

SE 4020: Mobile Application Design and Development

Lecture 03 – Functions, Closures and Classes

Topics

- Functions
- Closures
- Classes

Functions

```
print (min(9, -3, 12, 7))  
print (max("zoo", "barn", "cinema"))
```

← Experiment with your own examples here—for example, what happens if you enter "Zoo" instead of "zoo"?

We define a function using the `func` keyword.

It gets a function name, so we can call it to use it.

To work, our function needs to know two things: the type of pizza ordered, and a count of pizzas.

```
func pizzaOrdered(pizza: String, count: Int) {  
  
}  
}
```

The body of the function goes here. The body is the code that gets run when the function is called.

The named, typed values that the function takes as input are called parameters.

```
pizzaOrdered(pizza: "Hawaiian", count: 7)
```

Argument labels

```
func pizzaOrdered(thePizza pizza: String, theCount count: Int) { }
```

thePizza and theCount are the argument labels
for the parameter names pizza and count.

```
pizzaOrdered(thePizza: "Hawaiian", theCount: 7)
```

Avoiding argument Labels

```
func pizzaOrdered(_ pizza: String, count: Int)
```

```
    pizzaOrdered("Vegetarian", count: 5)
```

Function Parameters

- In Swift unlike in other languages function parameters are constants. You can't change them.

Exercise

- Write a function to input a start number, end number and step number and print the range of numbers.



`https://bit.ly/3wHlvik`

`https://replit.com/@NuwanKodagoda/2024-03-03`

Inout Parameters

```
func multiplyBy10(_ num: inout Int) {  
    num = num * 10  
    print("The number multiplied by 10 is: \(num)")  
}
```

```
var no = 10  
multiplyBy10(&no)  
print(no)
```


Exercise

- Write a function to find the shortest name and the longest name of a string array and return both.



`https://bit.ly/3wHlvik`

`https://replit.com/@NuwanKodagoda/2024-03-03`

Returning Values from a function

This syntax defines the return type. In this case, a string.

```
func welcome(name: String) -> String {  
    let welcomeMessage = "Welcome to the Swift Pizza shop, \(name)!"  
    return welcomeMessage  
}
```

This line returns the current value of the constant, welcomeMessage (which is a string, as per the return type we defined).

Making things simpler

```
func welcome(name: String) -> String {  
    return "Welcome to the Swift Pizza shop, \(name)!"  
}
```

Or using implicit return

```
func welcome(name: String) -> String {  
    "Welcome to the Swift Pizza shop, \(name)!"  
}
```

Default Parameters

This is the default value for the name, inside the function.



```
func welcome(name: String = "Customer") -> String {  
    let welcomeMessage = "Welcome to the Swift Pizza shop, \(name)!"  
    return welcomeMessage  
}
```

Exercise

- Write a function to return the grade given a mark. You can have Distiction, Merit, Pass and Fail grades



`https://bit.ly/3wHlvik`

`https://replit.com/@NuwanKodagoda/2024-03-03`

Variadic Parameters

```
func average(_ numbers: Double...) -> Double {  
    var total = 0.0  
    for number in numbers {  
        total += number  
    }  
    return total / Double(numbers.count)  
}
```

These three dots (called an ellipsis) indicate that this parameter can have any number of arguments.

```
let a = average(10, 21, 3.2, 16)  
print (average(2, 4, 6))
```

→ a gets set to 16.625

→ prints out 4.0

Here there seems to be a relaxation of conversions from Int to Double
Variadics can have zero elements, but your code should handle this

Exercise

- Calculate multiple grades for multiple student marks



`https://bit.ly/3wHlvik`

`https://replit.com/@NuwanKodagoda/2024-03-03`

Signatures

A function's signature is a combination of its *name*, plus its *argument names and types*, plus its *return type* (if it has one).

```
func test(no : Int)
func test(anotherNo : Int)
func test(no : Double)
func test( _ no: Double)
```

Are all unique signatures

However if the above are defined we can't have a function

```
func test(no : Double) -> Double
```


Exercise - Overload a validate function

- Function to input an integer for a range to and from
- Function to input a string which is less than a given number



