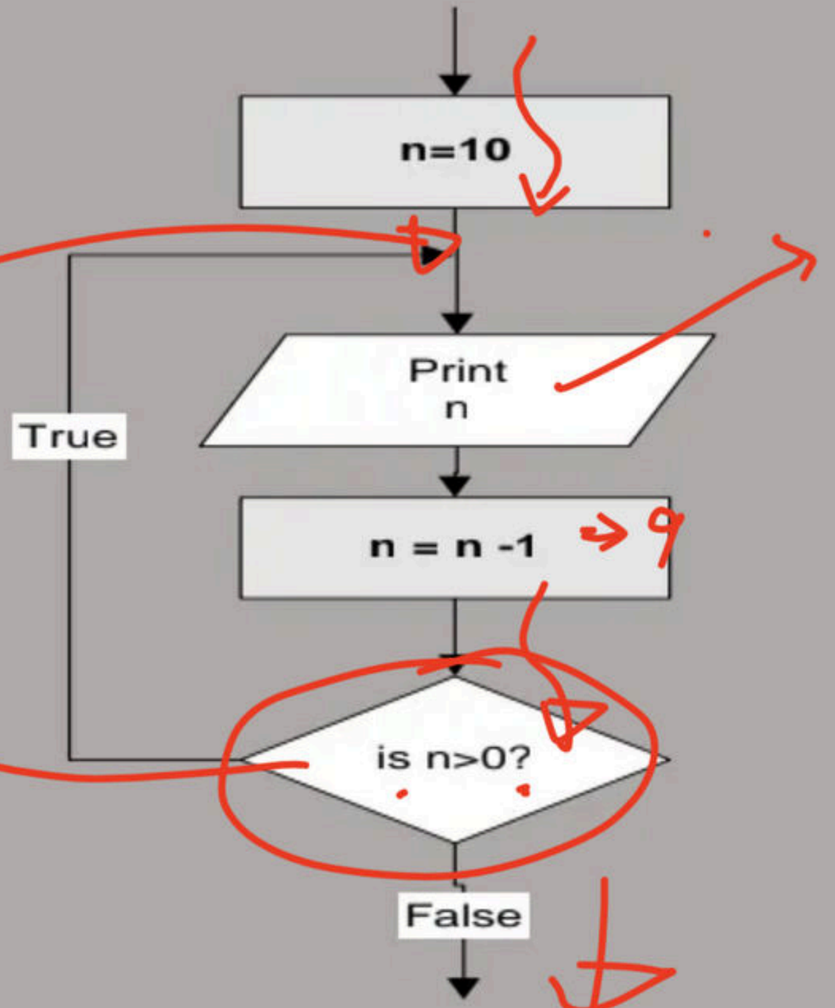
```
number=int(input('Enter a number: '))
match number:
    case 0:
        print("zero")
    case 1:
        print("one")
    case 2:
        print("two")
    case default:
        print("something")
```

Examples: Types of Loop

Pre-test Loop Example

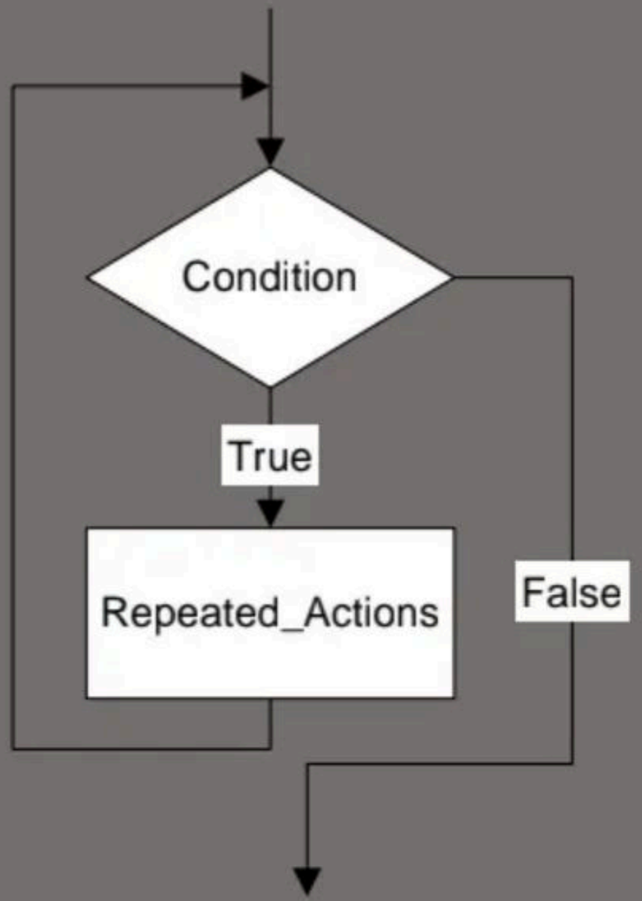


Post-test Loop Example



Pre-test Loop: While Loop

While Loop Flow Chart



While Loop

Syntax:

```
while <condition>:
        statement1
        statement2
        statement3
```

