巨匠線上真人

iOS行動程式基礎開發上架

第五堂：函式和閉鎖

本堂教學重點

1. 定義和呼叫函式
2. 參數和傳回值
3. 引數標籤和參數名
4. 函數資料類型
5. 巢狀函式
6. 閉鎖運算式
7. 尾端閉鎖
8. 截取值
9. 閉鎖是參考類型

1.定義和呼叫函式

```swift
func greet(person: String) -> String {
    let greeting = "Hello, " + person + "!"
    return greeting
}
```

```swift
print(greet(person: "Anna"))
// Prints "Hello, Anna!"
print(greet(person: "Brian"))
// Prints "Hello, Brian!"
```

```swift
func greetAgain(person: String) -> String {
    return "Hello again, " + person + "!"
}
print(greetAgain(person: "Anna"))
// Prints "Hello again, Anna!"
```

2.參數和傳回值

沒有參數的函式

```
•    func sayHelloWorld() -> String {
•        return "hello, world"
•    }
•    print(sayHelloWorld())
•    // Prints "hello, world"
•
```

2.參數和傳回值

多個參數的函式

```swift
func greet(person: String, alreadyGreeted: Bool) -> String {
    if alreadyGreeted {
        return greetAgain(person: person)
    } else {
        return greet(person: person)
    }
}
print(greet(person: "Tim", alreadyGreeted: true))
// Prints "Hello again, Tim!"
```

2.參數和傳回值

沒有傳回值的函式

```
•   func greet(person: String) {
•       print("Hello, \(person)!")
•   }
•   greet(person: "Dave")
•   // Prints "Hello, Dave!"

•   func printAndCount(string: String) -> Int {
•       print(string)
•       return string.count
•   }
•   func printWithoutCounting(string: String) {
•       let _ = printAndCount(string: string)
•   }
•   printAndCount(string: "hello, world")
•   // prints "hello, world" and returns a value of 12
•   printWithoutCounting(string: "hello, world")
•   // prints "hello, world" but does not return a value
•
```

2.參數和傳回值

傳回多個值的函式

```
•    func minMax(array: [Int]) -> (min: Int, max: Int) {
•        var currentMin = array[0]
•        var currentMax = array[0]
•        for value in array[1..<array.count] {
•            if value < currentMin {
•                currentMin = value
•            } else if value > currentMax {
•                currentMax = value
•            }
•        }
•        return (currentMin, currentMax)
•    }
•
```

```
•    let bounds = minMax(array: [8, -6, 2, 109, 3, 71])
•    print("min is \(bounds.min) and max is \(bounds.max)")
•    // Prints "min is -6 and max is 109"
```

2.參數和傳回值

傳回可nil多個值的函式

```
•    func minMax(array: [Int]) -> (min: Int, max: Int)? {
•        if array.isEmpty { return nil }
•        var currentMin = array[0]
•        var currentMax = array[0]
•        for value in array[1..<array.count] {
•            if value < currentMin {
•                currentMin = value
•            } else if value > currentMax {
•                currentMax = value
•            }
•        }
•        return (currentMin, currentMax)
•    }
•

•    if let bounds = minMax(array: [8, -6, 2, 109, 3, 71]) {
•        print("min is \(bounds.min) and max is \(bounds.max)")
•    }
•    // Prints "min is -6 and max is 109"
```

3.引數標籤和參數名

傳回可nil多個值的函式

```
•   func someFunction(firstParameterName: Int, secondParameterName: Int) {
•       // In the function body, firstParameterName and secondParameterName
•       // refer to the argument values for the first and second parameters.
•   }
•   someFunction(firstParameterName: 1, secondParameterName: 2)
•
```

