

# 國立虎尾科技大學

## 機械設計工程系

### 計算機程式 ag4 期末報告

#### PyQt5 事件導向計算器

#### PyQt5 Event-Driven Calculator Project

學生：

設計一甲 40623119 歐宗韋

設計一甲 40623120 蔡宗穎

設計一甲 40623121 蔡朝旭

設計一甲 40623128 張華偉

設計一甲 40623129 陳威誠 (組長)

設計一甲 40623130 陳鉅忠

指導教授：嚴家銘

2017.01.08

# 摘要

Fossil SCM

github

簡易 python 語法

網路自我診斷

免費開源軟體錄影

計算機程式語法

使用網路表單 Ethercalc 彙整所學

# 目錄

摘要	i
目錄	ii
表目錄	iv
圖目錄	v
第一章 前言	1
第二章 可攜程式系統介紹	2
2.1 啟動與關閉	2
第三章 Python 程式語法	5
3.1 變數命名	5
3.2 print 函式	6
3.3 重複迴圈	6
3.4 判斷式	8
3.5 數列	8
第四章 PyQt5 簡介	9
4.1 PyQt5 優點	9
4.2 PyQt5 基本撰寫知識	9
4.3 程式開發流程	9
第五章 Calculator 程式	11
5.1 建立案件	11
5.2 建立按鍵	13
5.3 編寫程式碼	13
第六章 心得	26
6.1 Fossil SCM	26
6.2 網誌心得	26
6.3 Github 協同倉儲	27
6.4 學員心得	28

第七章	結論.....	30
7.1	結論與建議 .....	30
第八章	參考文獻 .....	31

# 表目錄

## 圖目錄

圖 2.1	啟動與關閉 . . . . .	2
圖 2.2	Y 槽 . . . . .	2
圖 2.3	cmd . . . . .	2
圖 2.4	SciTE . . . . .	3
圖 2.5	把 star.bat 檔案拉進 SciTE . . . . .	4
圖 3.1	Print 簡易範例 . . . . .	6
圖 3.2	for 迴圈簡易範例 . . . . .	6
圖 3.3	while 迴圈 . . . . .	7
圖 4.1	程式開發流程圖 . . . . .	10
圖 5.1	Add-Project . . . . .	11
圖 5.2	Form . . . . .	12
圖 5.3	uiuiui . . . . .	13
圖 5.4	button . . . . .	14
圖 5.5	tag . . . . .	15
圖 5.6	py . . . . .	16
圖 5.7	1- . . . . .	17
圖 5.8	2- . . . . .	18
圖 5.9	3- . . . . .	19
圖 5.10	4- . . . . .	20
圖 5.11	5- . . . . .	21
圖 5.12	6- . . . . .	22
圖 5.13	7- . . . . .	23
圖 5.14	8- . . . . .	24
圖 5.15	9- . . . . .	25

圖 5.16	finish . . . . .	25
--------	------------------	----

# 第一章 前言

## 計算器程式期末報告前言

為了將來資訊化的時代，讓電腦幫你解決問題，幫你計算，幫你節省時間，幫你處理事情，而學習計算機相關資訊，雖然連入門都還不到，但已經踏出第一隻腳了，接著只要整理所學並擴充所學，一定可以適應將來資訊化的時代。

自己寫報告覺得很累嗎? 不願擔心，有了 `github`，幫你輕鬆大家一起同時寫報告，不用擔心版本新舊，自動幫你合併整合資料，想找舊版資料嗎? 別擔心 `github` 都幫你記住了，有了 `github` 再也不用擔心豬隊友把你的報告搞得一團亂。



## 第二章 可攜程式系統介紹

可攜程式系統介紹

### 2.1 啟動與關閉

請參考下圖

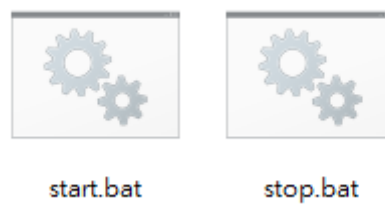


圖 2.1: 啟動與關閉

按下 start 後會產生新的磁碟:Y 槽



圖 2.2: Y 槽

並同時跳出多個 cmd 跟 SciTE



圖 2.3: cmd

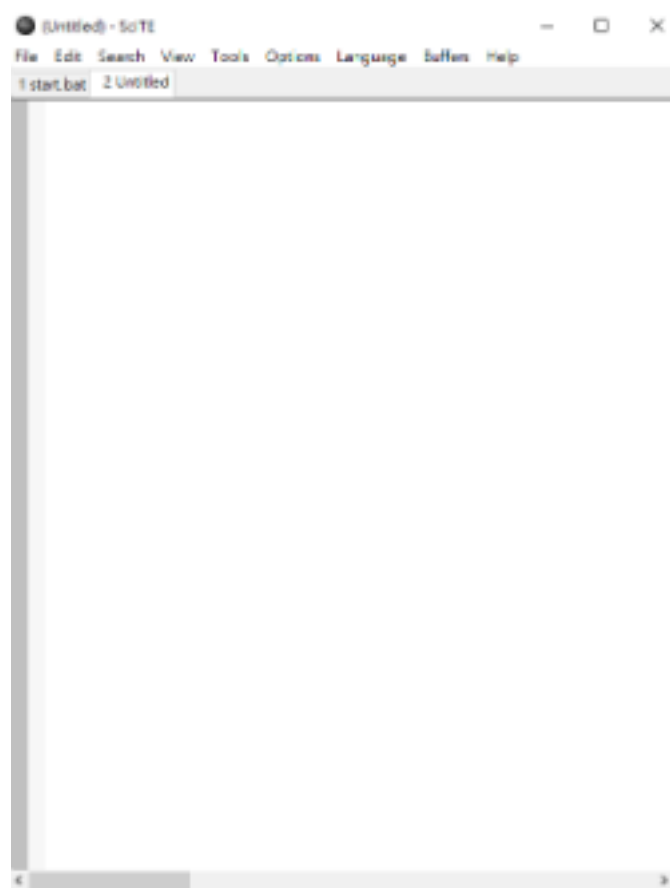


圖 2.4: SciTE

如果要設定跳出的數量只需要把 star.bat 拉進 SciTE



圖 2.5: 把 star.bat 檔案拉進 SciTE

## 第三章 Python 程式語法

### Python 程式語法

#### 3.1 變數命名

變數必須以英文字母大小寫或底線開頭

其餘字元可以是英文大小寫字母, 數字或底線

變數區分英文大小寫

變數不限字元長度

不可使用關鍵字當作變數名稱

變數命名時須避開下列字串

False, None, True, and, as, assert, break, class, continue, def, del, elif, else, except, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield

```
1 number = 10
2 Number = 20
3 _number = 30
4 _Number = 40
5 number_ = 50
6 Number_ = 60
7 print(number, Number, _number, _Number, number_, Number_)
```

