巨匠線上真人

iOS行動程式基礎開發上架

第四堂：流程控制

本堂教學重點

1. **For-in 迴圈**
2. **while 迴圈**
3. **Repeat-While**
4. **if**
5. **Switch**
   - 無貫穿
   - 區間符合
   - Tuples
   - Value Bindings
   - Where
   - Compound Cases
6. **改變流程控制**
   - continue
   - break
   - fallthrough
   - Early Exit

1.For-in 迴圈

```
•    let names = ["Anna", "Alex", "Brian", "Jack"]
•    for name in names {
•        print("Hello, \(name)!")
•    }
•    // Hello, Anna!
•    // Hello, Alex!
•    // Hello, Brian!
•    // Hello, Jack!

•    let numberOfLegs = ["spider": 8, "ant": 6, "cat": 4]
•    for (animalName, legCount) in numberOfLegs {
•        print("\(animalName)s have \(legCount) legs")
•    }
•    // ants have 6 legs
•    // cats have 4 legs
•    // spiders have 8 legs
•
```

1.For-in 迴圈

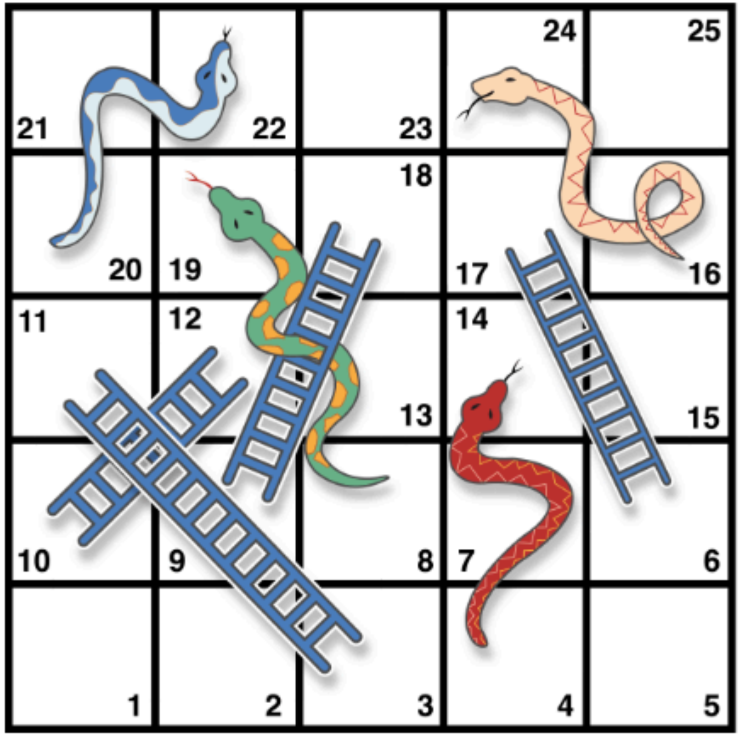
```
•   for index in 1...5 {
•       print("\(index) times 5 is \(index * 5)")
•   }
•   // 1 times 5 is 5
•   // 2 times 5 is 10
•   // 3 times 5 is 15
•   // 4 times 5 is 20
•   // 5 times 5 is 25
```

```
•   let base = 3
•   let power = 10
•   var answer = 1
•   for _ in 1...power {
•       answer *= base
•   }
•   print("\(base) to the power of \(power) is \(answer)")
•   // Prints "3 to the power of 10 is 59049"
•
```

