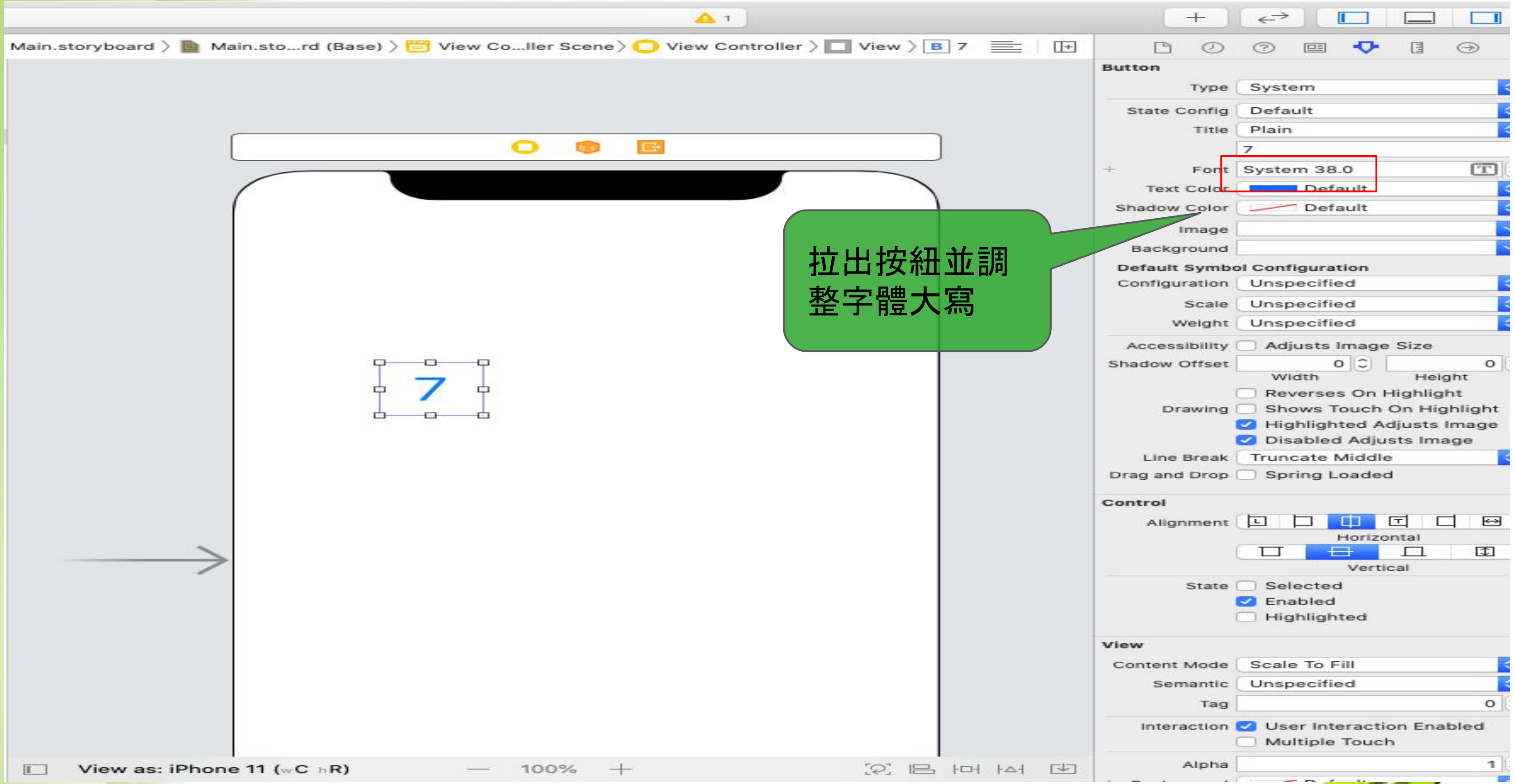
Calculator

National Taiwan University of Science and Technology

HPCLab
High-Performance Computing Laboratory

Calculator



拉出按鈕並調
整字體大寫

Main.storyboard > Main.sto...rd (Base) > View Co...ller Scene > View Controller > View > B 7

View as: iPhone 11 (wC hR) 100%

Button

Type	System
State Config	Default
Title	Plain
	7
Font	System 38.0
Text Color	Default
Shadow Color	Default
Image	
Background	

Default Symbol Configuration

Configuration	Unspecified
Scale	Unspecified
Weight	Unspecified
Accessibility	☐ Adjusts Image Size
Shadow Offset	Width 0, Height 0
Drawing	☐ Reverses On Highlight
	☐ Shows Touch On Highlight
	✓ Highlighted Adjusts Image
	✓ Disabled Adjusts Image
Line Break	Truncate Middle
Drag and Drop	☐ Spring Loaded

