

109-1: EE4052

通識課程： 計算機程式設計 之旅

Computer Programming

# Unit 06: 邏輯判斷與流程控制

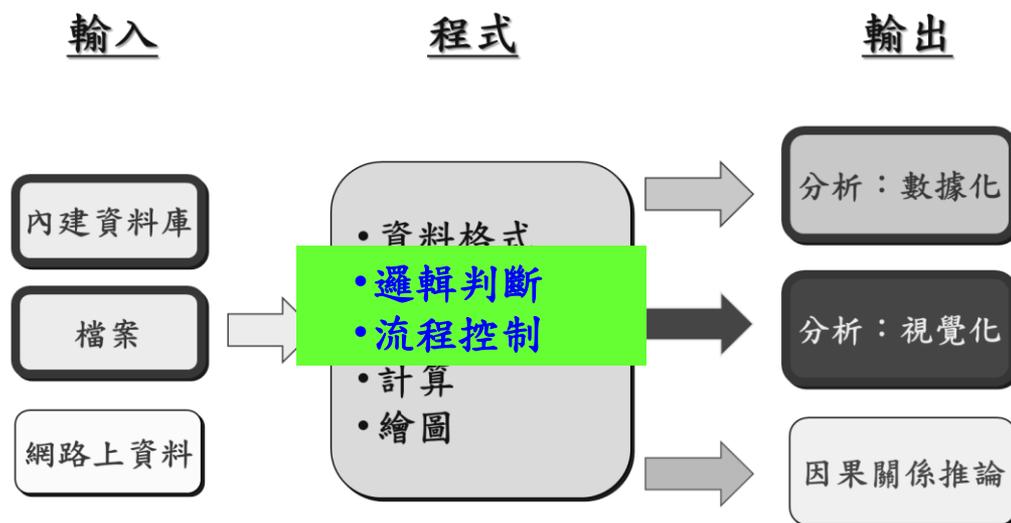
連 豐 力

臺大電機系

Sep 2020 - Jan 2021

# 課程主題進度

- **U01:** 課程介紹：討論主題，作業，報告，進行方式
- **U02:** 主題，案例，程式，演算法，資源
- **U03:** 設定軟體 R 與 Rstudio
- **U04:** 數據處理與繪圖指令功能
- **U05:** 資料類別與基本運算
- **U06:** 邏輯判斷與流程控制
- **U07:** 函數：計算與排序
- **U08:** 多維度資料格式
- **U09:** 檔案資料輸入與輸出
- **U10:** 繪圖功能與文字
- **U11:** 多重繪圖與顏色
- **U12:** 函數：動畫與動作
- **U13:** 探索性資料分析
- **U14:** 資料間的相關性
- **U15:** 資料連結分析



- 邏輯變數、判斷及運算
- 條件分支
- 迴圈

- $x \leftarrow c(-1.2, 3.4, 5.7, -5, 0, 2)$                       % 6 個數字
- $y \leftarrow c(-2.2, 4.4, -6.6, 8.8, 0, -3.3)$                       % 6 個數字
  
- 在  $x$  中，哪幾個大於或等於2，共有幾個，以及哪些數字？
  
- 在  $x$  與  $y$  中，哪幾個滿足  $x_i < y_i$ ，共有幾個，以及哪些數字？

# 作業

# HW04：邏輯判斷與流程控制

On 10/21, 2020

- 下表上學期修習 R 程式語言這門課的10位同學成績：

| ID    | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
|-------|----|----|----|----|----|----|----|----|----|----|
| Score | 92 | 74 | 85 | 60 | 45 | 83 | 66 | 78 | 95 | 55 |

- 為了要給每一位同學評語，如下所示：
  - 0 - 59分： **bad**
  - 60 - 74分： **ok**
  - 75 - 89分： **good**
  - 90 - 100分： **excellent**
  - 注意：要先判斷分數是否介於 0-100，  
也就是，負的與大於100的分數，需要給警示。
- 請參考，課程網站上的資料檔案，與範例程式。

