Booleans

When testing your knowledge in this course, you are often presented with two options, multiple choice and true or false. The latter is an example of a Boolean; a thing is or is not. In this reading, you will learn more about Boolean data structures and the critical features for working with them, conditional statements and logical operators.

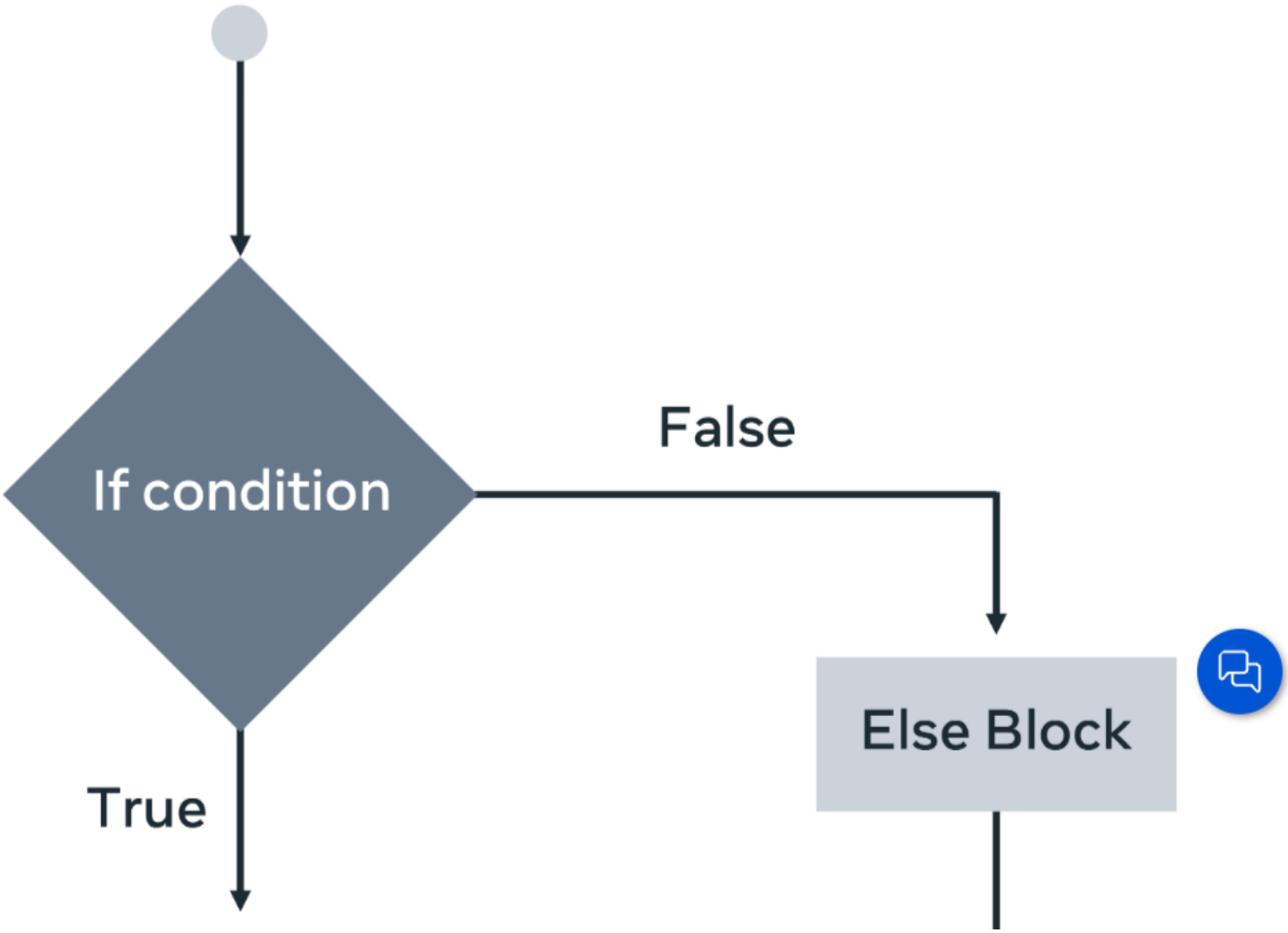
Conditional statements

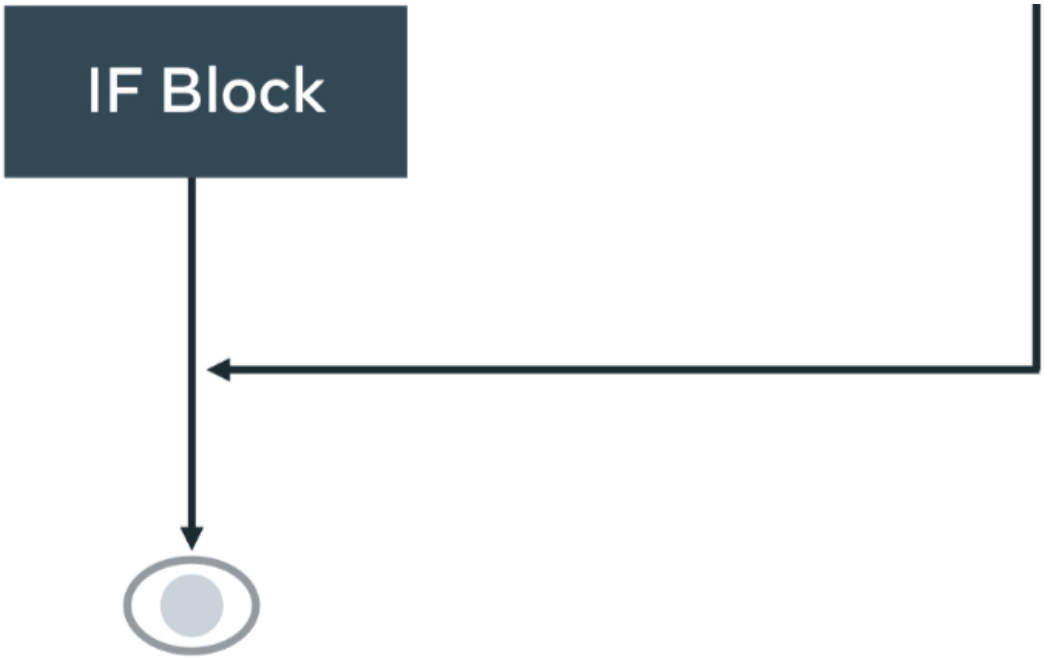
Boolean expressions can take a range of relational operators.

>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

These enable you to interrogate some data before running different code options. Therefore, Boolean expressions are often referred to as conditionals. Combine these with conditional statements, and you have an early implementation of AI.

Conditional statements are: `if`, `else`, `else if`, `while` and so on. A conditional block can enable you to execute one set of instructions in one condition and another if the conditions are different. Consider the following diagram.





That same diagram is represented in this Python code:

```
if right > wrong:
    doTheRightThing()
elif wrong > right:
    doTheWrongThing()
else:
    keepResearching()
```

Using Boolean expressions as indicators, you can then use relational operators to determine which line of code is to be executed. In this example, you have a computer evaluating between right and wrong. This has multiple applications and is commonly used. Finally, if you have not met any of your conditions, you might employ a catch-all code block. Roombas are a prime example of how this code can be used in everyday life. In place of right and wrong, you will have rudimentary sensors that determine which motors are to be activated based on distance and impact.

Logical operators

Additionally, you might want to expand the scope of your application by using logical operators.