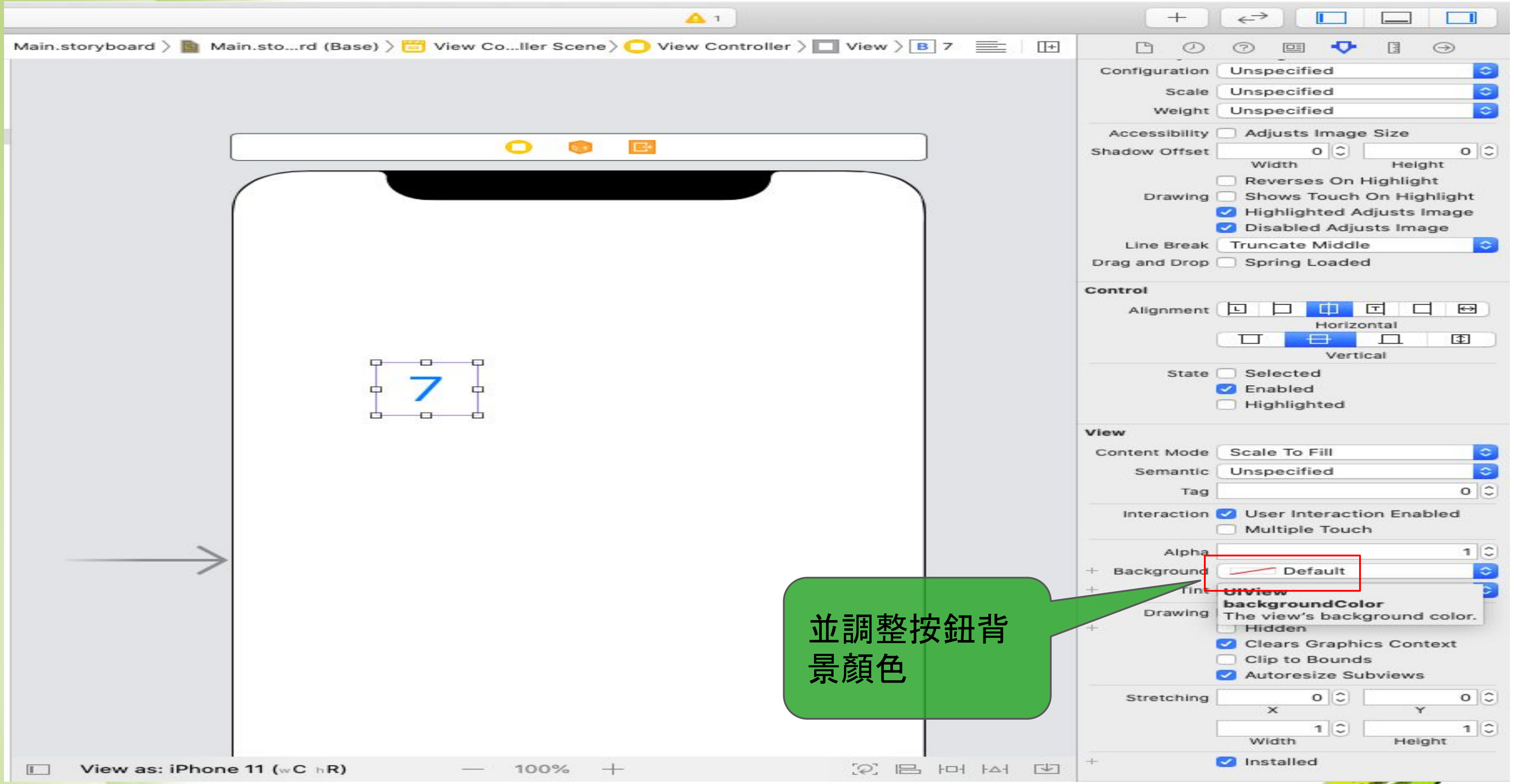
Control

Alignment	Horizontal
	Vertical
State	☐ Selected
	✓ Enabled
	☐ Highlighted

View

Content Mode	Scale To Fill
Semantic	Unspecified
Tag	0
Interaction	✓ User Interaction Enabled
	☐ Multiple Touch
Alpha	1

Calculator



Main.storyboard > Main.sto...rd (Base) > View Co...ller Scene > View Controller > View > B 7
7
View as: iPhone 11 (wC hR)
100%

Configuration: Unspecified
Scale: Unspecified
Weight: Unspecified
Accessibility: Adjusts Image Size
Shadow Offset: Width 0, Height 0
Drawing: Reverses On Highlight; Shows Touch On Highlight; ✓ Highlighted Adjusts Image; ✓ Disabled Adjusts Image
Line Break: Truncate Middle
Drag and Drop: Spring Loaded

Control

Alignment: Horizontal, Vertical
State: Selected; ✓ Enabled; Highlighted

View

Content Mode: Scale To Fill
Semantic: Unspecified
Tag: 0
Interaction: ✓ User Interaction Enabled; Multiple Touch
Alpha: 1
Background: Default
Tint

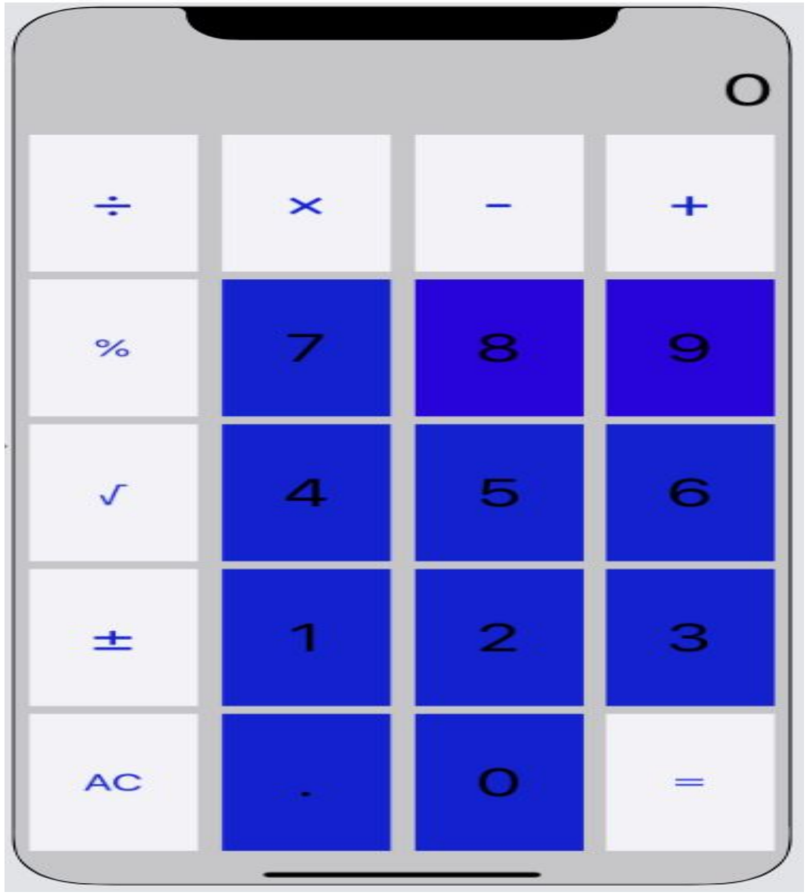
UIView
backgroundColor
The view's background color.

Drawing: Hidden; ✓ Clears Graphics Context; Clip to Bounds; ✓ Autoresize Subviews
Stretching: X 0, Y 0; Width 1, Height 1
✓ Installed

