Section 1 : Expressions, Variables & Constants

Exercise 1 : Declare two constants a and b of type Double and assign both a value. Calculate the average of a and b and store the result in a constant named average.

挑戰 1 ：宣告兩個常數 a 與 b，型別為 Double，並賦值。
計算 a 與 b的平均，並將結果存在一名為 average 的常數

Exercise 2 : Create a variable called answer and initialize it with the value 0. Increment it by 1. Add 10 to it. Multiply it by 10. After all of these operations, what's the answer?

挑戰 2：創建一個名為 answer 的變數，並賦予初始值 0 ，設值為原值加 1 ，再加 10 ，最後乘予 10，最後答案是多少？

Exercise 3 : A temperature expressed in °C can be converted to °F by multiplying by 1.8 then incrementing by 32. In this challenge, do the reverse: convert a temperature from °F to °C. Declare a constant named fahrenheit of type Double and assign it a value. Calculate the corresponding temperature in °C and store the result in a constant named celsius.

挑戰 3：以°C（攝氏溫度）作為計量單位的溫度在乘予 1.8 再 + 32 後可以得到以°F（華氏溫度）作為計量單位的溫度，在這個挑戰，將上述的計算邏輯進行反向操作，將溫度從°F（華氏溫度）轉換為°C（攝氏溫度）
宣告一個名為 fahrenheit ，型別為 Double 的常數，再賦以任意值。
計算對應的°C（攝氏溫度），並將結果放入一名為 celsius 的常數

Section 2 : Types & Operation

Exercise 1 : What's wrong with the following code?

挑戰 1 ：下面的程式中有什麼錯誤？

```
let nameExample = "Michael"
nameExample = nameExample + " Pan"
```

Exercise 2 : Create a constant called namedCoordinate with a row and column component.

挑戰 2 ：創建一個名為 namedCoordinate 的常數，它必須具備 row 和 column 兩個參數

Exercise 3 : Given the following constant. Please print out the following message “Currently, I’m 25 years old” and make sure to use the age constant in your code.

挑戰 3 ：給定下列變數，請將此信息print出 “Currently, I’m 25 years old” ，並請在你的程式裡面使用常數age

```
let age = 25
```

Section 3 : Basic Control Flow

Exercise 1 : Create a constant named myAge and initialize it with your age. Write an if statement to print out Teenager if your age is between 13 and 19, and Not a teenager if your age is not between 13 and 19.

挑戰 1 ：創建一個名為 myAge 的常數，並根據你的年紀賦予它初始值，接著，寫一個 if 判斷式，規則如下：如果你的年紀介於13歲和19歲，則print出”Teenager”，若不在此範圍，則print出”Not a teenager”

Exercise 2 : Create a variable named counter and set it equal to 0. Create a while loop with the condition counter < 10 which prints out counter is X (where X is replaced with counter value) and then increments counter by 1.

挑戰 2 ：創建一個名為 counter 的變數，並賦予0；寫一個while loop，規則如下：當counter < 10 時，print出counter的值，每次執行後counter的增值為1

Exercise 3 : What’s wrong with the following code?

挑戰 3：下面的程式中有什麼錯誤？

```
let firstName = "Matt"

if firstName == "Matt" {

  let lastName = "Galloway”

} else if firstName == "Ray" {

  let lastName = "Wenderlich”

}

let fullName = firstName + " " + lastName
```

Exercise 4 : Write a program to print out asterisks like the triangle shape below:

```
*
**
***
****
*****
******
*******
********
*********
**********
```

挑戰 4：寫一個程式，印出如下的圖形（一個由*號組成的三角形）：

```
*
**
***
****
*****
******
*******
********
*********
**********
```

提示：你可以利用下面的方法（Method）來完成這個挑戰

```
String(repeating: <String>, count: <Int>)
```

Exercise 5 : Given a number n, calculate the n-th Fibonacci number. (Recall Fibonacci is 1, 1, 2, 3, 5, 8, 13, ... Start with 1 and 1 and add these values together to get the next value. The next value is the sum of the previous two. So the next value in this case is 8+13 = 21.)

