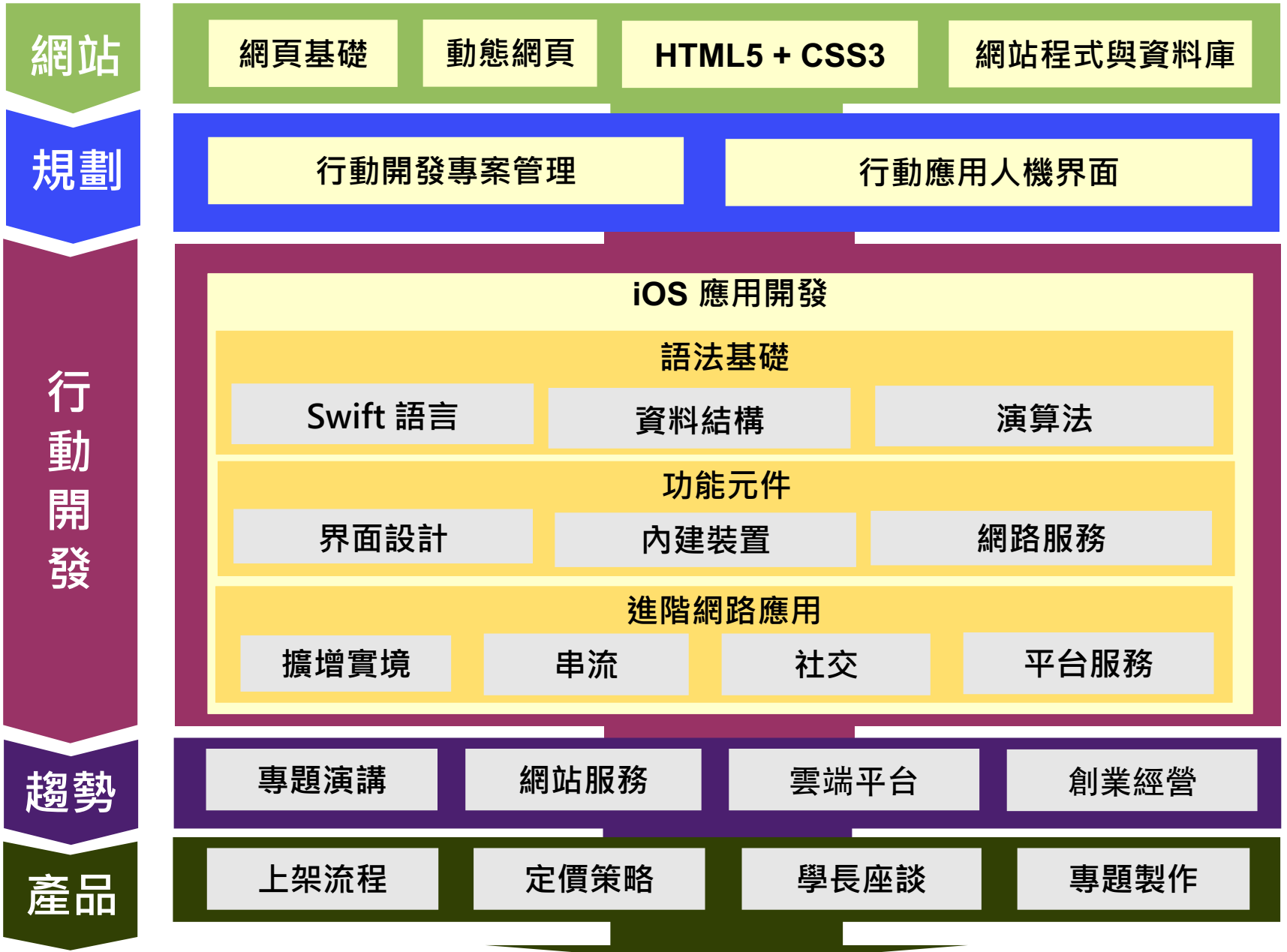
iOS APP Developer
Swift 語法入門

Ryan Chung

iOS APP 開發學習地圖



Swift

- 支援playground，可快速測試
- 效能最佳化
- 可以從Hello World到甚至作業系統都可開發

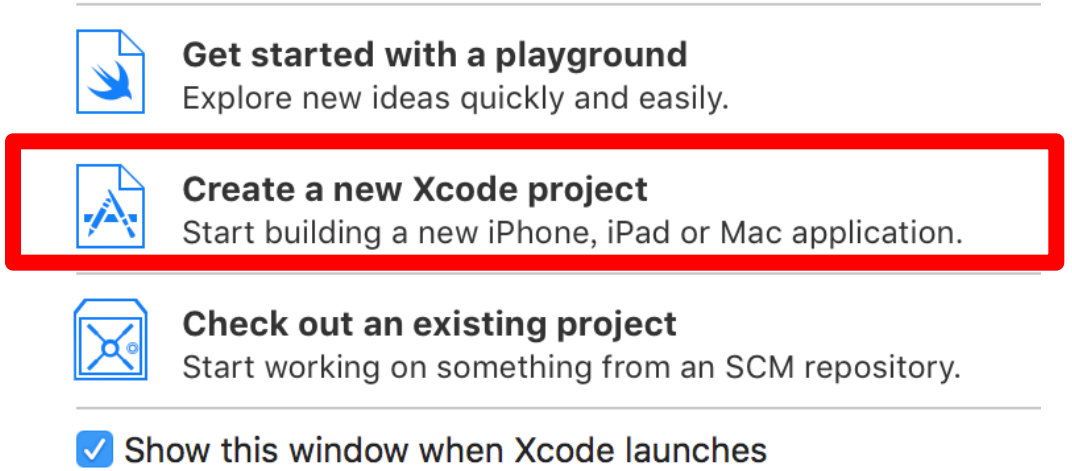
大綱

1. Hello Swift! Xcode環境認識
2. 變數與常數
3. 陣列與字典
4. 控制流程
5. 函數
6. 物件與類別

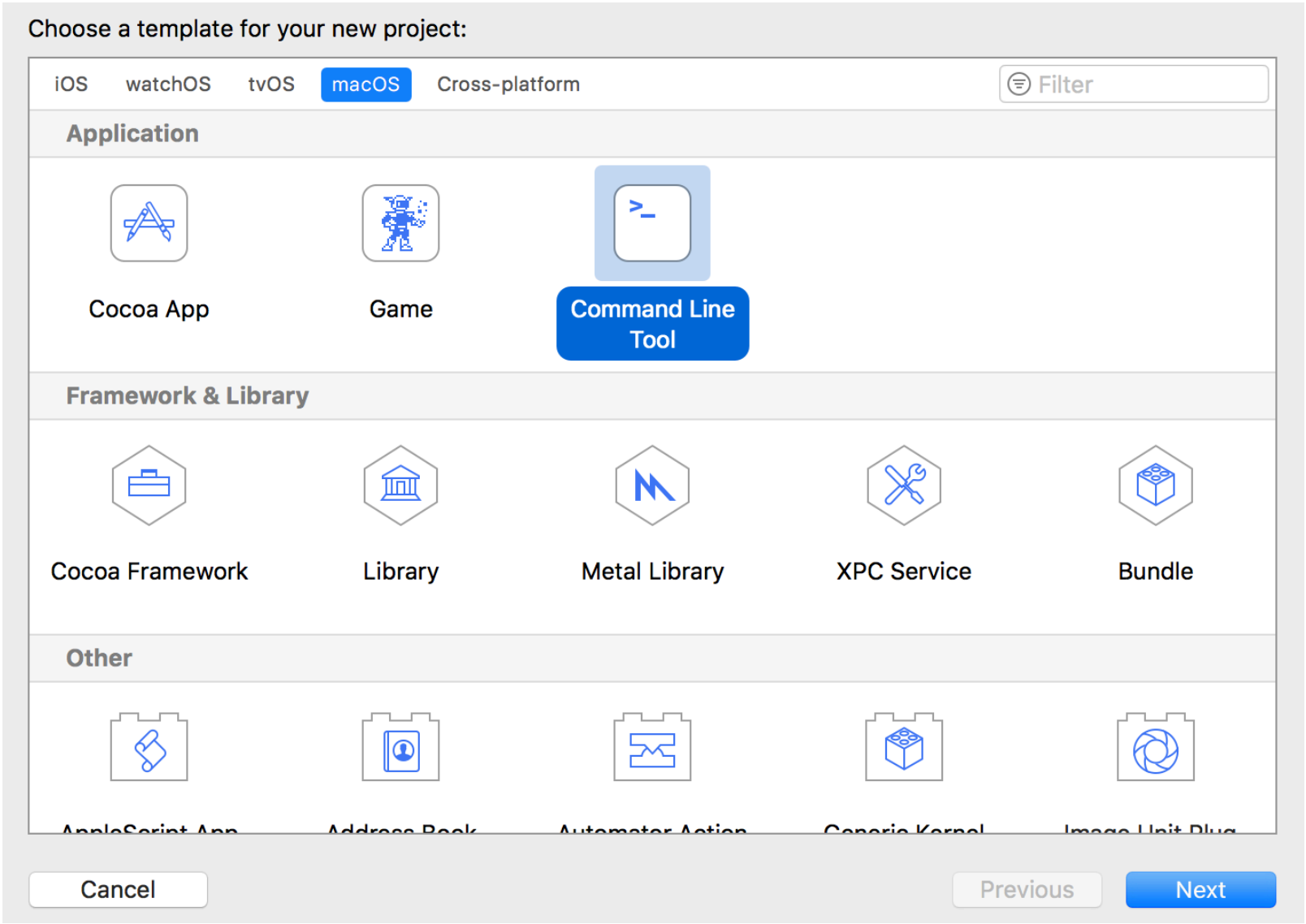
1.Hello Swift! Xcode環境認識

1.選擇Create a new Xcode project

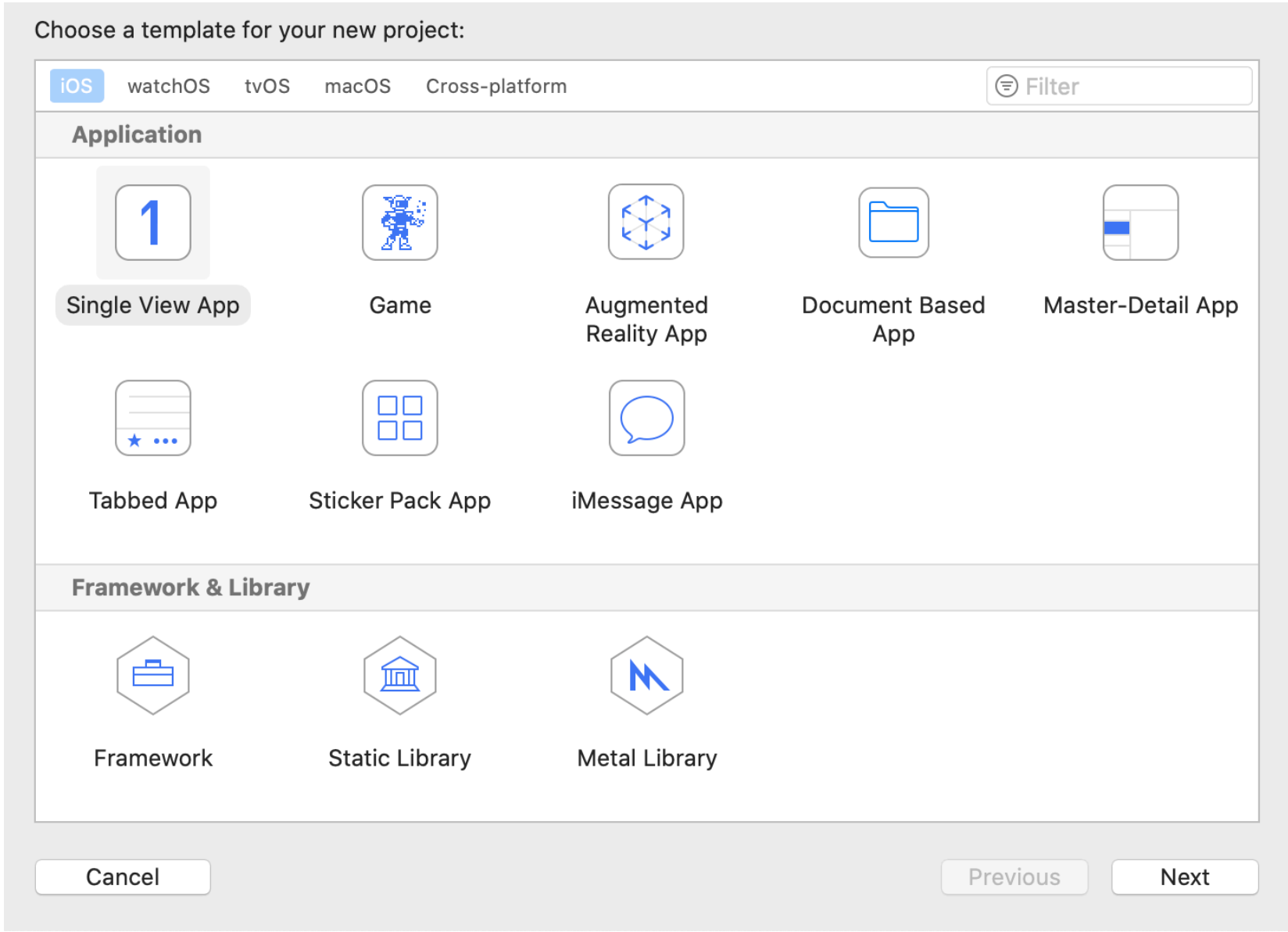
- File -> New Project



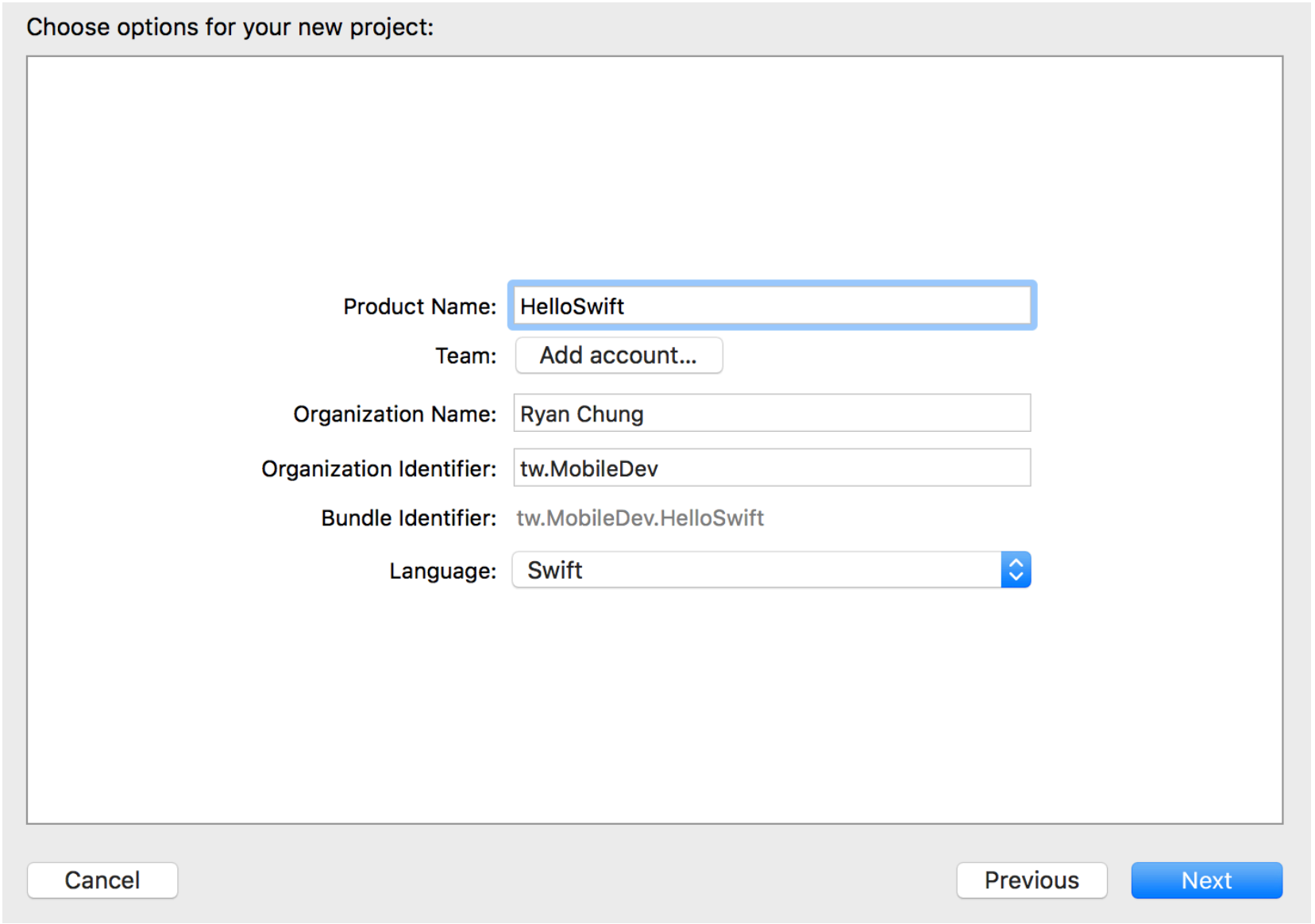
2.選擇macOS->Command Line Tool



Xcode的樣板專案分類介紹

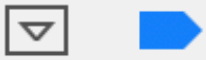


3.輸入專案名稱，語言選Swift，存檔



4.執行應用程式

```
 9 import Foundation
10
11 print("Hello, World!")
12
13
```



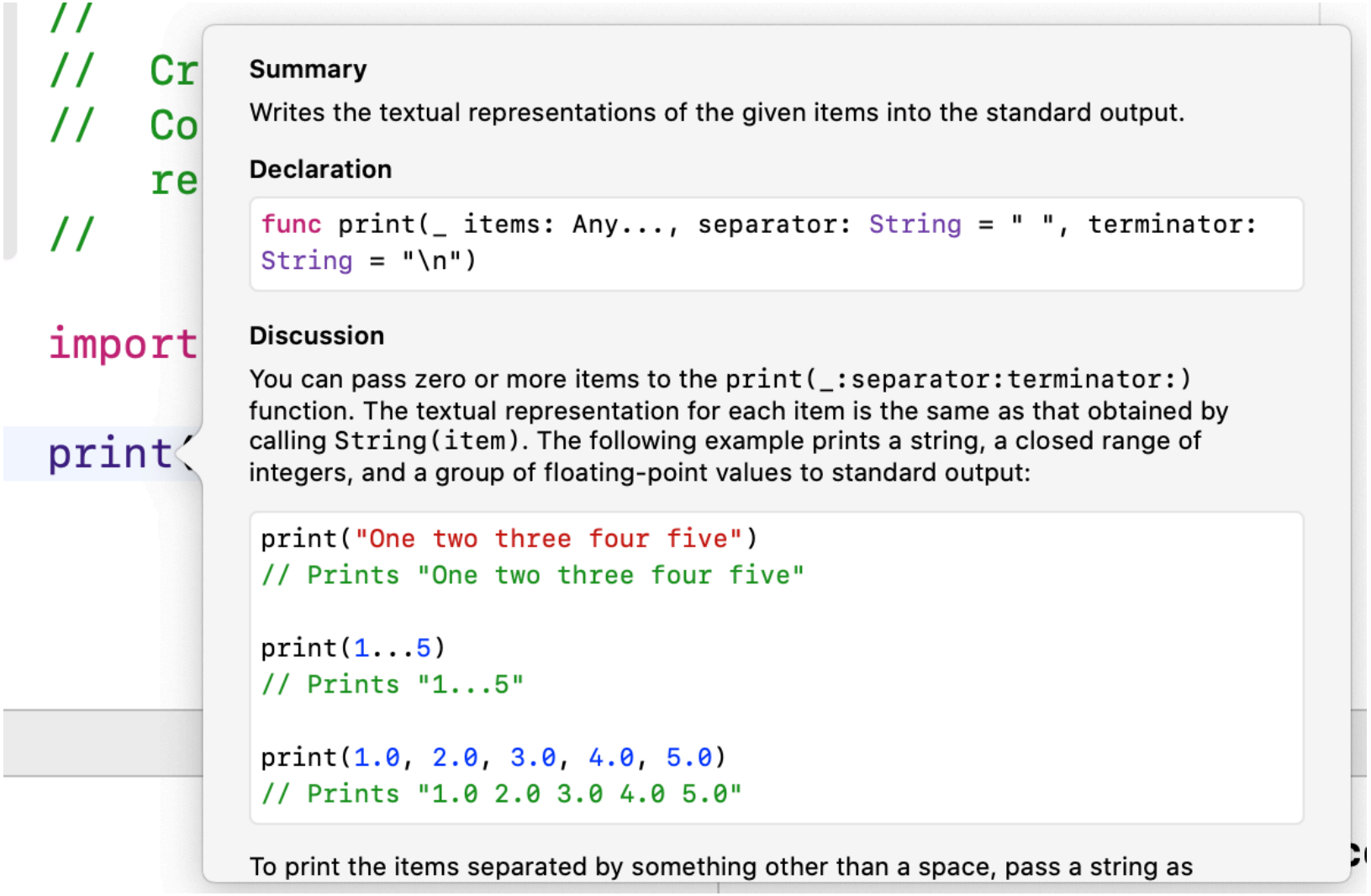
```
Hello, World!
Program ended with exit code: 0
```

重點理解

1. main.swift是程式的進入點
2. import匯入基本程式所需的Framework
3. 利用print來輸出資訊於console

What is print?

- 游標放在print上，按下alt，看到問號後按下滑鼠左鍵



任務：輸出相加結果

```
The sum of 50 and 25 is 75
Program ended with exit code: 0
```

你會不會變...

變數/常數宣告

變數與常數

- let：宣告常數
  - 很多地方要用且不會改變，只指定值一次
- var：宣告變數

```
11  var myVar=42
12  myVar=50 //可變
13  let myCons=100
14  myCons=20 //錯誤！不可變！
15
```

Cannot assign to value: 'myCons' is a 'let' constant

Fix-it Change 'let' to 'var' to make it mutable

指定變數/常數的資料型態

- 常數/變數名稱：資料型態
- 可先宣告再給值 或 直接給值

```
9  import Foundation
10
11 //宣告為常數、指定資料型態、並且給值：OK
12 let myDoubleCons:Double=70.0
13 let myIntCons:Int=5
14
15 //宣告為常數、指定資料型態：OK
16 let Cons2:Int
17
18 //直接給值、依據值決定了資料型態：OK
19 let thisCons=5
20
21 //只宣告、但不給值又不給資料型態：Wrong
22 let Cons3                    Type annotation missing in pattern
```

Convert value to different type

```
9  import Foundation
10
11 var sum:Int
12 sum=50+25
13
14 print("The sum of 50 and 25 is "+String(sum))
15
```

```
The sum of 50 and 25 is 75
Program ended with exit code: 0
```

Type conversion in string (string interpolation)

```
9  import Foundation