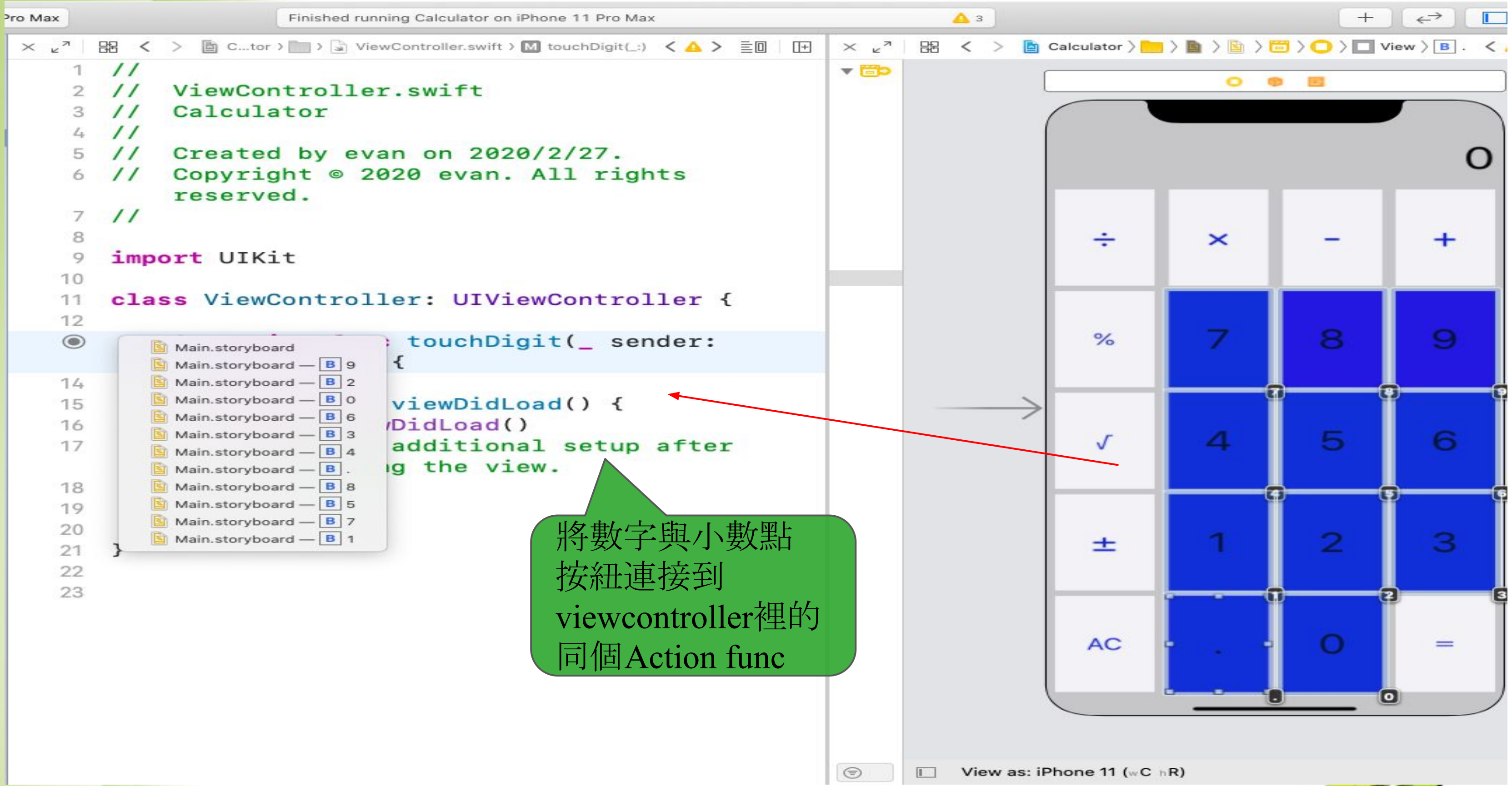
並調整按鈕背景顏色

Calculator

將計算機layout拉好



Calculator



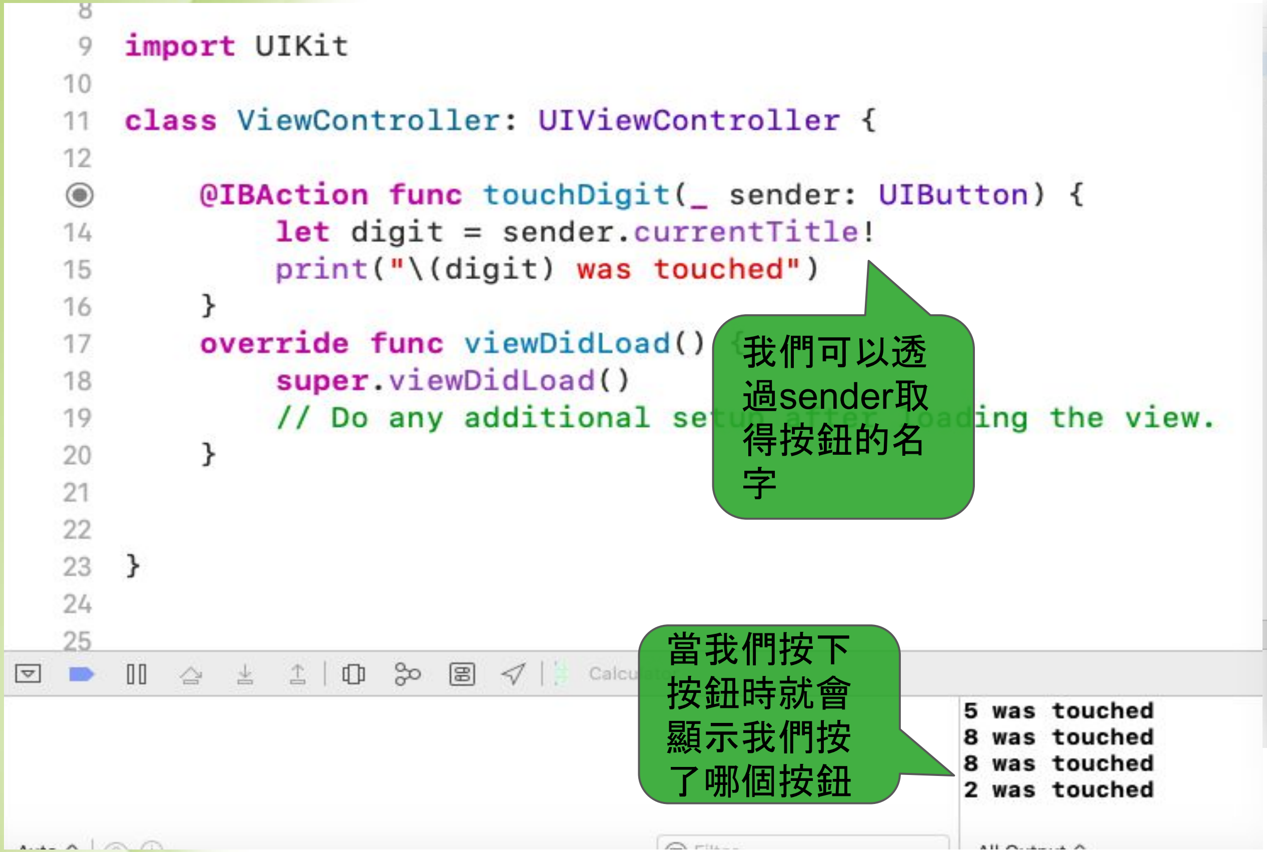
Finished running Calculator on iPhone 11 Pro Max

```
//
//  ViewController.swift
//  Calculator
//
//  Created by evan on 2020/2/27.
//  Copyright © 2020 evan. All rights
    reserved.
//

import UIKit

class ViewController: UIViewController {

                      : touchDigit(_ sender:
                       {

                       viewDidLoad() {
                      /DidLoad()
                       additional setup after
                      ng the view.




}
```

```
Main.storyboard
Main.storyboard — B 9
Main.storyboard — B 2
Main.storyboard — B 0
Main.storyboard — B 6
Main.storyboard — B 3
Main.storyboard — B 4
Main.storyboard — B .
Main.storyboard — B 8
Main.storyboard — B 5
Main.storyboard — B 7
Main.storyboard — B 1
```

