

國立虎尾科技大學

機械設計工程系

計算機程式 bg5 期末報告

PyQt5 事件導向計算器

PyQt5 Event-Driven Calculator Project

學生：

設計一乙 40623222 蔡博淮 (內文-可攜程式系統介紹、Python 程式語法)

設計一乙 40623223 任明彥 (摘要 fossil、github 倉儲)

設計一乙 40623224 鐘偉哲 (內文-Calculator 程式)

設計一乙 40623231 周駿麟 (內文-PyQt5 簡介)

設計一乙 40623232 余建杰 (前言-討論與建議)

設計一乙 40623233 謝宗宏 (結論、討論與建議)

指導教授：嚴家銘

2018.01.08

目錄

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
2-2 啟動與關閉-2	4
第三章 Python 程式語法	6
3-1 變數命名	6
3-2 prin 函式	7
3-3 重複迴圈	8
3-4 判斷式	9
3-5 數列	10
第四章 倉儲系統	11
4-1 Fossil SCM	11
4-2 Github 協同	14

第五章 Calculator 程式	18
5-1 計算機製作	18
5-2 計算機的命名	20
第六章 結論與建議	29

第一章 前言

本次報告的目的在於統整這學期的課程內容並加以檢視及複習，參照上課的日期依序介紹各教學內容：可攜程式系統介紹、Python 程式語法、fossil 網誌、github 倉儲、PyQt5 簡介、Calculator 程式。為多加練習多人協同合作之形式，本報告由各組學員分段同時編寫，以利未來行事之效率。根據結果，多人協同有助於減少個人作業的壓力並能與協同者有較多的交流，有利於討論及改進錯誤，使時間能在有效的運用下縮短整體作業時間。最後，歸納報告之結論，並整合各學員討論及建議，做為後續相關課程的參考資料。

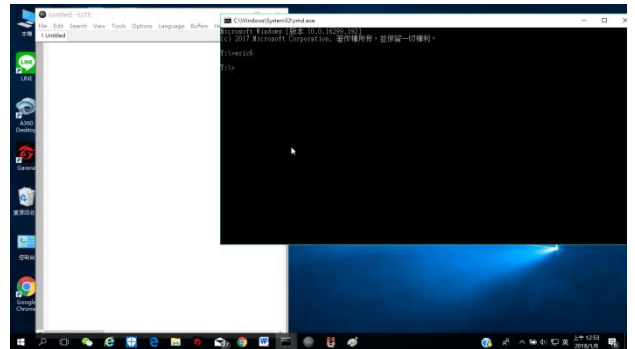
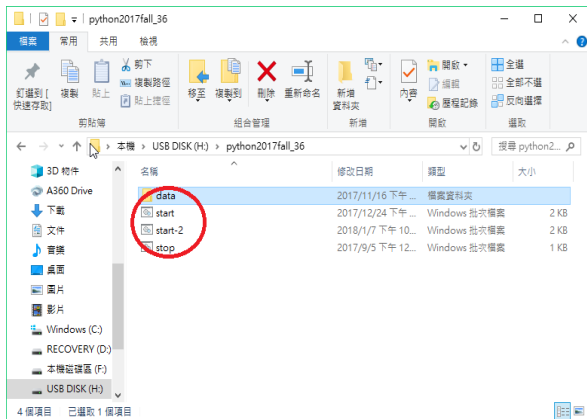
關鍵詞：檢視及複習、多人協同、行事之效率、討論及改進錯誤。

