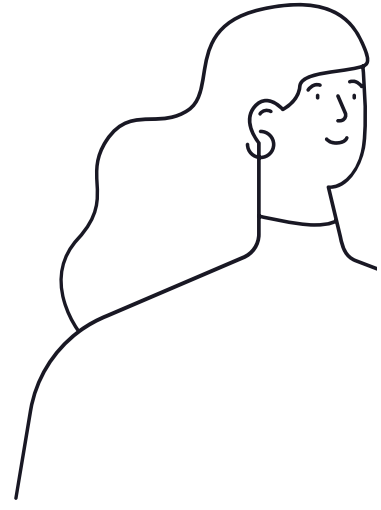
# Welcome Everyone!

## We will wait for others to join in!

KNOW ABOUT ME:



## We Will start in 10

# PYTHON BASICS

With Aaryan Kapur

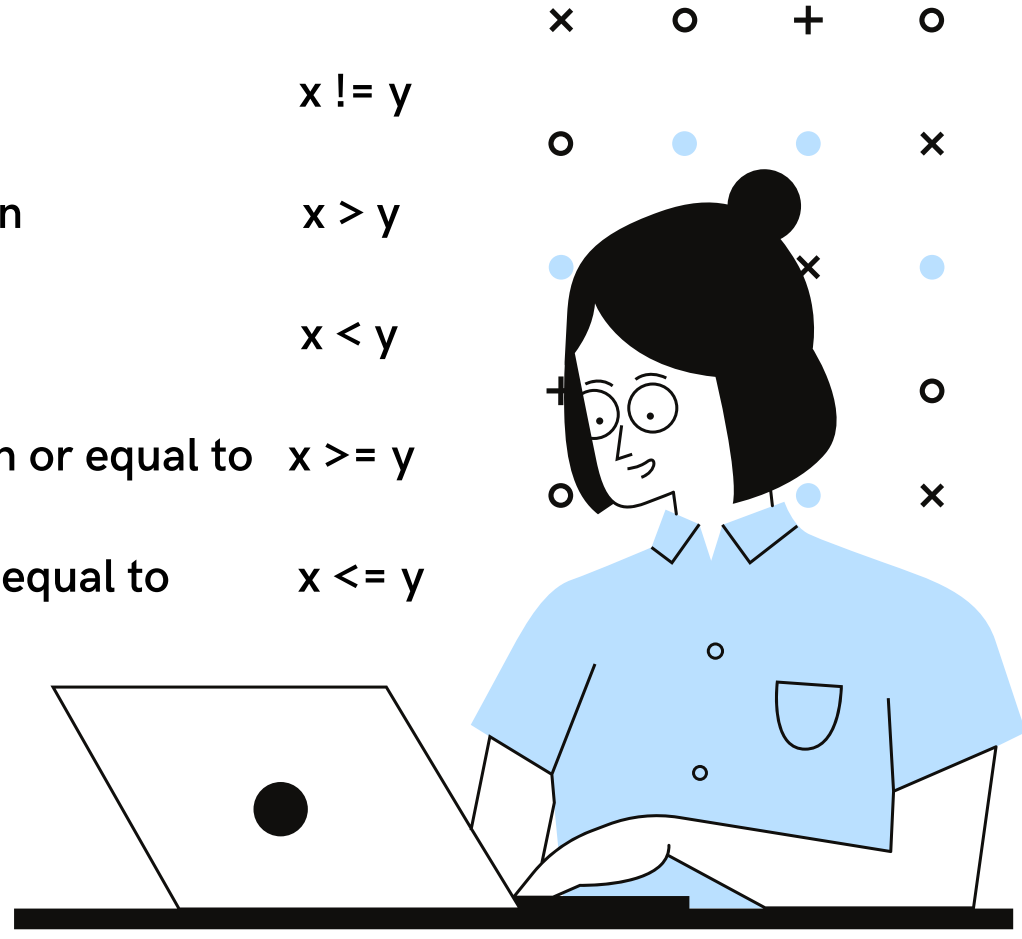
# TOP PERFORMER

## WEEK 2

**Ishita Agrawal**

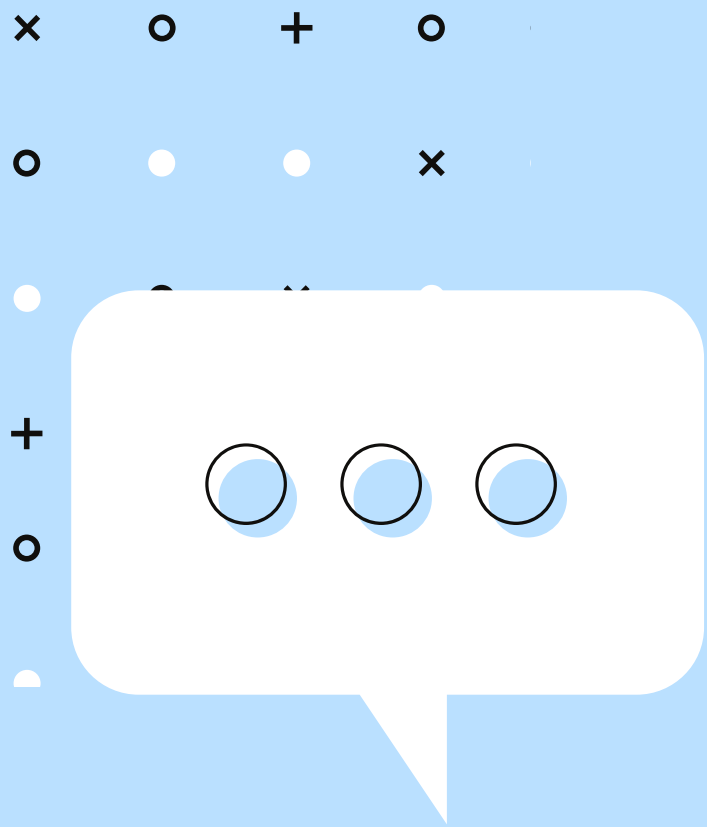
# Comparisons

| | | |
|---|---|---|
| == | 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 |



**To understand value we compare**

Python gives us several ways to compare values.



# Why Compare

**Compare values to understand and build conditions, so your system can respond based on varying inputs/conditions.**

# Comparison Operators

a == b Means a is equal to b

a != b Means a is not equal to b

a > b Means a is greater than b

a < b Means a is less than b

a >= b Means a is greater than or equal to b

a <= b Means a is lesser than or equal to b

# Assignment Operators

| | | |
|---|---|---|
| = | x = 5 | x = 5 |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| *= | x *= 3 | x = x * 3 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 3 | x = x % 3 |
| //= | x //= 3 | x = x // 3 |
| **= | x **= 3 | x = x ** 3 |
| &= | x &= 3 | x = x & 3 |
| \|= | x \|= 3 | x = x \| 3 |
| ^= | x ^= 3 | x = x ^ 3 |
| >>= | x >>= 3 | x = x >> 3 |
| <<= | x <<= 3 | x = x << 3 |