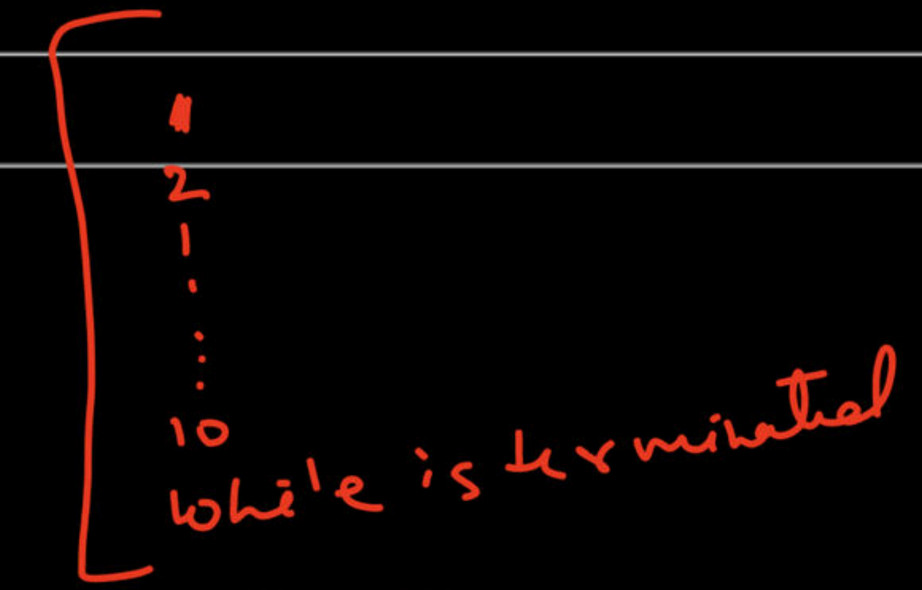
Example:

```
num=int(input('Enter a number'))
while i<=10:
    print(i*num)
    i=i+1
```

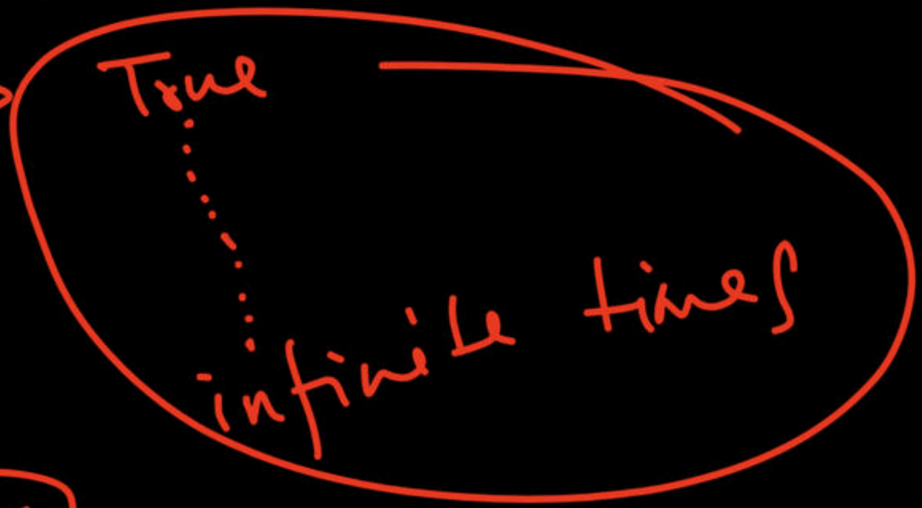
Let's Try…

Guess Output?

```
a=1
while a<=10:
  print(a)
  a=a+1
else:
  print('while is terminated')
```




```
while True:
   print('True')
else:
  print('False')
```




```
number = 5
while number <= 5:
    if number < 5:
        number = number + 1
    print(number)
```


```
a = "123789"
while x in a:
    print(x, end=" ")
```

