巨匠線上真人

iOS行動程式基礎開發上架

第三堂：集合物件

本堂教學重點

1. 陣列

- 型別簡要表示法
- 建立空陣列
- 建立有default value的陣列
- 陣列可相加
- 使用陣列表示法建立陣列
- 存取和修改陣列
- 讀取陣列內全部元素

2. 組合

- 型別表示法
- 建立空組合
- 使用陣列表示法建立組合
- 存取和修改組合
- 讀遍組合內全部元素
- 執行組合運算

3. 詞典物件

- 型別簡要表示法
- 建立空詞典
- 使用詞典簡易表示法
- 存取和修改詞典物件
- 讀遍詞典全部元素

1.陣列

型別簡要表示法-[Element]

建立空陣列

```
var someInts = [Int]()
print("someInts is of type [Int] with \(someInts.count) items.")
// Prints "someInts is of type [Int] with 0 items."

someInts.append(3)
// someInts now contains 1 value of type Int
someInts = []
// someInts is now an empty array, but is still of type [Int]
```

1.陣列

建立有default value的陣列

```
var threeDoubles = Array(repeating: 0.0, count: 3)
// threeDoubles is of type [Double], and equals [0.0, 0.0, 0.0]
```

陣列可相加

```
var anotherThreeDoubles = Array(repeating: 2.5, count: 3)
// anotherThreeDoubles is of type [Double], and equals [2.5, 2.5, 2.5]

var sixDoubles = threeDoubles + anotherThreeDoubles
// sixDoubles is inferred as [Double], and equals [0.0, 0.0, 0.0, 2.5, 2.5, 2.5]
```

1.陣列

使用陣列表示法建立陣列

```
var shoppingList: [String] = ["Eggs", "Milk"]
// shoppingList has been initialized with two initial items

var shoppingList = ["Eggs", "Milk"]
```

存取和修改陣列

```
print("The shopping list contains \(shoppingList.count) items.")
// Prints "The shopping list contains 2 items."

if shoppingList.isEmpty {
    print("The shopping list is empty.")
} else {
    print("The shopping list is not empty.")
}
// Prints "The shopping list is not empty."

```

1.陣列

存取和修改陣列

```
shoppingList.append("Flour")
// shoppingList now contains 3 items, and someone is making pancakes

shoppingList += ["Baking Powder"]
// shoppingList now contains 4 items
shoppingList += ["Chocolate Spread", "Cheese", "Butter"]
// shoppingList now contains 7 items

var firstItem = shoppingList[0]
// firstItem is equal to "Eggs"

shoppingList[0] = "Six eggs"
// the first item in the list is now equal to "Six eggs" rather than "Eggs"

shoppingList[4...6] = ["Bananas", "Apples"]
// shoppingList now contains 6 items
```

1.陣列

存取和修改陣列

```
shoppingList.insert("Maple Syrup", at: 0)
// shoppingList now contains 7 items
// "Maple Syrup" is now the first item in the list


let mapleSyrup = shoppingList.remove(at: 0)
// the item that was at index 0 has just been removed
// shoppingList now contains 6 items, and no Maple Syrup
// the mapleSyrup constant is now equal to the removed "Maple Syrup" string

firstItem = shoppingList[0]
// firstItem is now equal to "Six eggs"


let apples = shoppingList.removeLast()
// the last item in the array has just been removed
// shoppingList now contains 5 items, and no apples
// the apples constant is now equal to the removed "Apples" string
```

1.陣列

讀取陣列內全部元素

```
for item in shoppingList {
    print(item)
}
// Six eggs
// Milk
// Flour
// Baking Powder
// Bananas


for (index, value) in shoppingList.enumerated() {
    print("Item \(index + 1): \(value)")
}
// Item 1: Six eggs
// Item 2: Milk
// Item 3: Flour
// Item 4: Baking Powder
// Item 5: Bananas

```

2.組合

型別表示法-Set<Element>

建立空組合

```
var letters = Set<Character>()
print("letters is of type Set<Character> with \(letters.count) items.")
// Prints "letters is of type Set<Character> with 0 items."


letters.insert("a")
// letters now contains 1 value of type Character
letters = []
// letters is now an empty set, but is still of type Set<Character>
```

2.組合

使用陣列表示法建立組合

```
var favoriteGenres: Set<String> = ["Rock", "Classical", "Hip hop"]
// favoriteGenres has been initialized with three initial items

var favoriteGenres: Set = ["Rock", "Classical", "Hip hop"]
```