第二章 可攜程式系統介紹

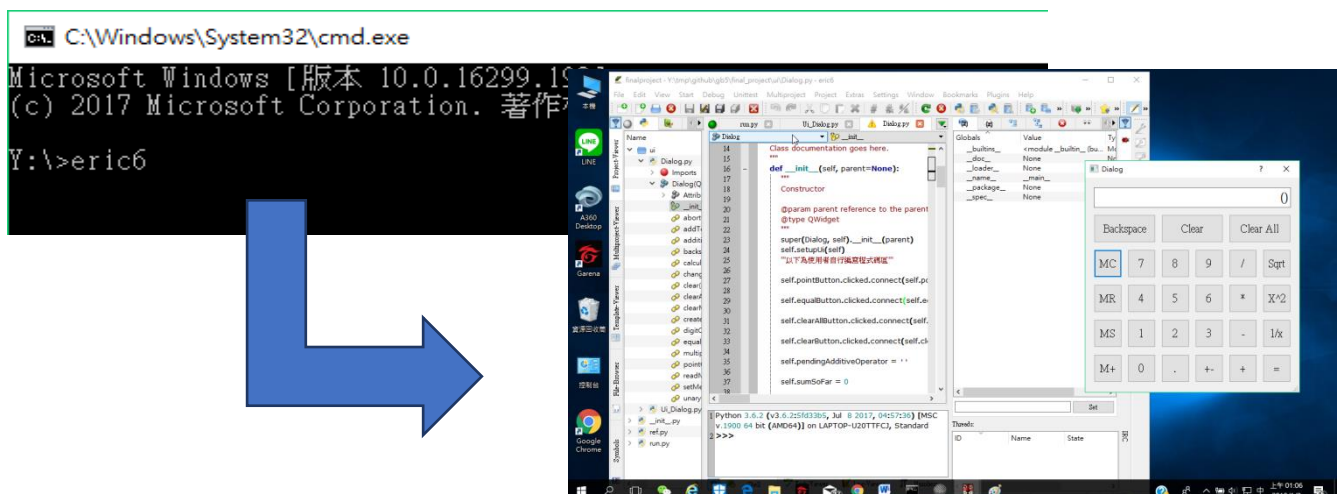
2-1 啟動與關閉

1. 啟動

按下在 python2017fall_36 裡的 star.bat，使用它來開啟小黑盒及 SciTE，來繼續後續的工作處理。

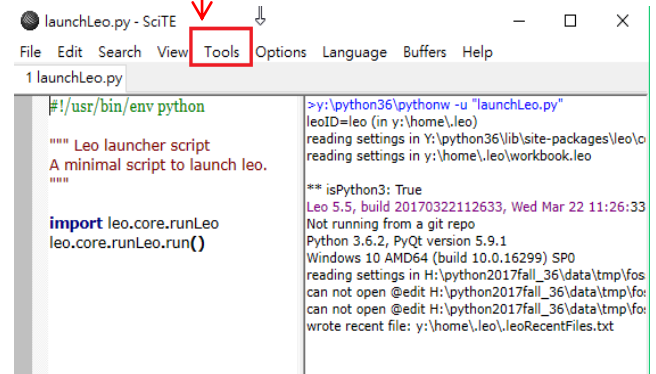
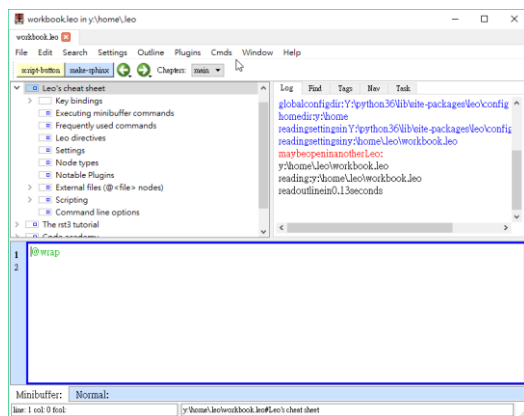