Filename: .py

10 20 30 40 50 60

## 3.2 print 函式

`print()` 為 Python 程式語言中用來列印數值或字串的函式

列印數值只需要在括號中打入數字，列印文字則需要加上”“來包住文字，也可以用逗號隔開兩個要列印的數值或文字，它就會直接列印兩個了。

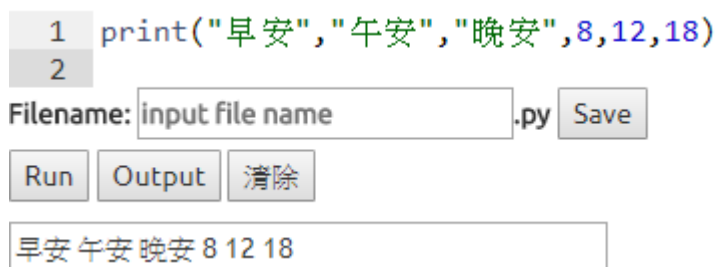


圖 3.1: Print 簡易範例

## 3.3 重複迴圈

重複迴圈分為 `while` 迴圈與 `for` 迴圈

`for` 迴圈:

在計算機科學中，`for` 迴圈（英語：for loop）是一種程式語言的疊代陳述，能夠讓程式碼反覆的執行。

它跟其他的迴圈，如 `while` 迴圈，最大的不同，是它擁有一個迴圈計數器，或是迴圈變數。這使得 `for` 迴圈能夠知道在疊代過程中的執行順序。(取自 wiki)

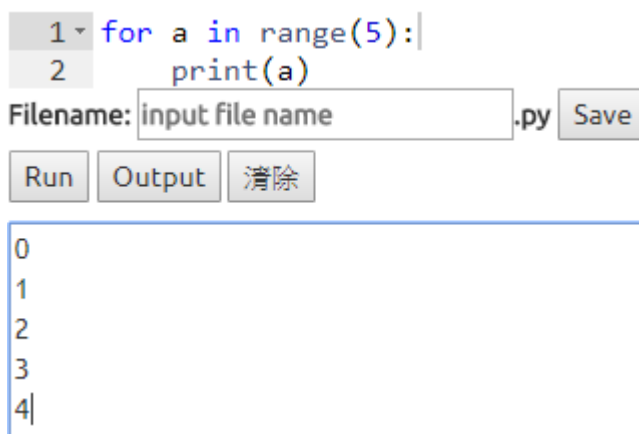


圖 3.2: for 迴圈簡易範例

while 迴圈:

while 迴圈與 for 迴圈最大的差異是 for 迴圈有固定會跑幾次，但 while 迴圈沒有特別限制會跑幾次，因此通常沒有固定跑幾次的迴圈常用 while 迴圈，while 迴圈可以搭配 continue、break、pass。

continue→ 使程式從迴圈的開始繼續執行

break→ 跳出迴圈

pass→ 什麼事都不做

```
1 i=1
2 ans=0
3 while i<=15:
4     ans+=i
5     i+=2
6 print(ans)
```

Filename: .py

64

圖 3.3: while 迴圈

### 3.4 判斷式

判斷式就是我們製作計算機時所使用的 `def calculator` 的內容裡有寫到當達成某一

```
def calculate(self, rightOperand, pendingOperator):
    if pendingOperator == "+":
        self.sumSoFar += rightOperand

    elif pendingOperator == "-":
        self.sumSoFar -= rightOperand

    elif pendingOperator == "x":
        self.factorSoFar *= rightOperand

    elif pendingOperator == "÷":
        if rightOperand == 0.0:
            return False

        self.factorSoFar /= rightOperand

    return True
```

條件時就會執行，反之則不執行

### 3.5 數列

數列有分為 `str`、`unicode`、`list`、`tuple`、`buffer`、`xrange`

`str` → 把括號內轉換成適合閱讀的形式

`unicode` → 有點像程式間的翻譯機每一種不同的語言都可以跟 `unicode` 互相轉換

`list` → 把一串你要的資料輸入中括號中，就可以幫你記憶你只有輸入相應位置的數字就會幫你輸入了

`tuple` → 跟 `list` 功能一樣，只差在不能夠修改

`buffer` → 在 `python3` 中改為 `memoryview`，有點像緩存功能

`xrange` → 用法跟 `range` 一樣，不同的是可以生成等差數列

## 第四章 PyQt5 簡介

說明 PyQt5 基本架構與程式開發流程

### 4.1 PyQt5 優點

PyQt5 有幾項特點

1. 容易撰寫

2. 功能強大

3. 跨平台

4. 容易擴充

5. 易於學習. 閱讀. 維護

### 4.2 PyQt5 基本撰寫知識

Python 程式會存成.py 檔，在 Windows 下的 Python 安裝程式會自動把這種副檔名和 Python 直譯器程式關聯起來，所以在指令行介面 (cmd.exe) 下輸入 “blahblah.py” 這樣的字眼，就會執行這個程式