10
11 var sum:Int
12 sum=50+25
13
14 print("The sum of 50 and 25 is \(sum)")
15
```

```
The sum of 50 and 25 is 75
Program ended with exit code: 0
```

常數/變數也可先做運算

```
9  import Foundation
10
11 var numberA=50
12 var numberB=25
13
14 print("The sum of 50 and 25 is \(numberA+numberB)")
```

```
The sum of 50 and 25 is 75
Program ended with exit code: 0
```

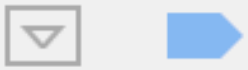
多種型態一起輸出

```
9  import Foundation
10
11 let myName="Ryan"
12 var age=37
13 var weight=78.5
14
15 print("Hi! My name is \(myName), I am \(age) years old, \(weight) kg.")
```

```
Hi! My name is Ryan, I am 37 years old, 78.5 kg.
Program ended with exit code: 0
```

字串相加

```
39 let width=100
40 var label="The width is "
41 label+=String(width)
42 print(label)
```



```
The width is 100
Program ended with exit code: 0
```

註解

- 單行：//
- 多行：/* …… */

Array & Dictionary

陣列與字典

字典 Dictionary

```
19 var numerDic=[
20     "Ryan":50,
21     "John":25
22 ]
```

改變字典某項目的值

```
19 var numerDic=[
20     "Ryan":50,
21     "John":25
22 ]
23
24 numerDic["Ryan"]=100
```

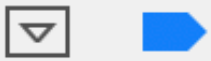
建立一個空的字典

```
9  import Foundation
10
11 var emptyDictionary=[String:Float]()
12
13 emptyDictionary["Mon"]=28.5
14 emptyDictionary["Tue"]=32.6
15 emptyDictionary["Wed"]=24.3
16
17 print(emptyDictionary["Wed"]!)
```

```
24.3
Program ended with exit code: 0
```

字典取值

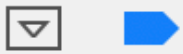
```
19 var numerDic=[
20     "Ryan":50,
21     "John":25
22 ]
23
24 numerDic["Ryan"]=100
25 print(numerDic["Ryan"])
26
```



```
Optional(100)
Program ended with exit code: 0
```

陣列取值

```
44 var shoppinglist=["milk","clothes","glasses"]
45 print(shoppinglist[1])
```



```
clothes
Program ended with exit code: 0
```

建立一個空的陣列

```
9  import Foundation
10
11 var myArray=[String]()
12 myArray.append("John")
13 myArray.append("Mary")
14 myArray.append("Ryan")
