Logic

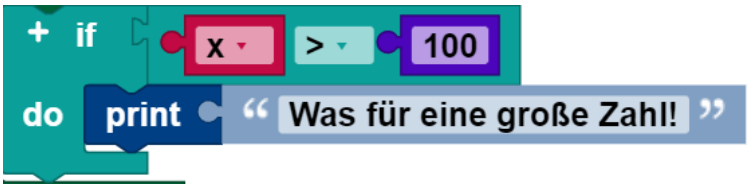
Conditional statements

Conditional instructions are essential for programming. They make it possible to formulate case differentiations, such as:

- If there is a path to the left, turn left.
- If the number of points = 100, press “Good job!”.

if blocks

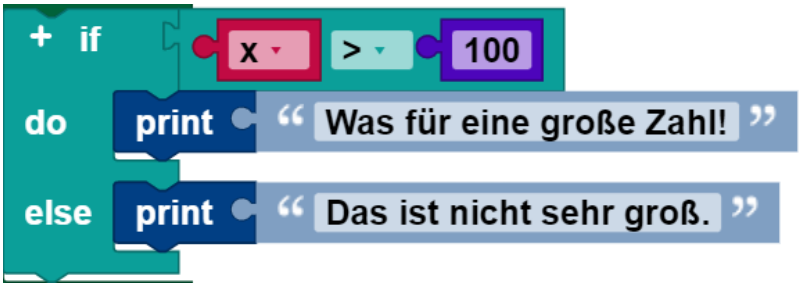
The simplest condition is an **if** block:



When it is executed, the value of the variable **x** is compared to 100. If it is larger, then “What a large number!” is output. Otherwise, nothing happens.

if else blocks

It is also possible to indicate that something should happen when the condition is false, as in this example:

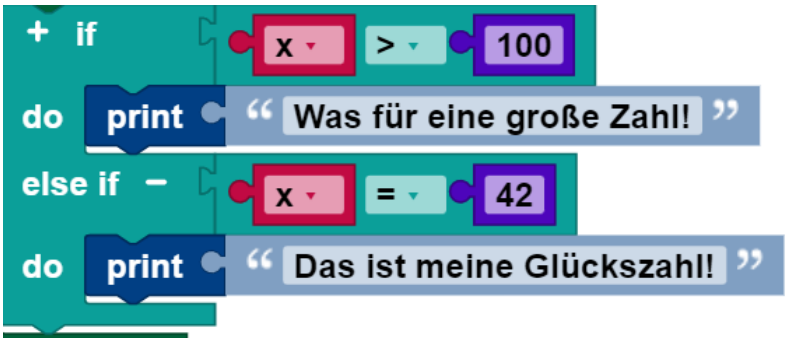


As in the previous block, “What a large number!” is output when **x** > 100. Otherwise “It's not very big’ is output.

An **if** block may have a **do** section, but not more than one.

if do else if blocks

It is also possible to test multiple conditions with a single **if** block, by adding **do else** clauses:

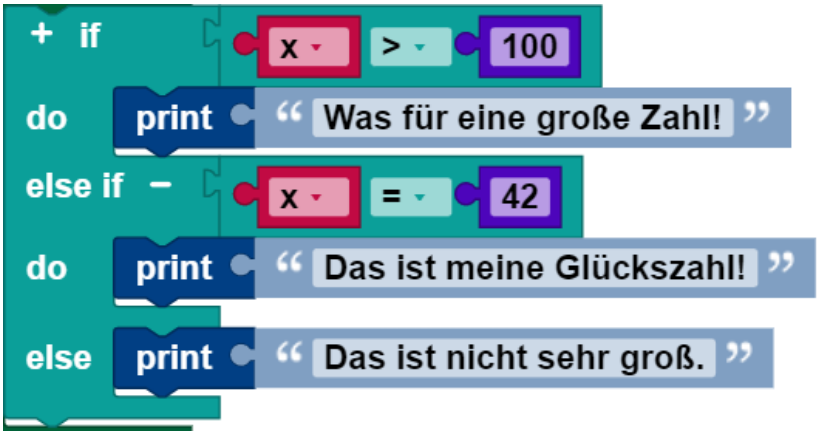


The block checks first whether **x** > 100, and outputs “What a large number!” if this is the case. If this is not the case, it then checks whether **x** = 42. If so, then it outputs “That is my lucky number!”. Otherwise, nothing happens.

An **if** block can have any number of **if do** sections. The conditions are evaluated from top to bottom, until one of them is fulfilled, or until there are no more conditions left.

if do else if do else blocks

if blocks can have both **if do** and **else if** sections:

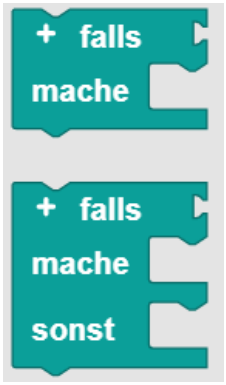


The **else if** section guarantees that an action is executed, even if none of the previous conditions is true.

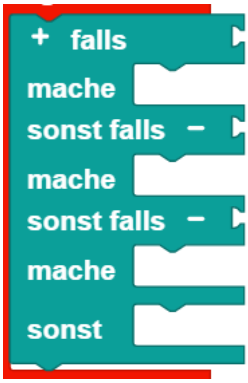
An **else if** section can also occur after any number of **if do** sections, including zero, which would then be a completely normal **if do** block.

Block modification

Only the simple **if** block and the **if do** block appear in the tool list:



To add **if do** and **else** clauses, click the (+) symbol. The (-) symbol can be used to remove **else if** clauses:



Note that the shapes of the blocks permit any number of **else if** sub-blocks to be added, but only up to one **if** block.