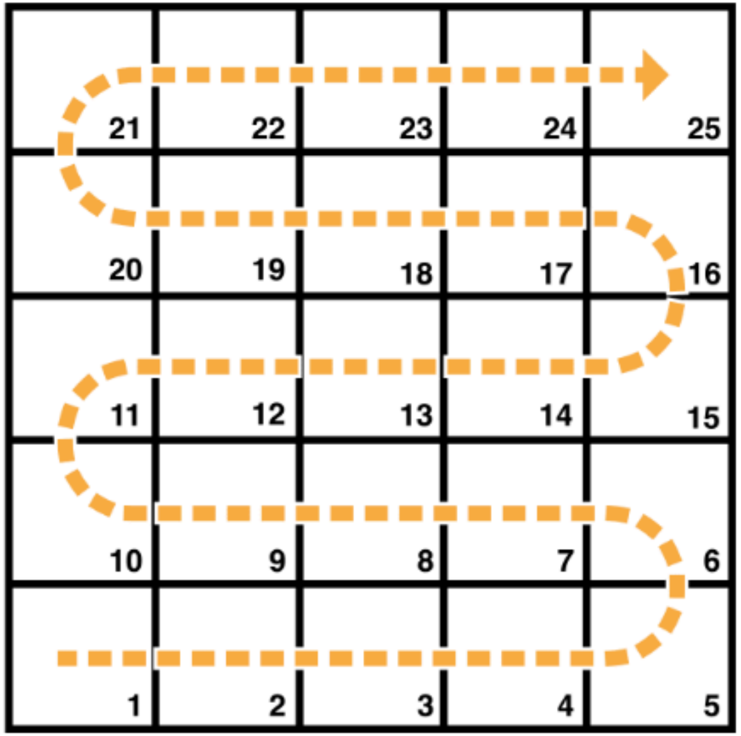
1.For-in 迴圈

```
•    let minutes = 60
•    for tickMark in 0..<minutes {
•        // render the tick mark each minute (60 times)
•    }

•    let minuteInterval = 5
•    for tickMark in stride(from: 0, to: minutes, by: minuteInterval) {
•        // render the tick mark every 5 minutes (0, 5, 10, 15 ... 45, 50, 55)
•    }


•    let hours = 12
•    let hourInterval = 3
•    for tickMark in stride(from: 3, through: hours, by: hourInterval) {
•        // render the tick mark every 3 hours (3, 6, 9, 12)
•    }
```

2.while 迴圈



21
22
23
24
25
20
19
18
17
16
11
12
13
14
15
10
9
8
7
6
1
2
3
4
5



21
22
23
24
25
20
19
18
17
16
11
12
13
14
15
10
9
8
7
6
1
2
3
4
5

2.while 迴圈

```
let finalSquare = 25
var board = [Int](repeating: 0, count: finalSquare + 1)

board[03] = +08; board[06] = +11; board[09] = +09; board[10] = +02
board[14] = -10; board[19] = -11; board[22] = -02; board[24] = -08


var square = 0
var diceRoll = 0
while square < finalSquare {
    // roll the dice
    diceRoll += 1
    if diceRoll == 7 { diceRoll = 1 }
    // move by the rolled amount
    square += diceRoll
    if square < board.count {
        // if we're still on the board, move up or down for a snake or a ladder
        square += board[square]
    }
}
print("Game over!")
```

3.Repeat-While 迴圈

```
•    let finalSquare = 25
•    var board = [Int](repeating: 0, count: finalSquare + 1)
•    board[03] = +08; board[06] = +11; board[09] = +09; board[10] = +02
•    board[14] = -10; board[19] = -11; board[22] = -02; board[24] = -08
•    var square = 0
•    var diceRoll = 0

•    repeat {
•        // move up or down for a snake or ladder
•        square += board[square]
•        // roll the dice
•        diceRoll += 1
•        if diceRoll == 7 { diceRoll = 1 }
•        // move by the rolled amount
•        square += diceRoll
•    } while square < finalSquare
•    print("Game over!")
•
```

4.if

- var temperatureInFahrenheit = 30
- if temperatureInFahrenheit <= 32 {
-     print("It's very cold. Consider wearing a scarf.")
- }
- // Prints "It's very cold. Consider wearing a scarf."

