

# Processing

## Types of operations

The operations to be performed with the data can be:

- **Arithmetic**: classic operations of addition, subtraction, multiplication and division or maths operations.
- **Logics**: comparisons, negation, AND, OR.
- **Concatenation**: union of several elements (strings of characters or variables of different types)
- **Loops**: Involves performing actions **repeatedly**. In this case it will be convenient to distinguish two types:
  - Previously known number of times to repeat the action: we will use the **For** or **Repeat** structures.
  - Number of times to repeat depending on values obtained: we will use the structures **While** (While...do) or **Repeat until** (Do...while), depending on whether we want to evaluate the condition before or after the first iteration. We will see that these structures are conditional as well as repetitive.
- **Conditional**: implies carrying out some actions or others by making **decisions**. Structures If-then (If-else), and Switch.

We have already seen the first three types of operations in the previous examples. We will now focus on the last two, although we will introduce various operations in the examples to delve into their use.

## Iterations and loops

To practice with these structures, we will create a simple program that will calculate the average of several numbers (marks, for example). In this case, since the number of iterations is known, we will use the **FOR/REPEAT** statement. Likewise, we will use **arithmetic** and **concatenation** operators.

### Program: Average of n marks

#### Statement:

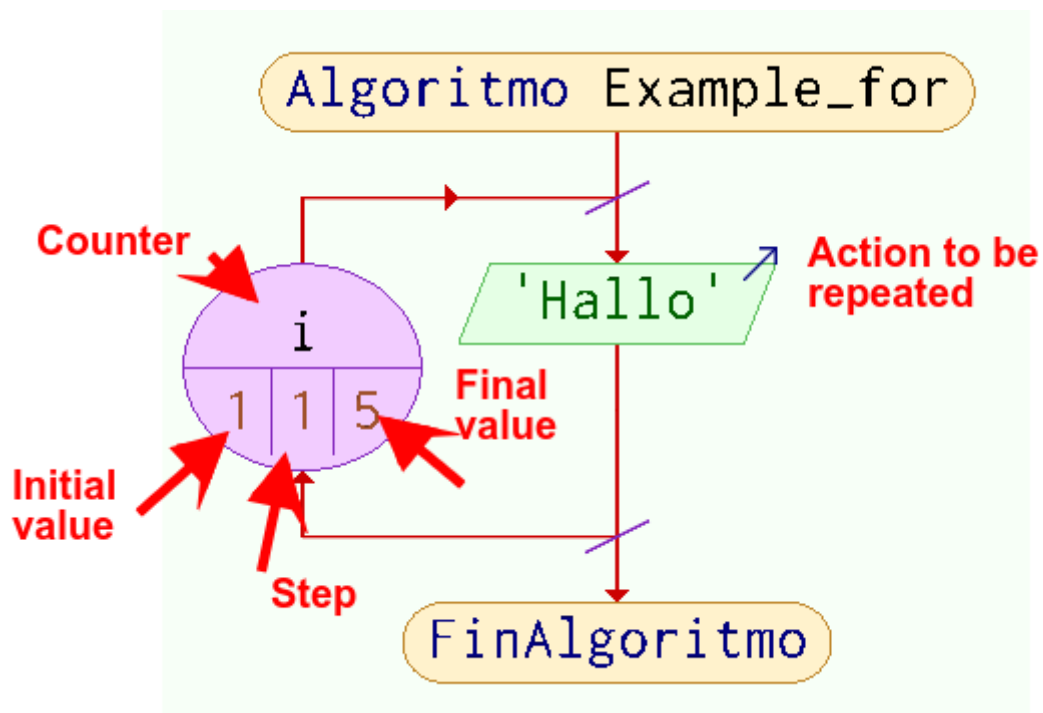
Create a program to calculate the average of several numbers . First, the program requests the number of marks to average, then it asks that the marks be entered as many times as we have told it (here is the repetition). At the end, it shows the average.