存取和修改組合

```
print("I have \(favoriteGenres.count) favorite music genres.")
// Prints "I have 3 favorite music genres."

if favoriteGenres.isEmpty {
    print("As far as music goes, I'm not picky.")
} else {
    print("I have particular music preferences.")
}
// Prints "I have particular music preferences."
```

2.組合

存取和修改組合

```
favoriteGenres.insert("Jazz")
// favoriteGenres now contains 4 items

if let removedGenre = favoriteGenres.remove("Rock") {
    print("\(removedGenre)? I'm over it.")
} else {
    print("I never much cared for that.")
}
// Prints "Rock? I'm over it."

if favoriteGenres.contains("Funk") {
    print("I get up on the good foot.")
} else {
    print("It's too funky in here.")
}
// Prints "It's too funky in here."
```

2.組合

讀遍組合內全部元素

```
for genre in favoriteGenres {
    print("\(genre)")
}
// Classical
// Jazz
// Hip hop
```

```
for genre in favoriteGenres.sorted() {
    print("\(genre)")
}
// Classical
// Hip hop
// Jazz
```

2.組合

執行組合運算

```
let oddDigits: Set = [1, 3, 5, 7, 9]
let evenDigits: Set = [0, 2, 4, 6, 8]
let singleDigitPrimeNumbers: Set = [2, 3, 5, 7]

oddDigits.union(evenDigits).sorted()
// [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
oddDigits.intersection(evenDigits).sorted()
// []
oddDigits.subtracting(singleDigitPrimeNumbers).sorted()
// [1, 9]
oddDigits.symmetricDifference(singleDigitPrimeNumbers).sorted()
// [1, 2, 9]
```

3.詞典物件

型別簡要表示法-Dictionary<Key, Value>, [Key: Value]

建立空詞典

```
var namesOfIntegers = [Int: String]()
// namesOfIntegers is an empty [Int: String] dictionary

namesOfIntegers[16] = "sixteen"
// namesOfIntegers now contains 1 key-value pair
namesOfIntegers = [:]
// namesOfIntegers is once again an empty dictionary of type [Int: String]
```

3.詞典物件

用詞典簡易表示法

```
var airports: [String: String] = ["YYZ": "Toronto Pearson", "DUB": "Dublin"]
var airports = ["YYZ": "Toronto Pearson", "DUB": "Dublin"]
```

存取和修改詞典物件

```
print("The airports dictionary contains \(airports.count) items.")
// Prints "The airports dictionary contains 2 items."

if airports.isEmpty {
    print("The airports dictionary is empty.")
} else {
    print("The airports dictionary is not empty.")
}
// Prints "The airports dictionary is not empty."
```

3.詞典物件

存取和修改詞典物件

```
airports["LHR"] = "London"
// the airports dictionary now contains 3 items

airports["LHR"] = "London Heathrow"
// the value for "LHR" has been changed to "London Heathrow"

if let oldValue = airports.updateValue("Dublin Airport", forKey: "DUB") {
    print("The old value for DUB was \(oldValue).")
}
// Prints "The old value for DUB was Dublin."

if let airportName = airports["DUB"] {
    print("The name of the airport is \(airportName).")
} else {
    print("That airport is not in the airports dictionary.")
}
// Prints "The name of the airport is Dublin Airport."
```

3.詞典物件

存取和修改詞典物件

```
airports["APL"] = "Apple International"
// "Apple International" is not the real airport for APL, so delete it
airports["APL"] = nil
// APL has now been removed from the dictionary


if let removedValue = airports.removeValue(forKey: "DUB") {
    print("The removed airport's name is \(removedValue).")
} else {
    print("The airports dictionary does not contain a value for DUB.")
}
// Prints "The removed airport's name is Dublin Airport."
```

3.詞典物件

讀遍詞典全部元素

```
for (airportCode, airportName) in airports {
    print("\(airportCode): \(airportName)")
}
// YYZ: Toronto Pearson
// LHR: London Heathrow

for airportCode in airports.keys {
    print("Airport code: \(airportCode)")
}
// Airport code: YYZ
// Airport code: LHR

for airportName in airports.values {
    print("Airport name: \(airportName)")
}
// Airport name: Toronto Pearson
// Airport name: London Heathrow
```

3.詞典物件

讀遍詞典全部元素

```
let airportCodes = [String](airports.keys)
// airportCodes is ["YYZ", "LHR"]

let airportNames = [String](airports.values)
// airportNames is ["Toronto Pearson", "London Heathrow"]
```