### 4.3 程式開發流程



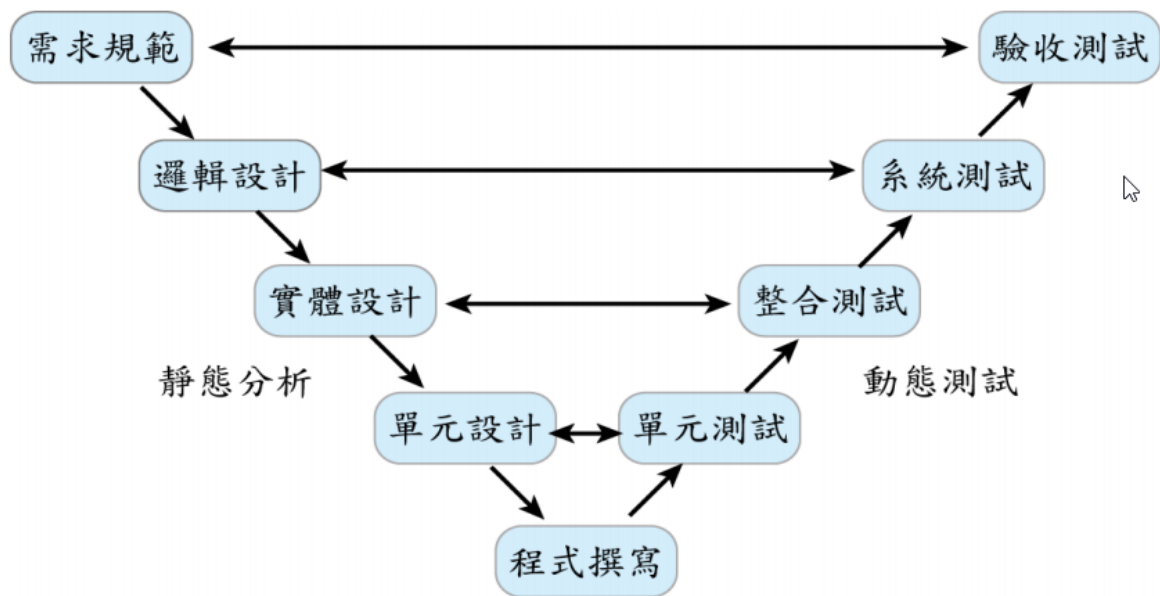


圖 4.1: 程式開發流程圖

## 第五章 Calculator 程式

### Calculator 程式細部說明

#### 5.1 建立案件

在黑盒子打開 eric6

接著如右圖一樣點 **project** → **new+** → 設定檔案位址、名稱與 Main-Scrip

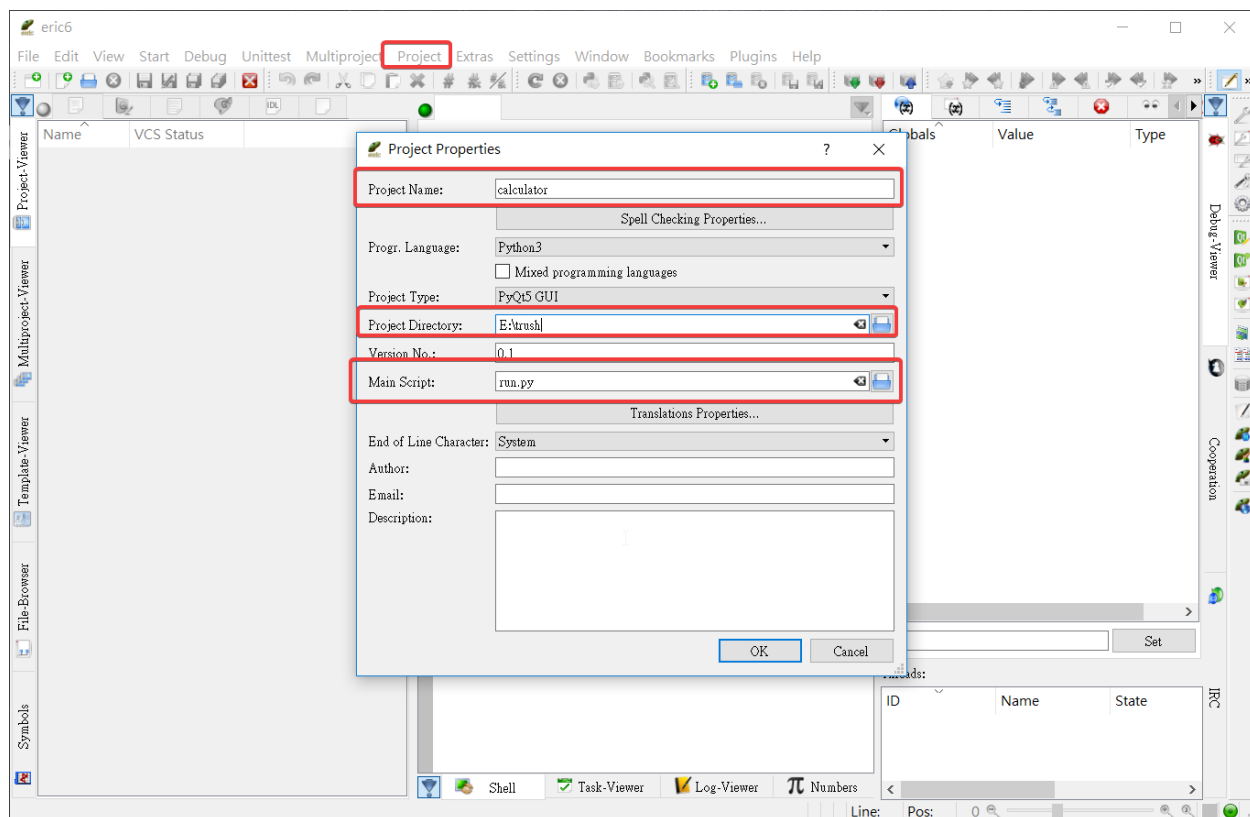


圖 5.1: Add-Project

接著點表單上方的圖 (左邊數來第二張圖有綠色方形與鉛筆) → 對下方空白處點右鍵 → 點按 **New-form** → 選擇 **Dialog**

建立表單專用資料夾與名稱 (ui)