#### Steps 1 and 2: Analysis and flow chart of the *Average of n marks* program

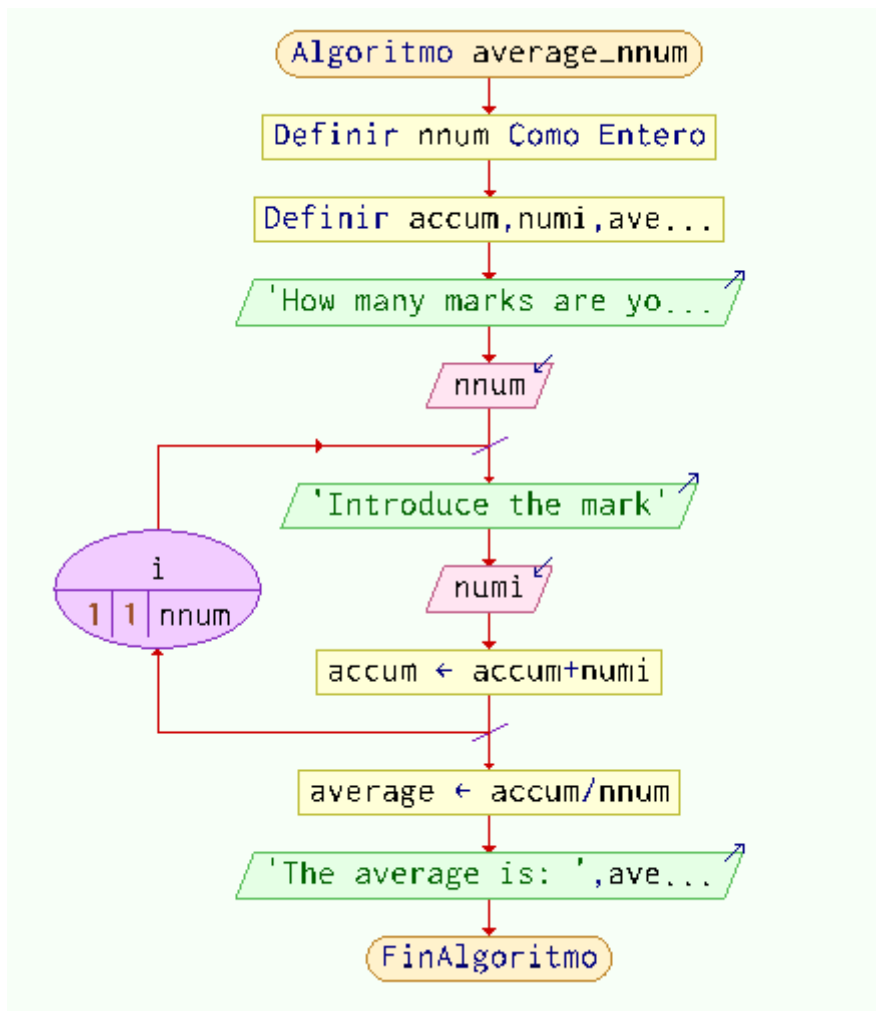
The elements involved are:

- **Start** and **end** of algorithm.
- **Outputs:** Request number of marks, show average.
- **Inputs:** number of marks and marks to be averaged.
- **Storage:** number of elements (integer), entered numbers (real number), cumulative sum of numbers (real number), and average (real number)
- **Processing:** loop, concatenation, addition and division.

The flowchart associated with the **FOR/REPEAT** instruction is as follows:

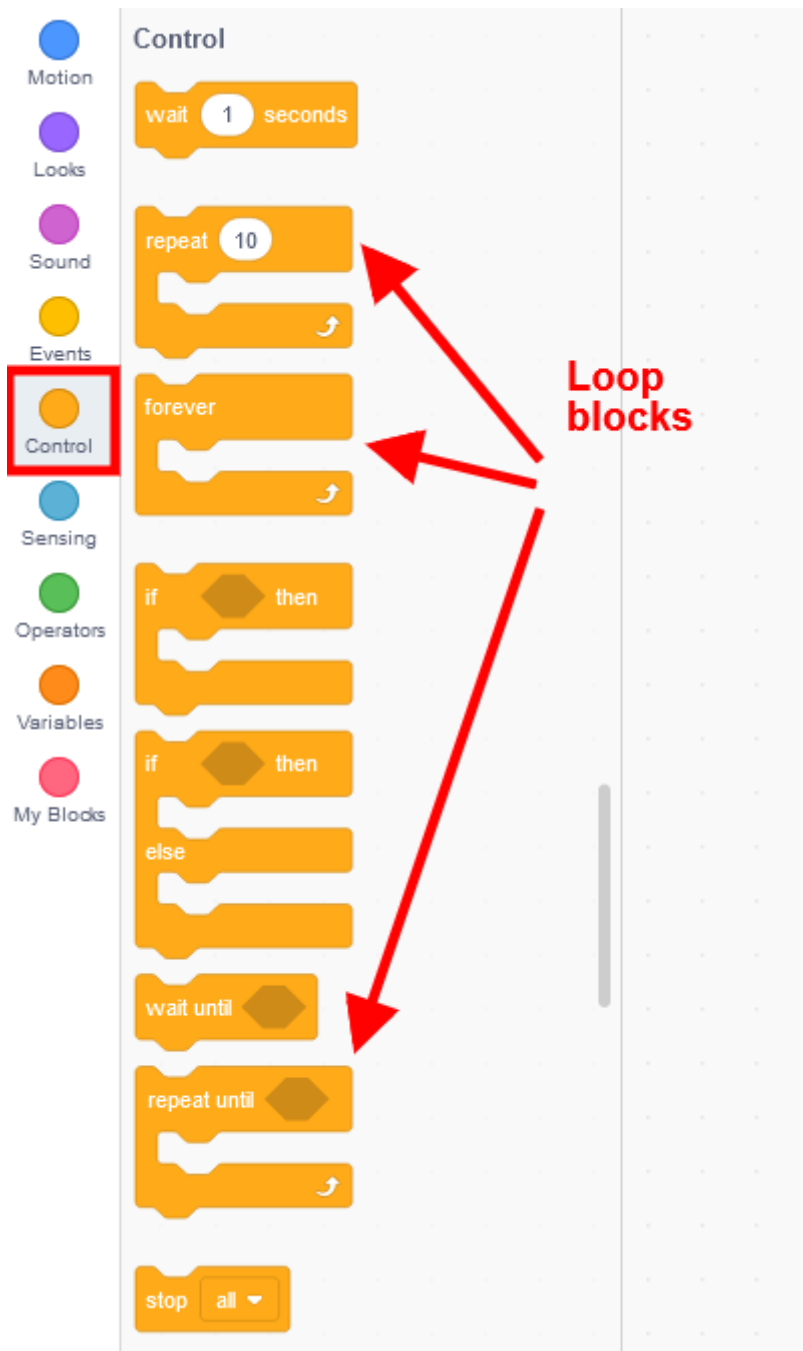


**Flowchart:**



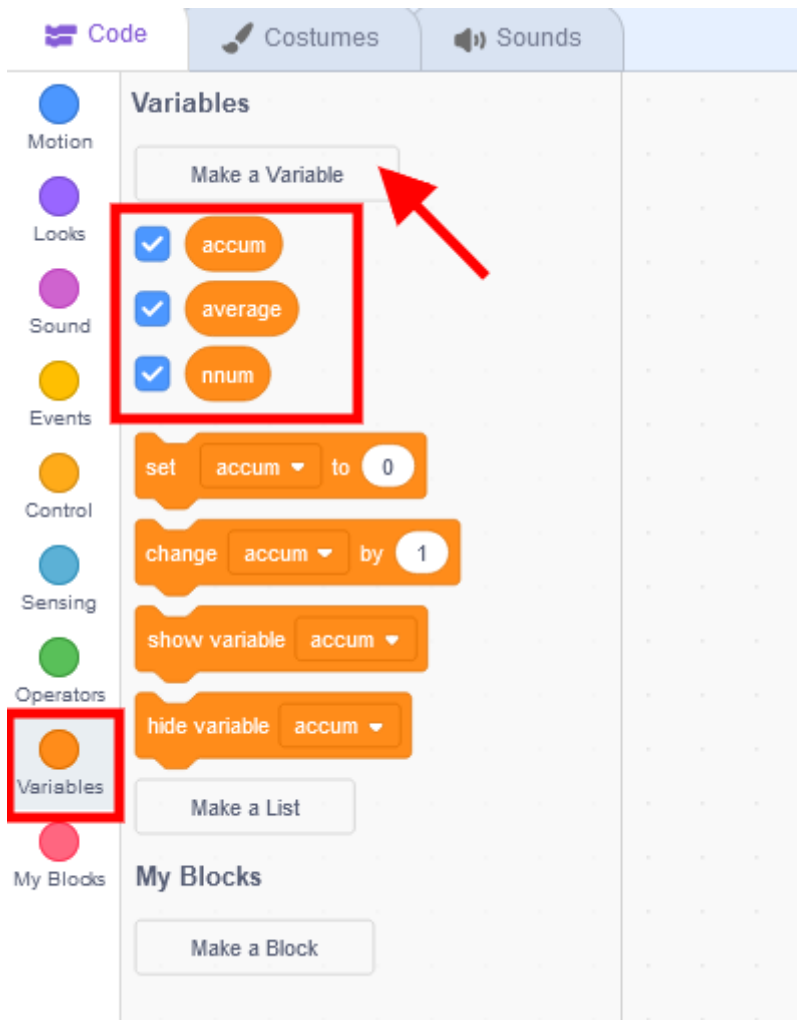
### Steps 3, 4 y 5: Coding, compilation and checking of the *Average of n marks* program with Scratch