(1) 利用小黑盒叫出 eric6 來做計算機程式的編寫

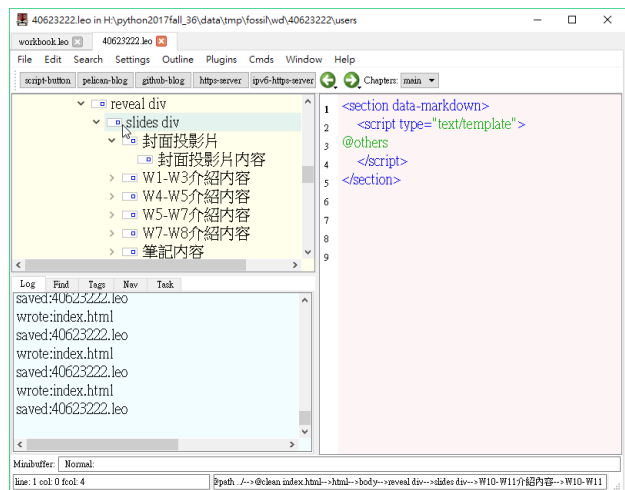


(2) 利用 SciTE 來開啟 launchLeo，但開啟時需再把自己的檔案打開。

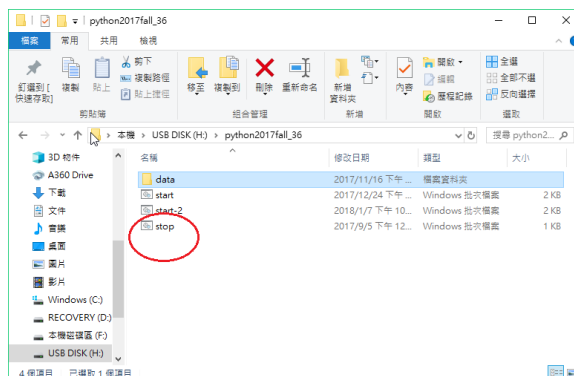
按下 tools 來驅動



打開自己的 leo 檔案

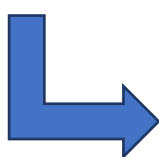


(3) 關閉時按下 python2017fall_36 的 stop 鍵，停止所有用 python 來開的程式。



2-2 啟動與關閉-2


1. 為了現代客製化的趨勢，我們把開啟時的步驟縮減一些，並且能達到快速的使用。而縮減步驟的部分為 launchLeo 的開啟。



```
start /MIN %Disk%\wsuite\SciTE.exe
start /MIN cmd.exe
start /MIN cmd.exe
start /MIN cmd.exe
REM 啟動 Leo 編輯器
REM %Disk%:\Miniconda3\python.exe %Disk%\apps\launchLeo.py
REM 啟動 stunnel
REM start /MIN fossil.exe server -P 127.0.0.1:8080 %Disk%\tmp\fossil_repo
REM start /MIN stunnel.exe
Exit

start /MIN %Disk%\wsuite\SciTE.exe
start /MIN cmd.exe
start /MIN cmd.exe
start /MIN cmd.exe
REM 啟動 Leo 編輯器
%Disk%:\Miniconda3\python.exe %Disk%\apps\launchLeo-2.py
REM 啟動 stunnel
REM start /MIN fossil.exe server -P 127.0.0.1:8080 %Disk%\tmp\fossil_repo
REM start /MIN stunnel.exe
Exit
```

2. 要讓程式知道我們要開的是哪個檔案，所以必須要在 launchLeo.py 中改寫開的檔案。



```
#!/usr/bin/env python

""" Leo launcher script
A minimal script to launch leo.
"""

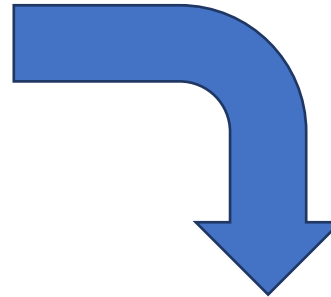
import leo.core.runLeo
leo.core.runLeo.run()

import leo.core.runLeo
leo.core.runLeo.run(fileName="y:/tmp/fossil/wd/40623222/user")
```