15
16 myArray.remove(at: 1)
17 print(myArray)
```

```
["John", "Ryan"]
Program ended with exit code: 0
```

大量調整陣列

```swift
var numberArray=[1,3,5,7,9,2,4,6,8,10]

let numberArrayAddTen=numberArray.map
    { (number:Int) -> Int in
        return number+10
    }

print(numberArrayAddTen)
```

```
[11, 13, 15, 17, 19, 12, 14, 16, 18, 20]
```

```swift
let numberArrayWithString=numberArray.map { (number:Int) -> String in
    return "This is number \(number)"
}

print(numberArrayWithString)
```

```
["This is number 1", "This is number 3", "This is number 5", "This is number 7", "This is number 9", "This is number 2", "This is number 4", "This is number 6", "This is number 8", "This is number 10"]
Program ended with exit code: 0
```

陣列排序

- 順著排

```
51  var numberArray = [1,3,5,7,9,2,4,6,8,10]
52  var sortedArray = numberArray.sorted()
53  print(sortedArray)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Program ended with exit code: 0
```

- 倒著排

```
55  var numberArray = [1,3,5,7,9,2,4,6,8,10]
56  var inversedSortedArray = numberArray.sorted(by: { s1, s2 in return s1 > s2 })
57  print(inversedSortedArray)
```

```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
Program ended with exit code: 0
```

You should got it NOW

- Xcode可以製作哪些種類的應用程式
- 如何查閱進一步解釋
- 如何跳至匯入程式碼的原始位置
- 如何使用print顯示變數的值
- 變數與常數該如何宣告
- 陣列與字典是什麼？要如何存取？

重複的工作請電腦幫你完成

控制流程

控制流程

- if
- switch
- for-in
- for
- while
- repeat-while

計算優秀與不及格的數量

```
73  let individualScores = [75,40,100,92,28]
74  var failCount = 0
75  var excellentCount = 0
76
77  for score in individualScores{
78      if score>80{
79          excellentCount+=1
80      }else if score<60{
81          failCount+=1
82      }
83  }
84
85  print("Excellent:\(excellentCount),Fail:\(failCount)")
```

```
Excellent:2,Fail:2
Program ended with exit code: 0
```

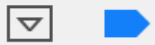
你若是不在了，我會...