3.引數標籤和參數名

指定引數標籤名稱

```
•   func someFunction(argumentLabel parameterName: Int) {
•       // In the function body, parameterName refers to the argument value
•       // for that parameter.
•   }

•   func greet(person: String, from hometown: String) -> String {
•       return "Hello \(person)!  Glad you could visit from \(hometown)."
•   }
•   print(greet(person: "Bill", from: "Cupertino"))
•   // Prints "Hello Bill!  Glad you could visit from Cupertino."
```

3.引數標籤和參數名

省略引數標籤名稱

```
func someFunction(_ firstParameterName: Int, secondParameterName: Int) {
    // In the function body, firstParameterName and secondParameterName
    // refer to the argument values for the first and second parameters.
}
someFunction(1, secondParameterName: 2)
```

3.引數標籤和參數名

預設參數值

```
•    func someFunction(parameterWithoutDefault: Int, parameterWithDefault: Int = 12) {
•        // If you omit the second argument when calling this function, then
•        // the value of parameterWithDefault is 12 inside the function body.
•    }
•    someFunction(parameterWithoutDefault: 3, parameterWithDefault: 6) // parameterWithDefault is 6
•    someFunction(parameterWithoutDefault: 4) // parameterWithDefault is 12
```

3.引數標籤和參數名

不限數量參數

```
•    func arithmeticMean(_ numbers: Double...) -> Double {
•        var total: Double = 0
•        for number in numbers {
•            total += number
•        }
•        return total / Double(numbers.count)
•    }
•    arithmeticMean(1, 2, 3, 4, 5)
•    // returns 3.0, which is the arithmetic mean of these five numbers
•    arithmeticMean(3, 8.25, 18.75)
•    // returns 10.0, which is the arithmetic mean of these three numbers
•
```

3.引數標籤和參數名

In-out 參數

```
•   func swapTwoInts(_ a: inout Int, _ b: inout Int) {
•       let temporaryA = a
•       a = b
•       b = temporaryA
•   }

•   var someInt = 3
•   var anotherInt = 107
•   swapTwoInts(&someInt, &anotherInt)
•   print("someInt is now \(someInt), and anotherInt is now \(anotherInt)")
•   // Prints "someInt is now 107, and anotherInt is now 3"
```

4.函式資料類型

```
(Int, Int) -> Int
```

```
•    func addTwoInts(_ a: Int, _ b: Int) -> Int {
•        return a + b
•    }
•    func multiplyTwoInts(_ a: Int, _ b: Int) -> Int {
•        return a * b
•    }
•
```

```
() -> Void
```

```
•    func printHelloWorld() {
•        print("hello, world")
•    }
•
```

4.函式資料類型

函式當作參數

```
•   func printMathResult(_ mathFunction: (Int, Int) -> Int, _ a: Int, _ b: Int) {
•       print("Result: \(mathFunction(a, b))")
•   }
•   printMathResult(addTwoInts, 3, 5)
•   // Prints "Result: 8"
```

4.函式資料類型

函式當作傳回值

```
•    func stepForward(_ input: Int) -> Int {
•        return input + 1
•    }
•    func stepBackward(_ input: Int) -> Int {
•        return input - 1
•    }

•    func chooseStepFunction(backward: Bool) -> (Int) -> Int {
•        return backward ? stepBackward : stepForward
•    }

•    var currentValue = 3
•    let moveNearerToZero = chooseStepFunction(backward: currentValue > 0)
•    // moveNearerToZero now refers to the stepBackward() function
```

```
•    print("Counting to zero:")
•    // Counting to zero:
•    while currentValue != 0 {
•        print("\(currentValue)... ")
•        currentValue = moveNearerToZero(currentValue)
•    }
•    print("zero!")
•    // 3...
•    // 2...
•    // 1...
•    // zero!
•
```