In Scratch the blocks related to repetitive structures are found in **Control** and are **Forever**, **Repeat** and **Repeat until**.

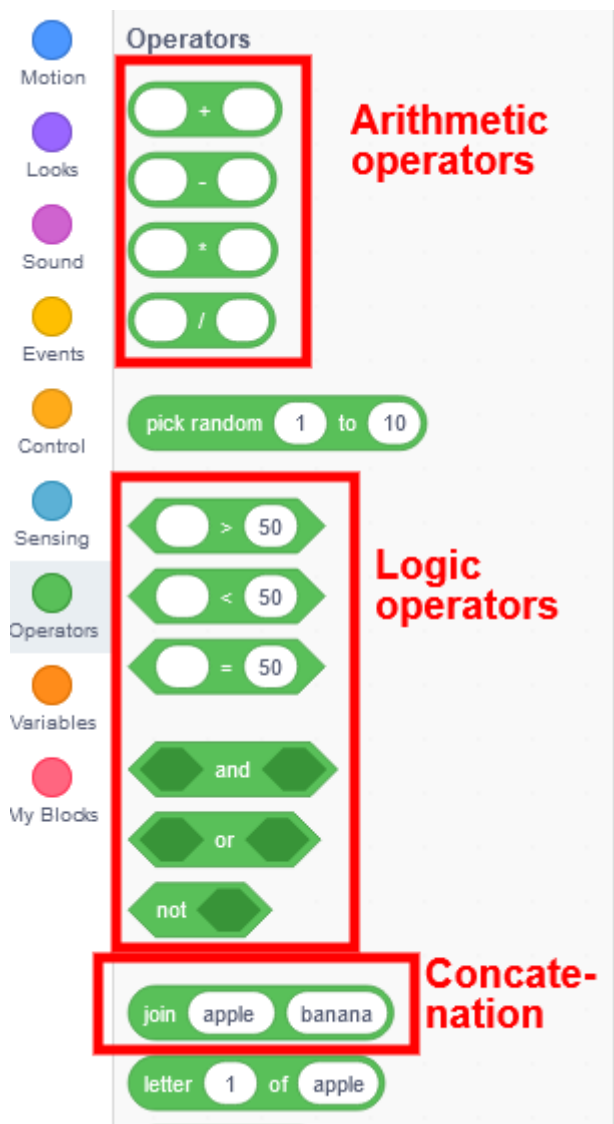


In our cas, as the number of iterations is defined, we will use **repeat**.