Optional Value

Optional Value

- 標示出可能為空值的風險
- 標示方式：？問號

```
62 var optionalString:String?="Hello"
63
64 print(optionalString)
65
66 optionalString=nil
67
68 print(optionalString)
69
```



```
Optional("Hello")
nil
Program ended with exit code: 0
```

如果有值才...的寫法 if let

```
72 var optionalName:String?="Ryan"
73
74 if let name=optionalName{
75     print("Hello,\(name)")
76 }
77
```

```
Hello,Ryan
Program ended with exit code: 0
```

```
72 var optionalName:String?=nil
73
74 if let name=optionalName{
75     print("Hello,\(name)")
76 }
77
```

Why Swift need optional?

1. 有些情況會讓方法本身無法回傳值
   - var x="ABC".toInt()
2. 有些物件建構時，還沒有辦法決定某個屬性值
   - 畫面上的按鈕要在哪裡

```
let imageView=UIImageView()
imageView.image=UIImage(named: "puppy_in_box")

if let image=imageView.image
{
    let size=image.size
    print(size)
}else
{
    print("This image hasn't been set")
}
```

```
let imageView=UIImageView()
//imageView.image=UIImage(named: "puppy_in_box")

if let image=imageView.image
{
    let size=image.size
    print(size)
}else
{
    print("This image hasn't been set")
}
```

安全地處理有值與無值的狀況

- 將 x可能的兩種情況分開處理

```
99   var x:Int?
100
101  let strings = ["ABC", "123"]
102  //產生0或1的亂數
103  let randomIndex = arc4random_uniform(2)
104
105  let string = strings[Int(randomIndex)]
106
107  x = Int(string)
108
109  if let intValue = x{
110      print(intValue*2)
111  }else{
112      print("Can't convert!")
113  }
```

```
246
Program ended with exit code: 0
```

```
Can't convert!
Program ended with exit code: 0
```

if不用括號，但是一定要是判斷式或布林值

- 在Swift中，if後面一定要接判斷式或是布林值

```
116  let studentScores=[30,50,60,100]
117  var passCount = 0
118
119  for score in studentScores{
120      if score{   Cannot convert value of type 'Int' to expected condition type 'Bool'
121
122      }
123  }
```

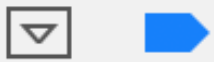
Swift 不用 ++

- 用+=1

```
116  let studentScores=[30,50,60,100]
117  var passCount = 0
118
119  for score in studentScores{
120      if score>=60{
121          passCount++      Use of unresolved operator '++'; did you mean '+= 1'?
122      }
123  }
```

很確定有值的情況，驚嘆號解開！

```
366 var hisName:String?
367 hisName="Ryan"
368 print(hisName!)
```



Ryan

Optional Chaining

- 問號除了用在宣告外，也可以繼續跟著變數走，持續為optional type

```
127  var hisName:String?
128  hisName = "Ryan"
129  if let nameLength = hisName?.count{
130      print("這名字共有\(nameLength)個字母")
131  }else{
132      print("沒有名字!")
133  }
134
135
```

```
這名字共有4個字母
Program ended with exit code: 0
```

```
127  var hisName:String?
128
129  if let nameLength = hisName?.count{
130      print("這名字共有\(nameLength)個字母")
131  }else{
132      print("沒有名字!")
133  }
134
135
```

```
沒有名字!
Program ended with exit code: 0
```

Switch..Case

- 支援資料型態更廣泛
- 可多個狀況合併在一起(逗號分開)
- 可使用判斷式
- 不用寫break，對應到執行完case就離開
- default一定要寫

Switch..Case

- 多種狀況對應用逗號隔開

```
11 //Ctrl+command+空白鍵 -> 叫出emoji輸入
12
13 let 水果=["Tomato","Banana","Orange","🍐"]
14 switch 水果[0]
15 {
16     case "Tomato","Orange":
17         print("不是番茄就是橘子")
18     case let x where x.hasSuffix("na"):
19         print("香蕉啦")
20
21     default:
22         print("就是水果")
23 }
```

```
不是番茄就是橘子
Program ended with exit code: 0
```

Switch..Case

- 可使用判斷式

```
40 let 水果=["Tomato","Banana","Orange","🍐"]
41
42 switch 水果[1]
43 {
44     case "Tomato","Orange":
45         print("不是番茄就是橘子")
46     case let x where x.hasSuffix("na"):
47         print("香蕉拉")
48
49     default:
50         print("就是水果")
51 }
```

```
香蕉拉
Program ended with exit code: 0
```

Switch..Case

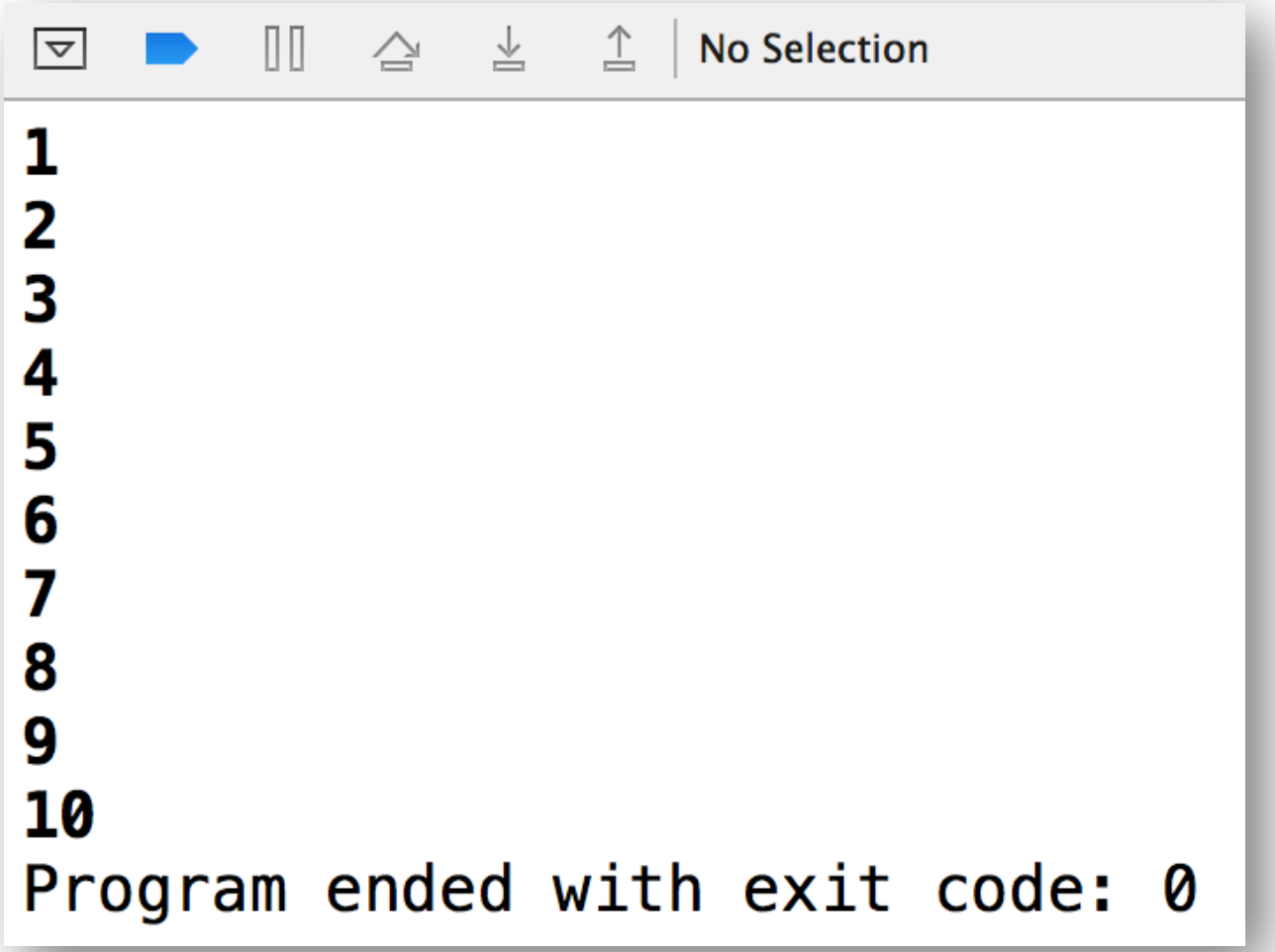
- 如果沒寫default會出什麼錯？

```
81 let fruit=["🍇","🍉","🍌","🍅"]
82 switch fruit[0]{
83     case "🍇":
84         print("葡萄！")
85     //default:
86       //  let x="水果"
87 }
```