Logical operators	
!!	Logical **`OR`**
&&	Logical **`AND`**
!	Logical **`NOT`**

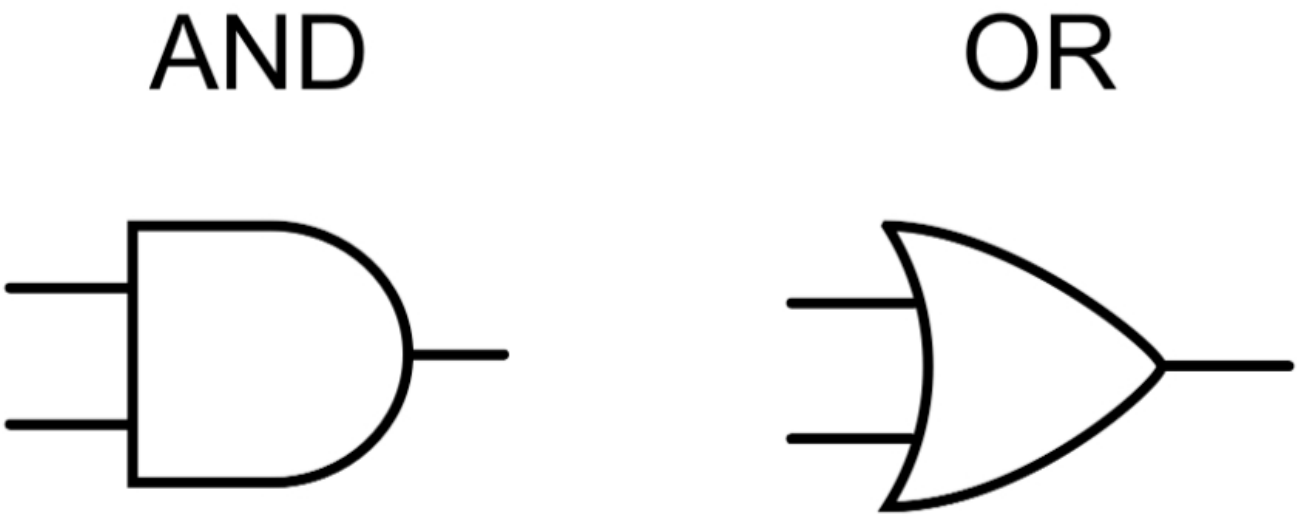
These logic expressions can be combined with Boolean expressions to give your code greater diversity.

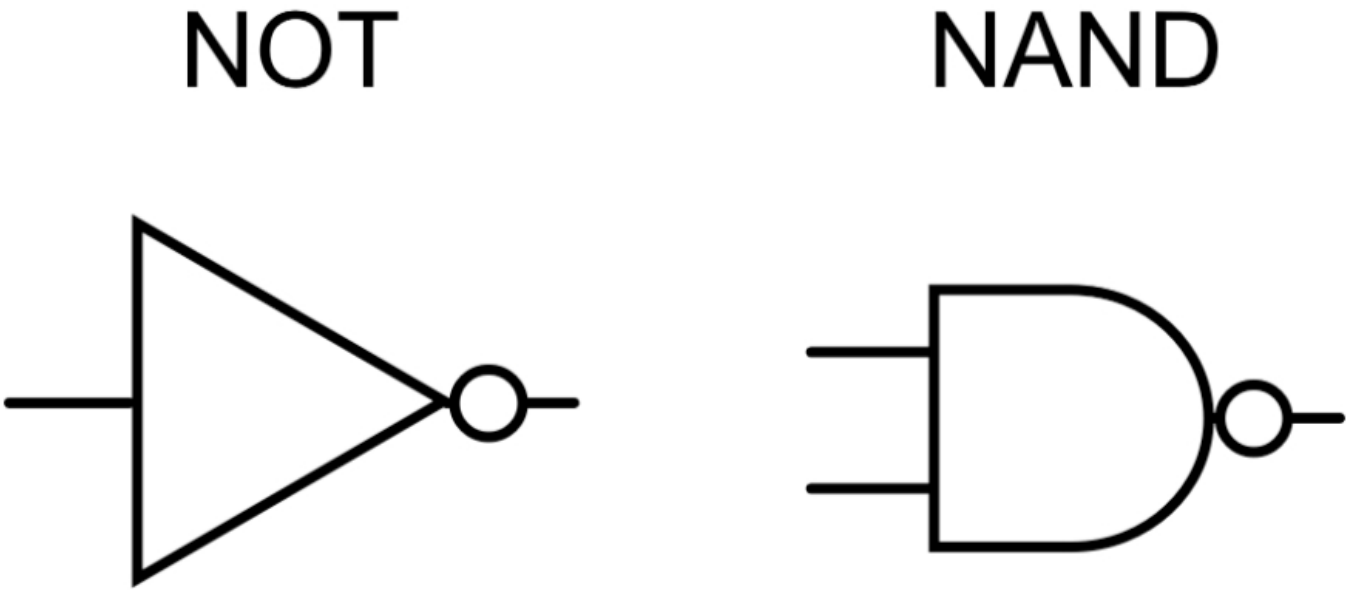
```
if condition_1 !! condition_2:
    doActionOne()
elif condition_1 && condition_2:
    doActionTwo()
elif !condition_1:
    doActionFour()
else:
    waitForInstruction()
```