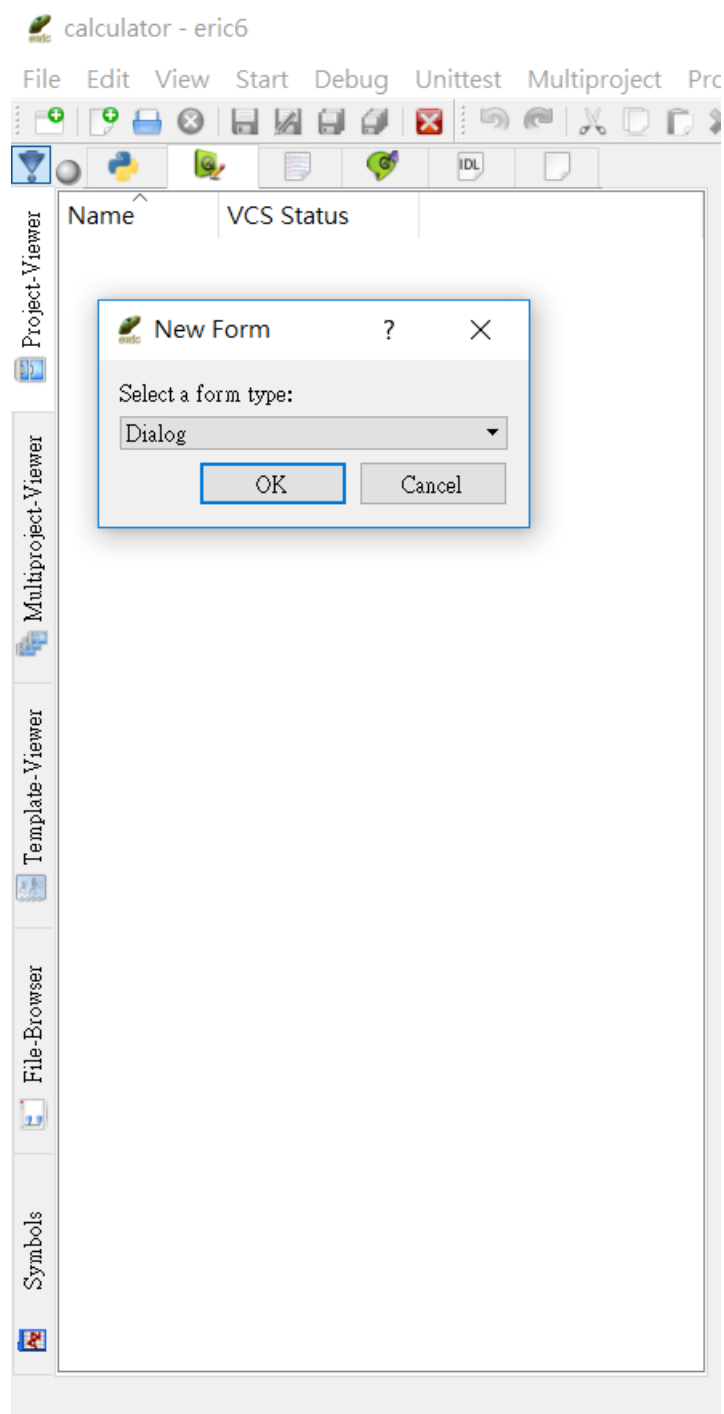


圖 5.2: Form

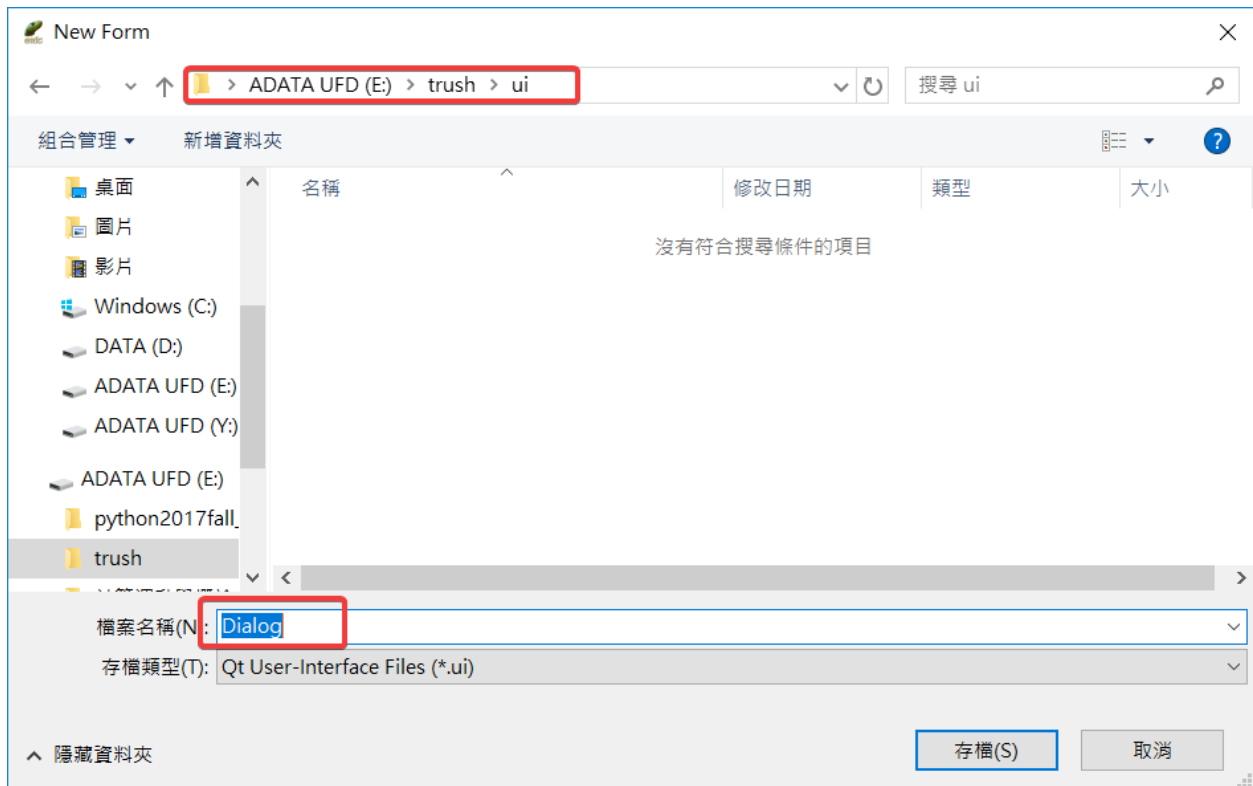


圖 5.3: uiuiui

## 5.2 建立按鈕

之後滑鼠雙擊剛剛建立的 Form → 雙擊後進入 Qt\_Designer → 開始拉左邊的物件到中間的空白格子

拉完排好後開始修改 tag 與物件大小 → 目的: 較好整理按鈕的程式碼和較好的視覺上觀感

## 5.3 編寫程式碼

準備好以上之後開始進入編寫程式碼步編 → 先處理 run.py

之後處理的有加減乘除、等於、數字、小數點、根號、平方、倒數、MS、MR、MC、M+、Allclear、clear、backspace、正負號、Line\_edit

