if Statement

Previously, we have learned that the actual work in a program is performed by expressions. This lesson further elaborates on that concept by explaining an important control structure that uses logical expressions.

WE'LL COVER THE FOLLOWING

Control statements

All of the expressions of all of the programs that we've seen so far have started with the `main()` function and were executed until the end of the main.

Control statements are features that affect the execution of expressions. These statements don't produce values and don't have side effects themselves. They determine whether and in what order the expressions are executed. Control statements sometimes use logical expressions when making such decisions.

> **Note:** Other programming languages may have different definitions for expression and statement, while some others may not have a distinction at all.

The `if` block and its scope

The **if statement** decides which piece of code to execute based on some condition becoming true. It makes this decision by evaluating a logical expression. It has the same meaning as the English word "if," as in the phrase

“if there is coffee, then I will drink coffee”.

`if` is a keyword in D that is followed by a logical expression in parentheses. If the value of that logical expression is *true*, then it executes the block of code that is enclosed in the curly brackets. Conversely, if the logical expression is *false*, it does not execute the statements within the curly brackets.

The area within the curly brackets is called a **scope**, and all of the code that is in that scope is called a **block of code**.
Here is the syntax of the `if` statement:

```
if (a_logical_expression) {
    // ... expression(s) to execute if true
}
```

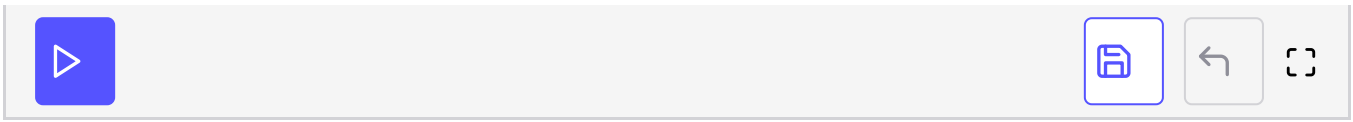
Example

This program displays “drink coffee and wash the cup” if “there is coffee”.

```
import std.stdio;

void main() {
    bool existsCoffee = true;

    if (existsCoffee) {
        writeln("Drink coffee");
        writeln("Wash the cup");
    }
}
```



if statement example

If the value of `existsCoffee` is *false*, then the expressions that are within the block would be skipped, and the program would not print anything.

The else block and its scope

Sometimes there are statements that need execution when the logical expression of the if statement is *false*.

Example

There is always an operation to execute in a decision, like “if there is coffee I will drink coffee, else I will drink tea” if the main condition is false.

The operations to execute in the *false* case are placed in scope after the `else` keyword. This is called the `else` block:

```
if (a_logical_expression) {
    // ... expression(s) to execute if true

} else {
    // ... expression(s) to execute if false
}
```

For example, under the assumption that there is always tea, the program appears:

```
import std.stdio;

void main() {
    bool existsCoffee = false;

    if (existsCoffee) {
        writeln("Drink coffee");

    } else {
        writeln("Drink tea");
    }
}
```

In that example, either the first or the second string would be displayed, depending on the value of `existsCoffee`.
`else` itself is not a statement but an optional clause of the `if` statement; it cannot be used alone.

> **Note:** Notice the placement of curly brackets of the `if` and `else` blocks above. Although it is official D style to place curly brackets on separate lines, this course uses a common style of inline curly brackets throughout.

Always use the scope curly brackets

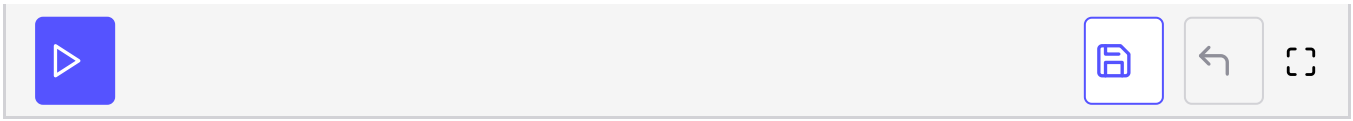
Though it is not recommended, it is actually possible to omit the curly brackets if there is only one statement within a scope. As both the `if` and the `else` scopes have just one statement above, that code can also be written as the following:

```
import std.stdio;

void main() {
    bool existsCoffee = false;

    if (existsCoffee)
        writeln("Drink coffee");

    else
        writeln("Drink tea");
}
```



Most experienced programmers use curly brackets even for single statements.

In the next lesson, we will see how `if` and `else` statements can be chained together for use in more complex ways.