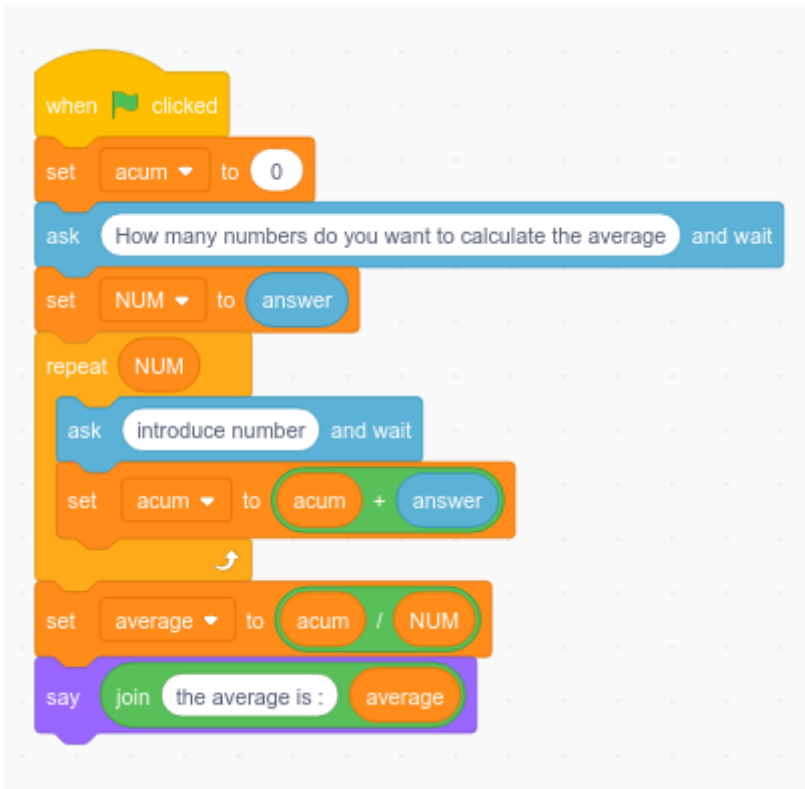
First of all, in the **Variables** block we will define the necessary variables.



In an analogous way, using the **Sensing**, **Looks**, and **Variables** blocks explained in the Inputs, Outputs and Storage sections, we build the program. In Operators we will find the blocks necessary to carry out arithmetic, logical and concatenation operations.



The final program with all the elements would look like this:



Save the program with the name ***averagenmarks.sb3***

Be careful with setting the initial value at the beginning of your variables, as it happens in the example with acum.

## Conditionals

To practice with these structures, we will carry out a simple program in which

### Program: Report

#### Statement:

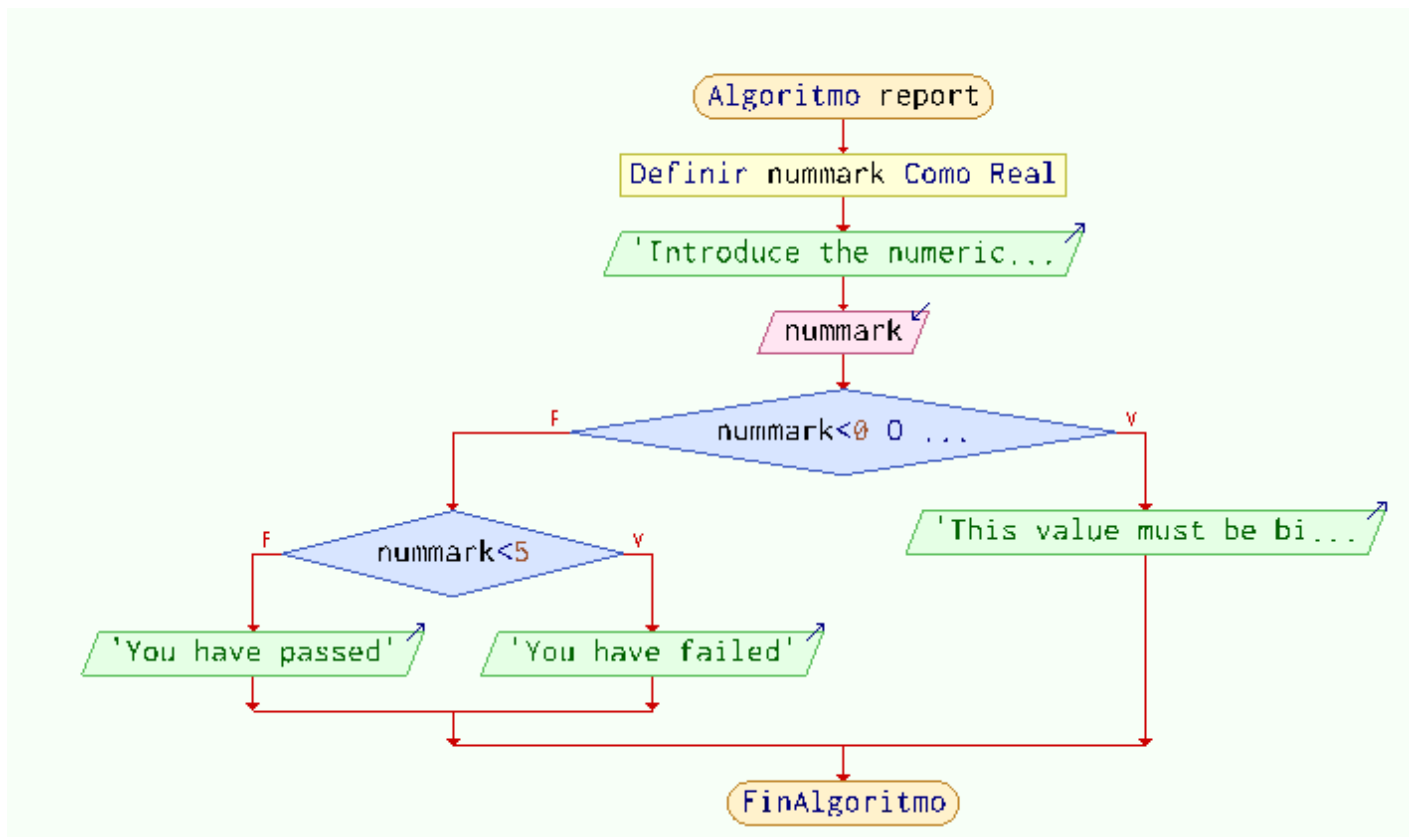
Create a program that requests a numerical grade, verifies if the value received is correct (between 0 and 10) and finally tells us whether the grade corresponds to a PASS or a FAIL.

