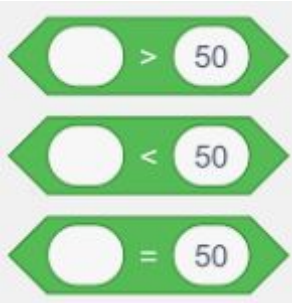
Lesson 1 – Relational Operators



The World of Operators

In computer science, an operator is a character or characters that determine the action that is to be performed or considered.

Types of Operators

There are three main types of operator that programmers use:

- Arithmetic operators.
- Relational operators.
- Logical operators.

We have seen the arithmetic operators in Level 1.

Relational Operators

What are Relational Operators

A relational operator:

- Tests, or
- Defines, or
- Establishes,

Some kind of relationship between two entities.

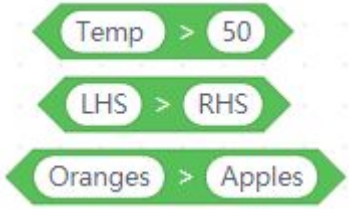
Operator	Example	Relationship
> 50	Price > 50	Greater than
< 50	Price < 50	Less than
= 50	Price = 50	Equal to

Ex, relationship between Price & its Value being <, >, or = to another.

Project 1. Greater Than – Automatic Temp Control in a room.

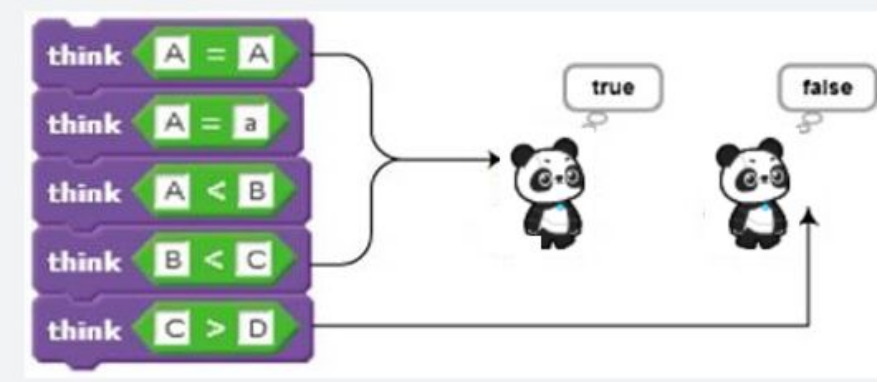
Its block statement is: 

The two roundels could be changed by us, as shown in these examples.



Important point to note is, this operator also works with letters.

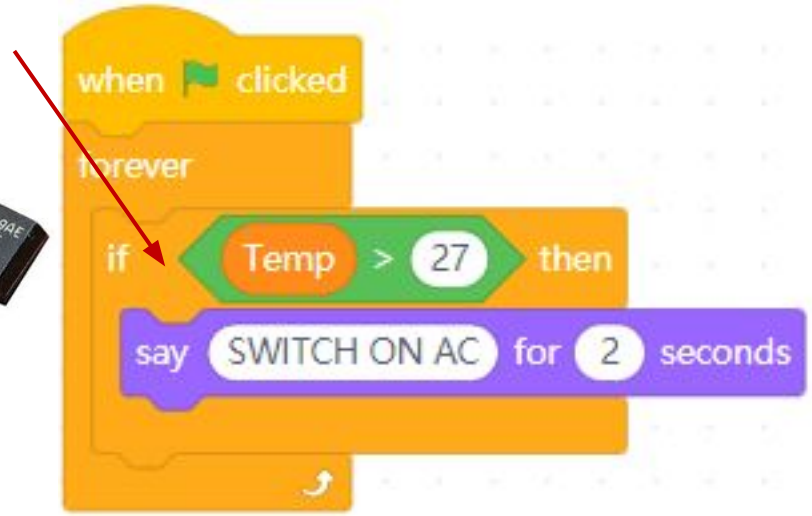
In Scratch, letters at the top of the alphabet stack (a, b, c) are valued less than letters at end (x, y, z).



Thus, RHS is greater than LHS & Apples are less than Oranges.

Its Typical Use:

- This is placed in the If Then control Statement block.
- This now checks if the first value, is greater than the other.
- In so doing, it works with the value of a variable (temp as shown by say a sensor (temp sensor monitoring the temp in the room.



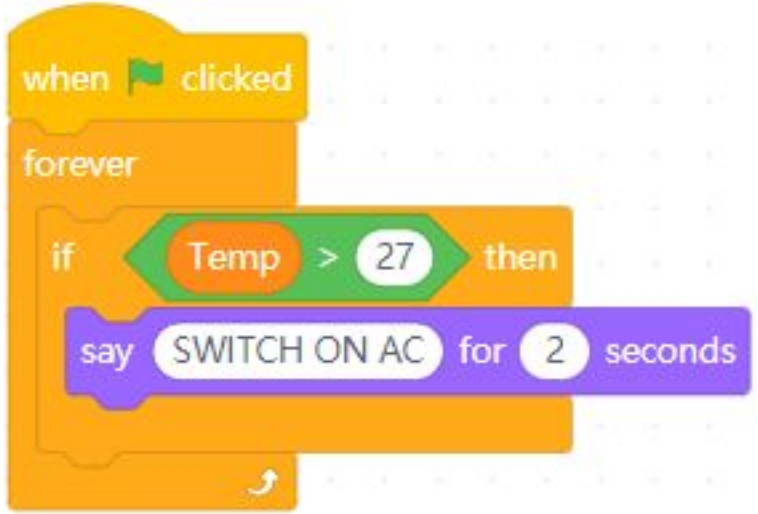
Note: We shall learn all about making variables in next lesson.

