Pragyee Nepal

**Naming, Scope and Bindings in Swift**

If we declare a variable in the main body and then declare another variable of same name inside a loop then the value gets replaced inside the loop but not outside. For example:

var x = 0

for index in 1...2{

print (x)

var x = 2

print (x)

}

print(x)

Output:

0

2

0

2

0

In the above example we can see that inside the loop it first prints 0 (value declared outside of the loop), then the variable is declared with different value (2). Now, when it is printed, it prints the value of the variable declared inside the loop (2). Then, when the code goes through the loop the second time, it prints 0 again instead of 2 and the same thing is repeated. When the loop ends, the value is again the initial value (0).

If we do the same thing but declare the other variable in a function rather than a loop then we get different results. Example:

func variable\_test(x: Int) -> Int{

print (x)

var x = 2

print (x)

return(x)

}

var x = 0

variable\_test(x:x)

print(x)

Output:

0

2

0

In this example you can see that anything that happens in the function doesn’t affect the main body. In the function it first prints out the value it took in (0), then when the variable with the same name is declared again but with different value (2), the value of the variable is overridden. But when the code return from the function, the value of the variable is unchanged from the initial value (0). If we want to change the value then in main while calling the function we must write x = variable\_test(x:x) which will then change the value of the variable in main and print out:

0

2

2

If we want a global variable then there are multiple ways to do it. One of the method is declaring a variable in a class which can be used within any function inside that class. For example:

var age: Int = 22

func Student(){

print(age)

}

Student()

Output:

22

Another way is by creating a struct and encapsulating all the variables inside it. This can be used in any class we want. Example:

struct Person{

var name: String = "Jane"

var age: Int = 22

}

class Student{

var person = Person()

lazy var a: Int = getInfo()

func getInfo()-> Int{

print(person.name)

print(person.age)

return (person.age)

}

}

var b = Student().getInfo()

Output:

Jane

22

In Swift, we can pass by both reference and value. Although we can use either method for variables, pass by value is favored for string, dictionary, array and struct. Pass by reference is favored for class. Below is the example that either method can be used:

//pass by reference

class fruit{

var isfav = false

}

let apple = fruit()

let banana = apple

banana.isfav = true

print(apple.isfav)

print(banana.isfav)

//we changed the isfav for banana and the value of isfav changed for apple too

//this is because they both are pointing to the same location in the memory

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

//pass by value

struct fruit1{

var isfav = false

}

var kiwi = fruit1()

var orange = kiwi

orange.isfav = true

print(kiwi.isfav)

print(orange.isfav)

Output:

true

true

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

false

true

In pass by reference, changing the value of one variable automatically changed the value of the other but in pass by value changing value of one did not affect the other. Using reference by value is also favored while comparing values.

If we write the following code in swift

var a : [Character] = ["c","a","t"]

var b : [Character] = ["d","o","g"]

a=b

b[1] = "u"

print (a)

print (b)

We get the following output:

["d", "o", "g"]

["d", "u", "g"]

It tells us that it handles assignments by passing them by value rather than by reference. If it was pass by reference, then the value of ‘a’ should have changed too when the value of ‘b’ was changed but it did not change.

Reference

* <https://www.tutorialspoint.com/how-to-create-and-use-global-variable-in-swift>
* <https://www.raywenderlich.com/9481-reference-vs-value-types-in-swift>
* <https://docs.swift.org/swift-book/LanguageGuide/StringsAndCharacters.html>