Switch must be exhaustive, consider adding a default clause

任務

- 請電腦幫我印出 1 ~ 10



For迴圈

```
for i in 1...10{
    print(i)
}
```

```
1
2
3
4
5
6
7
8
9
10
```

```
11  for(var i=1;i<=10;i+=1)
12  {    C-style for statement has been removed in Swift 3
13      print()
14  }
```

任務

- 抓出不及格的學生

```
11  var studentScores=[
12      "Ryan":45,
13      "John":95,
14      "Marry":50,
15      "Vinson":80
16  ]
```

```
Ryan不及格
Marry不及格
Program ended with exit code: 0
```

for in

```swift
var studentScores = [
    "Ryan":45,
    "John":95,
    "Marry":50,
    "Vinson":80
]

for scorePair in studentScores{
    if scorePair.value<60{
        print("\(scorePair.key)不及格")
    }
}
```

```
Marry不及格
Ryan不及格
Program ended with exit code: 0
```

任務

- 找到女生

```
var studentGender=[
    [
        "姓名":"萊恩",
        "性別":"男"
    ],
    [
        "姓名":"約翰",
        "性別":"男"
    ],
    [
        "姓名":"瑪麗",
        "性別":"女"
    ],
    [
        "姓名":"文森",
        "性別":"男"
    ]
]
```

```
瑪麗是女生
Program ended with exit code: 0
```

for in

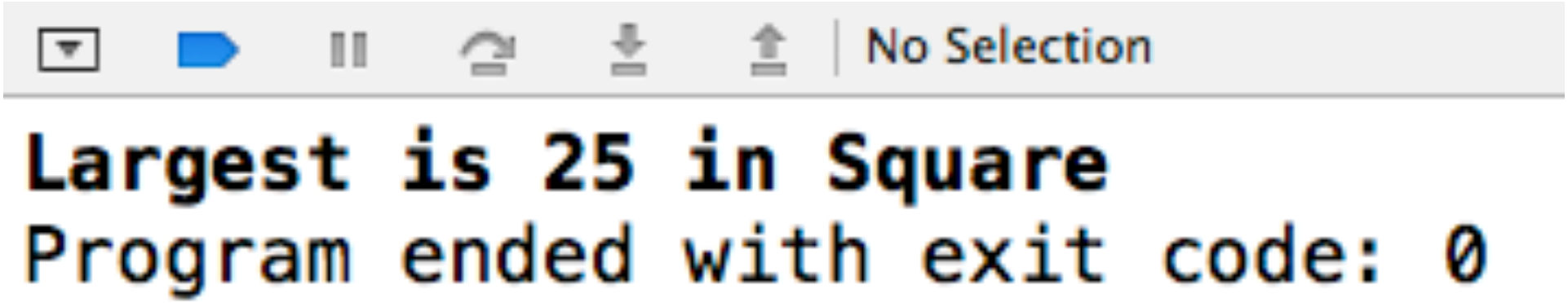
```
30  for person in studentGender
31  {
32      if person["性別"]=="女"
33      {
34          print(person["姓名"]!+"是女生")
35      }
36  }
```

找到最大值

```
let interestingNumbers = [
    "Prime":[2,3,5,7,11,13],
    "Fibonacci":[1,1,2,3,5,8],
    "Square":[1,4,9,16,25]
]

var largest = 0

for series in interestingNumbers{
    for number in series.value{
        if number > largest{
            largest = number
        }
    }
}

print(largest)
```

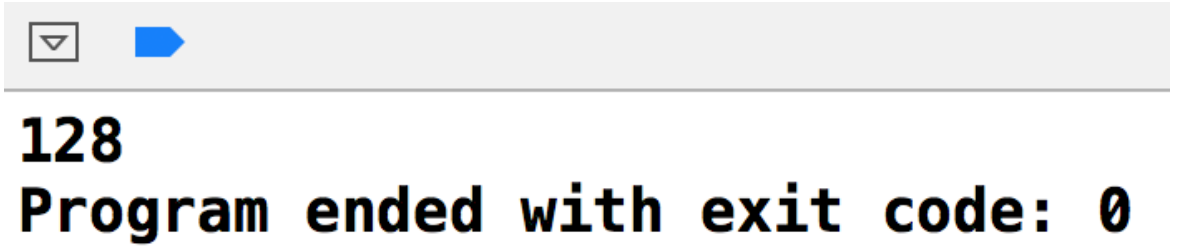
```
25
Program ended with exit code: 0
```

Lab：顯示最大值的Key值



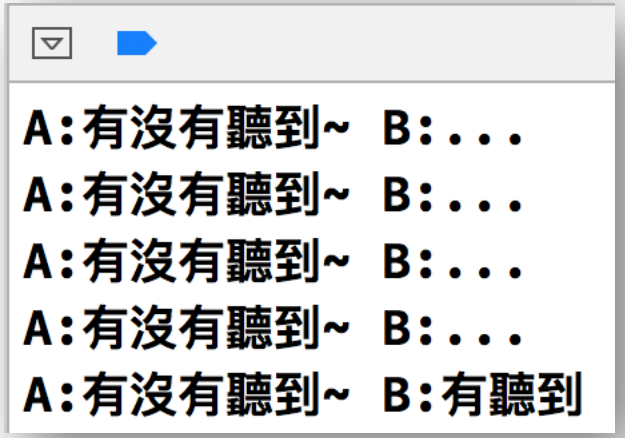
while迴圈

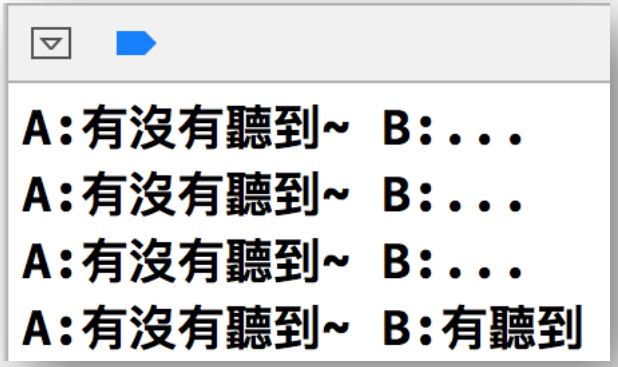
```
var n=2
while n<100{
    n=n*2
}
print(n)
```

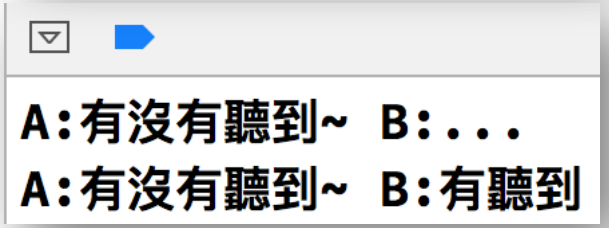


任務：repeat...while

- 當我們跟別人說話時，有些時候別人沒有聽到，所以我們會重複說一次，直到對方聽到為止
- 請用程式模擬，亂數決定對方有沒有聽到，並印出對對方說的話







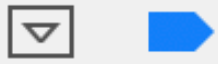
這個箱子有進有出，為了特定功能而存在著

Function 函數

最基本的函數：0進0出

- 關鍵字：func

```
194 func greetOnlyHi()
195 {
196     print("你好")
197 }
198
199 greetOnlyHi()
```



```
你好
Program ended with exit code: 0
```

有產出的函數：0進1出

- ->傳回值資料型態

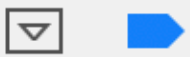
```
201 func greetOnly()->String
202 {
203     return "你好"
204 }
205
206 print(greetOnly())
207
208
```

```
你好
Program ended with exit code: 0
```

有輸入有產出的函數：1進1出

- 傳入變數名稱:變數資料型態

```
11 func greet(name:String)->String
12 {
13     return "你好 \(name)"
14 }
15
16 print(greet(name: "Ryan"))
17
```



```
你好 Ryan
Program ended with exit code: 0
```

多輸入有產出的函數：2進1出

- 多個輸入變數用逗號隔開

```
11 func greetWithCompany(name:String,company:String)->String
12 {
13     return "\(company)的\(name)你好"
14 }
15
16 print(greetWithCompany(name: "Ryan", company: "行動開發學院"))
17
```

```
行動開發學院的Ryan你好
Program ended with exit code: 0
```

多產出的函數：1進3出

- 多個輸出變數用逗號隔開，取值透過點運算子

```
239 func randomUnderX(x:UInt32)->(var1:UInt32,var2:UInt32,var3:UInt32){
240     return (arc4random_uniform(x),
241             arc4random_uniform(x),
242             arc4random_uniform(x))
243 }
244 print(randomUnderX(x: 10))
```

```
(var1: 4, var2: 0, var3: 6)
Program ended with exit code: 0
```