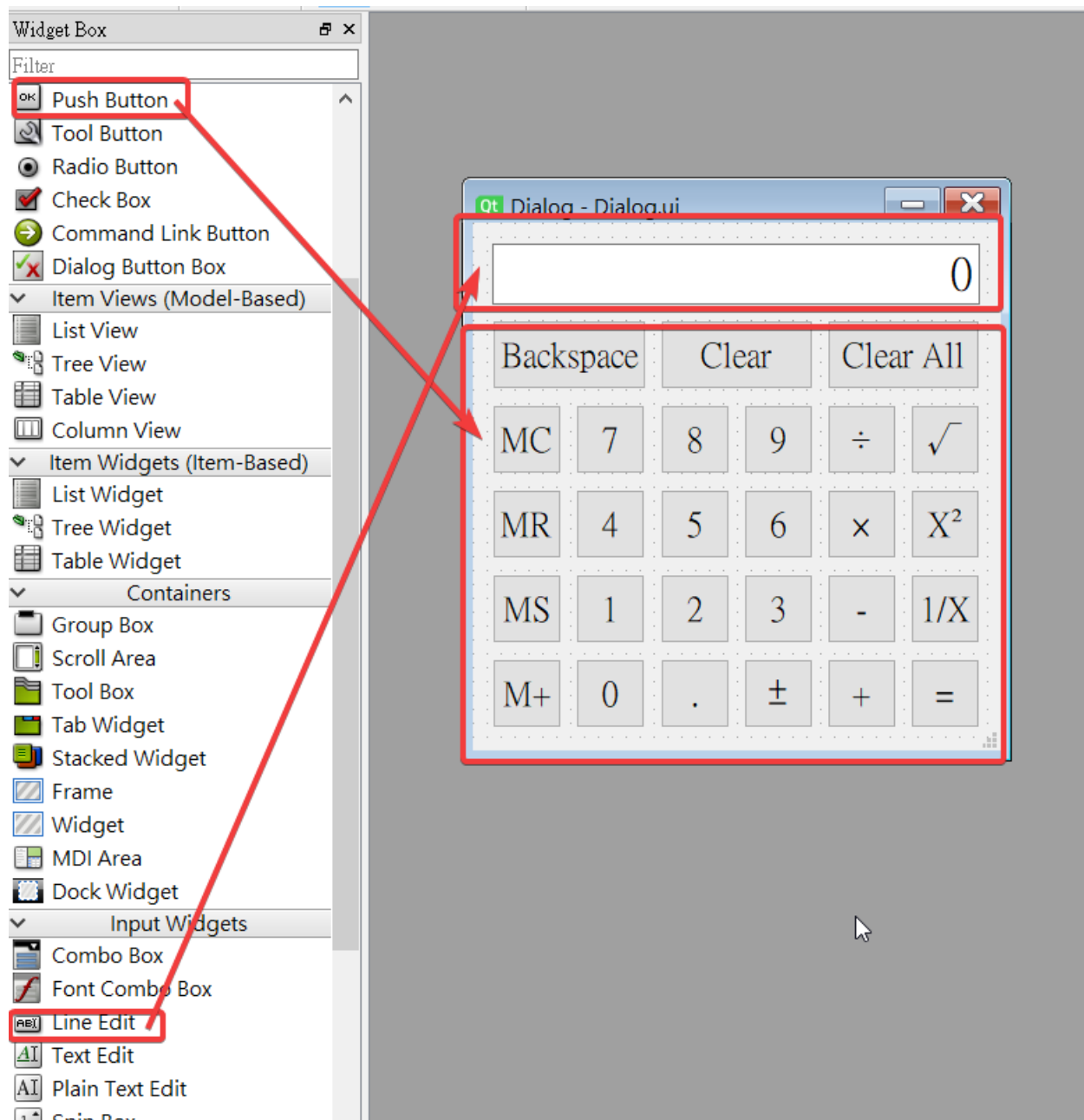


圖 5.4: button

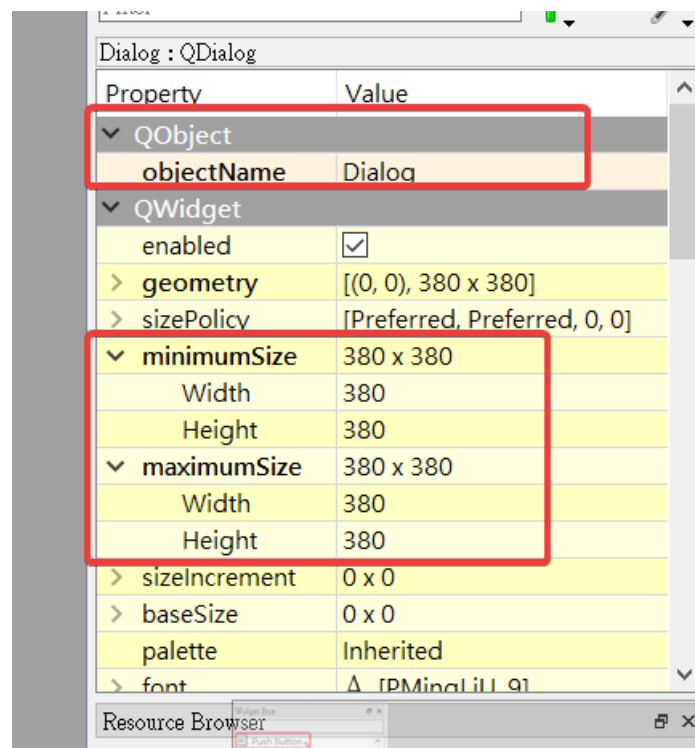
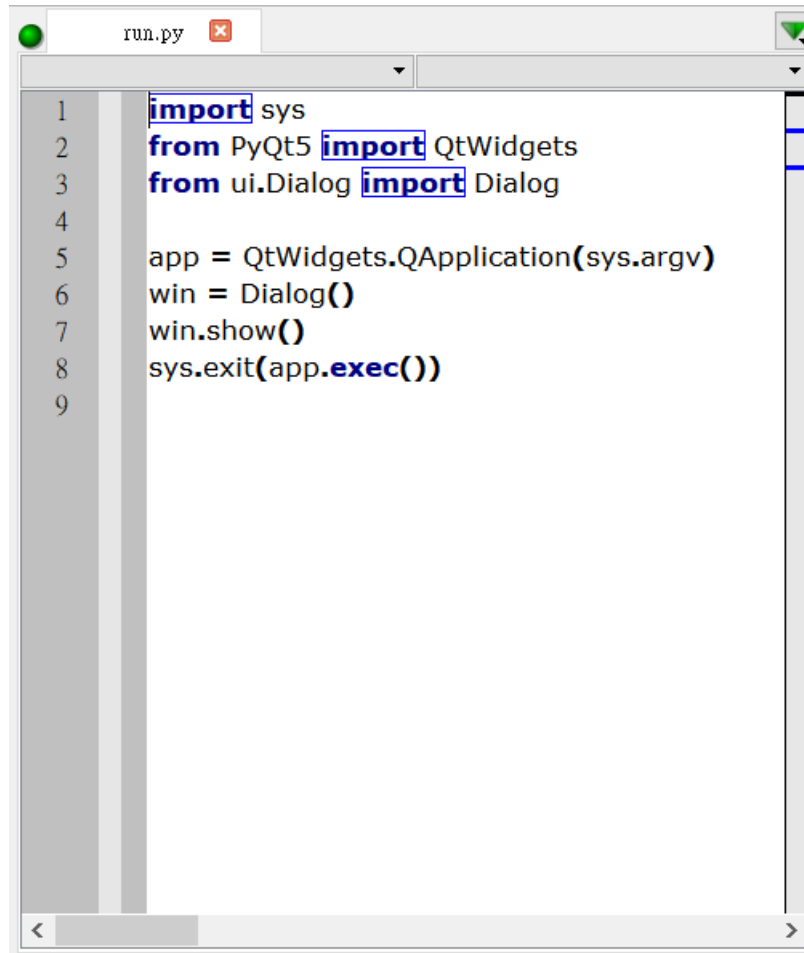