How does it Work?

The sensor keeps monitoring the environment.

As & when it becomes greater, this code returns *true*. If not, it returns *false*.

If it is true, it sends instructions to a switch that triggers the AC to go on.



This instruction is sent as a separate broadcast message to a switch that is coded separately to receive the broadcast message & act accordingly.

We shall learn more about broadcast ahead.

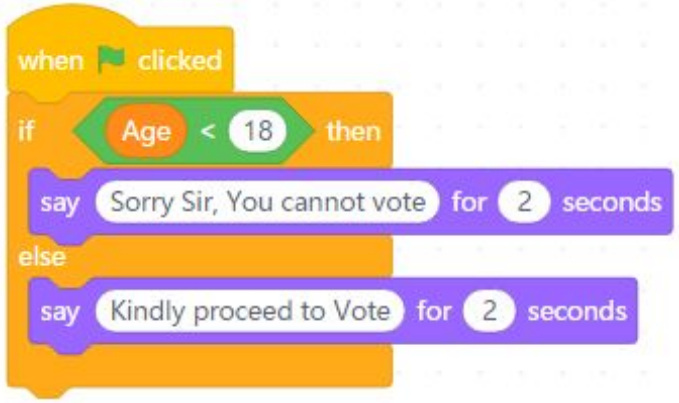
Project 2. Less Than – Making an Age check module for Voter eligibility

Its block statement is: 

Its changeable are similar to project 1.

A typical daily life project for this could be the making of the age check module for a voter eligibility app.

In the code note this block statement.



This will be capable of giving two outputs:

- If the voter is underage, it will broadcast a vote denial message to the eligibility app.
- If the age is ok, it will trigger a Proceed to vote message.

Point to note is, the code has taken a decision, based on the comparison of the two conditions.

This is how automation works.

Project 3. Equal to – Making a Basic Football Match Timer

Its block statement is: 

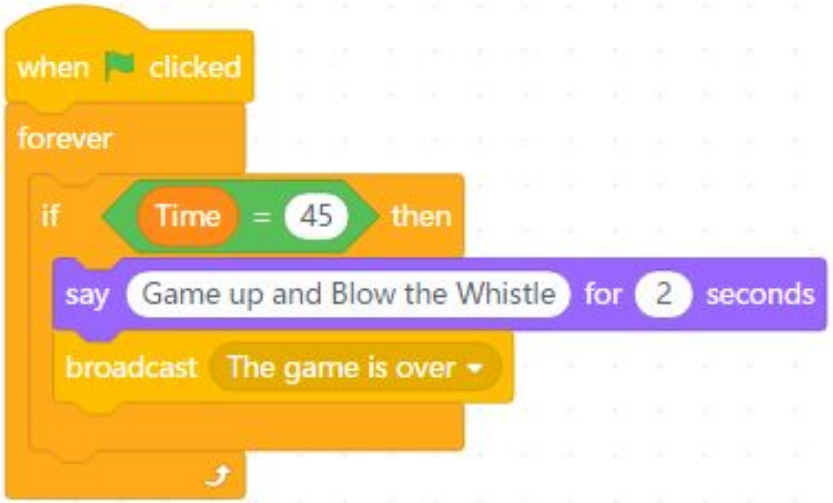
Its changeable are similar to project 1.

This is the code for a **basic football match timer**.

In this, the code, keeps checking if first (timer) value is equal to the other (specified) value.

When, the values become equal, the block returns *true* or displays given messages.

If not, it returns *false* and takes no action & game continues till 45 min are over.



Logical Operators

Logical Operators

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

Relational 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.

Operator	Meaning
and	Result true if both true
or	Result true if either true
not	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".

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**:

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?

This is done in two separate 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.

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.**

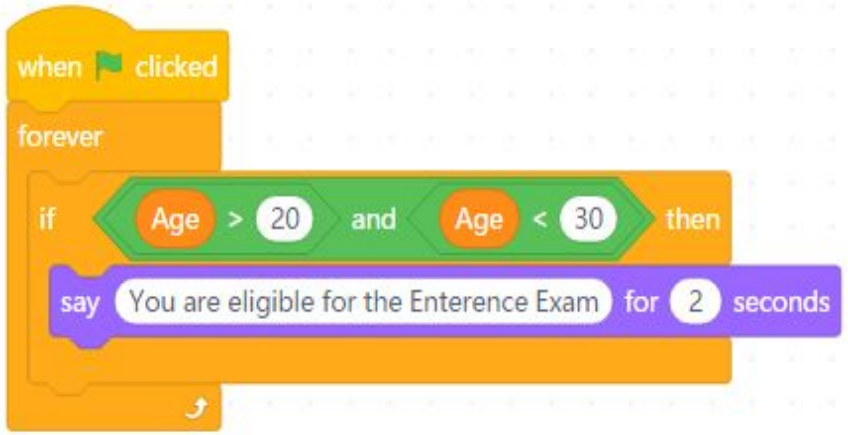
If as a child:

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

Project 4 . Making an Eligibility App for Exams.

This project explains the use of
AND Operator. < 50 and

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 5 . 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: If either value is non-zero the result is 1. If first is non-zero, then the second is skipped & not evaluated.

Project 6. 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

THANK YOU Children & THIS WAS A VERY IMPORTANT LESSON