把節點設定在自己的檔名

3. 關閉時為了要將所有的程式關閉，所以也必須改寫。

```
@echo off
set Disk=y
REM 關閉 SciTE
taskkill /IM SciTE.exe /F
REM 關閉 python
taskkill /IM python.exe /F
taskkill /IM pythonw.exe /F
REM 關閉 stunnel
taskkill /IM stunnel.exe /F
REM 關閉 fossil
taskkill /IM fossil.exe /F
REM 清除 log 資料
path=%PATH%;
REM del /Q /F V:\tmp\*.
REM 終止虛擬硬碟與目錄的對應
subst %Disk%: /D
REM 關閉 cmd 指令視窗
taskkill /IM cmd.exe /F
REM taskkill /IM mingw32 /F
EXIT
```



```
@echo off
set Disk=y\
taskkill /launchLeo-2.py_
REM 關閉 SciTE
taskkill /IM SciTE.exe /F
REM 關閉 python
taskkill /IM python.exe /F
taskkill /IM pythonw.exe /F
REM 關閉 stunnel
taskkill /IM stunnel.exe /F
REM 關閉 fossil
taskkill /IM fossil.exe /F
REM 清除 log 資料
path=%PATH%;
REM del /Q /F V:\tmp\*.
REM 終止虛擬硬碟與目錄的對應
subst %Disk%: /D
REM 關閉 cmd 指令視窗
taskkill /IM cmd.exe /F
REM taskkill /IM mingw32 /F
EXIT
```


第三章 Python 程式語法

3-1 命名

1. 通用命名規則：

函式命名，變數命名，文件命名要有描述性；少用縮寫，盡可能地使用描述性的字元來呈現，方便其他人來觀看，以及易於檢查。

2. 文件命名：

文件名要全部小寫，可以包含底線（`_`）或連字符（`-`）。如果是有專案約定的情況，則照專案約定來使用，如果沒有專案約定（`_`）會更好。

3. 變數命名：

變數名一律小寫，單詞之間用底線連接。類別的成員變量以下劃線結尾，但結構體的就不用

3-2 print 函式

函式為結構化程式，將相同功能的程式獨立出來，經由函式的呼叫，傳入資料與回傳處理後的結果，只要將函式寫好，可以不斷利用此函式做相同動作，可以達成程式碼不重複，要修改此功能，只要更改此函式。

函式的定義：

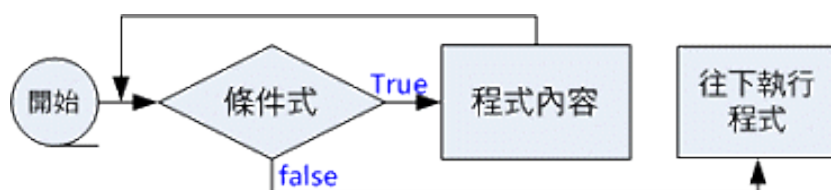
以 def 開頭，空一個空白字元，接函式名稱後，串接著一對小括號，小括號可以填入要傳入函式的參數，當參數有多個的時候以逗號隔開，右小括號後面須接上「:」，函式範圍為縮行固定個數空白字元的程式碼，縮行相同空白字元的程式碼就是函式的作用範圍。當函式需要傳回值使用指令 return，表示函式回傳資料給原呼叫函式，若不需要回傳值的函式就不需要加上 return。

```
def abortOperation(self):  
    """中斷運算"""  
    #pass  
    self.clearAll()  
    self.display.setText("fuck")  
def calculate(self, rightOperand, pendingOperator):  
    """計算"""  
    #pass  
    if pendingOperator == "+":  
        self.sumSoFar += rightOperand  
    elif pendingOperator == "-":  
        self.sumSoFar -= rightOperand  
    elif pendingOperator == "*":  
        self.factorSoFar *= rightOperand  
    elif pendingOperator == "/":  
        if rightOperand == 0.0:  
            return False  
        self.factorSoFar /= rightOperand  
    return True
```