挑戰 5：給定一個數字n，計算第n個費波那契數，以下為費波那契的定義：
一個簡單的費波那契數列為：1,1,2,3,5,8,13 …，從1,1開始，並將前、後的數字相加得出下一個數字，在上面的數列中，下一個數字即為 8 + 13 = 21

Section 4 : Advanced Control Flow

Exercise 1 : Create a constant named range and set it equal to a range starting at 1 and ending with 10 inclusive. Write a for loop which iterates over this range and prints the square of each number.

挑戰 1：創建一個名為 range 的常數，並賦予其為從1開始、10結束（包含10）的範圍；寫一個 loop，規則如下：將常數range的每一個值的平方print出

Exercise 2 : Write a switch statement that takes an age as an integer and prints out the life stage related to that age. You can make up the life stages, or use my categorization as follows: 0-2 years, Infant; 3-12 years, Child; 13-19 years, Teenager; 20-39, Adult; 40-60, Middle aged; 61+, Elderly.

挑戰 2：寫一switch語句，使用型別為Int的age作為判斷依據，並根據不同的人生階段將年紀print出，你可以自行定義不同的人生階段，或使用本題的規則： 0-2 歲, Infant; 3-12 歲, Child; 13-19 歲, Teenager; 20-39歲, Adult; 40-60歲, Middle aged; 61歲以上, Elderly.

Exercise 3 : In the following for loop, what will be the value of sum, and how many iterations will happen?

挑戰 3：下面的for loop語句中，sum的值是多少？有多少次的迴圈將會被執行？

```
var sum = 0
for i in 0...5 {
  sum = sum + i
}
```

Exercise 4 : Print 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0. (Note: do not use the stride(from:by:to:) function, which will be introduced later.)

挑戰 4：print出0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0.（請不要使用stride(from:by:to:)，這個語法後面才會講到）

Section 5 : Functions

Exercise 1 : Write a function named printFullName that takes two strings called firstName and lastName.

The function should print out the full name defined as firstName + " " + lastName. Use it to print out your own full name.

挑戰 1：寫一個名為printFullName的函式，其中包含兩個String參數，firstName和lastName這個函式將能print出完整的名字（full name），即firstName + " " + lastName，用這個函式將你的名字印出

Exercise 2 : Change the declaration of printFullName to have no external name for either parameter.

挑戰 2：修改printFullName的宣告方法，使得兩個參數都不使用外部描述（external name）

Exercise 3 : Write a function to determine whether or not a number is prime. First, write the following function:

```
func isNumberDivisible(_ number: Int, by divisor: Int) -> Bool
```

You'll use this to determine if one number is divisible by another. It should return true when number is divisible by divisor.

Hint: You can use the modulo (%) operator to help you out here.

挑戰 3：寫一個判斷質數的函式，首先宣告下列的函式

```
func isNumberDivisible(_ number: Int, by divisor: Int) -> Bool
```

你將用此函式判斷一個數字是不是能被另一個數字整除（即餘數為0）
提示：你可利用 % 來輔助你的函式進行判斷

Section 6 : Optionals

Exercise 1 : Make an optional String called myFavoriteSong.

If you have a favorite song, set it to a string representing that song.

If you have more than one favorite song or no favorite, set the optional to nil.

挑戰 1：創建一個名為 myFavoriteSong 、型別為Optional的常數
如果你有一首喜歡的歌，將歌名用賦予給myFavoriteSong
如果你有超過一首喜歡的歌，或沒有喜歡的歌，將nil賦予myFavoriteSong

Exercise 2 : Which of the following are valid statements?

```
var name: String? = "Ray"
var age: Int = nil
let distance: Float = 26.7
var middleName: String? = nil
```

挑戰 2：選出合法的 Swift 語句

```
var name: String? = "Ray"
var age: Int = nil
let distance: Float = 26.7
var middleName: String? = nil
```

Exercise 3 : First, create a function that returns the number of times an integer can be divided by another integer without a remainder. The function should return nil if the division doesn't produce a whole number. Name the function divideIfWhole.

Then, write code that tries to unwrap the optional result of the function. There should be two cases: upon success, print "Yep, it divides \(answer) times", and upon failure, print "Not divisible :[".
Finally, test your function:
1. Divide 10 by 2. This should print "Yep, it divides 5 times."
2. Divide 10 by 3. This should print "Not divisible :[."