# Assignment Operators

a = b Puts value of b in a

a += b Puts value of a + b in a

a -= b Puts value of a - b in a

a *= b Puts value of a * b in a

a /= b Puts value of a / b in a

a %= b Puts value of a % b in a

a //= b Puts value of a // b in a

a **= b Puts value of a ** b in a

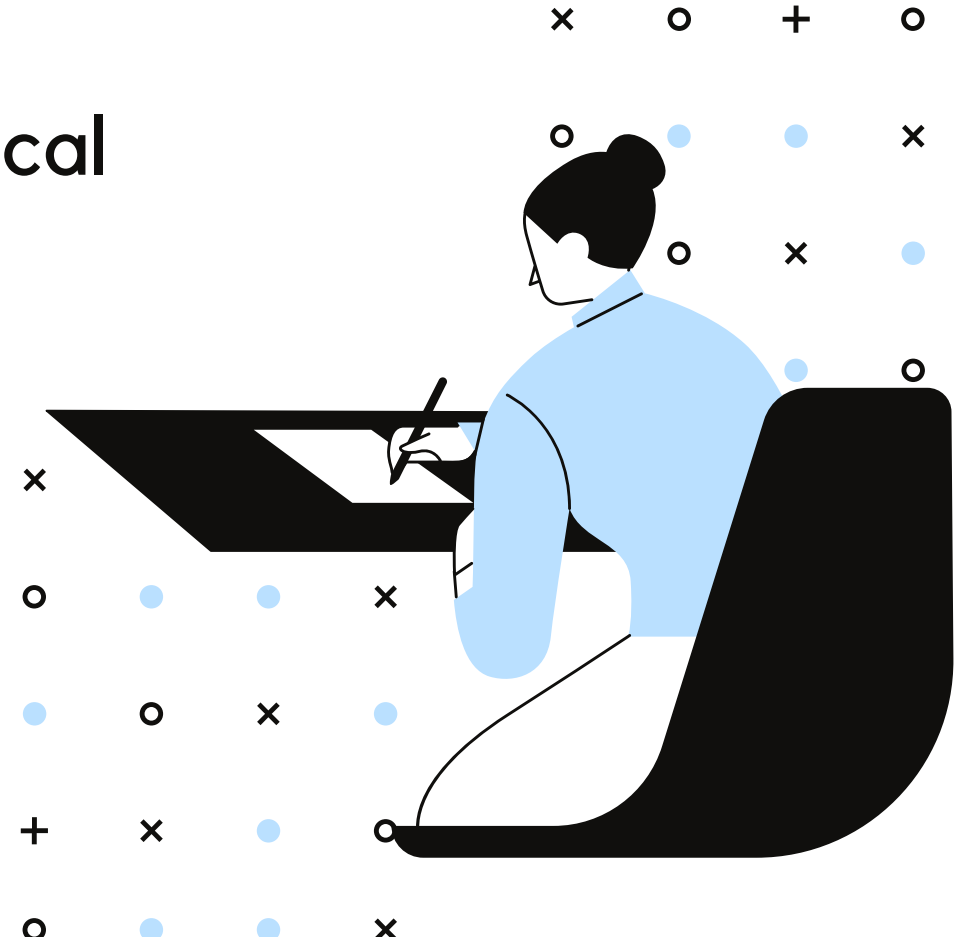
# Logical Operators

Compare based on logical operators

and

or

not



# Logical Operators

**AND**

**a and b: if a and b  both are true then the statement is True.**

**OR**

**a or b: if at least one of a or b  is true then the statement is True.**

**NOT**

**Not(a): it reverses the boolean value of a.**

**Not(b): it reverses the boolean value of b.**

# Identity Operators

**Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.**

**IS**

**IS NOT**

# Identity Operators

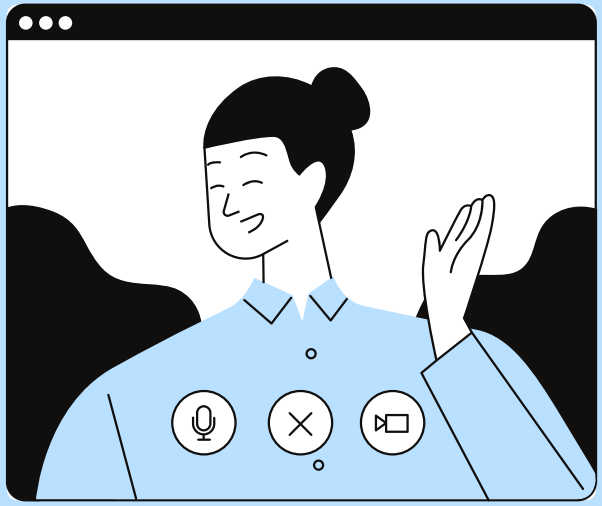
**IS**

**a is b: if a is equal to b  then the statement is True.**

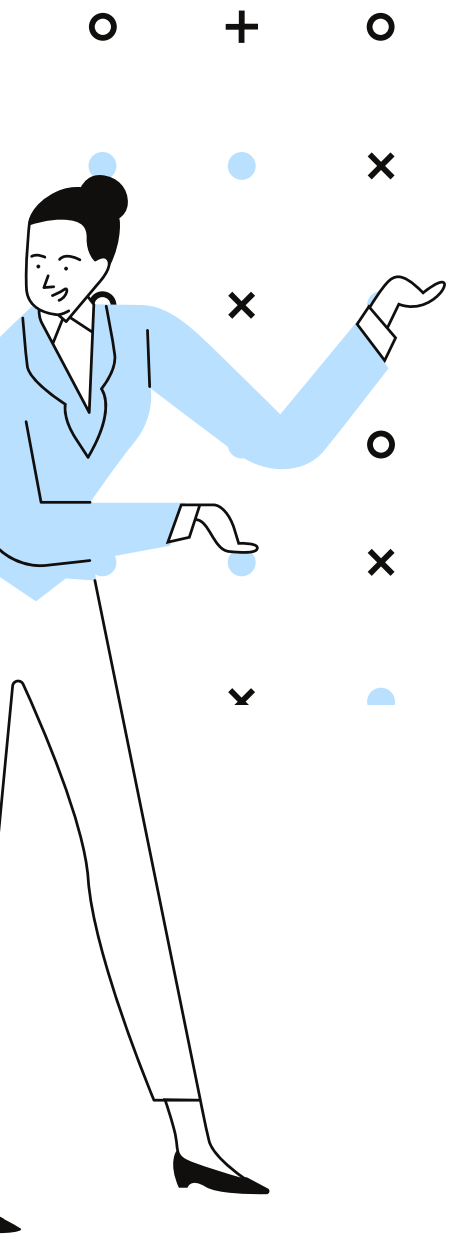