將數字與小數點按紐連接到viewcontroller裡的同個Action func

Calculator

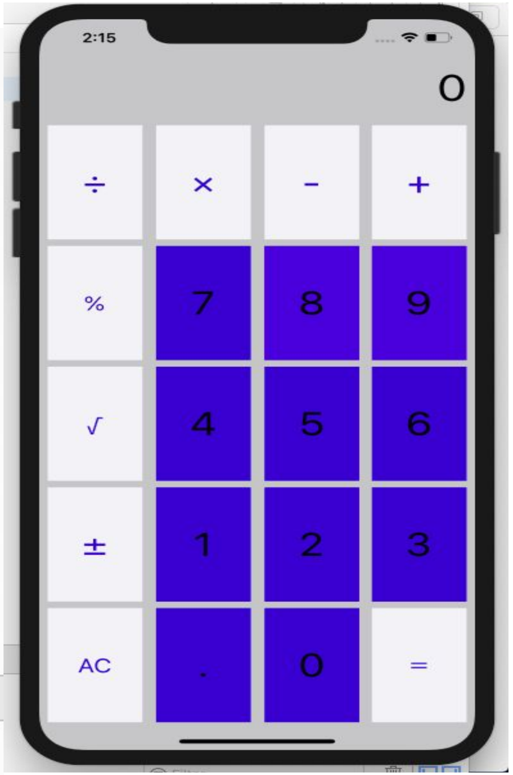


```
import UIKit

class ViewController: UIViewController {

    @IBAction func touchDigit(_ sender: UIButton) {
        let digit = sender.currentTitle!
        print("\(digit) was touched")
    }
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }


}
```

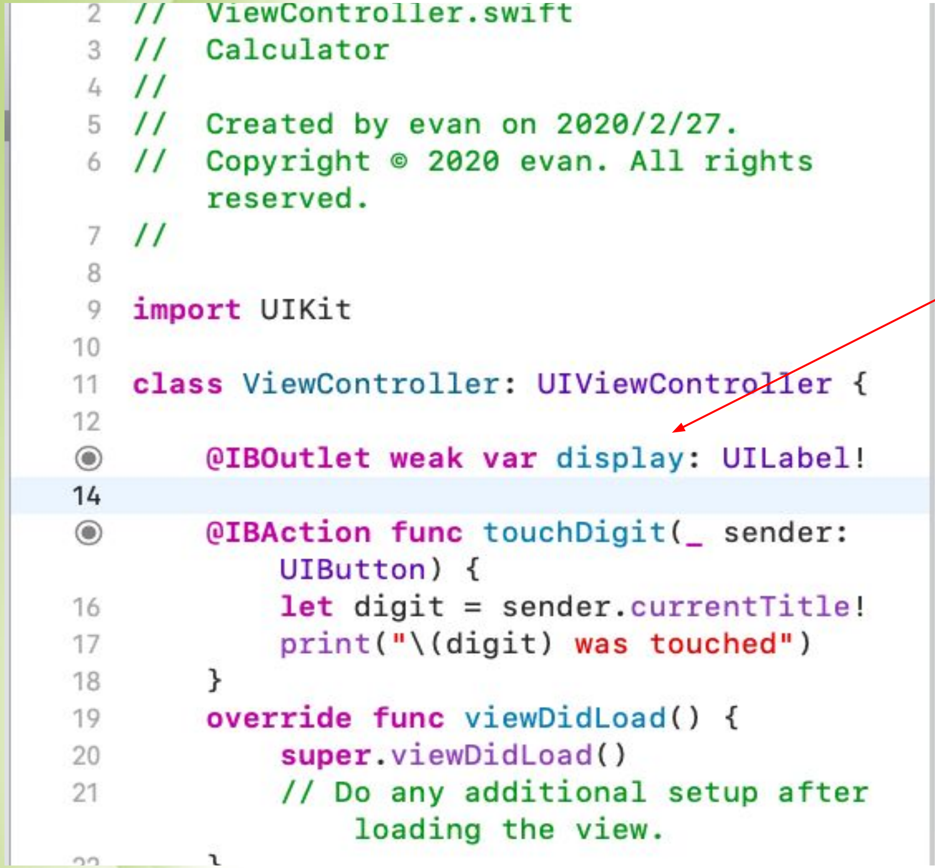
我們可以透過sender取得按鈕的名字

```
5 was touched
8 was touched
8 was touched
2 was touched
```

