Functions With Parameters

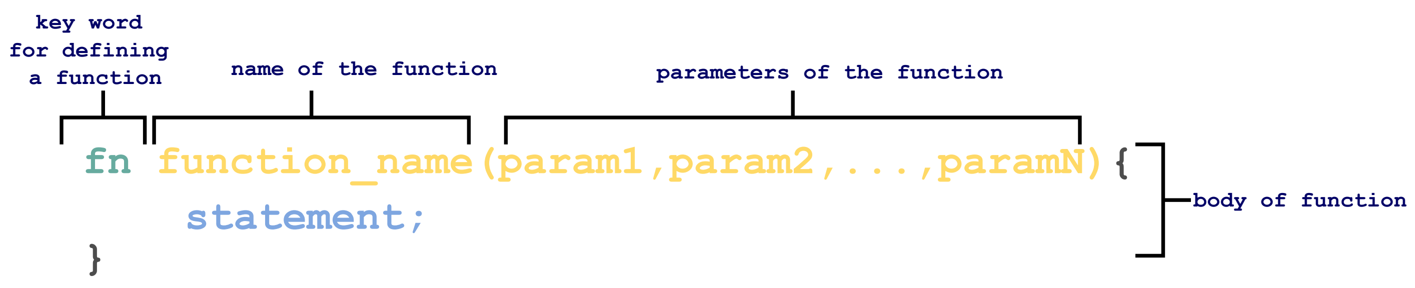
This lesson introduces parameterized functions.

We'll cover the following

In the previous example, a function was defined with nothing inside the round brackets. But certain functions require some information on which they should operate. For instance, a function that is expected to compute the square of a number needs to be provided with the number itself. That's what a parameter is.

What Are the Parameters?

Variable or values that go in the function definition are parameters.

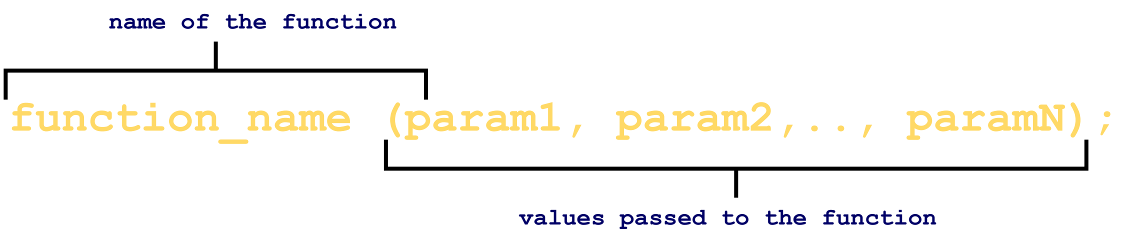


```
fn function_name(param1,param2,...,paramN) {
    statement;
}
```

Defining a function with parameters

What Are Arguments?

Variables or values that go in their place in the function invocation are known as arguments.


name of the function
function_name (param1, param2,.., paramN);
values passed to the function

Invoking a function with arguments

Example

To understand the above concept, let’s look at the example below:

```
//function definition
fn my_func(param_1:i32, param_2:i32) {
  println!("The first value passed inside function : {}", param_1);
  println!("The second value passed inside function : {}", param_2);
}
fn main() {
  let value_1 = 1;
  let value_2 = 2;
  //calling the function
  my_func( value_1, value_2 );
  println!("Function ended");
}
```

Explanation

The above program comprises two functions, the user defined function `my_func()` and the driver function `main()` where the function is being called.

User defined function

The function `my_func()` is defined from **line 2 to line 5**.

- Two parameters `param_1` and `param_2` are passed to the function.
- The values of passed parameters are printed on *line 3* and *line 4*.

