Functions in Swift

Organise your work!

Functions allow programming statements to be grouped in such a way as to represent a piece of functionality (or function) that can then be reused.

There are quite a few examples in Swift, so strap in as I show you one or two examples!



**Prerequisites:**

* Be able to produce a “Hello, World!” iOS application (guide [HERE](https://medium.com/@stevenpcurtis.sc/your-first-swift-application-without-a-mac-79598ad839f8))
* Perhaps be able to use Playgrounds to follow along (guide [HERE](https://medium.com/@stevenpcurtis.sc/coding-in-swift-playgrounds-1a5563efa089))

**Terminology**

function: a group of statements that together can perform a function

**The rationale**

You may have several statements that you want to re-use in your program.

An example of this is a program that draws a triangle on the screen

Repeat three times:  
 draw a line of length 10  
 turn right by 60 degrees

You might be happy to write this out once.

If you want two triangles on the screen…it quickly gets tiresome.

If you want to screen full of triangles, you must admit we need to find a better way.

**Practical functions**

**The random number written on the screen**

You can think of a function that prints a random number between 1 and 10. In Swift this is Int.random(in: 1 ... 10)and this is written in the following way for a function:

func writeNum() {

print ( Int.random(in: 1 ... 10) )

}

and this is the function **declaration**. This is were we declare the function, but it will never run because there is nothing **calling** this function.

To do so we write

writeNum()

as many times as we want, and we will get a new number written on the screen.

If you need some help coding in Swift Playgrounds follow [THIS](https://medium.com/@stevenpcurtis.sc/coding-in-swift-playgrounds-1a5563efa089) link, but for me I printed out 5 different random numbers (of course yours will be different as the numbers are random)



**The random number with a name**

Perhaps you want to give a random score to your friends. The output will be something like

*Dave*

*8*

and since the name is going to change everytime, we would like a function that takes in that name. Well, the taking in part is called the **parameter**, and we can express that with the following function:

func writeNum(name: String) {  
 print (name)  
 print ( Int.random(in: 0 ... 10) )  
}

We call this function with the following statement

writeNum(name: "Dave")

*Let us explain what is happening*

The first line (the function signature) of the writeNum function has certainly changed. func writeNum(name: "Dave") will now be explained in full:

**func** means that we are declaring a function

**writeNum** is where we name this function

**(name: String)** is the parameter — this is where we will pass the name of our friend and in this case it must be a String.

We call these components the **function signature**.

**Functions with multiple parameters**

You can have as many parameters as you require, you just separate them with a comma (,).

The following function joins two strings (followed by how to call the function:

func joinStrings(one: String, two: String) {  
 print (one + two)  
}  
joinStrings(one: "a", two: "b")

**Functions returning a value**

You can return a value by putting -> after the parameters, so you need to be sure that you do return a value.

So instead of printing our joined Strings to the screen, we can *return*them.

func joinStringsAndReturn(one: String, two: String) {  
 print (one + two)  
}  
joinStringsAndReturn(one: "a", two: "b")

Now this won’t actually show you’ve done anything on the screen. To do so, you need to print the result:

print (joinStringsAndReturn(one: "a", two: "b"))

which of course gives you the idea that returning the result is a good idea because you can decide what you are doing with the result *afterwards*.

**Functions returning an optional value**

You might be aware of [optionals](https://medium.com/@stevenpcurtis.sc/re-implement-optionals-in-swift-41129477934c). You can return them, which means your function either returns 0 or a value. A good example of this is division, since dividing by zero makes bad things happen.

Note the optional ? in the function signature.

func myDiv(one: Int, two: Int) -> Int? {  
 if two < 1 {return nil}  
 return one / two  
}  
myDiv(one: 1, two:0) // nil  
myDiv(one: 4, two: 2) // 2

**External and internal parameter names**

You can have different names for parameter names when they are called or internally within the function. this means that you can shorten some of the variables you use internally.

func myName(name nm: String) {  
 print (nm)  
}  
myName(name: "Steve") // Steve

It’s a simple example, but gives you some idea

**Variadic Parameters**

This is much more hard code. But you can accept an unknown number of parameters without resorting to using an array.

func sum(numbers: Int...) {  
 var sum = 0  
 for i in numbers {  
 sum += i  
 }  
 print (sum)  
}  
sum(numbers: 1,2,3) // 6  
sum(numbers: 1,2,3,4,5) // 15

A bit tricky — but in some situations worth it

**Default parameters**

You might sometimes want to have a parameter, other times to default to a value. No problem

func calcInterest(money: Int, InterestRate) -> Int {  
 return money + (money \* interestRate)  
}  
calcInterest(money: 1) // 1  
calcInterest(money: 2, interestRate: 1) // 4

A bad example with just Integers, but still it gives you an idea.

**Conclusion…**

You’ll be using functions alot in Swift. Go through these examples and I hope you have a good understanding of the functionality of functions in Swift.

And of course, I wish you happy coding.

**The repo link**

I’ve already prepared the following link for you. If you need a hand downloading look at the guide [HERE](https://medium.com/@stevenpcurtis.sc/downloading-repos-from-github-13a017951450).