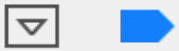
不定量的多變數輸入

```
247 func sumOf(numbers:Int...)->Int{
248     var sum=0
249     for number in numbers{
250         sum+=number
251     }
252     return sum
253 }
254 print(sumOf(numbers: 1,2,3))
255 print(sumOf(numbers: 1,2,3,4,5))
```

```
6
15
Program ended with exit code: 0
```

Lab：改寫上個範例，計算平均

```
23 print(avgOf(numbers: 1,2,3,4))
24 print(avgOf(numbers: 1,2,3,4,5,6,7,8,9,10))
25
```



```
2.5
5.5
Program ended with exit code: 0
```

Object & Class

物件與類別

從一個簡單的分數開始

指定固定的分子與分母

目前我們透過直接給值來顯示分數

顯示一個分數

```
10 import Foundation
11
12 var numerator=1
13 var denominator=3
14
15 print("The fraction is \(numerator)/\(denominator)")
```

```
The fraction is 1/3
Program ended with exit code: 0
```

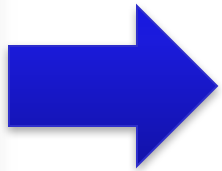
類別的初體驗

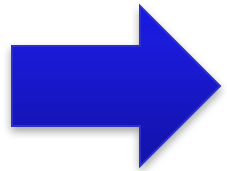
現在我們希望蓋一間工廠，給它分子與分母，它就產生分數給我們

建立類別

- 建立一個運算分數的類別
  - 裡面有分子、分母
  - 有輸入分子的方法
  - 有輸入分母的方法
  - 有輸出結果的方法

類別就像是生產工廠的運作機制，建立了類別，就可以根據這個類別，創造出一個個的物件來使用。





建立類別

```
19 class Fraction {
20
21     var numerator:Int
22     var denominator:Int
23
24     init(n:Int,d:Int)
25     {
26         numerator=n
27         denominator=d
28     }
29
30     func printFraction()
31     {
32         print("\(numerator)/\(denominator)")
33     }
34 }
35
36 var myFraction=Fraction(n: 1, d: 3)
37 myFraction.printFraction()
```

```
1/3
Program ended with exit code: 0
```

Swift 術語小字典：iVar

- 實體變數(instance variable)
  - 簡稱 iVar
  - 相當於一個類別中的屬性定義

就像是一個工廠的儲存空間。

在設計類別時，需要思考有哪些東西需要一個儲存空間，這些東西會被各個運作部門所使用。

你給我豬肉，我給你香腸

蓋一間工廠，需要說明有哪些成員、裡面有哪些機器在運作，並且實際把這些機器的運作流程都描述清楚

使用類別來生成實體

```
36 var myFraction=Fraction(n: 1, d: 3)
37 myFraction.printFraction()
38
```

```
1/3
Program ended with exit code: 0
```

讓我們一起再蓋一間更完善的工廠吧

來寫一個計算機！

來寫一個計算機！

- 建立一個計算機類別，裡面放一個iVar來儲存結果
- 初始化結果
- 計算用的方法
  - 加法
  - 減法
  - 乘法
  - 除法

建立類別

```
11 class Calculator {
12
13     var accumulator:Double
14
15     init(accu:Double) {
16         accumulator=accu
17     }
18
19     func add(value:Double){
20         accumulator+=value
21     }
22
23     func substract(value:Double){
24         accumulator-=value
25     }
27     func multiply(value:Double){
28         accumulator*=value
29     }
30
31     func divide(value:Double){
32         accumulator/=value
33     }
34
35 }
```

使用類別

```
37 var myCalculator=Calculator(accu: 0)
38 myCalculator.add(value: 5)
39 print(myCalculator.accumulator)
```

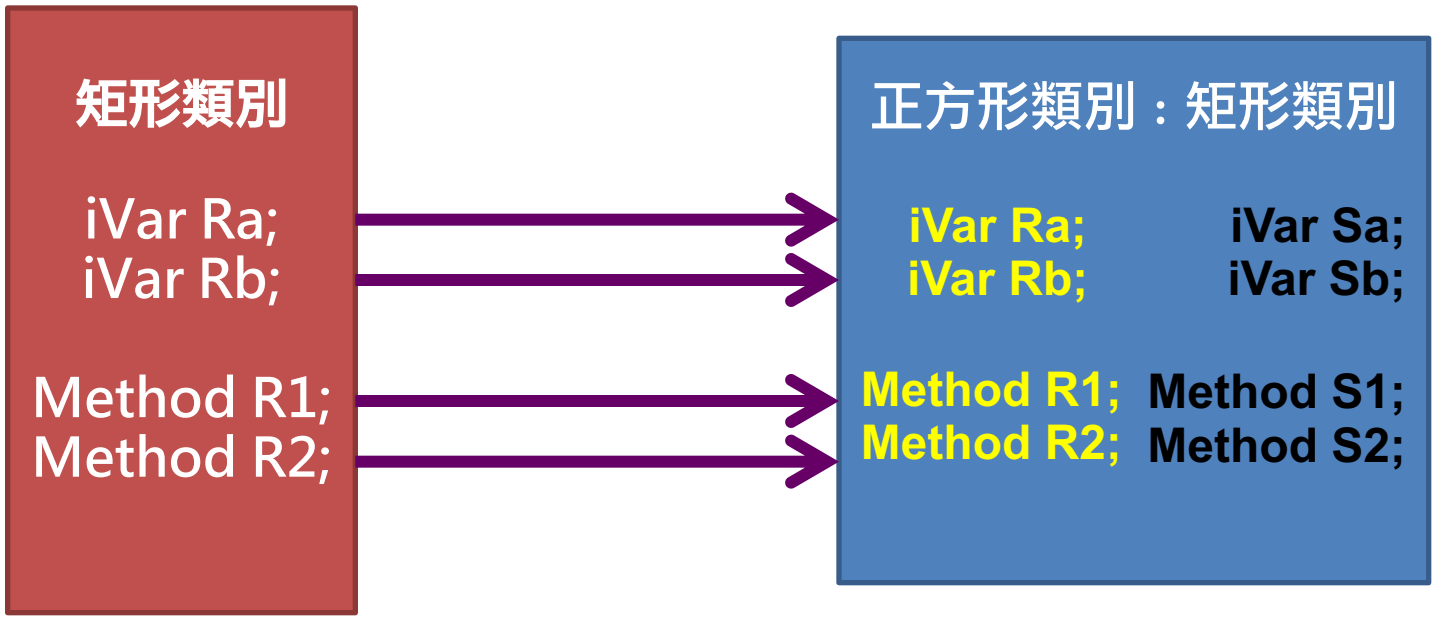
```
5.0
Program ended with exit code: 0
```

從0開始太累了，借用前人種的樹吧！

繼承的概念

L.運用繼承再客製化類別

- 設計一個正方形類別
  1. 若原先已經有矩形類別，那就繼承它，創一個正方形類別
  2. 再增加正方形需要用的屬性與方法
  3. 正方形這個類別，可以使用矩形類別中的屬性與方法



繼承的運作

- 父類別的所有實體變數與方法，都會成為子類別定義的一部份。
- 子類別可以直接存取這些方法與實體變數。
- 首先會先確認該變數或方法是否存在目前物件的類別定義中，若沒有，則會往父類別去尋找。

建立一個一般形狀的類別

```
258 class Shape {
259     var numberOfSides=0
260     func simpleDescription()->String
261     {
262         return "A shape with \(numberOfSides) sides."
263     }
264 }
265
266 var myShape=Shape()
267 myShape.numberOfSides=7
268 print(myShape.simpleDescription())
```

```
A shape with 7 sides.
Program ended with exit code: 0
```

Lab

1. 增加一個iVar為let資料型態，裡面儲存圖案名稱
2. 增加一個有輸入變數的方法，可設定為幾邊形

Lab

```
11 class Shape{
12     var numberOfSides=0
13     let shapeName="FixedShape Name"
14     func simpleDescription()->String{
15         return "A shape with \(numberOfSides) sides."
16     }
17     func setNumberOfSides(count:Int){
18         numberOfSides=count
19     }
20 }
21
22 var myShape=Shape()
23 myShape.setNumberOfSides(count: 4)
24 print(myShape.simpleDescription())
```

```
A shape with 4 sides.
Program ended with exit code: 0
```

自訂初始化方法

```
11 class Shape{
12     var numberOfSides:Int
13     var shapeName:String
14
15     init(name:String,count:Int) {
16         shapeName=name
17         numberOfSides=count
18     }
19
20     func simpleDescription()->String{
21         return
22             "This shape \(shapeName) with \(numberOfSides) sides."
23     }
24 }
25
26 var myShape=Shape(name: "Tri", count: 3)
27 print(myShape.simpleDescription())
```

```
This shape Tri with 3 sides.
Program ended with exit code: 0
```