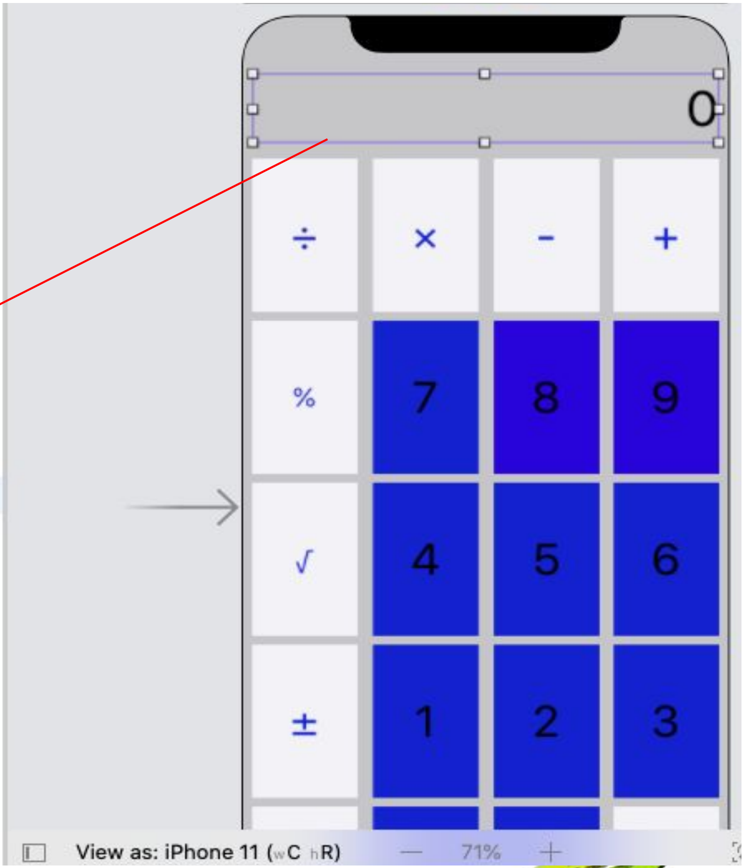
當我們按下按鈕時就會顯示我們按了哪個按鈕



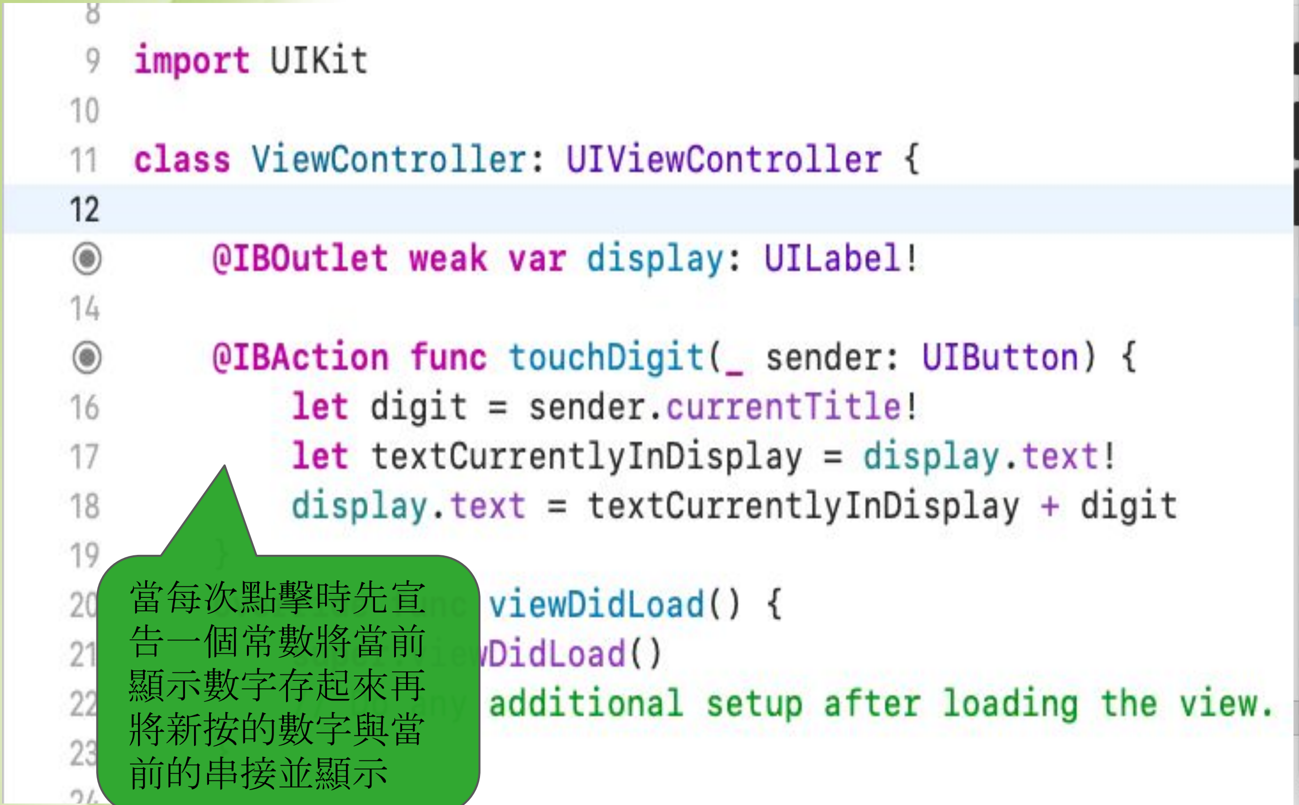
Calculator



```
//  ViewController.swift
//  Calculator
//
//  Created by evan on 2020/2/27.
//  Copyright © 2020 evan. All rights
    reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var display: UILabel!

    @IBAction func touchDigit(_ sender:
        UIButton) {
        let digit = sender.currentTitle!
        print("\(digit) was touched")
    }
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after
            loading the view.
```