#### Steps 1 and 2: Analysis and flowchart of the *Report* program

Involved elements are:

- **Start** and **end** of algorithm.
- **Outputs:** Request numerical mark, show pass or fail or error message.
- **Inputs:** numerical mark
- **Storage:** numerical mark (real number, may be decimal)
- **Processing:** logical (comparison, AND, OR) and conditional

#### Flowchart:

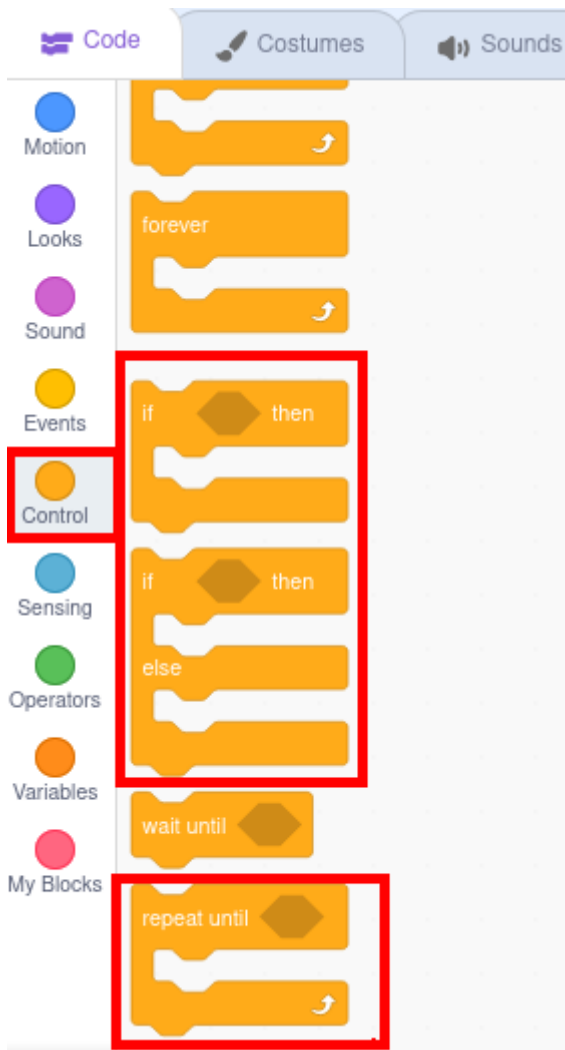


In this case we have defined the error condition using the logical operator **OR**, indicating that any mark less than zero **or** greater than 10 is considered erroneous. This program can be solved in an analogous way using the operator **AND**, indicating that it considers any numerical mark valid greater than or equal to zero **and** less than or equal to 10.

