Kotlin Functions

Welcome, in this lesson you'll learn about functions; what functions are, its syntax and how to create a user-function in Kotlin.

In programming, function is a group of related statements that perform a specific task.

Functions are used to break a large program into smaller and modular chunks. For example, you need to create and color a circle based on input from the user. You can create two functions to solve this problem:

- `createCircle()` Function
- `colorCircle()` Function

Dividing a complex program into smaller components makes our program more organized and manageable.

Furthermore, it avoids repetition and makes code reusable.

Types of Functions

Depending on whether a function is defined by the user, or available in standard library, there are two types of functions:

- **Kotlin Standard Library Function**
- **User-defined functions**

Kotlin Standard Library Function

The standard library functions are built-in functions in Kotlin that are readily available for use. For example,

- `print()` is a library function that prints message to the standard output stream (monitor).
- `sqrt()` returns square root of a number (`Double` value)

Example

```
fun main(args: Array<String>) {
    val number: Double = 5.5
    //find the square root
    val sqrt:Double = Math.sqrt(number)
    println("Result    $sqrt")
}//end
```

When you run the program, the output will be:

```
Result = 2.345207879911715
```

User-defined Functions

As mentioned, you can create functions yourself. Such functions are called **user-defined functions.**

How to create a user-defined function in Kotlin?

Before you can use (call) a function, you need to define it.

Here's how you can define a function in Kotlin:

```
fun callMe() {

    // function body
}
```

To define a function in Kotlin, **`fun`** keyword is used. Then comes the name of the function (**identifier**). Here, the name of the function is ***callMe***.

In the above program, the parenthesis `( )` is empty. It means, this function doesn't accept any argument. *You will learn about arguments and parameters later in this lesson*.

The codes inside curly braces `{ }` is the body of the function.

How to call a function?

You have to call the function to run codes inside the body of the function. Here's how you call a function:

**********Call your function in the **main** function*******

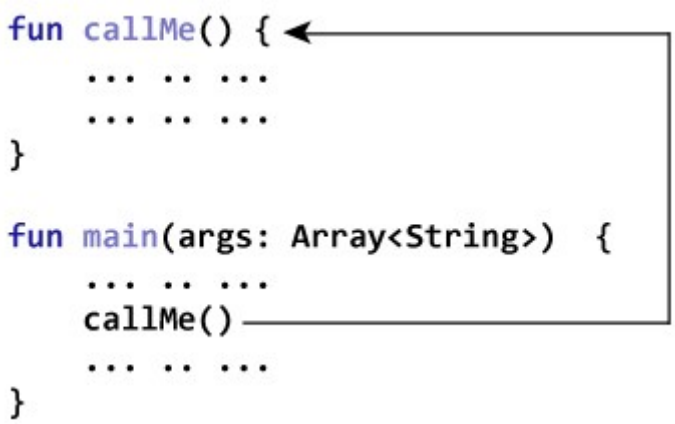
```
fun main(args: Array<String>) {

   //trigger the function

   callMe() //function called inside main function

}//end
```

This statement calls the `callMe()` function declared earlier.



Here is a full example

```
fun main(args: Array<String>) {

   //trigger the function

   callMe()

}//end

fun callMe(){

    //to do code

    println("This is a new function...")

}
```

Example Two
A function to add three numbers

```
fun main(args: Array<String>) {
  //trigger the function
  addition()
}//end


fun addition(){
   val num1: Double = 6.8
   val num2: Double = 6.2
   val num3: Double = 6.1

   val answer :Double= num1 + num2 + num3
   println("The total is $answer")
}
```

This is a function WITHOUT parameters

When you run the program, the output will be:

```
The total answer is 19.1
```

Above, we create a function named '**addition()**' inside the function body we declare 3 variables and find their sum.

Then we call/trigger the function '**addition()**' in the main function, ***t***

Example 3
A function to find if given number is **Negative, Positive or Zero**

```
fun main(args: Array<String>) {
  //trigger the function
  check()
}//end main()


fun check(){
   val num: Int = 12

   if(num < 0){
     println("Its Negative")
   }

   else if(num > 0){
     println("Its Positive")
   }

   else if(num == 0){
     println("Its Zero")
   }

   else {
     println("Its Invalid")
   }

}//end check()
```

This is a function WITHOUT parameters
When you run the program, the output will be:

```
Its Positive
```

Functions with parameters

Functions can have parameters as well , here we look on how to use parameters in kotlin.

Here is an example without parameters

```
/**
 * You can edit, run, and share this code.
 * play.kotlinlang.org
 */


fun main(args: Array<String>) {
  //trigger the function
  addNumbers()
}//end main()



fun addNumbers(){
   val num1: Double = 12.0
   val num2: Double = 56.0
   val answer: Double = num1 + num2
   println("Addition is $answer")
}//end check()
```

This is a function WITHOUT parameters

When you run the program, the output will be:

```
Addition is 68.0
```

Above addNumber() defines two(num1 & num2) variables in its body , meaning its tied to them values **12 & 56, any time we call our function we get 68.0 as the answer.**

Now,we can improve this function to include parameters and make the function more flexible to work with any variables, parameters are usually passes in function '()' parenthesis.

fun **addNumbers**(*parameters goes here, comma separated),*…..see below

See next **Example 1**

Here, two **parameters** *num1* and *num2* of type `Double` are placed to the `addNumbers(num1: Double, num2: Double)` function ,

Then during function call the actual values are provided and passed to the function where addition of the 2 variables happen

```
fun main(args: Array<String>) {

   //trigger the function

   addNumbers(num1=67.0, num2=45.0) //try new values here...

}//end main()


fun addNumbers(num1: Double, num2: Double){

    val answer: Double = num1 + num2

    println("Addition is $answer")

}//end addNumbers()
```

This is a function WITH parameters

Example 2

```
fun main(args: Array<String>) {

   //trigger the function

   check(marks=28.0, admno=3454)

}//end main()
```

These are parameters

```
fun check(marks: Double, admno:Int){

    if(marks< 30){

         println("Your Admission No is $admno , You Failed ")

    }

    else if(marks>= 30 && marks <=75){
```

```
        println("Your Admission No is $admno , You are Average ")
    }

    else if(marks> 75 && marks <=100){
        println("Your Admission No is $admno , You are Passed ")
    }

    else {
        println("Your Admission No is $admno , Invalid")
    }
}//end check()
```

This is a function WITH parameters

When you run the program, the output will be:

```
Your Admission No is 3454 , You Failed
```

Above, check() function accepts 2 parameters which are marks (Double) and admno(Int), see below

```
fun check(marks: Double, admno:Int)
```

```
In the main() function, the 2 variables are provided
 check(marks=28.0, admno=345)
```

***NB**: marks has a decimal because it was a Double, admno was an Int so no decimal*

Check Assignment4 for practice

END

Functions with return types, to be done Next topic 5