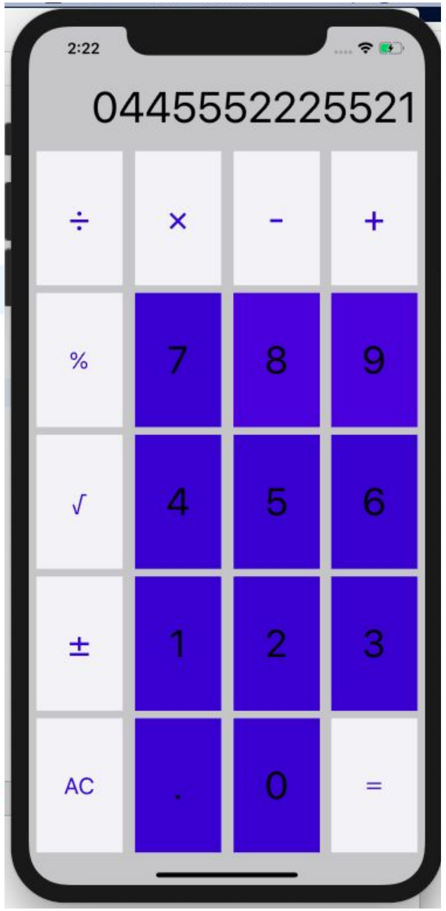


Calculator

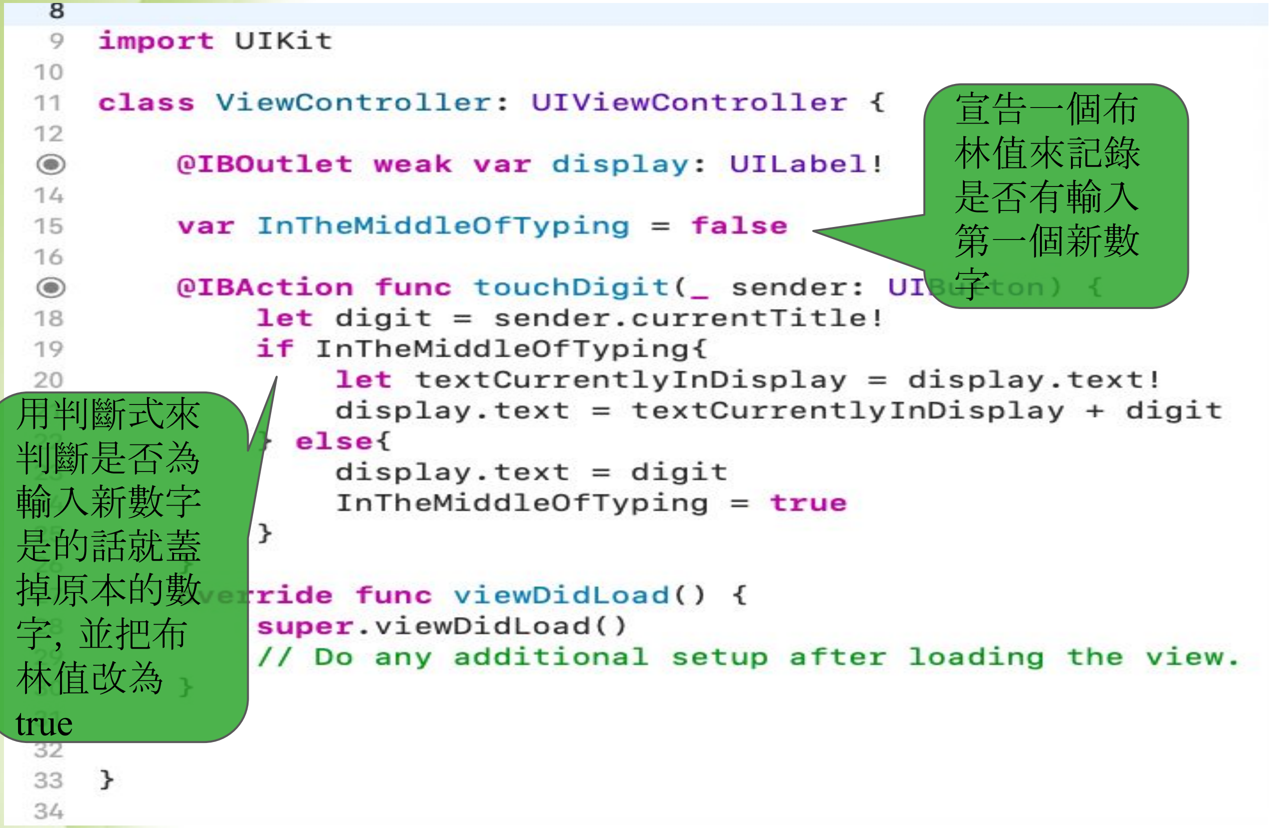


```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var display: UILabel!

    @IBAction func touchDigit(_ sender: UIButton) {
        let digit = sender.currentTitle!
        let textCurrentlyInDisplay = display.text!
        display.text = textCurrentlyInDisplay + digit
    }

        nc viewDidLoad() {
            ie wDidLoad()
            ny additional setup after loading the view.
```

