# INTRO TO *PROGRAMMING*: COMPARISONS

Imagine you are at your local fish and chips shop, staring at the specials board with ravenous eyes. 🐟🍟

With so many options available and limited money, you need to make a decision. How though? Probably by comparing the price and the contents of each of the available specials. 💰

For a human like you, identifying the lowest price is easy (I hope). But what if it was a computer making this decision for you?

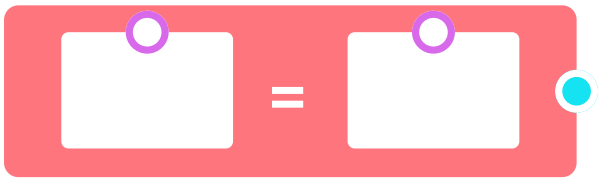
In programming, we don't ask the computer questions like "does special A cost less than special B?". Why not? Remember that programming is all about creating instructions for the computer to follow. A question is not an instruction – the computer will have no idea what steps it needs to go through to answer the question without you telling it.

Instead, we give the computer a statement like "special A is costs less than special B". The computer will read that statement and either go, "yeah, that statement makes sense. It is TRUE" or "hang on, special A costs $9, but special B costs $5. This statement doesn't make sense. It is FALSE".

If the computer agrees that the statement is correct, then we now know that yes, A must be less than B. If the computer identifies the statement as incorrect, we know that A must therefore not be less than B. 💡

We call these statements a comparison statement. A comparison statement compares two different values using a comparison operator. This statement will either be correct and return a TRUE value or incorrect and return a FALSE value.

You may be familiar with comparison operators from your math class. They include less than (<), greater than (>), less than or equal (<=), greater than or equal (>=), equal (==) and not equal (!=).



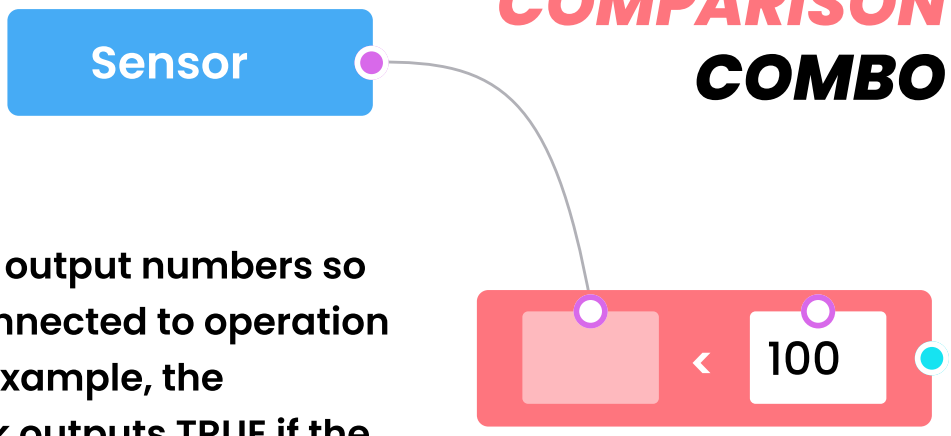

## COMPARISON BLOCKS

Comparison blocks compare two different input numbers using the operator in the middle. We can see an example of the equals comparison block above that checks if the two inputs have the same value.

Inputs can either be manually typed or connected to the output of another block.

If the comparison statement makes sense, it returns the value TRUE. Otherwise, it returns the value FALSE.

# SENSING + COMPARISON COMBO




Sensing blocks output numbers so they can be connected to operation blocks. In this example, the operation block outputs TRUE if the sensor reading is less than 100, and FALSE otherwise.

If you find a comparison operator and you are not sure how it works, try connecting a print block to the comparison output and experiment with the different combinations of inputs to try and figure it out.