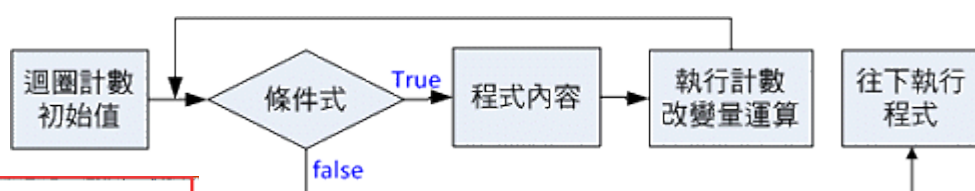
3-3 重複迴圈

1. 處理重複事件的敘述稱為迴圈，有[while]、[for]、[do...while]。
2. [for]是屬於固定次數的迴圈，[while]為不固定次數的迴圈。
3. [for]、[while]為測試在執行，屬於前測試迴圈；[do...while]為執行後在測試，屬於後測試迴圈。

While 迴圈：



For 迴圈：



```
1 #ex1 簡單的 for 迴圈範例
2 class w8():
3
4     def __init__(self, star):
5         self.star=star
6
7     def diamond(self,w):
8         for i in range(1,w):
9             print((w-i)*" "+i*self.star)
10        for i in range(w):
11            print(i*" " +(w-i)*self.star)
12 w=w8("ab")
13
14 w.diamond(5)
15
16
```

Filename: input file name .py Save

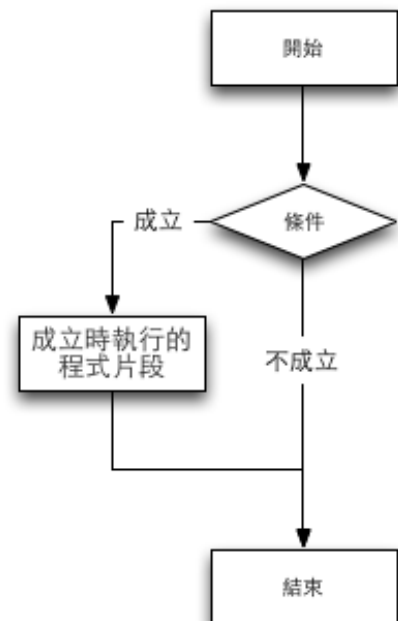
Run Output 清除

```
ab
abab
ababab
abababab
ababababab
ababababab
ababab
abab
ab
```

<completed in 6.00 ms>

3-4 判斷式

1. 定義：在程式中判別是否成立要或不成立的一種語法。



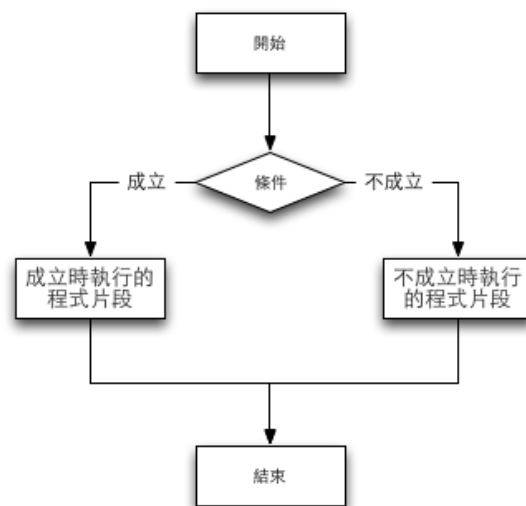
2. 判斷的條件：

比較運算 (A、B 比較)	語法
等於	$A==B$
不等於	$A!=B$
大於	$A>B$
小於	$A<B$
不大於	$A\leq B$
不小於	$A\geq B$