Hint 1: Use the following as the start of the function signature:

```
func divideIfWhole(_ value: Int, by divisor: Int)
```

Hint 2: You can use the modulo operator (%) to determine if a value is divisible by another; recall that this operation returns the remainder from the division of two numbers. For example, 10 % 2 = 0 means that 10 is divisible by 2 with no remainder, whereas 10 % 3 = 1 means that 10 is divisible by 3 with a remainder of 1.

挑戰 3：創建一個回傳「一個數字可被整除次數」的函式，如果這個函式無法產生一個整數，則其將回傳 nil，將它命名為divideIfWhole

接著，試著將此函式的結果進行解析（unwrapped），這會有兩種可能性，1）成功，print 出"Yep, it divides \(answer) times"；2）print 出"Not divisible :["

最後，對你的函式進行下列測試
1. 以2除10，這應該print出 "Yep, it divides 5 times."
2. 以3除10，這應該print出 "Not divisible :[."

提示1：利用下列的函式作為開頭

```
func divideIfWhole(_ value: Int, by divisor: Int)
```

提示2：你可利用 % 來判斷一個數字是不是能被另一個數字整除，例如：10 % 2 = 0 代表 10能夠被2整除，並無餘數，而10 % 3 = 1 ，代表10被3除後餘數並非0，而是1，10無法被3整除

Section 7 : Arrays, Dictionaries, Sets

Exercise 1 : Which of the following are valid statements?

```
1. let array1 = [Int]()
2. let array2 = []
3. let array3: [String] = []
```

挑戰 1：選出合法的 Swift 語句

```
1. let array1 = [Int]()
2. let array2 = []
3. let array3: [String] = []
```

Exercise 2 : Write a function that removes the first occurrence of a given integer from an array of integers. This is the signature of the function:

```
func removingOnce(_ item: Int, from array: [Int]) -> [Int]
```

挑戰 2：創建一個函式，能從一個存有多個整數的Array中「移除第一個給定值」（如[1,3,2,3]，給定值為3，則將Array中的第一個3移除），以下為此函式的基礎表述式

```
func removingOnce(_ item: Int, from array: [Int]) -> [Int]
```

Exercise 3 : Write a function that removes all occurrences of a given integer from an array of integers. This is the signature of the function:

```
func removing(_ item: Int, from array: [Int]) -> [Int]
```

挑戰 3：創建一個函式，能從一個存有多個整數的Array中「移除所有的給定值」（如 [1,3,2,3]，給定值為3，則將Array中所有值為3的資料都移除），以下為此函式的基礎表述式

```
func removing(_ item: Int, from array: [Int]) -> [Int]
```

Exercise 4 : Arrays have a reversed() method that returns an array holding the same elements as the original array, in reverse order. Write a function that does the same thing, without using reversed(). This is the signature of the function:

```
func reversed(_ array: [Int]) -> [Int]
```

挑戰 4：陣列（Arrays）是內建一個名為reversed的函式，這個函式能回傳一個相反順序的陣列，寫一個函式能做到和reversed一樣的功能，但不要使用reversed，下列為此函式的基礎表述式

```
func reversed(_ array: [Int]) -> [Int]
```

Exercise 5 : Given a dictionary with two-letter state codes as keys, and the full state names as values, write a function that prints all the states with names longer than eight characters.

For example, for the dictionary ["NY": "New York", "CA": "California"], the output would be California.

挑戰 5：給定一字典（dictionary），以 2 個字母為州碼作為鍵（keys），完整的州名作為值（values），寫一個函式將所有州名長於8個字母的州都print出
範例： dictionary ["NY": "New York", "CA": "California"], 此函式將print出 California

Exercise 6 : Write a function that combines two dictionaries into one. If a certain key appears in both dictionaries, ignore the pair from the first dictionary. This is the function's signature:

```
func merging(_ dict1: [String: String], with dict2: [String: String]) -> [String: String]
```

挑戰 6：寫一個能將兩個字典（dictionary）合而為一的函式，如果一特定的鍵（keys）在兩個字典都存在，忽略第一個字典中的鍵值，下列為此函式的基礎表述式

```
func merging(_ dict1: [String: String], with dict2: [String: String]) -> [String: String]
```

Exercise 7 : Given the dictionary:

```
var nameTitleLookup: [String: String?] = ["Mary": "Engineer", "Patrick": "Intern", "Ray": "Hacker"]
```

Set the value of the key "Patrick" to nil and completely remove the key and value for "Ray".

