Returning a Value From a Function

This lesson introduces you to returning a value from a function.

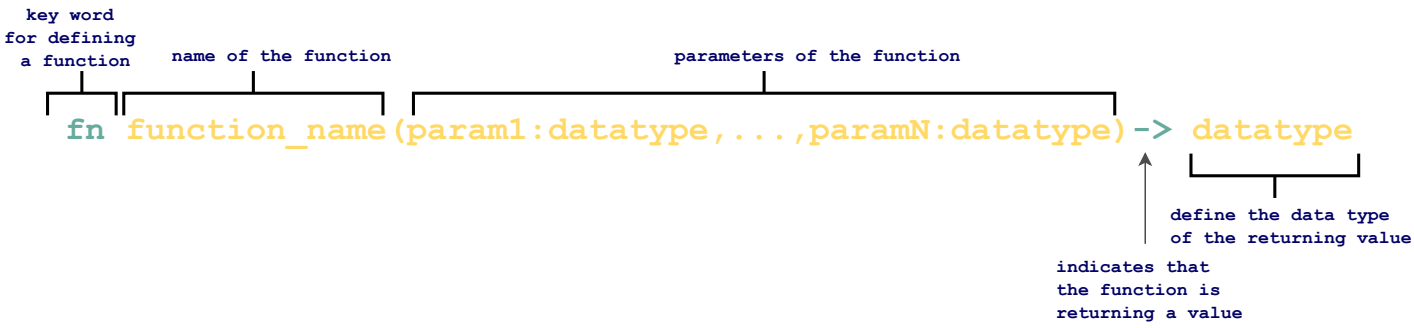
We'll cover the following

Returning Functions

The functions can return a value using the `return` keyword inside the function definition. After the return statement is executed, the control gets back to the caller. A function invocation is replaced with the value that the call returns. Thus, that value can be saved in a variable.

Syntax

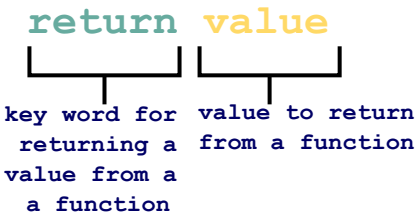
The **function definition for returning a value** from a function:



Defining a function with explicitly defining the return value

There are **two ways to actually return the value**.

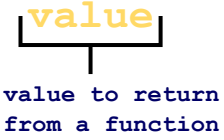
The general syntax for returning a value from a function **using the `return` keyword**:



Defining a function with explicitly defining the return value

The following syntax can be used to return a value from a function **without using the `return` keyword**:

> Just write the return value, the compiler will interpret it because of the `->` sign in the function definition.



Defining a function with implicitly defining the return value

Example 1

The following example makes a function `square()` that takes a number `n` as a parameter to the function and stores the square of `n` in the local variable `m` and returns the variable `m`.

```
fn square(n:i32)->i32{
  println!("The value of n inside function : {}", n);
  let m = n * n;
  m // return the square of the number n
}
fn main() {
  let  n = 4;
  println!("The value of n before function call : {}", n);
  println!("Invoke Function");
  println!("\nOutput : {}",square(n));
}
```

Example 2

The following example makes a function `square()` that takes a number `n` as a parameter to the function and returns the square of the number `n` by using the `return` keyword.

```
fn square(n:i32)->i32{
  println!("The value of n inside function : {}", n);
  return n * n;
}
fn main() {
  let  n = 4;
  println!("The value of n before function call : {}", n);
  println!("Invoke Function");
  println!("\nOutput : {}", square(n));
}
```

Explanation

The above program is of two parts, the user defined function `square()` and the driver function `main()` where the function is being called.

User defined function

The function `square()` is defined from **line 1 to line 4**.

- On *line 3* `n` is multiplied with itself and the value is saved in `n` and value of type `i32` is returned using the `return` keyword.

Driver function

The driver function `main()` is defined from **line 5 to line 10**.

- On *line 6*, a variable `n` is defined.
- On *line 9*, the function `square()` is invoked which takes `n` as an argument to the function and the value of the square is printed using the `println!()` macro.