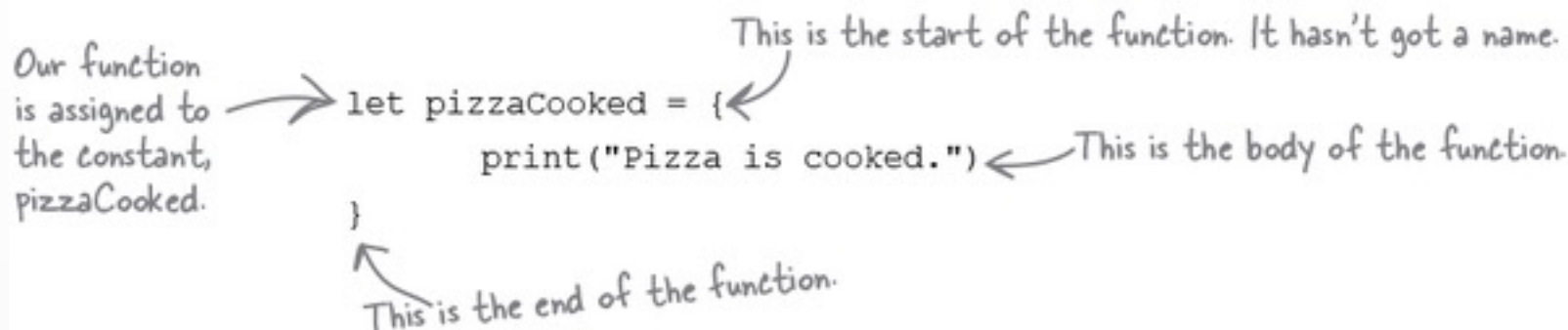
`https://bit.ly/3wHlvik`

`https://replit.com/@NuwanKodagoda/2024-03-03`

Closures

- **Functions are useful, but sometimes you need more flexibility.** Swift allows you to **use a function as a type**, just like an integer or string. This means **you can create a function and assign it to a variable**. Once it's assigned to a variable, you can call the function using the variable, or pass the function to other functions as a parameter. When you create and use a function in this manner, it's called a **closure**.
Closures are useful because they can **capture references** to constants and variables from the context in which they are defined. This is called **closing over a value**, hence the name.

Closures



```
let pizzaCooked = {  
  print("Pizza is cooked.")  
}
```

Our function is assigned to the constant, pizzaCooked. →

← This is the start of the function. It hasn't got a name.

← This is the body of the function.

← This is the end of the function.

Closures are really useful when you want to store some functionality and use it later, or use it again.

Particular occasions when you often need to store some functionality include:

- When you want to wait some time, then run code afterward
- When you have a user interface, and you need some sort of animation or interaction to finish before you run code
- When you are performing some sort of indeterminately long network operation (like a download) and want your code to run when it's done

Closures

```
func cooked(pizza: String, minutes: Int) {  
    print("\(pizza) Pizza is cooked in \(minutes) minutes.")  
}
```

```
let pizzaCooked = { (pizza: String, minutes: Int) in  
    print("\(pizza) Pizza is cooked in \(minutes) minutes.")  
}
```

Functions and closures

- Write a program that takes an array of integers and returns the sum of all even numbers in the array using both a function and a closure.



`https://bit.ly/3wHlvik`

`https://replit.com/@NuwanKodagoda/2024-03-03`

Stored properties

```
class Circle {  
    var radius: Double = 0  
}  
  
let circle = Circle()  
circle.radius = 10  
  
print("radius: \(circle.radius)")
```

Property Observers

```
class Circle {  
    var radius: Double = 0 {  
        didSet {  
            if radius < 0 {  
                radius = oldValue  
            }  
        }  
    }  
}  
  
let circle = Circle()  
  
circle.radius = -10  
print("radius: \(circle.radius)")  
circle.radius = 10  
print("radius: \(circle.radius)")
```

Property Observers

```
class Circle {  
    var radius: Double = 0 {  
        willSet {  
            print("About to assign the new value \$(newValue)")  
        }  
        didSet {  
            if radius < 0 {  
                radius = oldValue  
            }  
        }  
    }  
}
```

```
let circle = Circle()  
circle.radius = 10
```


Computed Properties

```
class Circle {  
  var radius: Double = 0 {  
    didSet {  
      if radius < 0 {  
        radius = oldValue  
      }  
    }  
  }  
  var area: Double {  
    get {  
      return radius * radius * Double.pi  
    }  
  }  
}  
  
let circle = Circle()  
circle.radius = 5  
print("area: \(circle.area)")
```

Computed Properties

```
import Foundation
```

```
class Circle {
```

```
    var radius: Double = 0 {
```

```
        didSet {
```

```
            if radius < 0 {
```

```
                radius = oldValue
```

```
            }
```

```
        } }
```

```
    var area: Double {
```

```
        get {
```

```
            return radius * radius * Double.pi
```

```
        }
```

```
        set(newArea) {
```

```
            radius = sqrt(newArea / Double.pi)
```

```
        }
```

```
    } }
```

```
let circle = Circle()
```

```
circle.area = 25
```

```
print("radius: \(circle.radius)")
```

Initialization of stored properties

- Initialize the value inside the init method
- Set a default value for the property

```
class Circle {  
    var radius: Double  
    var identifier: Int = 0  
  
    init(radius: Double) {  
        self.radius = radius  
    }  
}
```

```
var circle = Circle(radius: 5)
```

Exercise

- A Temperature class that can store values internally as Celcius but handle temperatures in Fahrenheit and Kelvin



`https://bit.ly/3wHlvik`

`https://replit.com/@NuwanKodagoda/2024-03-03`