unacademy

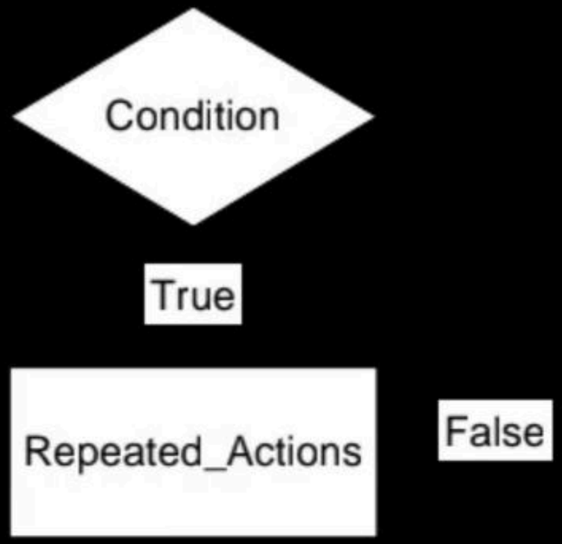
$a = 5$

Python Programming

Control Statements Part II

Pre-test Loop: For Loop

For Loop Flow Chart



For Loop

Syntax:

```
for<condition>:
    statement1
    statement2
    statement3
```

Example:

```
Msg='Welcome to Bharat'
for i in Msg:
    print(i)
```

Let's Try…

Guess Output?

```
for i in range(0,5):
    print(i)

for k in [0,1,2,3,4,5]:
    print(k)

 for l in range(0,5,1):
    print(l)

for k in range(0.0,5.5,0.5):
    print(k)

x = "abcd"
for i in range(len(x)):
print(i)

x = 12
for i in x:
    print(i)
```

Jumping Statement: break

break statement

Syntax:
for<condition>:
statement1
if <condition>:
break
Example:
Msg='Welcome to Bharat'
for i in Msg:
print(i)
if(i==‘e’):
break

Let's Try…

Guess Output?

```
n=7
c=0
while(n):
   if(n>5):
      c=c+n-1
      n=n-1
   else:
      break
print(n)
print(c)

i = 1
while True:
     if i%3 == 0:
        break
     print(i)
     i + = 2
i = 5
while True:
   if i%0O9 == 0:
      break
   print(i)
   i += 1
```

Jumping Statement: continue

continue statement

Syntax:
for<condition>:
if <condition>:
continue
statement1
Example:
Msg='Welcome to Bharat'
for i in range (len(Msg)):
if(i%2==0):
continue
print(i)

Let's Try…

Guess Output?

```
Msg='Welcome to IISC'
for i in range(len(Msg)):
 if(i//2==0):
   continue
 print(i)

for letter in 'Python':
   if letter == 'h':
         continue
   print('Current Letter : ' + letter)
```

Jumping Statement: return

return statement

Syntax:
def function_name(argumentlist):
return statement
var=function_name(argumentlist)
Example:
x,y=35,25
def sum(a,b):
return a+b
z=sum(x,y)

Sample Problems

Problem-1:
Write python code to read an integer N from the user and try to print the following: 1234…N Example N=4 Print the string: 1234

Sample Problems

Problem-1:
Write python code to read an integer N from the user and try to print the following: 1234…N Example N=4 Print the string: 1234

```
# Read an integer N from the user
N = int(input("Enter an integer N: "))

# Iterate through numbers from 1 to N and print them without a newline
for i in range(1, N + 1):
    print(i, end="")

# Print a newline character to move to the next line
print()
```

Sample Problems

Problem-2:

Write a program that asks the user for a number and prints out that many stars on a line. Continue asking the user for new numbers until they decide not to.

Example:

```
 Please enter a number: 4
****
Do you want to try again? (y/n) y
Please enter a number: 7
*******
Do you want to try again? (y/n) n
Goodbye!
```

Sample Problems

Problem-2:

Write a program that asks the user for a number and prints out that many stars on a line. Continue asking the user for new numbers until they decide not to.

Example:

Please enter a number: 4
****

Do you want to try again? (y/n) y
Please enter a number: 7
****

Do you want to try again? (y/n) n
Goodbye!

```
while True:
    try:
        num = int(input("Please enter a number: "))
    except ValueError:
        print("Invalid input. Please enter a valid number.")
        continue

    stars = '*' * num
    print(stars)

    choice = input("Do you want to try again? (y/n): ").lower()
    if choice != 'y':
        print("Goodbye!")
        break
```

Sample Problems

Problem-3:

Write a program that asks the user for a number, and then outputs all the factors of the number.

Example:

Sample Input: Please enter a number: 18

Sample Output: Factors are 1 2 3 6 9 18

Sample Problems

Problem-3:

Write a program that asks the user for a number, and then outputs all the factors of the number.
Example:
Sample Input: Please enter a number: 18
Sample Output: Factors are 1 2 3 6 9 18

```python
# Get user input
try:
    num = int(input("Please enter a number: "))
except ValueError:
    print("Invalid input. Please enter a valid number.")
    exit(1)

# Initialize a list to store factors
factors = []

# Find factors of the number
for i in range(1, num + 1):
    if num % i == 0:
        factors.append(i)

# Output the factors
print("Factors are:", *factors)
```

Sample Problems

Problem-4:

Write a program that asks the user for a number, and then outputs a pyramid of ‘*’s whose largest layer is the entered number.

Example:

Sample Input: Please enter a number: 4

```
*
**
***
****
```

Sample Problems

Problem-4:

Write a program that asks the user for a number, and then outputs a pyramid of ‘*’s whose largest layer is the entered number.

Example:

Sample Input: Please enter a number: 4

```
*
**
***
****
```

```python
# Get user input
try:
    num = int(input("Please enter a number: "))
except ValueError:
    print("Invalid input. Please enter a valid number.")
    exit(1)

# Print the pyramid
for i in range(1, num + 1):
    print('*' * i)
```