**IS NOT**

**a is not b: if a is not equal to b then the statement is True.**

# Python Operators Precedence



Python operators  operate with precedence that means that they are executed in a priority!



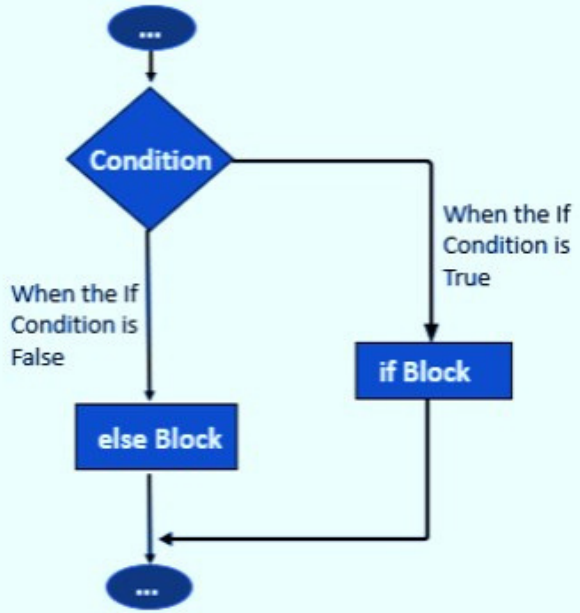
| Operator | Description |
|---|---|
| ** | Exponentiation (raise to the power) |
| ~ + - | Complement, unary plus and minus (method names for the last two are +@ and -@) |
| * / % // | Multiply, divide, modulo and floor division |
| + - | Addition and subtraction |
| >> << | Right and left bitwise shift |
| & | Bitwise 'AND' |
| ^ \| | Bitwise exclusive `OR' and regular `OR' |
| <= < > >= | Comparison operators |
| <> == != | Equality operators |
| = %= /= //= -= += *= **= | Assignment operators |
| is is not | Identity operators |
| in not in | Membership operators |
| not or and | Logical operators |