挑戰 7：給定一字典（dictionary）如下：

```
var nameTitleLookup: [String: String?] = ["Mary": "Engineer", "Patrick": "Intern", "Ray": "Hacker"]
```

將鍵 "Patrick"設為nil，並完全移除"Ray"的鍵值

Section 8 : Collection Iteration with Closure

Exercise 1 : Create a constant array called names which contains some names as strings. Any names will do — make sure there's more than three.

Now use reduce to create a string which is the concatenation of each name in the array.

挑戰 1：創建一個名為的names的常數陣列（Array），names中含有一些型別為字串的名字，任何名字都可以，至少賦予超過3個名字
接著，利用reduce來創建一個「以所有陣列中名字為組成元件」的字串

Exercise 2 : Using the same names array, first filter the array to contain only names which have more than four characters in them, and then create the same concatenation of names as in the above exercise.

(Hint: you can chain these operations together.)

挑戰 2：利用Exercise 1中的names陣列，將超過4個字符的名字留下，接著同上題，利用reduce來創建一個「以所有陣列中名字為組成元件」的字串

提示：你可以將所有的執行步驟串在一起

Exercise 3 : Write a function that will run a given closure a given number of times. Declare the function like so:

```
func repeatTask(times: Int, task: () -> Void)
```

The function should run the task closure, times number of times.
Use this function to print "Swift Apprentice is a great book!" 10 times.

挑戰 3：創建一個函式，能執行根據參數times 決定其執行閉包（closure） task() 的次數，以下述形式宣告你的函式

```
func repeatTask(times: Int, task: () -> Void)
```

利用此函式print出 "Swift Apprentice is a great book!" 10次

Exercise 4 : In this challenge, you're going to write a function that you can reuse to create different mathematical sums. Declare the function like so:

```
func mathSum(length: Int, series: (Int) -> Int) -> Int
```

The first parameter, length, defines the number of values to sum.
The second parameter, series, is a closure that can be used to generate a series of values. Series should have a parameter that is the position of the value in the series and return the value at that position.
mathSum should calculate length number of values, starting at position 1, and return their sum.

Use the function to find the sum of the first 10 square numbers, which equals 385. Then use the function to find the sum of the first 10 Fibonacci numbers, which equals 143. For the Fibonacci numbers, you can use the function you wrote in the functions chapter — or grab it from the solutions if you're unsure your solution is correct.

挑戰 4：在這個挑戰中，創建一個能重複利用、計算數字總和的函式，以下述形式宣告你的函式：

```
func mathSum(length: Int, series: (Int) -> Int) -> Int
```

第一個參數Length，決定了要進行加總的數字長度
第二個參數series，是一個能用於產生一連串數字的閉包（closure）
series應該有一個參數名為position，代表著這一連串數字中的特定數字位置
mathSum應從位置1開始計算這串數字的總和

Section 9 : Strings

Exercise 1 : Write a function which tells you how many words there are in a string. Do it without splitting the string but rather iterating through the string yourself.

挑戰 1：創建一個函式，它能告訴你一個字串（String）中有多少個字，透過不斷遞迴此字串，而不是拆開字串的方式來進行計算

Exercise 2 : Write a function which takes a string which looks like ”Pan, Michael" and returns one which looks like ”Michael Pan"; i.e. the string goes from "<LAST_NAME>, <FIRST_NAME>" to "<FIRST_NAME> <LAST_NAME>".

挑戰 2：創建一個函式能接收 “Pan, Michael“ 這樣形式的字串 ，並回傳 ”Michael Pan"，即此字串從 “<LAST_NAME>, <FIRST_NAME>” 轉變為 "<FIRST_NAME> <LAST_NAME>”

Exercise 3 : Write a function which reverse a string which looks like ”abcdefg" and returns one which looks like ”gfedcba"

挑戰 3：創建一個函式，能將字串倒序排列，如輸入”abcdefg"，則回傳”gfedcba"

Exercise 4 : There exists a method on string named split(separator: <character>) that will split the string into chunks delimited by the given string and return an array containing the results. Your challenge is to implement this yourself.