The following illustration shows how the code is executed:

```
fn square(n:i32)->i32{
  println!("The value of n inside function : {}",n);
  return n*n;
}
fn main() {
  let  n=4;
  println!("The value of n before function call : {}",n);
  println!("Invoke Function");
  println!("\nOutput : {}",square(n));
  println!("Function ended");
}
```

Output:


```
fn square(n:i32)->i32{
  println!("The value of n inside function : {}",n);
  return n*n;
}
fn main() {
  let  n=4;
  println!("The value of n before function call : {}",n);
  println!("Invoke Function");
  println!("\nOutput : {}",square(n));
  println!("Function ended");
}
```

Output:

```
fn square(n:i32)->i32{
  println!("The value of n inside function : {}",n);
  return n*n;
}
fn main() {
  let  n=4;
  println!("The value of n before function call : {}",n);
  println!("Invoke Function");
  println!("\nOutput : {}",square(n));
  println!("Function ended");
}
```

Output: **The value of n before function call : 4**


```
fn square(n:i32)->i32{
  println!("The value of n inside function : {}",n);
  return n*n;
}
fn main() {
  let  n=4;
  println!("The value of n before function call : {}",n);
  println!("Invoke Function");
  println!("\nOutput : {}",square(n));
  println!("Function ended");
}
```

Output: **The value of n before function call : 4**
Invoke function

```
fn square(n:i32)->i32{
  println!("The value of n inside function : {}",n);
  return n*n;
}
fn main() {
  let  n=4;
  println!("The value of n before function call : {}",n);
  println!("Invoke Function");
  println!("\nOutput : {}",square(n));
  println!("Function ended");
}
```

```
Output: The value of n before function call : 4
        Invoke function
```


n=4

```
fn square(n:i32)->i32{
  println!("The value of n inside function : {}",n);
  return n*n;
}
fn main() {
  let  n=4;
  println!("The value of n before function call : {}",n);
  println!("Invoke Function");
  println!("\nOutput : {}",square(n));
  println!("Function ended");
}
```

```
Output: The value of n before function call : 4
        Invoke function
```

```
fn square(n:i32)->i32{
  println!("The value of n inside function : {}",n);
  return n*n;
}
fn main() {
  let  n=4;
  println!("The value of n before function call : {}",n);
  println!("Invoke Function");
  println!("\nOutput : {}",square(n));
  println!("Function ended");
}
```

```
Output: The value of n before function call : 4
        Invoke function
        The value of n inside function :4
```



```
fn square(n:i32)->i32{
  println!("The value of n inside function : {}",n);
  return n*n; return 4*4
}
fn main() {
  let  n=4;
  println!("The value of n before function call : {}",n);
  println!("Invoke Function");
  println!("\nOutput : {}",square(n));
  println!("Function ended");
}
```

```
Output: The value of n before function call : 4
        Invoke function
        The value of n inside function :4
```

```
fn square(n:i32)->i32{
  println!("The value of n inside function : {}",n);
  return n*n; return 4*4
}
fn main() {
  let  n=4;
  println!("The value of n before function call : {}",n);
  println!("Invoke Function");
  println!("\nOutput : {}",square(n));
  println!("Function ended");
}
```

```
Output: The value of n before function call : 4
        Invoke function
        The value of n inside function :4
        Output : 16
```



```
fn square(n:i32)->i32{
  println!("The value of n inside function : {}",n);
  return n*n; return 4*4
}
fn main() {
  let  n=4;
  println!("The value of n before function call : {}",n);
  println!("Invoke Function");
  println!("\nOutput : {}",square(n));
  println!("Function ended");
}
```

```
Output: The value of n before function call : 4
        Invoke function
        The value of n inside function :4
        Output : 16
        Function ended
```

Quiz

Test your understanding of returning a value from a function in Rust.

Quick Quiz on Functions with a Return Value!

Q

What is the output of the following code?

```
fn main(){
   println!("Area of rectangle is {}", get_area(2, 2));
}
fn get_area(x:i32, y:i32) -> i32 {
   x  *  y
}
```

Retake Quiz

Now that you have studied functions with a single return value, let’s learn to return multiple values from a function.