# Conditional Statements

**We use conditions to build statements based on our comparisons to further make our system dynamic and respond based on input and conditions!**

If Else in

**Python**



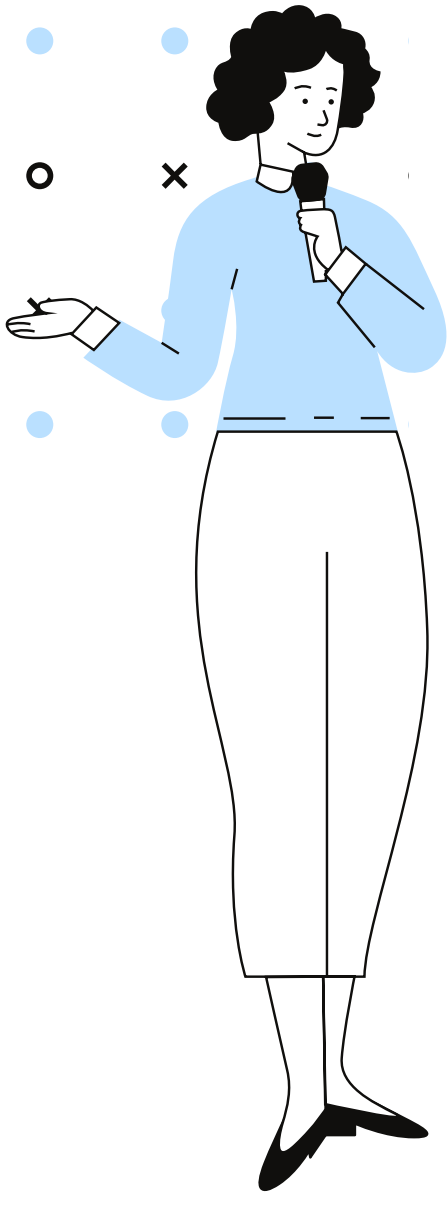
# Conditional Statements

```
a = 10
b = 20

if(a > b):
    print("a is greater than b")

elif(b>a):
    print("b is greater than a")

else:
    print("a is equal to b")
```