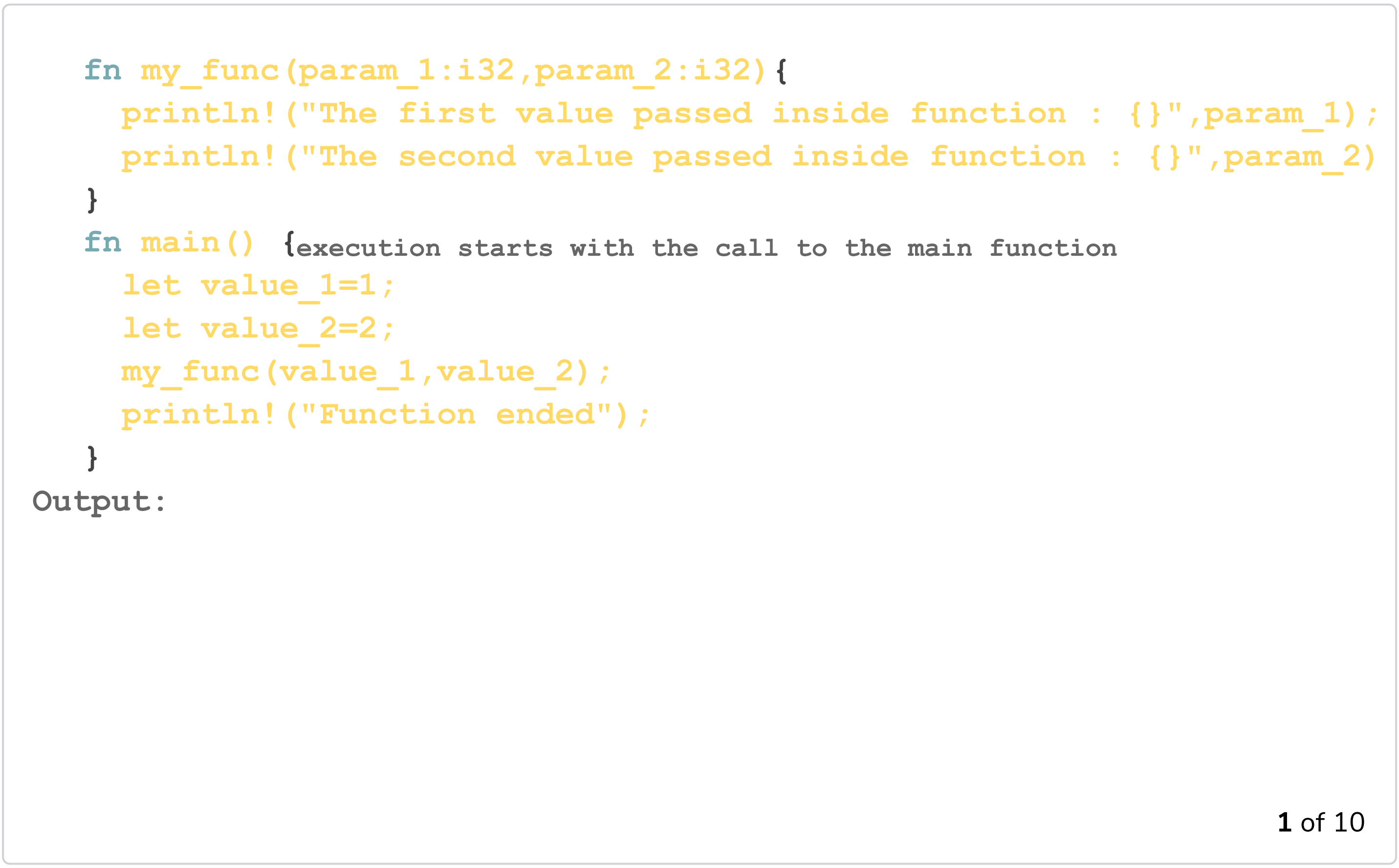
Driver function

The driver function `main()` is defined from **line 6 to line 12**.