In this code, four outcomes are checked. First, take **`condition_1`** or **`condition_2`**, and let's stick with the Roomba example. If proximity detection or impact detection is true, then action one might be stop and reverse. If there's

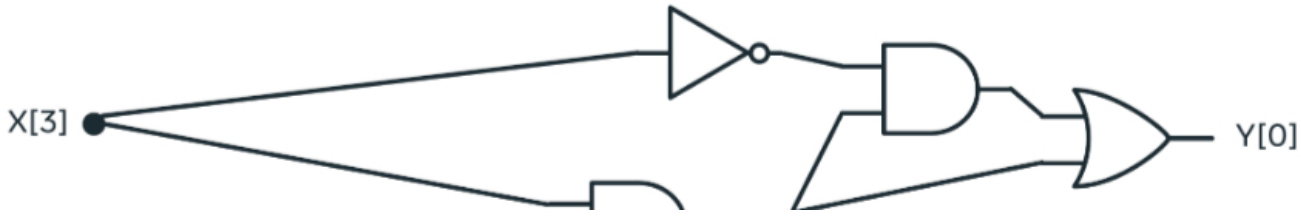
proximity and impact detection, then it'll trigger an alarm. If there is no proximity detection, it will continue to send power to the forward motors. Finally, if no conditions are met, there should be some fail-safe code ready to activate.

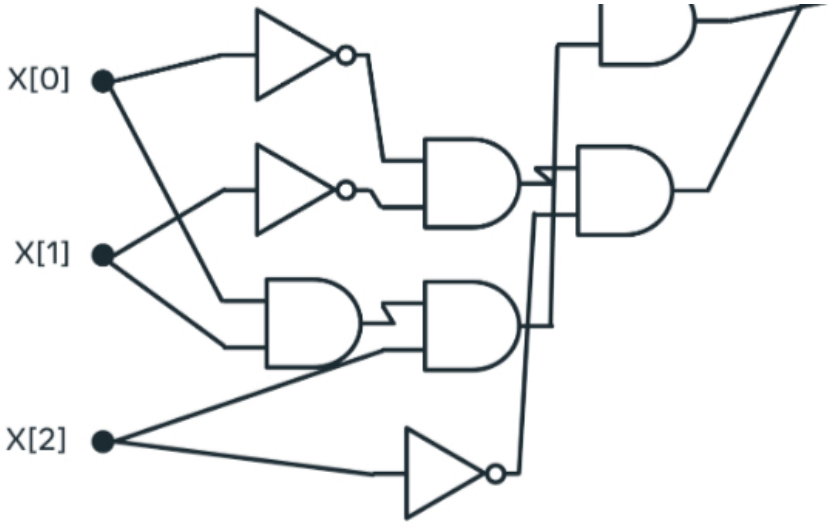
The use of Boolean logic is the backbone of circuit design. It is a way of interpreting Boolean operators together. The shapes in the diagram below are gates triggered when certain Boolean operators are fired. A signal is sent through the wires (depicted by the incoming and outgoing lines). The `AND` operator says if both wires are true, then continue this line of execution. At the same time, the `NAND` specifies that only two false assertions will trigger a reaction.





Combined, they can be used to make complex decisions and cause a device to operate in diverse ways, depending on the input from the sensors.





Conclusion

To conclude, Boolean expressions are either true or false, and their equivalence in binary would be 0 or 1. Just like with binary, you might think that only having two values results in limited capability. But, they can achieve a surprisingly elevated level of complexity when combined with other mathematical constructs like conditional statements and logical operators. Boolean can be used in your computer programs to inform which operations should run automatedly and form the backbone of circuitry diagrams.

In this reading, you learned more about Boolean data structures and the critical features for working with them, conditional statements and logical operators.