當每次點擊時先宣告一個常數將當前顯示數字存起來再將新按的數字與當前的串接並顯示



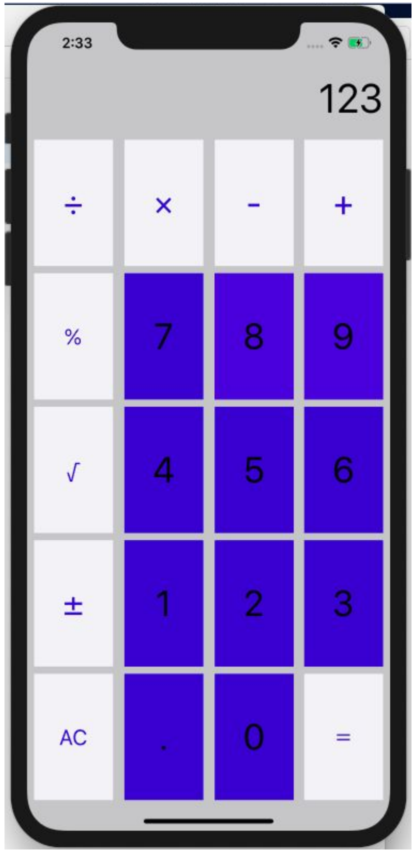
Calculator



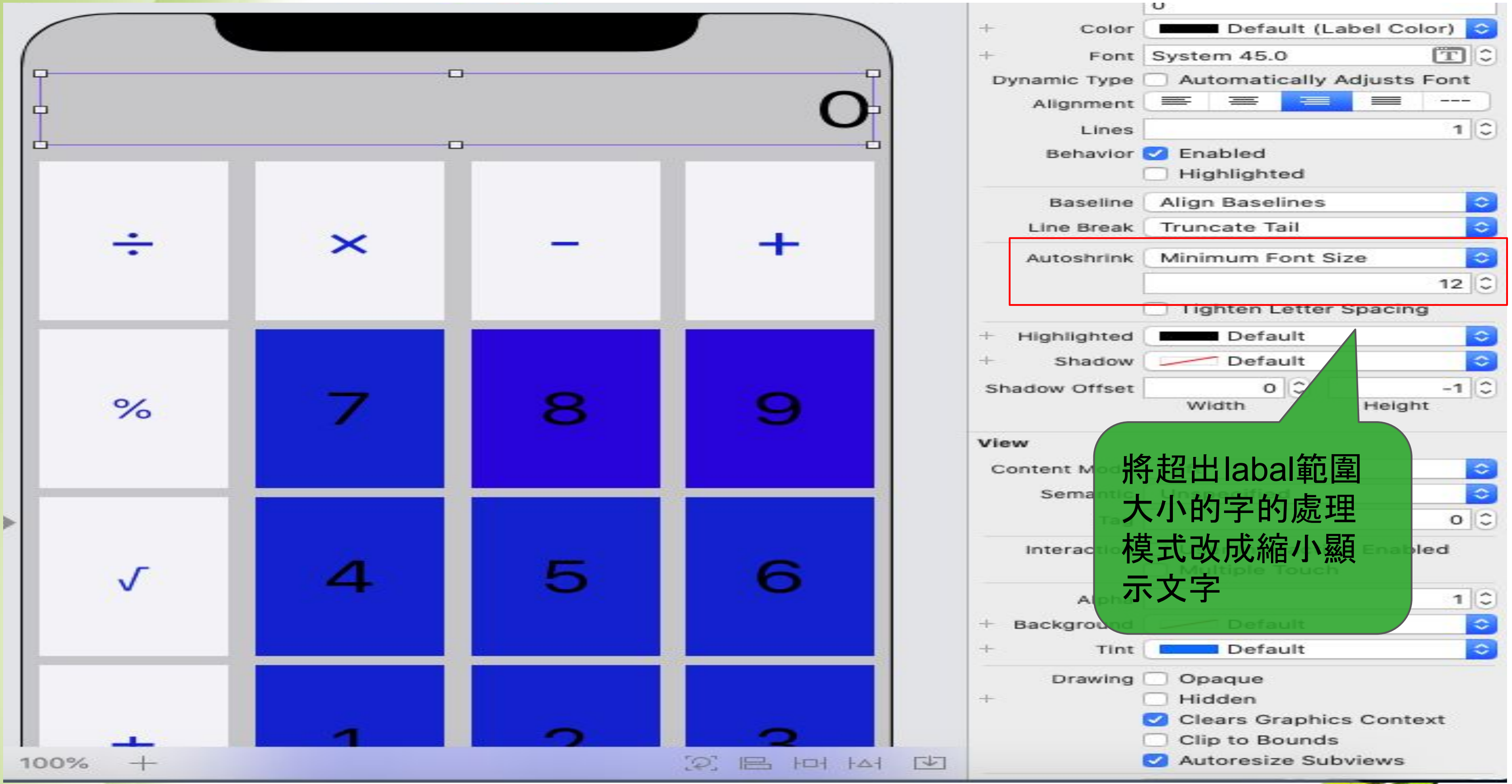
```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var display: UILabel!

    var InTheMiddleOfTyping = false

    @IBAction func touchDigit(_ sender: UIButton) {
        let digit = sender.currentTitle!
        if InTheMiddleOfTyping{
            let textCurrentlyInDisplay = display.text!
            display.text = textCurrentlyInDisplay + digit
        } else{
            display.text = digit
            InTheMiddleOfTyping = true
        }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

}
```

宣告一個布林值來記錄是否有輸入第一個新數字

用判斷式來判斷是否為輸入新數字是的話就蓋掉原本的數字，並把布林值改為true



Calculator



Color: Default (Label Color)
Font: System 45.0
Dynamic Type: Automatically Adjusts Font
Alignment
Lines: 1
Behavior: ✓ Enabled
Highlighted
Baseline: Align Baselines
Line Break: Truncate Tail
Autoshrink: Minimum Font Size
12
Tighten Letter Spacing
Highlighted: Default
Shadow: Default
Shadow Offset: 0 Width, -1 Height

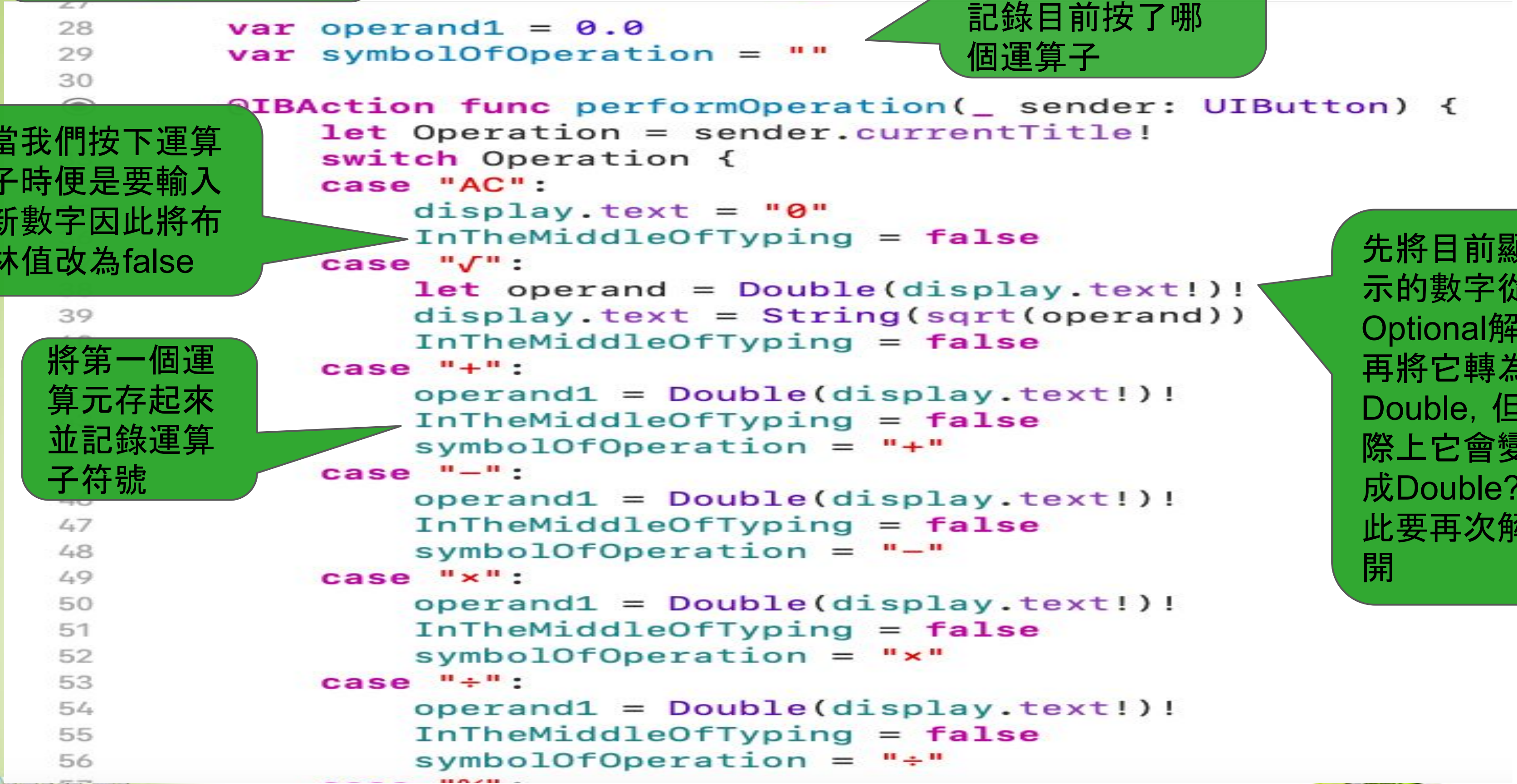
View
Content Mode
Semantic
Tag: 0
Interaction: Enabled, Multiple Touch
Alpha: 1
Background: Default
Tint: Default
Drawing: Opaque
Hidden
✓ Clears Graphics Context
Clip to Bounds
✓ Autoresize Subviews