繼承：建立正方形類別

```
29 class Square: Shape {
30
31     var sideLength:Double
32
33     init(length:Double,squareName:String) {
34
35         sideLength=length
36         super.init(name: squareName, count: 4)
37     }
38
39     func area()->Double
40     {
41         return sideLength*sideLength
42     }
43
44     override func simpleDescription() -> String {
45         return
46             "This square \(shapeName) with side of length \(sideLength),
                the area is \(self.area())"
47     }
48 }
49
50 let mySquare=Square(length: 5.2, squareName: "mySquare")
51 print(mySquare.simpleDescription())
```

```
This square mySquare with side of length 5.2, the area is 27.04
Program ended with exit code: 0
```

Lab：建立一個圓形的類別

1. 製作一個新的類別叫Circle
2. 初始化中輸入半徑、名稱
3. 實作area、simpleDescription方法

Lab

```
53 class Circle: Shape {
54
55     var radius:Double
56
57     init(r:Double,circleName:String) {
58
59         radius=r
60         super.init(name: circleName, count: 0)
61     }
62
63     func area()->Double{
64         return M_PI*radius*radius
65     }
66
67     override func simpleDescription() -> String {
68         return "This circle \(shapeName) with radius \(radius)
               which area is \(String(format: "%.2f",self.area()))"
69     }
70 }
71
72 let myCircle=Circle(r: 5.5, circleName: "myCircle")
73 print(myCircle.simpleDescription())
```

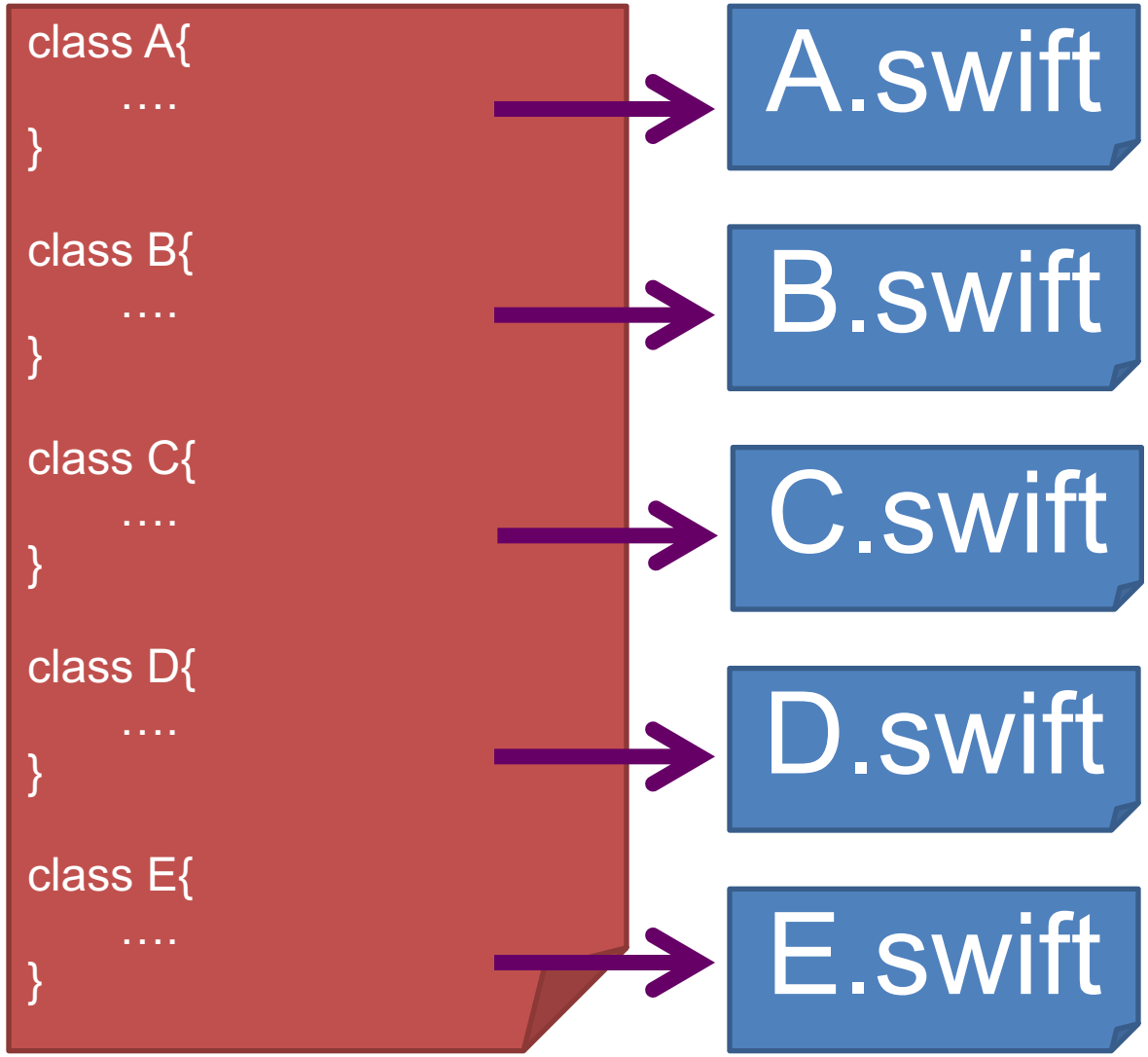
```
This circle myCircle with radius 5.5 which area is 95.03
Program ended with exit code: 0
```

