iOS行動程式基礎開發上架

第二堂：基本運算子、字串和字元

本堂教學重點

基本運算子

1. 指定運算子
2. 算數運算子
3. 一元運算子
4. 組合指定運算子
5. 比較運算子
6. 三元運算子
7. nil連結運算子
8. 範圍運算子
9. 邏輯運算子
10. 優先運算

本堂教學重點

字串和字元

11. 字串表示法
12. 多行文字
13. 建立空字串
14. 字串變動能力
15. 字元的操作
16. 字串和字元的整合
17. 字串插補
18. 字串的修改和存取
19. Substrings

1指定運算子

- `let b = 10`
- `var a = 5`
- `a = b`
- `// a is now equal to 10`

- `let (x, y) = (1, 2)`
- `// x is equal to 1, and y is equal to 2`

2算數運算子

```
1 + 2       // equals 3
5 - 3       // equals 2
2 * 3       // equals 6
10.0 / 2.5  // equals 4.0
```

```
"hello, " + "world"  // equals "hello, world"
```

```
9 % 4
```

```
let three = 3
let minusThree = -three       // minusThree equals -3
let plusThree = -minusThree   // plusThree equals 3, or "minus
```

4組合指定運算子

- `var a = 1`
- `a += 2`
- `// a is now equal to 3`

5比較運算子

- `1 == 1  // true because 1 is equal to 1`
- `2 != 1  // true because 2 is not equal to 1`
- `2 > 1  // true because 2 is greater than 1`
- `1 < 2  // true because 1 is less than 2`
- `1 >= 1  // true because 1 is greater than or equal to 1`
- `2 <= 1  // false because 2 is not less than or equal to 1`

```
let name = "world"
if name == "world" {
    print("hello, world")
} else {
    print("I'm sorry \(name), but I don't recognize you")
}
// Prints "hello, world", because name is indeed equal to "world".
```

6三元運算子

```
let contentHeight = 40
let hasHeader = true
let rowHeight: Int
if hasHeader {
    rowHeight = contentHeight + 50
} else {
    rowHeight = contentHeight + 20
}
// rowHeight is equal to 90
```

```
let contentHeight = 40
let hasHeader = true
let rowHeight = contentHeight + (hasHeader ? 50 : 20)
// rowHeight is equal to 90
```

7nil連結運算子

```
let defaultColorName = "red"
var userDefinedColorName: String?   // defaults to nil
var colorNameToUse = userDefinedColorName ?? defaultColorName
// userDefinedColorName is nil, so colorNameToUse is set to the default of "red"
```

```
userDefinedColorName = "green"
colorNameToUse = userDefinedColorName ?? defaultColorName
// userDefinedColorName is not nil, so colorNameToUse is set to "green"
```

8範圍運算子

```
for index in 1...5 {
    print("\(index) times 5 is \(index * 5)")
}
// 1 times 5 is 5
// 2 times 5 is 10
// 3 times 5 is 15
// 4 times 5 is 20
// 5 times 5 is 25
```

```
let names = ["Anna", "Alex", "Brian", "Jack"]
let count = names.count
for i in 0..<count {
    print("Person \(i + 1) is called \(names[i])")
}
// Person 1 is called Anna
// Person 2 is called Alex
// Person 3 is called Brian
// Person 4 is called Jack
```

```
for name in names[2...] {
    print(name)
}
// Brian
// Jack
```

```
for name in names[...2] {
    print(name)
}
// Anna
// Alex
// Brian
```

```
for name in names[..<2] {
    print(name)
}
// Anna
// Alex
```

9邏輯運算子

```
let allowedEntry = false
if !allowedEntry {
    print("ACCESS DENIED")
}
// Prints "ACCESS DENIED"
```

```
if enteredDoorCode && passedRetinaScan || hasDoorKey || knowsOverridePassword {
    print("Welcome!")
} else {
    print("ACCESS DENIED")
}
// Prints "Welcome!"
```

```
let enteredDoorCode = true
let passedRetinaScan = false
if enteredDoorCode && passedRetinaScan {
    print("Welcome!")
} else {
    print("ACCESS DENIED")
}
// Prints "ACCESS DENIED"
```

```
let hasDoorKey = false
let knowsOverridePassword = true
if hasDoorKey || knowsOverridePassword {
    print("Welcome!")
} else {
    print("ACCESS DENIED")
}
// Prints "Welcome!"
```

11字串表示法

- `let someString = "Some string literal value"`

12多行文字

```
let quotation = """
The White Rabbit put on his spectacles.  "Where shall I begin,
please your Majesty?" he asked.

"Begin at the beginning," the King said gravely, "and go on
till you come to the end; then stop."
"""
```

```
let softWrappedQuotation = """
The White Rabbit put on his spectacles.  "Where shall I begin, \
please your Majesty?" he asked.

"Begin at the beginning," the King said gravely, "and go on \
till you come to the end; then stop."
"""
```

13建立空字串

```
var emptyString = ""               // empty string literal
var anotherEmptyString = String()  // initializer syntax
// these two strings are both empty, and are equivalent to each other
```

```
if emptyString.isEmpty {
    print("Nothing to see here")
}
// Prints "Nothing to see here"
```

14建立空字串

```
var variableString = "Horse"
variableString += " and carriage"
// variableString is now "Horse and carriage"

let constantString = "Highlander"
constantString += " and another Highlander"
// this reports a compile-time error - a constant string cannot be modified
```

15字元的操作

```
for character in "Dog!🐶" {
    print(character)
}
// D
// o
// g
// !
// 🐶

let catCharacters: [Character] = ["C", "a", "t", "!", "🐱"]
let catString = String(catCharacters)
print(catString)
```

16字串和字元的整合

```
let string1 = "hello"
let string2 = " there"
var welcome = string1 + string2
```

```
var instruction = "look over"
instruction += string2
// instruction now equals "look over there"
```

```
let exclamationMark: Character = "!"
welcome.append(exclamationMark)
// welcome now equals "hello there!"
```

17字串插補

```
let multiplier = 3
let message = "\(multiplier) times 2.5 is \(Double(multiplier) * 2.5)"
// message is "3 times 2.5 is 7.5"
```

18字串的修改和存取

```
let greeting = "Guten Tag!"
greeting[greeting.startIndex]
// G
greeting[greeting.index(before: greeting.endIndex)]
// !
greeting[greeting.index(after: greeting.startIndex)]
// u
let index = greeting.index(greeting.startIndex, offsetBy: 7)
greeting[index]
// a

greeting[greeting.endIndex] // Error
greeting.index(after: greeting.endIndex) // Error



for index in greeting.indices {
    print("\(greeting[index]) ", terminator: "")
}
// Prints "G u t e n   T a g ! "
```

18字串的修改和存取

```
var welcome = "hello"
welcome.insert("!", at: welcome.endIndex)
// welcome now equals "hello!"

welcome.insert(contentsOf: " there", at: welcome.index(before: welcome.endIndex))
// welcome now equals "hello there!"
```

```
welcome.remove(at: welcome.index(before: welcome.endIndex))
// welcome now equals "hello there"

let range = welcome.index(welcome.endIndex, offsetBy: -6)..<welcome.endIndex
welcome.removeSubrange(range)
// welcome now equals "hello"
```

19Substrings

```
let greeting = "Hello, world!"
let index = greeting.firstIndex(of: ",") ?? greeting.endIndex
let beginning = greeting[..<index]
// beginning is "Hello"

// Convert the result to a String for long-term storage.
let newString = String(beginning)
```