5.巢狀函式

函式當作傳回值

```
•    func chooseStepFunction(backward: Bool) -> (Int) -> Int {
•        func stepForward(input: Int) -> Int { return input + 1 }
•        func stepBackward(input: Int) -> Int { return input - 1 }
•        return backward ? stepBackward : stepForward
•    }
•    var currentValue = -4
•    let moveNearerToZero = chooseStepFunction(backward: currentValue > 0)
•    // moveNearerToZero now refers to the nested stepForward() function
•    while currentValue != 0 {
•        print("\(currentValue)... ")
•        currentValue = moveNearerToZero(currentValue)
•    }
•    print("zero!")
•    // -4...
•    // -3...
•    // -2...
•    // -1...
•    // zero!
•
```

6.閉鎖運算式

陣列排序方法

```
let names = ["Chris", "Alex", "Ewa", "Barry", "Daniella"]
```

-
  ```
  func backward(_ s1: String, _ s2: String) -> Bool {
  ```

-
  ```
      return s1 > s2
  ```

-
  ```
  }
  ```

-
  ```
  var reversedNames = names.sorted(by: backward)
  ```

-
  ```
  // reversedNames is equal to ["Ewa", "Daniella", "Chris", "Barry", "Alex"]
  ```
-

6.閉鎖運算式

閉鎖語法

```
•    { (parameters) -> return type in
•        statements
•    }
```

```
•    reversedNames = names.sorted(by: { (s1: String, s2: String) -> Bool in return s1 > s2 } )
```

6.閉鎖運算式

推測類型

- `reversedNames = names.sorted(by: { s1, s2 in return s1 > s2 } )`

6.閉鎖運算式

省略return的單行運算式閉鎖

-
  ```
  reversedNames = names.sorted(by: { s1, s2 in s1 > s2 } )
  ```

6.閉鎖運算式

省略引數標籤名

-
```
reversedNames = names.sorted(by: { $0 > $1 } )
```

6.閉鎖運算式

運算子函式

-
  ```
  reversedNames = names.sorted(by: >)
  ```

6.尾端閉鎖

```
•    func someFunctionThatTakesAClosure(closure: () -> Void) {
•        // function body goes here
•    }

•    // Here's how you call this function without using a trailing closure:

•    someFunctionThatTakesAClosure(closure: {
•        // closure's body goes here
•    })

•    // Here's how you call this function with a trailing closure instead:

•    someFunctionThatTakesAClosure() {
•        // trailing closure's body goes here
•    }
```

```
reversedNames = names.sorted() { $0 > $1 }

reversedNames = names.sorted { $0 > $1 }
```

6.尾端閉鎖

```swift
let digitNames = [
    0: "Zero", 1: "One", 2: "Two",   3: "Three", 4: "Four",
    5: "Five", 6: "Six", 7: "Seven", 8: "Eight", 9: "Nine"
]
let numbers = [16, 58, 510]
```

```swift
let strings = numbers.map { (number) -> String in
    var number = number
    var output = ""
    repeat {
        output = digitNames[number % 10]! + output
        number /= 10
    } while number > 0
    return output
}
// strings is inferred to be of type [String]
// its value is ["OneSix", "FiveEight", "FiveOneZero"]
```

7.截取值

```
•   func makeIncrementer(forIncrement amount: Int) -> () -> Int {
•       var runningTotal = 0
•       func incrementer() -> Int {
•           runningTotal += amount
•           return runningTotal
•       }
•       return incrementer
•   }

•   func incrementer() -> Int {
•       runningTotal += amount
•       return runningTotal
•   }

let incrementByTen = makeIncrementer(forIncrement: 10)
```

```
•   incrementByTen()
•   // returns a value of 10
•   incrementByTen()
•   // returns a value of 20
•   incrementByTen()
•   // returns a value of 30
```

```
•   let incrementBySeven = makeIncrementer(forIncrement: 7)
•   incrementBySeven()
•   // returns a value of 7
```

```
•   incrementByTen()
•   // returns a value of 40
```

8.閉鎖是參考類型

- let alsoIncrementByTen = incrementByTen
- alsoIncrementByTen()
- // returns a value of 50

- incrementByTen()
- // returns a value of 60