- On *line 7* and *line 8*, two variables `value_1` and `value_2` are defined.
- On *line 10*, the function is invoked while passing the value of the variable

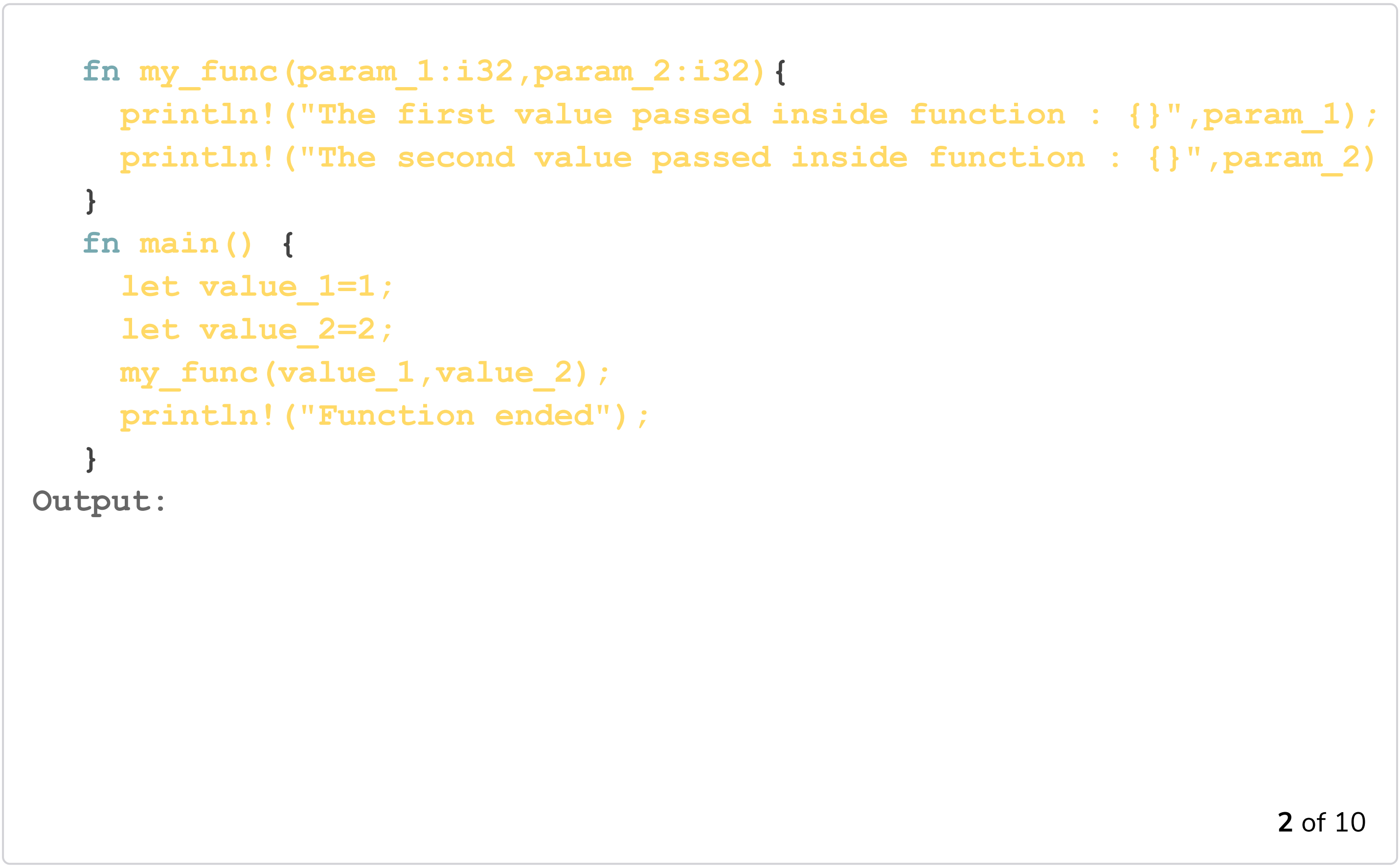
`value_1` as the first argument and that of `value_2` as the second.