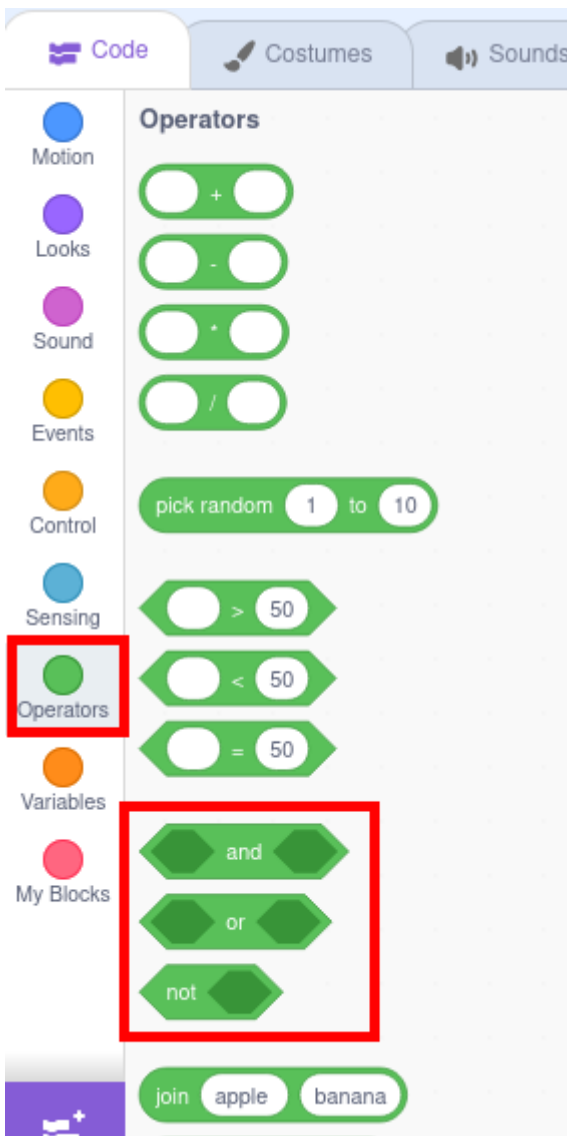
### Steps 3, 4 y 5: Coding, compilation and checking of the *Report* program with Scratch

In Scratch the blocks related to conditional structures are found in **Control**, next to the loops.





The logical operators AND and OR are found in **Operators**, along with the comparison operators and those already seen previously: arithmetic, concatenation...