Hint: There exists a view on String named indices which lets you iterate through all the indices (of type String.Index) in the string. You will need to use this.

“for i in string.indices” can iterate through all the index in string

挑戰 4：有一個既存的方法（Method）名為split(separator: <character>)，它能根據特定的符號將字串拆成多塊，並回傳成一個陣列（Array）的型態，你的任務是實現這樣的功能，而不使用split

提示：有一個String的property為indices，透過它你能得到每一個String的Index，具體方法為“for i in string.indices”，這能遍歷每一個String的Index

Section 10 : Structures

Exercise 1 : Write a structure that represents a pizza order. Include toppings, size and any other option you'd want for a pizza.

挑戰1：創建一個建構式（structure），它代表著一個披薩訂單的結構，包括佐料（toppings）、尺寸（sizes）和其他你認為一個的披薩應有的屬性

Exercise 2 : Create a T-shirt structure that has size, color and material options.

Provide methods to calculate the cost of a shirt based on its attributes.

挑戰2：創建一個建構式（structure），它代表著一個T-Shirt的結構，包括尺寸（sizes）、顏色（color）和材質（material options），創建一個方法（methods）來根據不同的屬性進行T-Shirt成本的計算

Section 11 : Properties

Exercise 1 : Rewrite the IceCream structure below to use default values and lazy initialization:

```
struct IceCream {
        let name: String
        let ingredients: [String]
    }
```

1. Use default values for the properties.
2. Lazily initialize the ingredients array.

挑戰1：重寫下列的建構式（structure）IceCream，將「默認值（default values）」和「延遲儲存屬性的初始化（Lazy initialization）加入到IceCream中

```
struct IceCream {
        let name: String
        let ingredients: [String]
    }
```

1.替屬性加入默認值
2.延遲初始化ingredients這個陣列

Exercise 2 : Dive into the inner workings of the car and rewrite the FuelTank structure below with property observer functionality:

```
struct Car {
    let make: String
    let color: String
}

struct FuelTank {
    var level: Double // decimal percentage between 0 and 1
}
```

1. Add a lowFuel stored property of Boolean type to the structure.
2. Flip the lowFuel Boolean when the level drops below 10%.
3. Ensure that when the tank fills back up, the lowFuel warning will turn off.
4. Set the level to a minimum of 0 or a maximum of 1 if it gets set above or below the expected values.
5. Add a FuelTank property to Car.

挑戰2：將下列屬性監視器（Property Observers）的功能加入進 FuelTank 建構式

```
struct Car {
    let make: String
    let color: String
}

struct FuelTank {
    var level: Double // decimal percentage between 0 and 1
}
```

1.加入一名為 lowFuel 的布林（Boolean）屬性到 FuelTank 建構式
2.當 level 低於10%時，將 lowFuel 從 true 改為 false
3.當 level 重回10%以上時，將 lowFuel的值改回來
4.設定 level 的最小值為0、最大值為1，當 level的數值超過這個區間時，賦予前一個值
5.將 FuelTank 作為一屬性加到 Car 建構式

Exercise 3 : In the light bulb example, the bulb goes back to a successful setting if the current gets too high. In real life, that wouldn't work. The bulb would burn out!

```
struct LightBulb {
  static let maxCurrent = 40
  var current = 0 {
  didSet {
  if current > LightBulb.maxCurrent {
  print("Current too high, falling back to previous setting.")
  current = oldValue
          }
     }
  }
}

var light = LightBulb()
light.current = 50
var current = light.current // 0
light.current = 40
current = light.current // 40
```

Rewrite the structure so that the bulb turns off before the current burns it out. You'll need to use the willSet observer for this.
This observer gets called before the value is changed.
The value that is about to be set is available in the constant newValue.
You can't change this newValue, and it will still be set, so you'll have to do more than just add a willSet observer. :]