# HW04：邏輯判斷與流程控制

On 10/21, 2020

- 請寫一個程式，分別使用下面三種方式，來達到這個目的：
  1. 使用 **if/else** 的設計
  2. 使用 **which( )** 的設計
  3. 使用 **switch( )** 的設計
- 輸出結果到：**scoreIF**，**scoreWHICH**，**scoreSWITCH**，內容至少包含：

|         |           |    |      |    |     |      |    |      |           |     |
|---------|-----------|----|------|----|-----|------|----|------|-----------|-----|
| ID      | 1         | 2  | 3    | 4  | 5   | 6    | 7  | 8    | 9         | 10  |
| Score   | 92        | 74 | 85   | 60 | 45  | 83   | 66 | 78   | 95        | 55  |
| Grading | excellent | ok | good | ok | bad | good | ok | good | excellent | bad |

- 比較一下這三個結果是否一致？
- 可以先用一組已經結果的分數，  
例如：from -10 to 110，測試看看結果是否正確！！！
- 在 .R 的程式碼中，註解您所加註的程式碼的意義或想法。

# HW04：邏輯判斷與流程控制

計算機程式設計 - 2020F

U06: 邏輯判斷-流程控制

Feng-Li Lian @ NTU-EE

On 10/21, 2020

- 繳交下面檔案，檔案名稱：**HW04\_學號\_關鍵字.xxx**
  - R 程式檔案：**HW04\_B01921001\_LogicFlow.R**
  - 報告檔案：**HW04\_B01921001\_LogicFlow.pdf**
- 繳交方式與期限：
  - 上傳檔案到：**<https://cool.ntu.edu.tw>**
  - 繳交期限：**10/26 (Mon), 11pm 以前**
- 學習方式：
  - 請至下面網址輸入此次的學習方式所花的時間：
  - **<https://forms.gle/TGYXj2uLoL4HwqLHA>**

# 邏輯變數、判斷及運算



# 將其他類型轉為邏輯類型

- `as.logical( )` 將其他類型轉為邏輯類型之物件
- 將數字向量轉為邏輯類型之物件
  - `as.logical( c(0, 1) )`
  - `as.logical( c( -2.2, -1, 0, 1, 2.2 ) )`
- 將文字轉為邏輯類型之物件
  - `as.logical( c("T", "TRUE", "True", "true") )`
  - `as.logical( c("F", "FALSE", "False", "false") )`
  - `as.logical( "handsome" )`

# 判斷一個物件是否為邏輯類型

- `is.logical( )`                      判斷一個物件是否為邏輯類型
  - `is.logical( 3 < 5 )`
  - `is.logical( c(TRUE, FALSE, FALSE, TRUE) )`
  - `is.logical( c(-2.2, -1, 0, 1, 2.2) )`
  - `is.logical( "handsome" )`

# 邏輯函數 - any(), all()

- `x <- c( -1.2, 0.5, 1.0, 1.3, 2.4, 5, 6.3 )`

- `any( 1 < x )`

- `any( x < 5 )`

- `any( 1 < x & x < 5 )`

- 是否有任何數字介於1到5之間？

- `any( 1 < x ) & any( x < 5 )`

- `all( 1 < x )`

- `all( x < 5 )`

- `all( 1 < x & x < 5 )`

- 是否所有數字介於1到5之間？

- `all( 1 < x ) & all( x < 5 )`

# 邏輯函數 - which()

- `x <- c( 1.2, -3.4, 5.7, -6, 0, 3 )`
- `which( x >= 1 )`
  - 哪幾個數字大於等於1?
- `which( (x >= 1) & (x <= 4) )`
  - 哪幾個數字大於等於1且小於等於4?
- `which( x >= 6 )`
  - 哪幾個數字大於等於6?
  
- `class( which( x >= 1 ) )`
  
- `x[ which( x >= 1 ) ]`
  - 哪些數字大於等於1?
  