圖 5.5: tag



```
1 import sys
2 from PyQt5 import QtWidgets
3 from ui.Dialog import Dialog
4
5 app = QtWidgets.QApplication(sys.argv)
6 win = Dialog()
7 win.show()
8 sys.exit(app.exec())
9
```

圖 5.6: py

```
1  # -*- coding: utf-8 -*-
2
3  """
4  Module implementing Dialog.
5  """
6  import math
7  from PyQt5.QtCore import pyqtSlot
8  from PyQt5.QtWidgets import QDialog
9
10 from .Ui_Dialog import Ui_Dialog
11
12
13 - class Dialog(QDialog, Ui_Dialog):
14     """
15     Class documentation goes here.
16     """
17 -     def __init__(self, parent=None):
18         """
19         Constructor
20
21         @param parent reference to the parent widget
22         @type QWidget
23         """
24         super(Dialog, self).__init__(parent)
25         self.setupUi(self)
26         '''以下為使用者自行編寫程式碼區'''
27 -         num = [self.one, self.two, self.three, \
28                 self.four, self.five, self.six, \
```

圖 5.7: 1-



```

29         self.seven, self.eight, self.nine, self.zero]
30     for i in num:
31         i.clicked.connect(self.digitClicked)
32     self.clearAllButton.clicked.connect(self.clearAll)
33     self.wait = True
34     self.plusButton.clicked.connect(self.additiveOperatorClicked)
35     self.minusButton.clicked.connect(self.additiveOperatorClicked)
36     self.divisionButton.clicked.connect(self.multiplicativeOperatorClicked)
37     self.timesButton.clicked.connect(self.multiplicativeOperatorClicked)
38     self.temp = 0
39     #正負號
40     self.changeSignButton.clicked.connect(self.changeSignClicked)
41     self.equalButton.clicked.connect(self.equalClicked)
42     self.clearButton.clicked.connect(self.clear)
43     self.backspaceButton.clicked.connect(self.backspaceClicked)
44     self.pushButton_22.clicked.connect(self.pointClicked)
45     self.clearMemoryButton.clicked.connect(self.clearMemory)
46     self.readMemoryButton.clicked.connect(self.readMemory)
47     self.setMemoryButton.clicked.connect(self.setMemory)
48     self.addToMemoryButton.clicked.connect(self.addToMemory)
49     self.sumInMemory = 0.0
50     #等待運算的加或減符號
51     self.pendingAdditiveOperator = ''
52     self.pendingMultiplicativeOperator = ''
53     #+ -的運算值
54     self.sumSoFar = 0.0
55     self.factorSoFar = 0.0
56     unaryOperator = [self.squareRootButton, self.powerButton, self.reciprocalButton ]
57     for i in unaryOperator:

```

圖 5.8: 2-

```

58         i.clicked.connect(self.unaryOperatorClicked)
59     def digitClicked(self):
60         """
61         使用者按下數字鍵, 必須能夠累積顯示該數字
62         當顯示幕已經為 0, 再按零不會顯示 00, 而仍顯示 0 或 0.0
63
64         """
65         #pass
66         clickedButton = self.sender()
67         digitValue = int(clickedButton.text())
68         if self.display.text() == '0' and digitValue == 0:
69             return
70         if self.wait:
71             self.display.clear()
72             self.wait = False
73         self.display.setText(self.display.text() + str(digitValue))
74     def unaryOperatorClicked(self):
75         """單一運算元按下後處理方法"""
76         clickedButton = self.sender()
77         clickedOperator = clickedButton.text()
78         operand = float(self.display.text())
79
80         if clickedOperator == "√":
81             if operand < 0.0:
82                 self.abortOperation()
83             return
84
85         result = math.sqrt(operand)
86         elif clickedOperator == "x²":

```

圖 5.9: 3-

```

87         result = math.pow(operand, 2.0)
88     elif clickedOperator == "1/x":
89         if operand == 0.0:
90             self.abortOperation()
91             return
92         result = 1.0 / operand
93     self.display.setText(str(result))
94     self.waitingForOperand = True
95     def additiveOperatorClicked(self):
96         """加或減按下後進行的處理方法"""
97         clickedButton = self.sender()
98         clickedOperator = clickedButton.text()
99         operand = float(self.display.text())
100         if self.pendingMultiplicativeOperator:
101             if not self.calculate(operand, self.pendingMultiplicativeOperator):
102                 self.abortOperation()
103             return
104
105         self.display.setText(str(self.factorSoFar))
106         operand = self.factorSoFar
107         self.factorSoFar = 0.0
108         self.pendingMultiplicativeOperator = ''
109         if self.pendingAdditiveOperator:
110             if not self.calculate(operand, self.pendingAdditiveOperator):
111                 self.abortOperation()
112             return
113         self.display.setText(str(self.sumSoFar))
114     else:
115         self.sumSoFar = operand

```

圖 5.10: 4-

```

116 self.pendingAdditiveOperator = clickedOperator
117 self.wait = True
118 - def multiplicativeOperatorClicked(self):
119     '''乘或除按下後進行的處理方法'''
120     #pass
121     clickedButton = self.sender()
122     clickedOperator = clickedButton.text()
123     operand = float(self.display.text())
124     - if self.pendingMultiplicativeOperator:
125     -     if not self.calculate(operand, self.pendingMultiplicativeOperator):
126         self.abortOperation()
127         return
128     self.display.setText(str(self.factorSoFar))
129     - else:
130         self.factorSoFar = operand
131     self.pendingMultiplicativeOperator = clickedOperator
132     self.wait = True
133
134 - def equalClicked(self):
135     '''等號按下後的處理方法'''
136     #pass
137     operand = float(self.display.text())
138     - if self.pendingMultiplicativeOperator:
139     -     if not self.calculate(operand, self.pendingMultiplicativeOperator):
140         self.abortOperation()
141         return
142     # factorSoFar 為乘或除運算所得之暫存數值
143     operand = self.factorSoFar
144     self.factorSoFar = 0.0

