Functions

A function is a set of statements to perform a specific task. Functions organize the program into logical blocks of code. Once defined, functions may be called to access code. This makes the code reusable. Moreover, functions make it easy to read and maintain the program's code.

Rust code uses snake case as the conventional style for function and variable names, in which all letters are lowercase and underscores separate words.

We define a function in Rust by entering fn followed by a function name and a set of parentheses. The curly brackets tell the compiler where the function body begins and ends.

We can call any function we've defined by entering its name followed by a set of parentheses

Defining a Function

```
fn main() {
    println!("Hello, world!");

}

fn another_function() {
    println!("Another function.");
}
```

Calling a Function

A function must be called so as to execute it. This process is termed as function calling.

```
fn main() {
    println!("Hello, world!");

    another_function();
}

fn another_function() {
    println!("Another function.");
}
```

Returning values from a function.

Functions may also return a value along with control, back to the caller. Such functions are called returning functions.

```
fn five() -> i32 {
    5
}

fn main() {
    let x = five();

    println!("The value of x is: {x}");
}
```

Parameters

Parameters are a mechanism to pass values to functions

```
fn main() {
    another_function(5);
}

fn another_function(x: i32) {
    println!("The value of x is: {x}");
}
```