- `length( which( x >= 1 ) )`
  - 有幾個數字大於等於1?
- `length( which( x >= 6 ) )`
  - 有幾個數字大於等於6?

# 條件分支

# 關係運算子

## ■ 常用的關係運算子：

- $x > y$                       大於
- $x \geq y$                       大於等於
- $x < y$                         小於
- $x \leq y$                       小於等於
- $x == y$                       等於
- $x != y$                       不等於

# 關係運算範例

- $3 < 5$
- $3 > 5$
  
- `class( 3 < 5 )`
- `as.integer( 3 < 5 )`
- `as.integer( 3 > 5 )`
  
- $2.5 * ( 3 < 5 )$
- $2.5 * ( 3 > 5 )$

# 關係運算範例

- `x <- c( 1.2, -3.4, 5.7, -6, 0, 3 )`
- `x >= 0`
- `as.integer( x >= 0 )`
- `sum( x >= 0 )`
- `table( x >= 0 )`
- `x[ x >= 0 ]`

# 關係運算範例 - 數列比較

- `x <- c( 1.2, -3.4, 5.7, -6, 0, 3 )`
- `y <- c( 2.2, -4.4, 6.6, -8.8, 0, 3.3 )`
  
- `x < y`
  
- `sum( x < y )`
  
- `table( x < y )`
  
- `x[ x < y ]`

# 關係運算範例 - 數值相等比較

- `x <- 0.5 - 0.3`
- `y <- 0.3 - 0.1`
- `x == y`
  
- `print( x )`
- `print( y )`
  
- `print( x, digits = 10 )`
- `print( x, digits = 20 )`
- `print( y, digits = 10 )`
- `print( y, digits = 20 )`
  
- `all.equal( x, y )`
  
- `identical( all.equal( x, y ), TRUE )`
  
- `round( x, 10 ) == round( y, 10 )`
- `round( x, 20 ) == round( y, 20 )`
- `round( x, 30 ) == round( y, 30 )`

# 關係運算範例 - 數值相等比較

- `a <- 1:10 / 16`
- `a`
- `print( a )`
- `print( a, digits = 20 )`
  
- `b <- 1:10 / 10`
- `b`
- `print( b, digits = 20 )`

# 條件分支函數 – if / else

- `if( statement_1 )`  
`{`  
`statement_2`  
`}`
- `if( statement_1 ){`  
`statement_2`  
`} else {`  
`statement_3`  
`}`

- `statement_1`
  - 條件判斷成立與否，例如：
    - `x > 10`
    - `word == "good"`
- `statement_2`
  - 進行的計算或動作，例如：
    - `data <- data + 1`
    - `plot( data )`

# 條件分支函數範例

- `x <- 2.5; y <- 4.7`
  
- `if ( x < y ) { z1 <- x } else { z1 <- y }`
- `z1`
  
- `if ( x > y ) { z2 <- x } else { z2 <- y }`
- `z2`
  
- `min( x, y )`
- `max( x, y )`

# 條件分支函數範例

- `x <- "handsome"`
- `y <- 2`
- `z <- 2`
  
- `if( x == "handsome" ) { y <- y + 1 }`
- `y`
  
- `if( x == "beautiful" ) { y <- y - 1 }`
- `y`

# 條件分支函數 – ifelse( )

- ifelse( logical condition, value.true if TRUE, value.false if FALSE )
  
- `x <- -1.5`
- `ifelse( x > 0, x, -x )`
  
- `abs( x )`

# 條件分支函數 - switch()

- `switch( expression, statement_1, statement_2, statement_3, ... )`

- 如果：`expression = 1, 2, 3, ..., 整數`

- 分別對應到：

`statement_1, statement_2, statement_3, ...` 的動作

- `x <- 3`

- `switch( x, 2 + 2, mean(1:10), 1:5 )`

- `switch( 6, 2 + 2, mean(1:10), 1:5 )`

# 條件分支函數 - switch()

- `switch( expression, statement_1, statement_2, statement_3, ..... )`

- 如果：`expression = 某些關鍵字`，
  - `statement_1, statement_2, statement_3, ...`，  
需要對應的關鍵字與動作