把類別放在不同的檔案中

全部都寫在一起，萬一程式碼很多怎麼辦

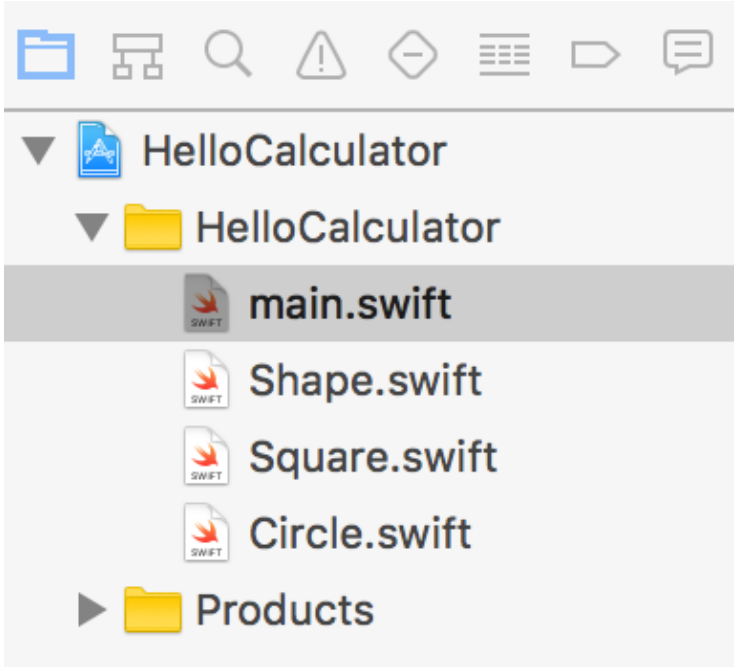
用檔案來區分用途

- 全部寫在一起難以管理
- 一個類別一個檔案



Lab

- 請將Shape, Square, Circle分別獨立成檔案，並且測試是否仍可在main.swift中順利建立實體

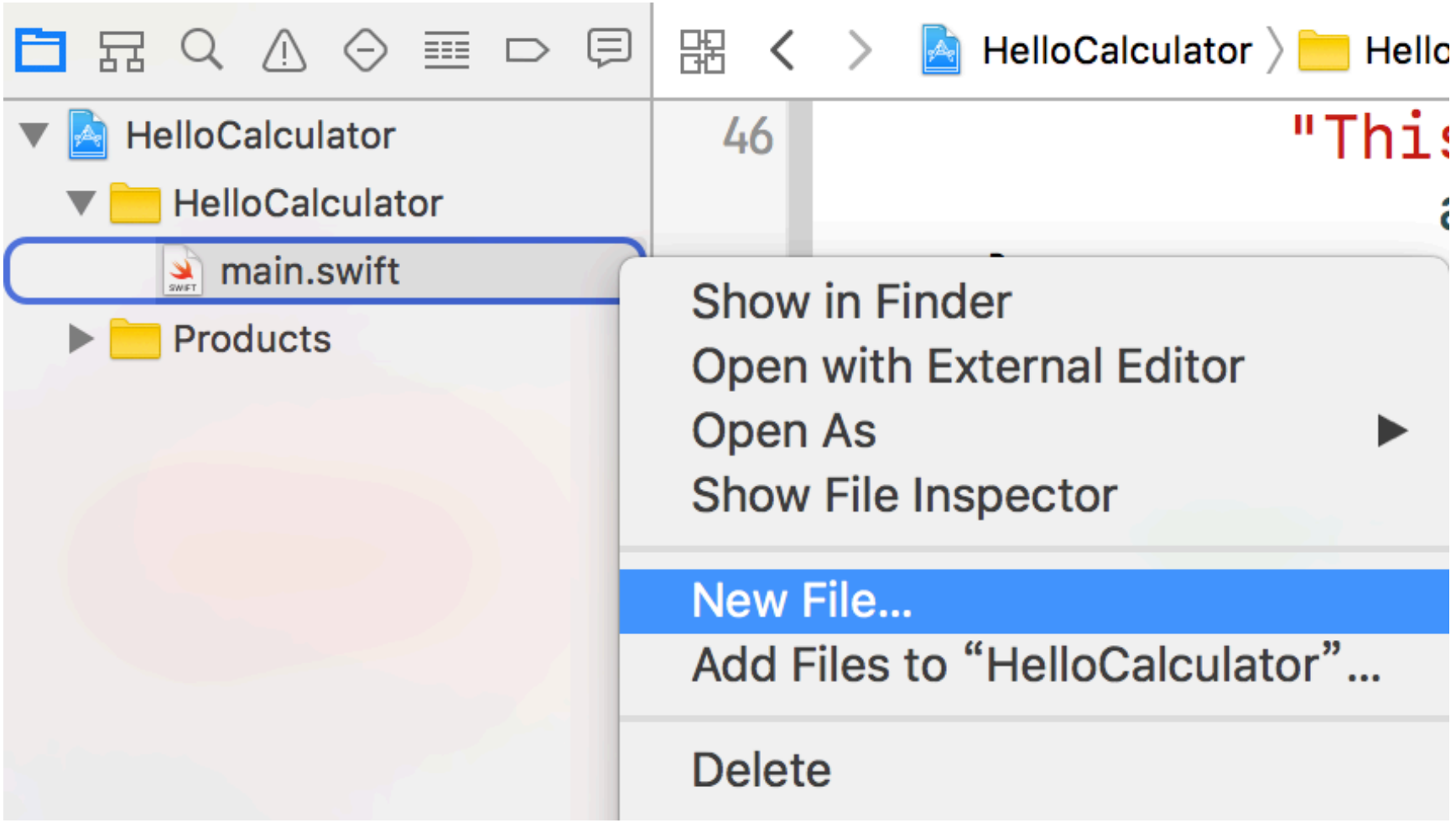



```
9  import Foundation
10
11 var myShape=Shape(name: "Tri", count: 3)
12 print(myShape.simpleDescription())
13
14 let mySquare=Square(length: 5.2, squareName: "mySquare")
15 print(mySquare.simpleDescription())
16
17 let myCircle=Circle(r: 5.5, circleName: "myCircle")
18 print(myCircle.simpleDescription())
19
```

```
This shape Tri with 3 sides.
This square mySquare with side of length 5.2, the area is 27.04
This circle myCircle with radius 5.5 which area is 95.03
Program ended with exit code: 0
```

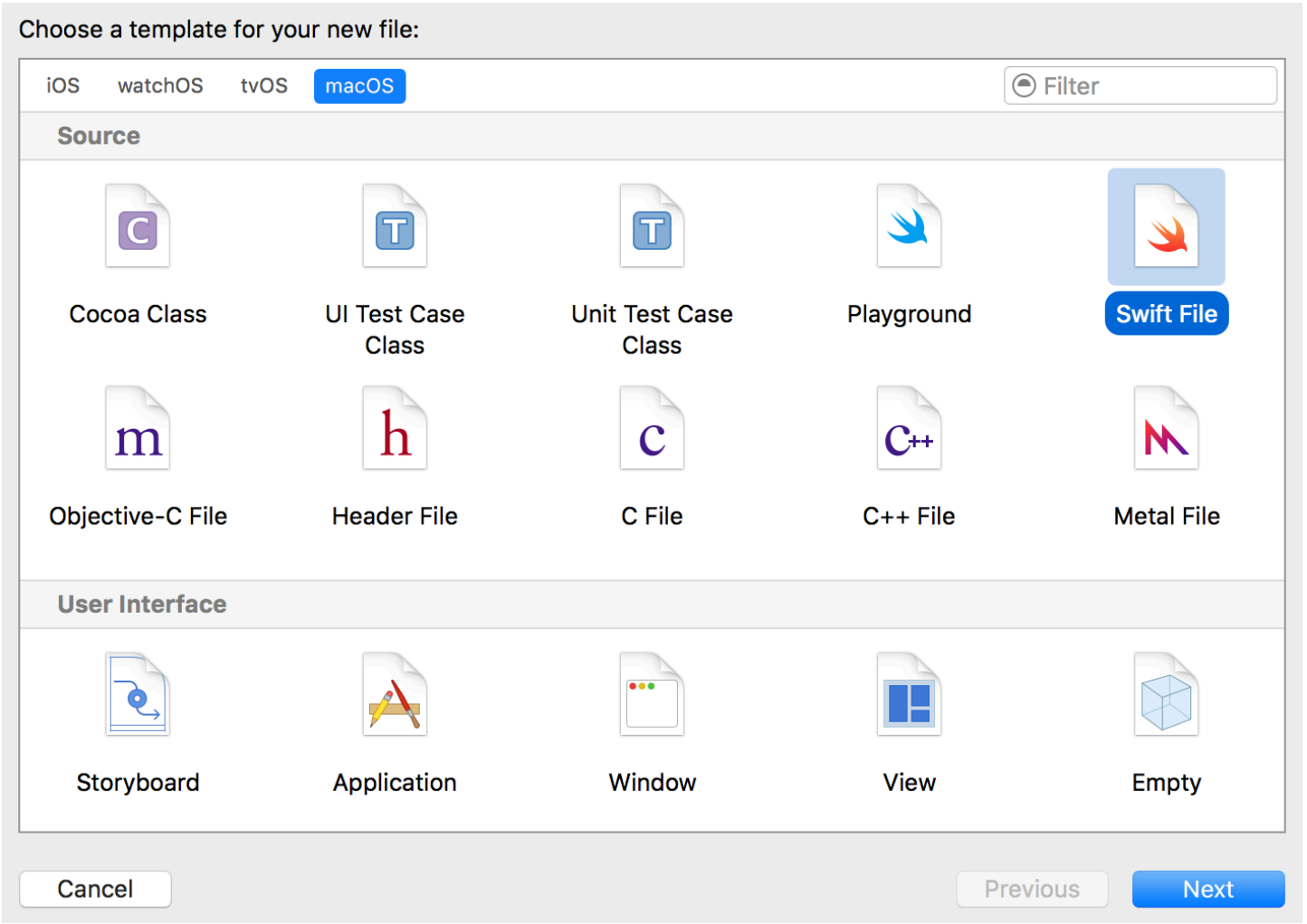
如何增加檔案?

- Step1.點選欲新增檔案的位置，滑鼠右鍵，New File...



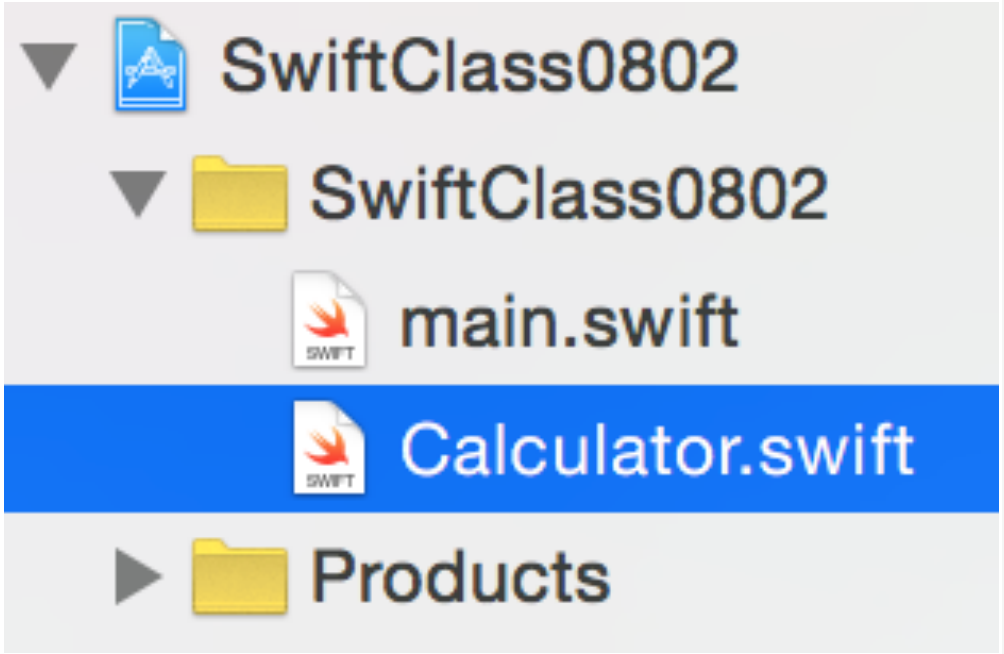
如何增加檔案?

- Step2.決定要製作的檔案類型，再按下Next



如何增加檔案?

- Step3.命名為該Class名稱，然後把類別宣告程式碼搬移過去




```swift
import Foundation

class Calculator {

    var accumulator:Double

    init(accu:Double)
    {
        accumulator=accu
    }

    func add(value:Double)
    {
        accumulator+=value
    }

    func substract(value:Double)
    {
        accumulator-=value
    }
```