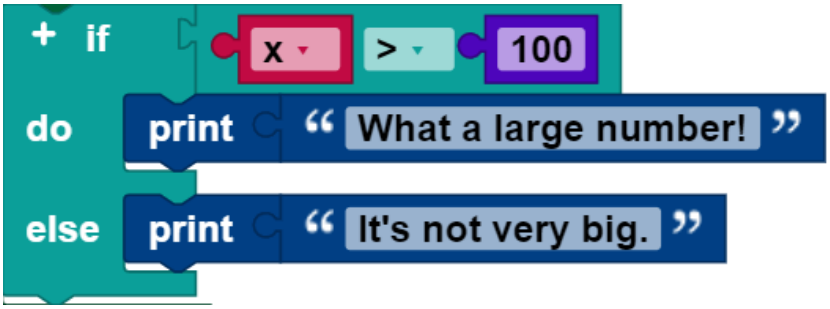
Boolean logic

Boolean logic is a simple mathematical system with two values:

- **true**
- **incorrect**

Logic blocks in ROBO Pro Coding are generally there to control conditions and loops.

Here is an example:



If the value of x is not greater than 100, then the condition is false, and “It’s not very big.” is output. If the value of x is not greater than 100, then the condition is false and “It’s not very big.” is output. Boolean values can also be saved in variables and transmitted to functions, just like numbers, texts, and list values.

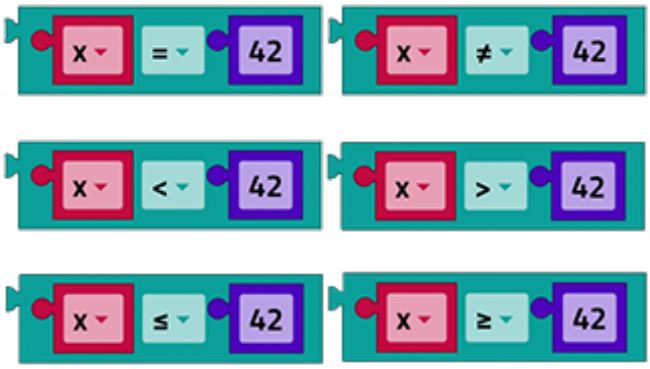
If a block expects a Boolean value as an input, then no input will be interpreted as **false**. Non-Boolean values cannot be inserted directly where Boolean values are expected, although it is possible (but not advisable) to save a non-Boolean value in a variable and then insert this into the condition input. This method is not recommended, and its behavior can change in future versions of ROBO Pro Coding.

Values

An individual block with a drop down list that either indicates **true** or **false** can be used to access a Boolean value:

Comparative operators

There are six comparative operators. Two inputs are entered into each (normally two numbers), and the comparative operator returns **true** or **false**, depending on how the inputs are compared to one another.



The six operators are: equal, not equal, less than, greater than, less than or equal, greater than or equal.

Logical operators

The **and** block returns **true** if and only if its two input values are true.

The **or** block returns **true** if at least one of its two input values is true.

do

The **not** block converts a Boolean input into its opposite. For example, the result of:

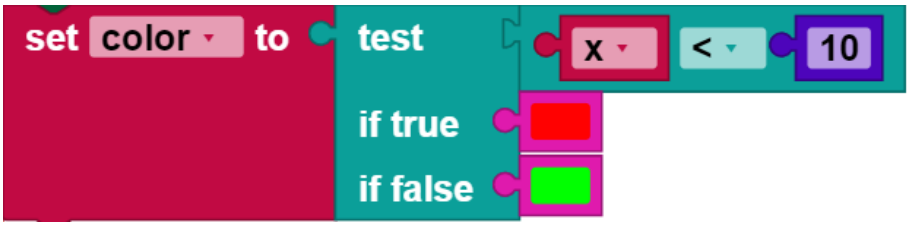
is **false**.

If there is no input, then the value **true** is assumed, so that the following block will generate the value **false**:

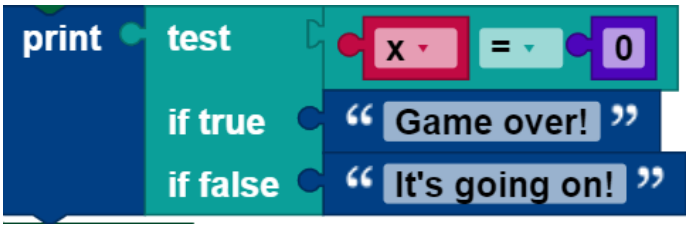
However, leaving an input empty is not recommended.

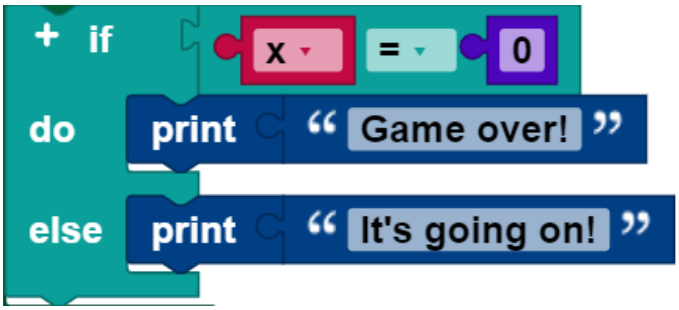
Three-part operator

The three-part operator acts like a miniature **if do** block. It uses three input values The first Boolean condition to be tested is the first input value, the second input value is the value returned if the test is **true**, and the third input value is the value returned if the test is false. In the following example, the variable **color**is set to red if the variable **x** is less than 10, otherwise the variable **color**is set to green.



A three-part block can always be replaced by an **if do** block. The following two examples are just the same.





Revision #7
Created 21 February 2022 15:50:51 by Admin
Updated 8 November 2024 16:56:02 by phuesing