```

圖 5.11: 5-

```

145         self.pendingMultiplicativeOperator = ''
146
147         # 若有等待加或減的運算子, 執行運算
148         if self.pendingAdditiveOperator:
149             if not self.calculate(operand, self.pendingAdditiveOperator):
150                 self.abortOperation()
151                 return
152
153         self.pendingAdditiveOperator = ''
154     else:
155         self.sumSoFar = operand
156         self.display.setText(str(self.temp + self.sumSoFar))
157         self.sumSoFar = 0.0
158         self.waitingForOperand = True
159
160     def pointClicked(self):
161         """小數點按下後的處理方法"""
162         #pass
163         if self.wait:
164             self.display.setText('0')
165
166         if "." not in self.display.text():
167             self.display.setText(self.display.text() + ".")
168
169         self.wait = False
170     def changeSignClicked(self):
171         """變號鍵按下後的處理方法"""
172         text = self.display.text()
173         value = float(text)

```

圖 5.12: 6-

```

174
175 -     if value > 0.0:
176         text = "-" + text
177 -     elif value < 0.0:
178         text = text[1:]
179
180     self.display.setText(text)
181
182 - def backspaceClicked(self):
183     """回復鍵按下的處理方法"""
184     #pass
185     text = self.display.text()[:-1]
186 -     if not text:
187         text = '0'
188         self.waitingForOperand = True
189         self.display.clear()
190         self.wait = True
191
192     self.display.setText(text)
193
194 - def clear(self):
195     """清除鍵按下後的處理方法"""
196     #pass
197     self.wait = True
198     self.display.setText('0')
199
200 - def clearAll(self):
201     """全部清除鍵按下後的處理方法"""
202     #pass

```

圖 5.13: 7-

```

203     self.wait = True
204     self.temp = 0
205     self.display.setText('0')
206     self.sumSoFar = 0.0
207
208     def clearMemory(self):
209         """清除記憶體鍵按下後的處理方法"""
210         self.sumInMemory = 0.0
211
212     def readMemory(self):
213         """讀取記憶體鍵按下後的處理方法"""
214         self.display.setText(str(self.sumInMemory))
215         self.waitingForOperand = True
216
217     def setMemory(self):
218         """設定記憶體鍵按下後的處理方法"""
219         self.equalClicked()
220         self.sumInMemory = float(self.display.text())
221
222     def addToMemory(self):
223         """放到記憶體鍵按下後的處理方法"""
224         self.equalClicked()
225         self.sumInMemory += float(self.display.text())
226
227     def createButton(self):
228         """ 建立按鍵處理方法, 以 Qt Designer 建立對話框時, 不需要此方法"""
229         pass
230
231     def abortOperation(self):

```

圖 5.14: 8-

```

231 - def abortOperation(self):
232 -     """中斷運算"""
233 -     pass
234 -
235 - def calculate(self, rightOperand, pendingOperator):
236 -     if pendingOperator == "+":
237 -         self.sumSoFar += rightOperand
238 -
239 -     elif pendingOperator == "-":
240 -         self.sumSoFar -= rightOperand
241 -
242 -     elif pendingOperator == "×":
243 -         self.factorSoFar *= rightOperand
244 -
245 -     elif pendingOperator == "÷":
246 -         if rightOperand == 0.0:
247 -             return False
248 -
249 -         self.factorSoFar /= rightOperand
250 -
251 -     return True
252 -

```

圖 5.15: 9-



圖 5.16: finish



## 第六章 心得

### 期末報告心得

#### 6.1 Fossil SCM

40623119 歐宗韋: 最開始老師講到 Fossil SCM 時, 我聽都聽不懂, 可是經過幾次上課以後我就慢慢了解怎麼使用, 只要有 USB 在哪裡都可以做, 不需用到網路就能工作, 是一件非常酷的事情, 雖然還不怎麼熟悉, 但是還是懂得裡面操縱內容。

40623121 蔡朝旭: Fossil SCM 可以說是讓我更快了解 github 指令的基石, 只要有隨身碟, 不管有沒有網路, 或是在哪裡的電腦都能查看. 上傳. 修改倉儲內容, 而且 Fossil SCM 還能使用 html 的語法來客製自己的倉儲使其改成自己的風格。

40623128 張華偉: fossil SCM 是我們在接觸 github 之前先接觸的網路倉儲, 這讓我了解到網路倉儲的方便性, 用別人電腦也可以更改倉儲資料, 還可以檢視我在何時改了什麼, 因此更加容易找到錯誤, 只要對照之前的就知道我哪裡多改了。

40623129 陳威誠: 雖然剛接觸 fossil SCM 覺得很陌生很困難, 現在我知道只要先將資料取出並存放至使用者的工作目錄, 等到完成了新增、修改等各種工作, 再將本地端的資料回存, 就可完成 fossil 的處理作業。

40623130 陳鉅忠: fossil SCM 是一開始接觸的課程倉儲, 跟雲端很像很方便, 一開始大家都不是很懂都必須要看著影片一步一步做, 在學會一些指令以後才比較了解, 而且在跟其他有學程式語言的朋友說 fossil 他們都不懂, 也滿開心的。

#### 6.2 網誌心得

40623119 歐宗韋: 小時候看著哥哥在用無名小站來記錄生活, 覺得非常的厲害, 但是到了大學我是利用網誌來記錄今天學到了什麼, 而且更新版本後想看以前的做了什麼事情還可以去 Timeline 看看舊的版本, 是一個非常好用的東西。

40623120 蔡宗穎: fossil 跟 github 的用法很像, 所以很快就學會了, 加上有查看 commit 這個功能, 所以可以很快地找到更改內容與衝突, 還能拉回以前的版本,

雖然語法花了些時間理解，但理解好後就覺得這是個很好用的東西。

40623121 蔡朝旭: 回想當年看著 facebook 的崛起，使人們逐漸將生活大小事以及想分享的東西放置在個人專屬的部落格或檔案裡，而我也是個富有想像力並且常常記錄學習到的知識的人，然而在一些因緣際會下我學會在網誌加入各種特效或是更改字體顏色之類的語法，雖然還有些不足，但我會一一克服的。