- temperatureInFahrenheit = 40
- if temperatureInFahrenheit <= 32 {
-     print("It's very cold. Consider wearing a scarf.")
- } else {
-     print("It's not that cold. Wear a t-shirt.")
- }
- // Prints "It's not that cold. Wear a t-shirt."
-

4.if

```
•    temperatureInFahrenheit = 90
•    if temperatureInFahrenheit <= 32 {
•        print("It's very cold. Consider wearing a scarf.")
•    } else if temperatureInFahrenheit >= 86 {
•        print("It's really warm. Don't forget to wear sunscreen.")
•    } else {
•        print("It's not that cold. Wear a t-shirt.")
•    }
•    // Prints "It's really warm. Don't forget to wear sunscreen."
```

```
•    temperatureInFahrenheit = 72
•    if temperatureInFahrenheit <= 32 {
•        print("It's very cold. Consider wearing a scarf.")
•    } else if temperatureInFahrenheit >= 86 {
•        print("It's really warm. Don't forget to wear sunscreen.")
•    }

•
```

5.switch

```swift
let someCharacter: Character = "z"
switch someCharacter {
case "a":
    print("The first letter of the alphabet")
case "z":
    print("The last letter of the alphabet")
default:
    print("Some other character")
}
// Prints "The last letter of the alphabet"
```

5.switch

無貫穿

```
•    let anotherCharacter: Character = "a"
•    switch anotherCharacter {
•    case "a": // Invalid, the case has an empty body
•    case "A":
•        print("The letter A")
•    default:
•        print("Not the letter A")
•    }
•    // This will report a compile-time error.
•
```

```
•    let anotherCharacter: Character = "a"
•    switch anotherCharacter {
•    case "a", "A":
•        print("The letter A")
•    default:
•        print("Not the letter A")
•    }
•    // Prints "The letter A"
•
```

5.switch

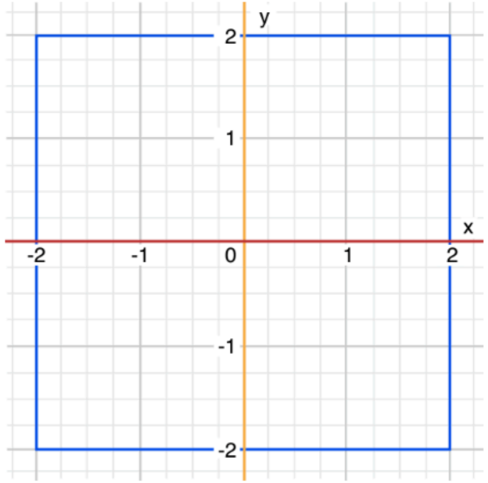
區間符合

```swift
let approximateCount = 62
let countedThings = "moons orbiting Saturn"
let naturalCount: String
switch approximateCount {
case 0:
    naturalCount = "no"
case 1..<5:
    naturalCount = "a few"
case 5..<12:
    naturalCount = "several"
case 12..<100:
    naturalCount = "dozens of"
case 100..<1000:
    naturalCount = "hundreds of"
default:
    naturalCount = "many"
}
print("There are \(naturalCount) \(countedThings).")
// Prints "There are dozens of moons orbiting Saturn."
```

5.switch

Tuples

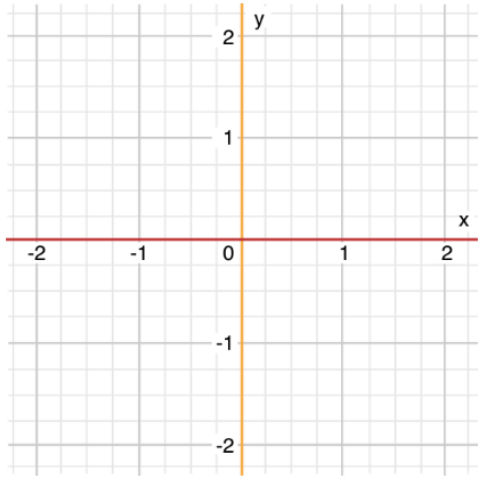
```
•    let somePoint = (1, 1)
•    switch somePoint {
•    case (0, 0):
•        print("\(somePoint) is at the origin")
•    case (_, 0):
•        print("\(somePoint) is on the x-axis")
•    case (0, _):
•        print("\(somePoint) is on the y-axis")
•    case (-2...2, -2...2):
•        print("\(somePoint) is inside the box")
•    default:
•        print("\(somePoint) is outside of the box")
•    }
•    // Prints "(1, 1) is inside the box"
•
```