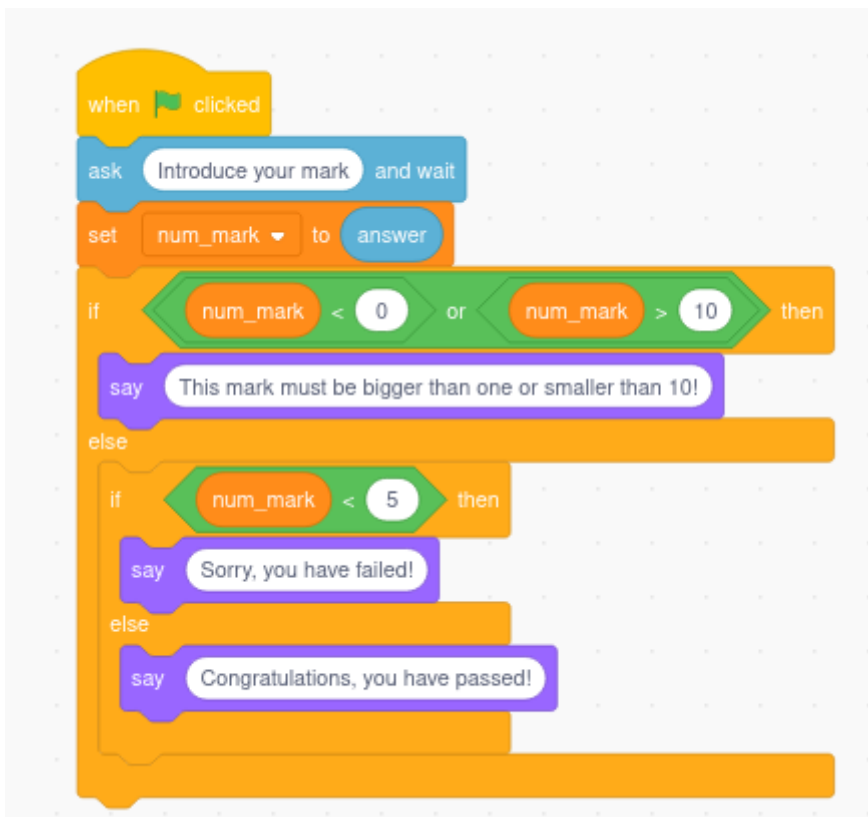


As in all programs, we will start by defining the necessary variables, in our case only one from the **Variables** blocks.

Next we will request the rating using the block available in **Sensing** and we will store its response in the newly created variable.

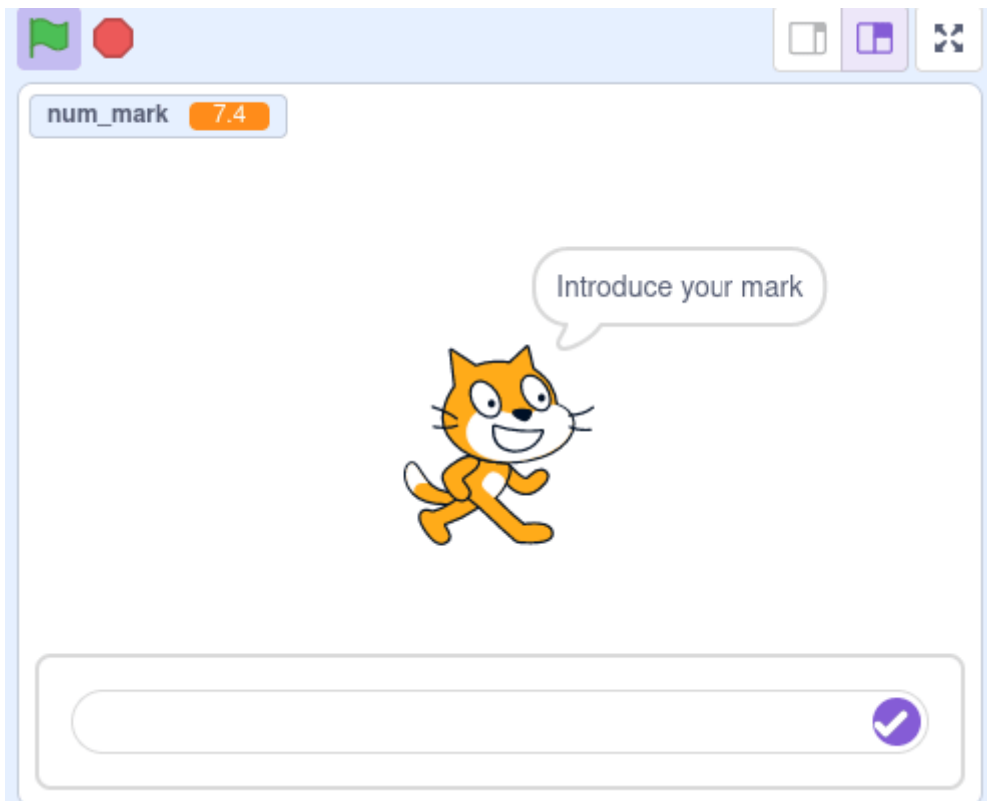
Finally, we will evaluate the response obtained through the corresponding **Control** block and using the logical operators of **Operators**, depending on their value, we will offer the error message or write the final grade with the corresponding **Looks** block.

The resulting code would be the following:



In this case we use the **OR** operator because Scratch does not have the combined operators less than or equal and greater than or equal, and thus we save ourselves from having to combine both operators with a logical operator OR. Remember the KISS principle.

Finally, we verify the correct functioning of the program and its robustness against the most common errors. In Scratch we **execute** by clicking on the green flag.



Save the file with the name **report.sb3**

---

Revision #16

Created 15 October 2023 18:58:22 by Ana López Floría

Updated 25 January 2025 13:43:14 by Ana López Floría