40623128 張華偉: 這應該是我人生第一次寫網誌，記錄下每一堂課上了什麼，雖然一開始使用 Markdown 編寫時很多不懂，但經過很長時間的努力，我應該可以算是小高手，知道出問題如何解決，我花了很多時間在上網尋找 Markdown 的編寫語法，雖然還有很多不足，但我會再接再厲的。

40623129 陳威誠: 雖然在其他課程會用抄筆記的方式記錄上課內容，但是我還是第一次使用電腦紀錄今天上了什麼，雖然在製作網誌的時候還要學習一些程式語法，但隨著多次練習認識的程式語言也漸漸增加慢慢熟悉了。

40623130 陳鉅忠: 雖然網誌不是第一次打，但是像這樣需要打那麼正經又充滿專業術語的還是第一次，基本上每周的都會有更新。

### 6.3 Github 協同倉儲

40623119 歐宗章: 這是我第一次與同學一起利用協同倉儲來完成作業，雖然摸索的很久，但是協同倉儲是一個很好用的東西，能夠與同學共同討論問題並修改。

40623121 蔡朝旭: github，是一個能讓人與人之間共享或是共同完成一項要開發的東西的地方，雖然一開始就遇到同時上船導致一些衝突的問題，但是組員們都會一起去爭鮮吃飯並且討論. 規劃和互相勉勵，所以我們組比其他組做的進度快了許多，而在 github 上面也有很多不同人創作. 開發很多東西，可以讓我學習到更多我想學到的知識。

40623128 張華偉: github，我們協同作業的開始，一開始使用 github 時，時常打成 fossil 的指令，經過了一段時間的適應後，我才不會一直打成 fossil 的指令，用 github 來協同作業真的很方便，可以看誰做了什麼，誰是豬隊友什麼都沒做，誰亂改程式碼，真的很方便，讓被抓到的人百口莫辯，上面寫的一清二楚，刪了什麼，加了什麼，一看就知道，我相信 github 可以幫我很多，上面也有很多別人協同製作的程式，可以幫助我自我學習。

40623129 陳威誠: github 是個能讓許多人一起完成協同開發同一項程式，就算許多人一起製作過程中有錯誤，github 協同作業也能抓出做錯的那個人是誰，也能知道自己的錯誤在哪，讓我們一起討論錯誤，漸漸地就能把一個城市變得更完整。

40623130 陳鉅忠: github 也很方便而且比 fossil 的倉儲功能多了一些，規模也比較大，雖然很容易發生衝突但通過溝通討論我們很可以地把老師要我們完成的工作在時間內完成。

## 6.4 學員心得

40623119 歐宗韋: 上完這學期的課程後，我學到了很多，而且跟小組成員們一起完成計算機程式，小組加減乘除有問題時我們一起討論與修改，完成後蠻有成就感的，有問題成員們會一起討論這一點我覺得非常的好，不只可以增長知識，也增進組員的感情。

40623120 蔡宗穎: 我是組別裡面比較不擅長的人，所以常常要請教我們組比較會的人，幸好大家都很樂於教導不會不耐煩，讓我更快了解課程的精隨。

40623121 蔡朝旭: 我負責的是語法. 簡介以及開發計算機程式碼，因為我也是從頭開始學的，所以有時候都要自己理解或是問別人，然後我對資訊本來就有一些興趣，所以學習起來也不會乏味，加上教導的教授又是個英文好長得帥脾氣又好的人，讓我更有興趣去鑽研這堂課。

40623128 張華偉: 我在製作 pdf 時我負責的任務是編輯一部分資料與教導組員如何使用 Markdown 編輯 pdf，因為有很多東西不懂，我花了一點時間上網解決問題，我發現使用 Markdown 編輯 pdf 有個很不方便的事情，圖片沒辦法調整大小，只能去調整原圖的大小，真的很麻煩，希望如果有辦法調整的話可以教教我。

40623129 陳威誠: 因為我學習的速度跟組員們相比落後的很多，所以在開發計算機程式的時候也只能做比較簡單的工作，但是組裡的人在製作的過程我都會去旁邊看慢慢學習，希望日後我能跟上他們，在一起做出更棒的程式。

40623130 陳鉅忠: 製作需要分組協同的計算機時很有趣, 或許有人會懶惰不想做, 或許有人會在分配時起爭議, 在最後還是順利的完成了, 但這只是一個開始只是一個里程碑, 即使這學期我的學習態度有點虎頭蛇尾的感覺而我們只不過是接觸了 Python 的一點點小角連 c 程式語言都沒摸到, 還在一年級, 要高興還太早了, 希望

能在老師以後的課上學到許多而不是在還沒來不及吸收的完就準備離開了。

## 第七章 結論

### 期末報告結論

#### 7.1 結論與建議

結論: 老師非常專業, 英文講得流利, 富有國際觀, 又會在課堂中講解人生的道理, 讓人不會討厭早八並且非常專心的上課, 在這個 **international** 而且資訊化的社會, 勢必會轉變現代的產業類型, 這堂課剛好是上資訊類型的東西, 對於出社會有非常大的幫助。

建議: 拍課程影片是個很不錯的方法, 只是如果把進度放慢些會更好。

## 第八章 參考文獻

<https://marco79423.net/articles/%E6%B7%BA%E8%AB%87-python-%E7%9A%84-for-%E8%BF%B4%E5%9C%88/>

[http://www.tablesgenerator.com/markdown\\_tables](http://www.tablesgenerator.com/markdown_tables)

<https://zh.wikipedia.org/wiki/Wikipedia:%E9%A6%96%E9%A1%B5>

<https://ithelp.ithome.com.tw/articles/10158587>

<https://docs.python.org/2.4/lib/typeseq.html>

<http://www.runoob.com/python/python-func-str.html>

<https://www.programiz.com/python-programming/methods/built-in/str>

<http://python.ez2learn.com/basic/unicode.html>

[https://www.uuu.com.tw/Public/content/Edm/171115\\_brochure/pdf/Python.pdf](https://www.uuu.com.tw/Public/content/Edm/171115_brochure/pdf/Python.pdf)

[https://yungyuc.github.io/oldtech/python/python\\_intro.html](https://yungyuc.github.io/oldtech/python/python_intro.html)

<https://kaochenlong.com/2011/10/12/python-introduction/>