# **Logic gates**

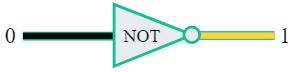
We send information through computers using wires that represent 1s and 0s. Computers need a way to manipulate those 1s and 0s, so that they can eventually do more complicated operations like calculating the 50th digit of π.

Computers use logic gates to transform the 1s and 0s from input wires. A logic gate accepts inputs and then outputs a result based on their state.

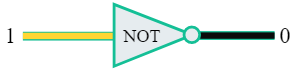
# NOT gate

The simplest gate is the NOT gate, also known as an inverter. It accepts a single input and outputs the opposite value.

If the input is 0, the output is 1:



If the input is 1, the output is 0:



Inverting a value may seem like a trivial operation, but in computers, we can build highly sophisticated logic by combining many small operations.

# AND gates

All other logic gates operate on multiple inputs. The AND gate accepts two wires, and if both of those wires are "on" (representing 1), right parenthesis, it outputs 1:



If either of those wires are "off" (representing 0), then it outputs 0:



What do you think this combination will output?

## CHECK YOUR UNDERSTANDING

### Choose 1 answer:

1. 0
2. 1

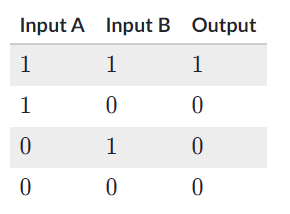
AND is a Boolean operation, an operation that takes values that are either "true" or "false", and then outputs "true" or "false" based on a logical manipulation of those inputs. In logic gates, we consider 111 to be true and 000 to be false.

One way to understand Boolean operations is to make a truth table of all the possible inputs and outputs. Here's a truth table for the AND gate:

| **Input A** | **Input B** | **Output** |
| --- | --- | --- |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

Notice there's only one row where the output is true - that happens only when both inputs are true.

We can also write the truth table using 111 and 000 to think of it in more computer-y terms:



Once again, there's only one row where the output is 1.

What if we want the opposite, a gate that outputs 1 almost every time? There's a gate for that!

# OR gates

The OR logic gate accepts two inputs, and as long as either of those inputs is a 1, it outputs a 1:



## CHECK YOUR UNDERSTANDING

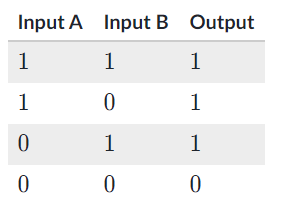
Based on that description, what value do you think this OR gate outputs?



### Choose 1 answer:

1. 0
2. 1

Let's look at the truth table for OR gates:



As you can see, the output is a 1 for every row except one. The only time that an OR gate outputs a 0 is if both inputs are 0.

# Thinking logically

It might help you to think about what these gates do in human speech.

Consider this OR gate. The first input represents "it's raining outside", the second input represents "we're going hiking", and the output represents "we should wear boots".



If it's raining outside OR we're going hiking, then we should wear boots. We can say that more verbosely: If it is true that "it's raining outside" OR it is true that "we're going hiking", then it's true that "we should wear boots".

That means that if it's raining outside (regardless of whether we're going hiking), we should wear boots. If we're going hiking (regardless of whether it's raining), we should wear boots. The only time we shouldn't wear boots (according to this gate) is when it's not raining outside and we're not going hiking.