5.switch

Value bindings

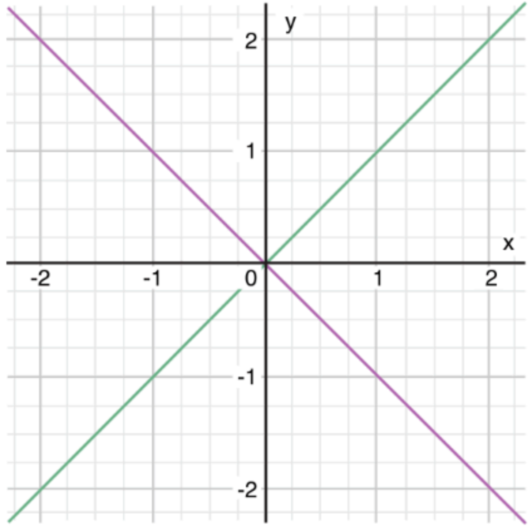
```
let anotherPoint = (2, 0)
switch anotherPoint {
case (let x, 0):
    print("on the x-axis with an x value of \(x)")
case (0, let y):
    print("on the y-axis with a y value of \(y)")
case let (x, y):
    print("somewhere else at (\(x), \(y))")
}
// Prints "on the x-axis with an x value of 2"
```



5.switch

Where

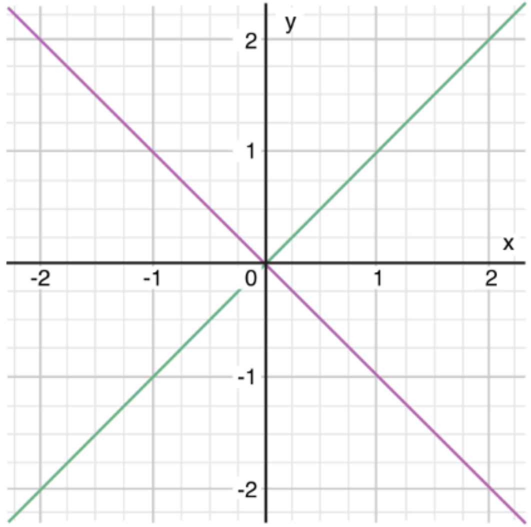
```
let yetAnotherPoint = (1, -1)
switch yetAnotherPoint {
case let (x, y) where x == y:
    print("(\(x), \(y)) is on the line x == y")
case let (x, y) where x == -y:
    print("(\(x), \(y)) is on the line x == -y")
case let (x, y):
    print("(\(x), \(y)) is just some arbitrary point")
}
// Prints "(1, -1) is on the line x == -y"
```



5.switch

Compound Cases

```
let someCharacter: Character = "e"
switch someCharacter {
case "a", "e", "i", "o", "u":
    print("\(someCharacter) is a vowel")
case "b", "c", "d", "f", "g", "h", "j", "k", "l", "m",
     "n", "p", "q", "r", "s", "t", "v", "w", "x", "y", "z":
    print("\(someCharacter) is a consonant")
default:
    print("\(someCharacter) is not a vowel or a consonant")
}
// Prints "e is a vowel"
```



6.改變流程控制

Continue

```
let puzzleInput = "great minds think alike"
var puzzleOutput = ""
let charactersToRemove: [Character] = ["a", "e", "i", "o", "u", " "]
for character in puzzleInput {
    if charactersToRemove.contains(character) {
        continue
    }
    puzzleOutput.append(character)
}
print(puzzleOutput)
// Prints "grtmndsthnklk"
```

6.改變流程控制

Early Exit

```swift
func greet(person: [String: String]) {
    guard let name = person["name"] else {
        return
    }

    print("Hello \(name)!")

    guard let location = person["location"] else {
        print("I hope the weather is nice near you.")
        return
    }

    print("I hope the weather is nice in \(location).")
}

greet(person: ["name": "John"])
// Prints "Hello John!"
// Prints "I hope the weather is nice near you."
greet(person: ["name": "Jane", "location": "Cupertino"])
// Prints "Hello Jane!"
// Prints "I hope the weather is nice in Cupertino."
```