Lesson 5

The Hard Way: Copying and pasting the same logic over and over again. It's long and error-prone.

The Smart Way: Building a reusable function and calling it with a single, clean line of code.

01

Programming Without Tools

02

Anatomy of a Function

03

Function Inputs

04

Getting a Result Back

05

Building Your Toolkit

1

Programming Without Tools

Repetitive Code

Repeating code is dangerous.
A bug in one spot means the same bug is likely hidden in many other places, making your program unreliable.

2

Anatomy of a Function

Let's Build a Tool

Function

A programmer-defined, reusable block of code that performs a specific task.

Every function follows a basic structure: a return type, a name, parameters for input, and a body with instructions.

The Parts of a Function

- return_type: The data type the function sends back when it's done.
- functionName: The unique name used to call the function.
- (parameters): The inputs the function needs to do its work.
- { body }: The block of code containing the step-by-step instructions.

Example of a Function

Function Header (Declaration/Prototype)

The header provides the compiler with information about the function [6]. It typically goes at the top after libraries and namespaces [6].

```
return_type Name_of_the_Function (parameters);
// Example where variable names are optional in the header:
double sum(double , double ); // [6]
```

Function Definition

The definition contains the actual code executed by the function. You cannot define a function inside another function [6]. The definition must match the header [7].

```
return_type Name_of_the_Function (parameters) {
    // function code goes here
    // A function with a return type must include a return statement
    return result;
} // [7]
```

3

Function Inputs

Making Tools Adaptable

How do we make functions work with different data?

A rigid function that can only perform one specific, hardcoded task.

A flexible function that accepts inputs (parameters) to work on new data.

4

Getting a Result Back

What Your Tool Produces

Return Value

The value a function sends back to the calling code using the 'return' statement.

Changing the return type from `void` (returns nothing) to a data type like `bool` or `int` means the function will now produce a result.

The value returned by a function can be stored in a variable for later use, just like any other value or literal.

When a program runs, a function call is effectively replaced by the value it returns to the calling code.

5

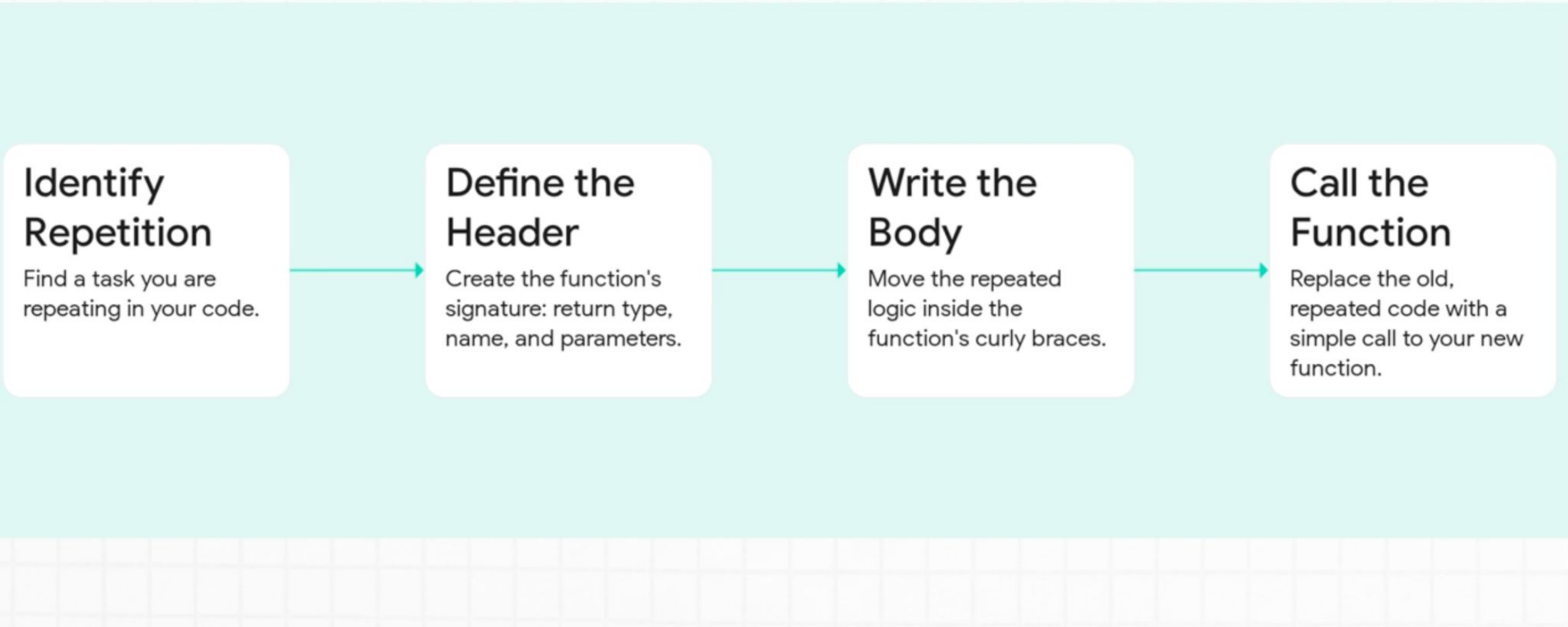
Building Your Toolkit

From Mess to Masterpiece

Repetitive, hard to read, and difficult to maintain.

Clean and readable. The complex logic is abstracted away into a reusable tool.

Function Blueprint



The Power of Functions

- Reusability: Write the logic once, call it many times.
- Readability: Give complex operations a simple, meaningful name.
- Maintainability: Fix a bug in one single location, not ten.
- Abstraction: Hide complex details behind a simple interface.

What repetitive task will you turn into a function first?