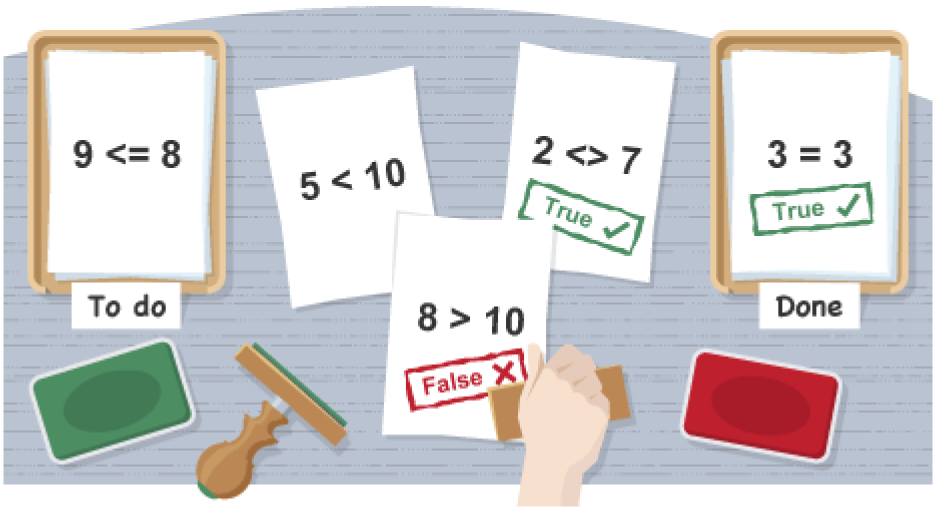
Boolean logic

When designing programs, there are often points where a condition needs to be tested in order to make a decision. Conditions are formed using Boolean logic.

What is Boolean logic?

Programs use simple comparisons to help make decisions. Boolean logic is a form of algebra where all values are either True or False. These values of true and false are used to test the conditions that selection and iteration are based around.



Boolean logic uses algebra and algebraic expressions. We use these expressions in algorithms and programs.

Expression	Boolean equivalent
Equals	=
Greater than	>
Less than	<
Greater than or equal to	>=
Less than or equal to	<=
Does not equal	<>
And	AND
Or	OR

Expression	Boolean equivalent
Not	NOT

Most programming languages use these equivalent Boolean expressions. However, some, such as Python, have slightly different equivalents:

Expression	Boolean equivalent	In Python
Equal to	=	==
Does not equal	<>	!=
And	AND	and
Or	OR	or
Not	NOT	not

Boolean expressions are represented using algebra.

Consider these statements:

- 5 < 10
- x < 10
- x < y

Each of these statements is a Boolean expression in the form of algebra. The only difference between them is that the first expression uses numbers and the second and third use variables. If we give x the value 5 and y the value 10, then each statement is identical.

Each statement is also a comparison. The statements compare the first value with the second. In this case we are saying that 5 is less than 10.

Boolean values

In Boolean logic, each statement is a comparison, and each comparison gives a Boolean value – True or False.

When **x = 5 and y = 10** then:

Statement	Expression	Boolean value
y > x	y is greater than x	**True.** When x is 5 and y is 10, then y is greater than x.
x < y	x is less than y	**True.** When x is 5 and y is 10, then x is less than y.
x = y	x equals y	**False.** When x is 5 and y is 10, then x does not equal y.
x<>y	x does not equal y	**True.** When x is 5 and y is 10, then x does not equal y.

When **x = 5 and y = 5**, we get a different set of Boolean values:

Statement	Expression	Boolean value
y > x	y is greater than x	**False.** When x is 5 and y is 5, then y is not greater than x.
x < y	x is less than y	**False.** When x is 5 and y is 5, then x is not less than y.
x = y	x equals y	**True.** When x is 5 and y is 5, then x is equal to y.
x<>y	x does not equal y	**False.** When x is 5 and y is 5, then x is equal to y.

Each Boolean expression gives a result that we can use in selection and iteration.

Using Boolean logic in programming

Boolean logic is used in selection to test conditions.

Consider this simple Python (3.x) program that prints out a different message depending on how old you are:

```
age = int(input("How old are you?")) if age >= 70: print("You are aged to perfection!")
else: print("You are a spring chicken!")
```

This program uses selection to determine whether to print one message or the other:

- The program examines the condition of the Boolean expression in line 2.
- If the inputted age is greater than or equal to 70, then the condition is **True**. As a result the program prints “You are aged to perfection!”
- If the inputted age is less than 70, then the condition is **False**. As a result, the program prints “You are a spring chicken!”

Building up complex decisions with Boolean expressions

The following Python (3.x) program works as above but has the added feature of checking to see if the inputted age is 50:

```
age = int(input("How old are you?")) if age >= 70: print("You are aged to perfection!") elif
age == 50: print("Wow, you are half a century old!") else: print("You are a spring
chicken!")
```

As well as checking the condition of the Boolean expression in line 2, this program also checks the condition of the Boolean expression in line 4:

- The program examines the condition of the Boolean expression in line 2.
- If the age that is input is greater than or equal to 70, then the condition is **True**. As a result the program prints “You are aged to perfection!”
- If the age that is input is less than 70, then the condition is **False**. The program then examines the condition of the expression in line 4.
- If the age that is input is equal to 50, then the first condition (age >= 70) is **False**, but the second condition (age == 50) is **True**. As a result the program prints “Wow, you are half a century old!”)
- If the age that is input is not greater than or equal to 70 and not equal to 50, then both expressions are **False**. As a result the program prints “You are a spring chicken!”

Boolean logic does not just work with numbers. Boolean expressions can also compare text, for example to check if a password is correct.

Consider this Python (3.x) program, which repeats if a password has been entered incorrectly:

```
answer = "" while answer != "ilovecomputing": answer = input("Type in the password: ")
print("Password correct”)
```

This program uses selection to determine whether to repeat, but this time compares text, not numbers.

- The program examines the condition of the Boolean expression in line 2. It is looking to see if the value of ‘answer’ does not equal “ilovecomputing”
- If the value of ‘answer’ does not equal “ilovecomputing”, then the condition is **True**. As a result the program asks the user to input the password and repeats the comparison.
- If the value of ‘answer’ equals “ilovecomputing” then the condition is **False**. As a result, the program skips the loop and proceeds to the last line of the program, which prints “Password correct”.