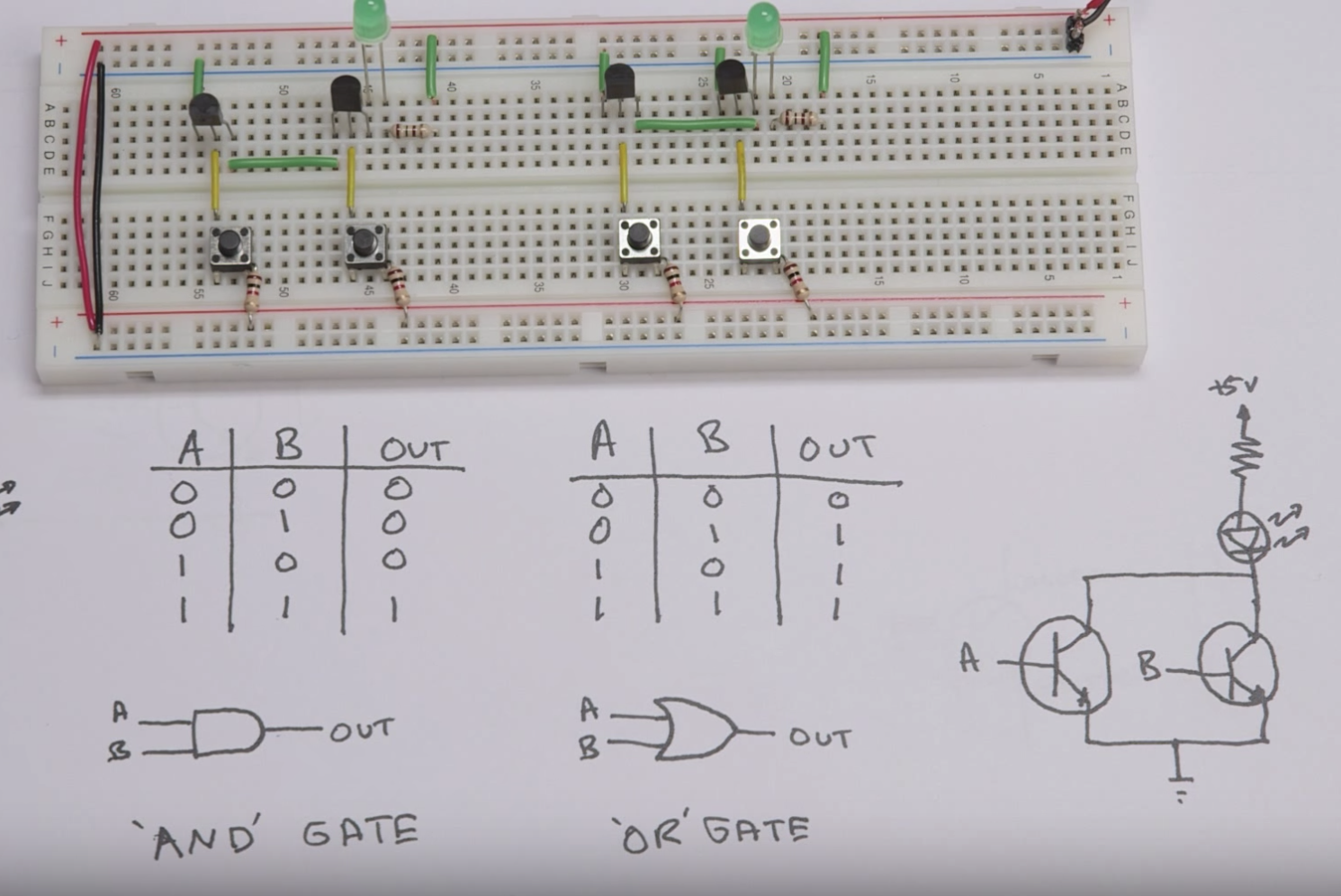
That was a vast simplification of the complicated logic that humans compute when we decide whether to wear boots, but it shows you that logic is something that relates very much to the "real world" outside the computer.

# Behind the abstraction

The logic gates that we've presented here are abstract representations of real devices. A logic gate describes any device that can take in values of 0 or 1 and output a 0 or 1 according to its truth table.

In most modern computers, logic gates are built using transistors combined with other electrical components like resistors and diodes. Those are all wired together to make sure that they transform the inputs in the way we expect.

With a little electronic equipment, you could put together your own logic gates, like [this video shows](https://www.youtube.com/watch?v=sTu3LwpF6XI&t=481s). Here are the homemade circuits for AND and OR gates from that video:



If you break into your own computer (don't!), you won't see anything like that. Our powerful computers now require billions of gates, so manufacturers have figured out how to make electronic parts very small. My own mac has 5.6 billion transistors that are just 14 nanometers wide.

We can understand and use logic gates without needing to know exactly how they're implemented. That's the power of an abstraction, enabling us to ignore the details and focus on the higher-level functionality.

Further learning

Computers also use other logic gates like [NAND](https://en.wikipedia.org/wiki/NAND_logic), [NOR](https://en.wikipedia.org/wiki/NOR_logic), and [XOR](https://en.wikipedia.org/wiki/XOR_gate). Each logic gate operates on inputs in a slightly different way; they output 1 and 0 in different situations.

In fact, the NAND and NOR gates are known as universal logic gates, which means that we can build any of the other gates with only NAND gates or only NOR gates. Computer hardware manufacturers prefer to use NAND gates due to their universality and ease of fabrication, so your computer likely has millions of NAND gates inside its circuitry.