- `y <- "fruit"`

- `switch( y, fruit = "banana", vegetable = "broccoli" )`

- `y <- "vegetable"`

- `switch( y, fruit = "banana", vegetable = "broccoli" )`

- `y <- "meat"`

- `switch( y, fruit = "banana ", vegetable = "broccoli", "Neither" )`

# 迴圈

# 迴圈指令 – for( )

- for( )
- while( )
- repeat( )
  
- next( )
- break( )

# 迴圈指令 – for( )

- `for( index in index.set ) {`

`statement`

`}`

- `x <- 0`

- `for ( k in 1:10 ) {`

`x <- x + 1`

`}`

- `x`

- `x <- rep( 0, times = 10 )`

- `for ( k in 1:10 ) {`

`x[ k ] <- k*k`

`}`

- `x`

# 迴圈指令 - for( )

- for( index in index.set ) {

statement

}

- x <- rep( 0, times = 10 )

- for ( k in c( 1, 3, 5 ) ) {

x[ k ] <- k\*k\*k

}

- x

- x <- rep( 0, times = 10 )

- for ( k in c( 2, 4, 6, 8 ) ) {

x[ k ] <- k\*k

}

- x

# 迴圈指令 - for( )

- `for( index in index.set ) {`  
     `statement`  
     `}`
- $x(k+1) = 4x(k)(1 - x(k))$
- $k = 1, 2, 3, \dots, x(1) = 0.2$
- `x <- 0.2`
- `for ( k in 2:5 ) {`  
     `x[ k ] <- 4* x[ k-1 ] * ( 1 - x[ k-1 ] )`  
     `}`
- `round( x, 4 )`

# 迴圈指令 - for( )

- 給定一個長度為10之向量，計算奇數項之和與偶數項之和的差。
- `x <- 1:10`
- `odd <- seq( from = 1, to = 9, by = 2 )`
- `even <- seq( from = 2, to = 10, by = 2 )`
- `mysum <- mysum.odd <- mysum.even <- 0`
- `for( i in x ) { mysum <- mysum + x[i] }`
- `for( j in odd ) { mysum.odd <- mysum.odd + x[j] }`
- `for( k in even ) { mysum.even <- mysum.even + x[k] }`
- `mysum.odd - mysum.even`
- `sum( x )`
- `sum( x[odd] ) - sum( x[even] )`

# 迴圈指令 – while( )

- while( statement\_1 ) {
  - statement\_2
- }
- x <- c( 1, 3, 2, 5, 4, 2, 5, 3 )
- total <- x[ 1 ]
- count <- 1
- while ( total <= 12 ) {
  - count <- count + 1
  - total <- total + x[ count ]
- }
- count; total

• 總數只有12元，可以買幾個？

# 迴圈指令 - while( )

```

■ while( statement_1 ) {
    statement_2
}
    
```

```

■ x <- c( 1, 3, 2, 5, 4, 2, 5, 3 )
    
```

```

■ total <- x[ 1 ]
    
```

```

■ count <- 1
    
```

```

■ while ( total + x[ count+1 ] <= 12 ) {
    
```

```

    count <- count + 1
    
```

```

    total <- total + x[ count ]
    
```

```

■ }
    
```

```

■ count; total
    
```

• 總數只有12元，可以買幾個？

# 迴圈指令 - while( )

- count; total

- 總數只有12元，可以買幾個？

- $y \leftarrow \text{cumsum}(x)$

- $\text{sum}(y \leq 12)$

# 迴圈指令 – repeat( )

- repeat { statements
  - if ( statement\_1 ) break
- }
- x <- c( 1, 3, 2, 5, 4, 2, 5, 3 )
- total <- x[ 1 ]
- count <- 1
- repeat{ count <- count + 1; total <- total + x[ count ]
- if ( total + x[ count + 1 ] >= 12 ) break
- }
- count; total

下課了！