將超出labal範圍大小的字的處理模式改成縮小顯示文字

Calculator

再來將+ - * / 等剩餘沒有連接的按紐連結到viewcontroller

先宣告兩個變數一個來記錄第一個運算元，一個來記錄目前按了哪個運算子



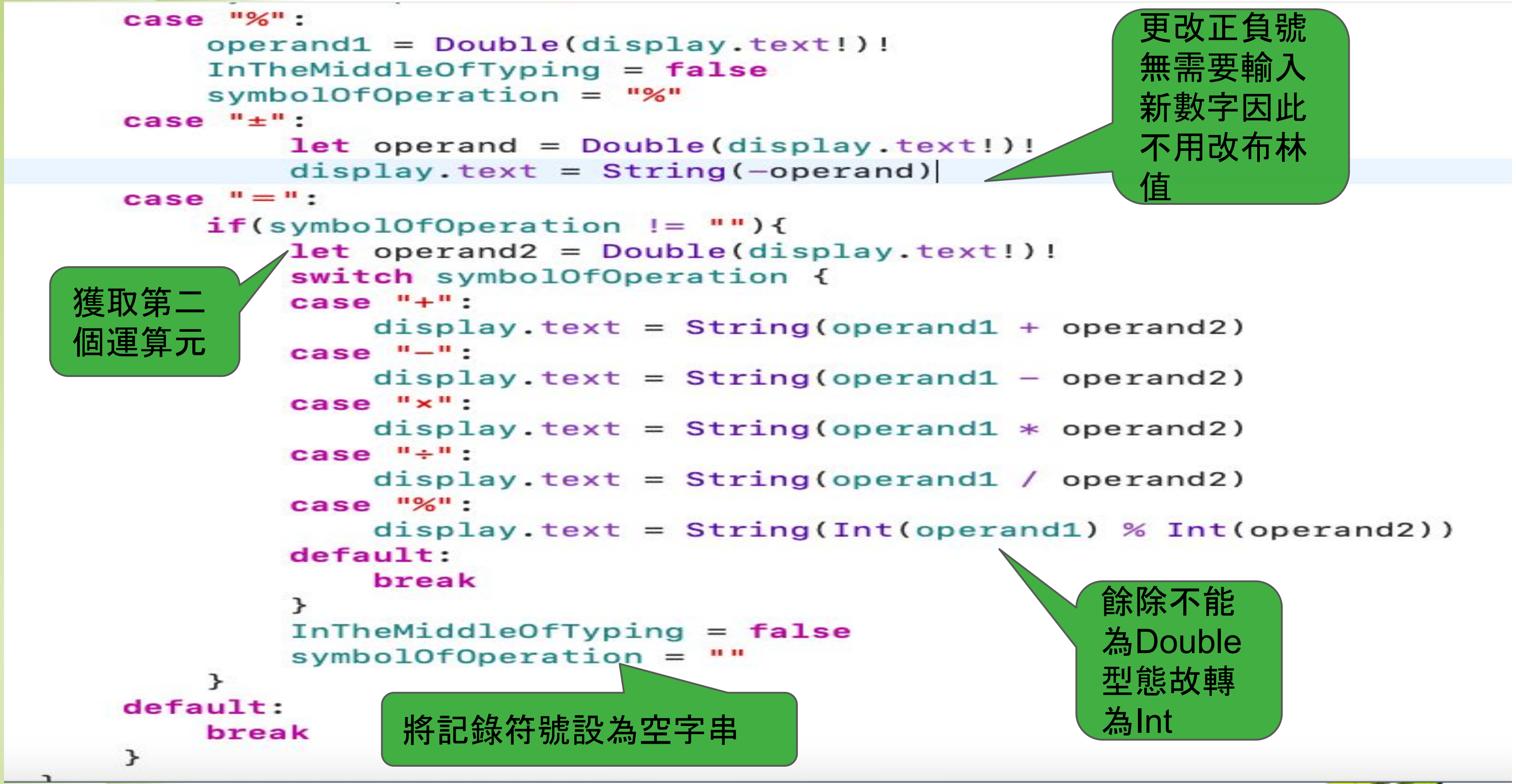
```swift
var operand1 = 0.0
var symbolOfOperation = ""

@IBAction func performOperation(_ sender: UIButton) {
    let Operation = sender.currentTitle!
    switch Operation {
    case "AC":
        display.text = "0"
        InTheMiddleOfTyping = false
    case "√":
        let operand = Double(display.text!)!
        display.text = String(sqrt(operand))
        InTheMiddleOfTyping = false
    case "+":
        operand1 = Double(display.text!)!
        InTheMiddleOfTyping = false
        symbolOfOperation = "+"
    case "—":
        operand1 = Double(display.text!)!
        InTheMiddleOfTyping = false
        symbolOfOperation = "—"
    case "×":
        operand1 = Double(display.text!)!
        InTheMiddleOfTyping = false
        symbolOfOperation = "×"
    case "÷":
        operand1 = Double(display.text!)!
        InTheMiddleOfTyping = false
        symbolOfOperation = "÷"
```

當我們按下運算子時便是要輸入新數字因此將布林值改為false

將第一個運算元存起來並記錄運算子符號

先將目前顯示的數字從Optional解開再將它轉為Double，但實際上它會變成Double?因此要再次解開

Calculator



```
    case "%":
        operand1 = Double(display.text!)!
        InTheMiddleOfTyping = false
        symbolOfOperation = "%"
    case "±":
            let operand = Double(display.text!)!
            display.text = String(-operand)
    case "=":
        if(symbolOfOperation != ""){
            let operand2 = Double(display.text!)!
            switch symbolOfOperation {
            case "+":
                display.text = String(operand1 + operand2)
            case "−":
                display.text = String(operand1 − operand2)
            case "×":
                display.text = String(operand1 * operand2)
            case "÷":
                display.text = String(operand1 / operand2)
            case "%":
                display.text = String(Int(operand1) % Int(operand2))
            default:
                break
            }
            InTheMiddleOfTyping = false
            symbolOfOperation = ""
        }
    default:
        break
    }
```

更改正負號無需要輸入新數字因此不用改布林值

獲取第二個運算元

餘除不能為Double型態故轉為Int

將記錄符號設為空字串