The following illustration shows how program execution proceeds in the above code:



```
fn my_func(param_1:i32,param_2:i32){
  println!("The first value passed inside function : {}",param_1);
  println!("The second value passed inside function : {}",param_2)
}
fn main() {execution starts with the call to the main function
  let value_1=1;
  let value_2=2;
  my_func(value_1,value_2);
  println!("Function ended");
}
Output:
```

1 of 10



```
fn my_func(param_1:i32,param_2:i32){
  println!("The first value passed inside function : {}",param_1);
  println!("The second value passed inside function : {}",param_2)
}
fn main() {
  let value_1=1;
  let value_2=2;
  my_func(value_1,value_2);
  println!("Function ended");
}
Output:
```

2 of 10

```
   fn my_func(param_1:i32,param_2:i32){
     println!("The first value passed inside function : {}",param_1);
     println!("The second value passed inside function : {}",param_2)
   }
   fn main() {
     let value_1=1;
     let value_2=2;
     my_func(value_1,value_2);
     println!("Function ended");
   }
Output:
```

```
   fn my_func(param_1:i32,param_2:i32){
     println!("The first value passed inside function : {}",param_1);
     println!("The second value passed inside function : {}",param_2)
   }
   fn main() {
     let value_1=1;
     let value_2=2;
     my_func(value_1,value_2);funtion
                              invoked
     println!("Function ended");
   }
Output:
      fn my_func(param_1:i32,param_2:i32){


       println!("The first value passed inside function : {}",param_1
       println!("The second value passed inside function : {}",param_
      }
      fn main() {
        let value_1=1;
        let value_2=2;
        my_func(value_1,value_2);
        println!("Function ended");
      }
```

Output:

```
      fn my_func(param_1:i32,param_2:i32){


       println!("The first value passed inside function : {}",param_1
       println!("The second value passed inside function : {}",param_
      }
      fn main() {
        let value_1=1;
        let value_2=2;
        my_func(value_1,value_2);
        println!("Function ended");
      }
```

```
Output:The first value passed inside function : 1
```

```
fn my_func(param_1:i32,param_2:i32){


 println!("The first value passed inside function : {}",param_1
 println!("The second value passed inside function : {}",param_
}
fn main() {
  let value_1=1;
  let value_2=2;
  my_func(value_1,value_2);
  println!("Function ended");
}
```

```
Output:The first value passed inside function : 1
       The second value passed inside function : 2
```

```
fn my_func(param_1:i32,param_2:i32){


 println!("The first value passed inside function : {}",param_1
 println!("The second value passed inside function : {}",param_
}end of user defined function
fn main() {
  let value_1=1;
  let value_2=2;
  my_func(value_1,value_2);
  println!("Function ended");
}
```

```
Output:The first value passed inside function : 1
       The second value passed inside function : 2
       Function ended
```

```
      fn my_func(param_1:i32,param_2:i32){


       println!("The first value passed inside function : {}",param_1
       println!("The second value passed inside function : {}",param_
      }
      fn main() {
        let value_1=1;
        let value_2=2;
        my_func(value_1,value_2);
        println!("Function ended");
      }
Output:The first value passed inside function : 1
       The second value passed inside function : 2
       Function ended
```

9 of 10

```
      fn my_func(param_1:i32,param_2:i32){


       println!("The first value passed inside function : {}",param_1
       println!("The second value passed inside function : {}",param_
      }
      fn main() {
        let value_1=1;
        let value_2=2;
        my_func(value_1,value_2);
        println!("Function ended");
      }end of program code
Output:The first value passed inside function : 1
       The second value passed inside function : 2
       Function ended
```

10 of 10

Types of Arguments

Arguments can be passed to a function in two different ways:

- Pass by value
- Pass by reference

Quiz

Test your understanding of parameterized functions in Rust!

Quick Quiz on Parameterized Functions!

Q

What is the output of the following code?

```
fn my_func(param1:i32, param2:i32) {
  println!("The first value passed inside function : {}",  param1
);
}
fn main() {
  let value1 = 1;
  let value2 = 2;
  my_func(value1, value2);
}
```

Retake Quiz

Now that you have learned about functions with parameters, let’s learn what it means to pass an argument by value and how it is done in Rust.