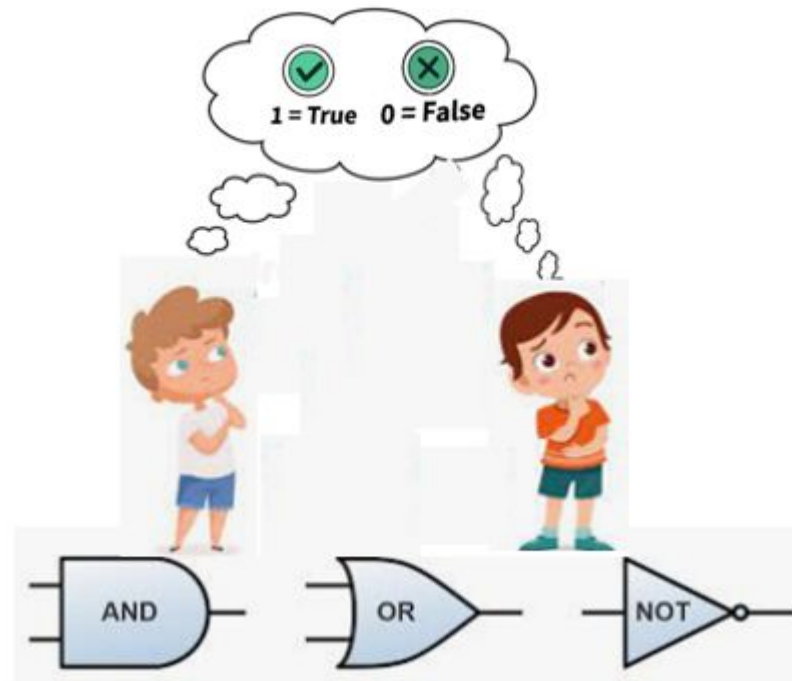


## Lesson 2 – Logical Operators



## Logical Operators




In a way

these are also considered as **rational operators** that belong to the **boolean category** of coding.

**Rational operators** discussed above were analysing **only one condition**, & declaring the result based on the outcome of that condition.

**Logical operators** analyse **two conditions**, & declare the result, only after **further refinement** of their analysis.

**Thus, we have 3 types of logic operators.**

<u>Operator</u>	<u>Meaning</u>
 The icon for the 'and' operator, consisting of a green hexagon with the word 'and' in white text.	Result true if both true
 The icon for the 'or' operator, consisting of a green hexagon with the word 'or' in white text.	Result true if either true
 The icon for the 'not' operator, consisting of a green hexagon with the word 'not' in white text.	Result true if neither true

## Application of Logical Operators

In computer science, Logical operators define a very important operation of electronics called **Gates**.

While normal gates control movement of men & material on roads, these **gates control the flow of current** in a circuit by either allowing it or stopping it.

In so doing, we decide the conditions under which to switch on or switch off, a device or a circuit, from one stage to the other.

To do this the code **keeps monitoring the conditions**, till they become **True** or **False** as the case may be.

To understand more, let us take the example of **“Eating a Dosa”**.

In this ex, we want to **Make a Decision** on eating the dosa, based on **evaluation of two conditions, that have been specified by us:**

- **First condition** is - Am I feeling hungry?
- **Second condition** is - Has the Dosa Arrived?

This **Decision Making** takes place using three options or methods (also referred to as three different gates).

These **Methods or Gates** would then give us three distinct outcomes or results.



**AND Gate.** This gate allows me to eat only If the Dosa has arrived **AND** I am hungry.

Both first, **AND** second Conditions must be True. Thus its name.

**OR Gate.** This will allow me to eat the Dosa if either the Dosa has arrived, **OR** I am hungry.

Needs **first, OR second condition to be True**. Thus its name.

**NOT Gate.** This is like the crazy man, who does **double evaluation**.  
First evaluation as a **NOT** gate  
& the second as a **AND** gate.

The result is announced as per the second evaluation & **NOT**  
as per the first. Thus its name.

## How Exactly does NOT Gate Work?

Evaluation under NOT is done in two sperate parts.

In Part 1, this gate first checks the first condition - if Dosa has arrived as a NOT gate.

Doing so it automatically **reverses** its outcome. No questions asked.

**In part 2**, it then compares this **reversed outcome** with second condition as a **AND** gate & gives the result based on AND

Thus in this ex:

If the Dosa has indeed arrived, the NOT gate will first take the outcome as **NOT** Arrived.

It will now compare Not arrived with Am I hungry to give the decision.

Thus, even if the Dosa has actually arrived, & I am actually hungry, NOT gate will **Not grant me access** to the Dosa & I will **Not get** to eat.

This is because the second evaluation is being done as an **AND** gate, requiring both conditions after its first evaluation, to be true.

**It is important to understand this logic of NOT gate.**

**Code is changing the World.**

**Typical application of a NOT operator is to do the inversion process in an inverter using a NOT gate.**

**NOT gate would take low voltage as an input, & reverse or invert it into high voltage at the output.**

**Simple as that.**

**Till a few years back all domestic inverters were made using hardware.**

**They were thus bulky, heavy & had a big cost attached.**





These days you may have heard of a thing called **“An Inverter Refrigerator.”**

Have you ever wondered what exactly it is?

It is nothing but the **normal refrigerator that is using a device based on a NOT gate** instead of the bulky external inventor of the past.

In other words, **it is controlled by a Code you write.**



**If as a child:**

- You can **understand this important value of coding**  
You will start **Loving it.**
- And then  
You can become a **part of the change.**

## Project 1 . Making an Eligibility App for Exams

This project explains the use of **AND** Operator.



We have made a block containing **two conditions**. →

If both are true, it returns true; otherwise, it returns false.



**Forever loop keeps checking the conditions**  
till both become true. After that it announces the result.

**Important Note:** In coding result of evaluation:

- **True** is represented by a **1**.
- **False** by a **0**.

## Project 2 . Making an Examination Eligibility App Method 2.

This uses the OR operator.

It **also takes two expressions** as **evaluation conditions**.

If either expression is true  
the operator returns true.  
It returns false  
only When the two expressions are both false.



***Note: In OR, if either value is non-zero the result is 1.***

***If first is non-zero, then the second is skipped & not evaluated.***

### Project 3. Making an Examination Eligibility App Method 3.

Kindly study ex carefully.

In this, the code will **forever** check the value inside score variable.

Being NOT, the comparison will be true if its value is not greater than 100.



Thus, **as long as comparison is true**, the sprite will keep saying **“Sorry, you cannot go to Level 2”**.

## Enhancing the Code

The code could have been extended to - When the condition becomes false, ie the score exceeds 100 the sprite could have said **'Congrats. Proceed to level 2'**.

Try doing it **yourself**.

Remember in case of not block **False is true & True is false**.



# To Consolidate

- A relational operator works on the relationship between two entities.  
Ex between Price & its Value being  $<$ ,  $>$ , or  $=$  to another.
- They work with numbers, as well as with letters.
- In Scratch, letters at the top of the alphabet stack (a, b, c), are valued less than letters at end (x, y, z).
- Logical operators analyse one or two conditions, & declare the result, only after further refinement of their analysis.
- Condition True is represented by 1, & condition False by 0.

- Logical operators also define a very important operation of electronics called **Gates**, that allow or stop the flow of current in a circuit.
- **AND** evaluation results in True, if **both** conditions are True.
- **Or** evaluation results in True, if **either** condition is True.
- **NOT** evaluation results in True, if **neither** condition is True.
- In **NOT**, False finally results in True, & True results in False.



**Code Karega India Badhega**