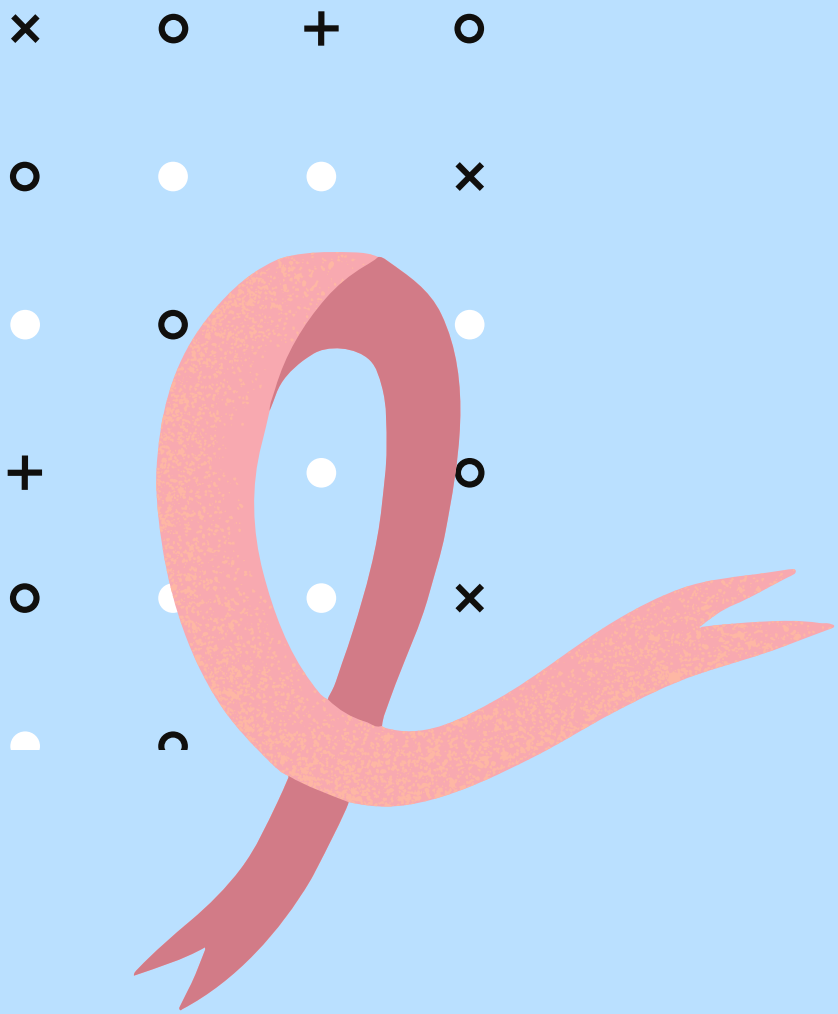
# Multiple Conditions

We can use multiple conditions together to respond.

```
a = 20
b = 30
c = 40
if(a > b and a > c):
  print("a is the largest")

if(b > a and b > c):
  print("b is the largest")

if(c > b and c > a):
  print("c is the largest")
```



# Loops

## For Loop

Once for each item in range.

```
for i in range(0,10):
  print("Cycle:",i)
```

## While Loop

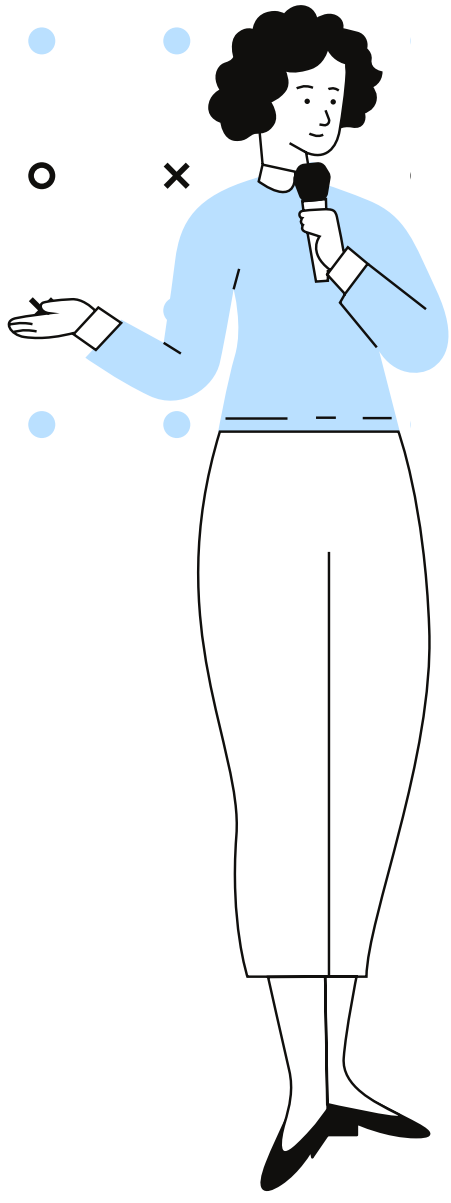
Executes as long as a condition is true

```
i = 0
while(i<10):
  print("Cycle:",i)
  i+=1
```

## Do While Loop

Works with loop and a conditional

```
i = 0
while True:
  print("Cycle:",i)
  i+=1
  if(i>=10):
    break
```



# Loop Control Statements

## Break Statement

Terminates the loop statement and transfers execution to the statement immediately following the loop.

## Continue Statement

Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

## Pass Statement

The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

# Let's get to work

# Let's Build a calculator

# Thank You

Join Here!