挑戰3：在下列的燈泡範例中，如果電流過高，燈泡將回到之前的狀態，繼續運作，但在現實世界中，燈泡會燒壞！

將LightBulb建構式重寫，在電流超過最大乘載導致「燈泡燒壞之前」將燈泡關閉，你會用到willSet監視器，這個監視器會在電流改變之前就被呼叫，即將要被設定的數值可以使用內建的常數newValue
你無法改變newValue，所以你會需要加入除了willSet監視器以外的判斷

```
struct LightBulb {
  static let maxCurrent = 40
  var current = 0 {
  didSet {
  if current > LightBulb.maxCurrent {
  print("Current too high, falling back to previous setting.")
  current = oldValue
          }
     }
  }
}

var light = LightBulb()
light.current = 50
var current = light.current // 0
light.current = 40
current = light.current // 40
```

Section 12 : Methods

Exercise 1 : Given the following code :

```
let months = ["January", "February", "March",
                  "April", "May", "June",
                  "July", "August", "September",
                  "October", "November", "December"]

struct SimpleDate {
    var month: String

    func monthsUntilWinterBreak() -> Int {
    return months.index(of: "December")! -
           months.index(of: self.month)!
         }
}
```

Since monthsUntilWinterBreak( ) returns a single value and there's not much calculation involved, transform the method into a computed property with a getter component.

挑戰1：給定下列程式：

```
let months = ["January", "February", "March",
                  "April", "May", "June",
                  "July", "August", "September",
                  "October", "November", "December"]

struct SimpleDate {
    var month: String

    func monthsUntilWinterBreak() -> Int {
    return months.index(of: "December")! -
           months.index(of: self.month)!
         }
}
```

monthsUntilWinterBreak( )僅回傳一單一值，也並未牽涉到太多計算，將此方法 (methods) 改寫為一有著getter的計算屬性 (Computed Property)

Exercise 2 : Given the Circle structure below:

```
struct Circle {
    var radius = 0.0
    var area: Double {
        return .pi * radius * radius
    }
    init(radius: Double) {
        self.radius = radius
    }
}
```

Write a method that can change an instance's area by a growth factor.
For example if you call circle.grow(byFactor: 3), the area of the instance will triple.

挑戰2：給定下列 Circle 建構式（structure）：

```
struct Circle {
    var radius = 0.0
    var area: Double {
        return .pi * radius * radius
    }
    init(radius: Double) {
        self.radius = radius
    }
}
```

創建一個方法（method），能使得 area 的值呈指定倍數增長
即呼叫circle.grow(byFactor: 3)，area增加3倍

Exercise 3 : Below is a naive way of writing advance( ) for the SimpleDate structure you saw earlier in the chapter:

```
let months = ["January", "February", "March",
                  "April", "May", "June",
                  "July", "August", "September",
                  "October", "November", "December"]

struct SimpleDate {
    var month: String
    var day: Int

    mutating func advance() {
        day += 1
    }
}

var date = SimpleDate(month: "December", day: 31)
date.advance()
date.month // December; should be January!
date.day // 32; should be 1!
```

What happens when the function should go from the end of one month to the start of the next? Rewrite advance() to account for advancing from December 31st to January 1st.

挑戰3：給定下列 SimpleDate 建構式（structure），其中對於advance ()的設置是比較粗糙的（day 的最大值應為31）

```
let months = ["January", "February", "March",
                  "April", "May", "June",
                  "July", "August", "September",
```

```
                  "October", "November", "December"]

struct SimpleDate {
    var month: String
    var day: Int

    mutating func advance() {
        day += 1
    }
}

var date = SimpleDate(month: "December", day: 31)
date.advance()
date.month // December; should be January!
date.day // 32; should be 1!
```

將advance ( )改成能正確判斷跨月後的月份 (month) 和日期 (day)

Exercise 4 : Given the following code:

```
struct Math {

    static func plusTen(of number: Int) -> Int {
        return number + 10
    }
```

Add type methods named isEven and isOdd to Math namespace that return true if a number is even or odd respectively.

挑戰4：給定下列程式

```
struct Math {

    static func plusTen(of number: Int) -> Int {
        return number + 10
    }
```

加入類別方法(type method) ，名為isEven，用於判斷數字是否為「偶數」，另一為isOdd，用於判斷數字是否為「奇數」

Exercise 5 :
It turns out that Int is simply a struct.
Add the computed properties isEven and isOdd to Int
using an extension.

挑戰5：事實上，在Swift也是一建構式（structure），利用extension，將挑戰4的 isEven 和 isOdd 以「計算屬性（Computed Property）的型式」加入Int 建構式中