3. if-else 的使用

If 的條件:成立時通過成立時要跑的程式。

Else 的條件:不成立時通過不成立要跑的程式。



3-5. 數列

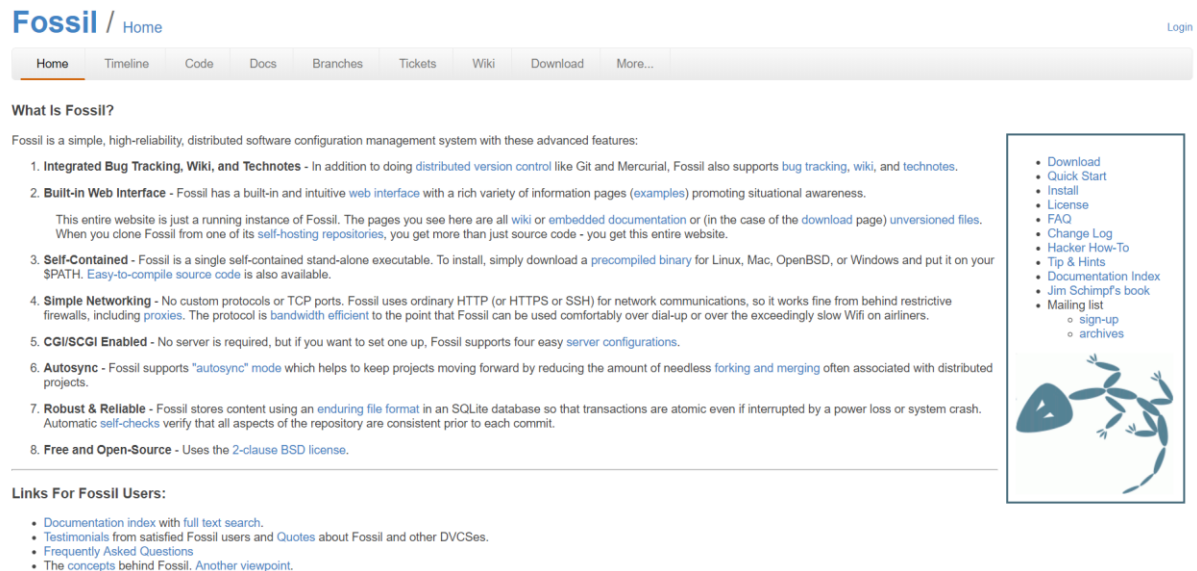
簡單的解釋就是說我把我所需要的資源全部放在一塊，然後讓我在後續的工作不需要再去一個一個找，而是直接重我所放資源的地方找。

```
digits = [self.one, self.two, self.three, \
self.four, self.five, self.six, \
self.seven, self.eight, self.nine, self.zero]
```

[] 中的部分就是數列

第四章 倉儲系統

4-1 Fossil SCM



The screenshot shows the Fossil SCM website. At the top, there's a navigation bar with links: Home, Timeline, Code, Docs, Branches, Tickets, Wiki, Download, and More... A 'Login' link is on the right. Below the navigation bar, the heading 'What Is Fossil?' is followed by a paragraph describing Fossil as a simple, high-reliability, distributed software configuration management system. This is followed by a list of 8 features: 1. Integrated Bug Tracking, Wiki, and Technotes; 2. Built-in Web Interface; 3. Self-Contained; 4. Simple Networking; 5. CGI/SCGI Enabled; 6. Autosync; 7. Robust & Reliable; 8. Free and Open-Source. To the right of the features list is a sidebar with a list of links: Download, Quick Start, Install, License, FAQ, Change Log, Hacker How-To, Tip & Hints, Documentation Index, Jim Schimpf's book, and a Mailing list with sub-links for sign-up and archives. Below the sidebar is a small illustration of a blue lizard.

Fossil scm(化石)是本學期為使學員熟悉管理版次及編寫 blog 所練習之程式。

以下為練習之過程：

1. 在 cmd(提示指令字元)的 fossil 目錄下 clone 線上倉儲網址，並在 https://後輸入綁定帳號之身分以利於之後推送。

```
Y:\tmp\fossil\wd>cd ..  
Y:\tmp>fossil clone https://40623223@cpb.kmol.info/40623223_40623223.fossil
```

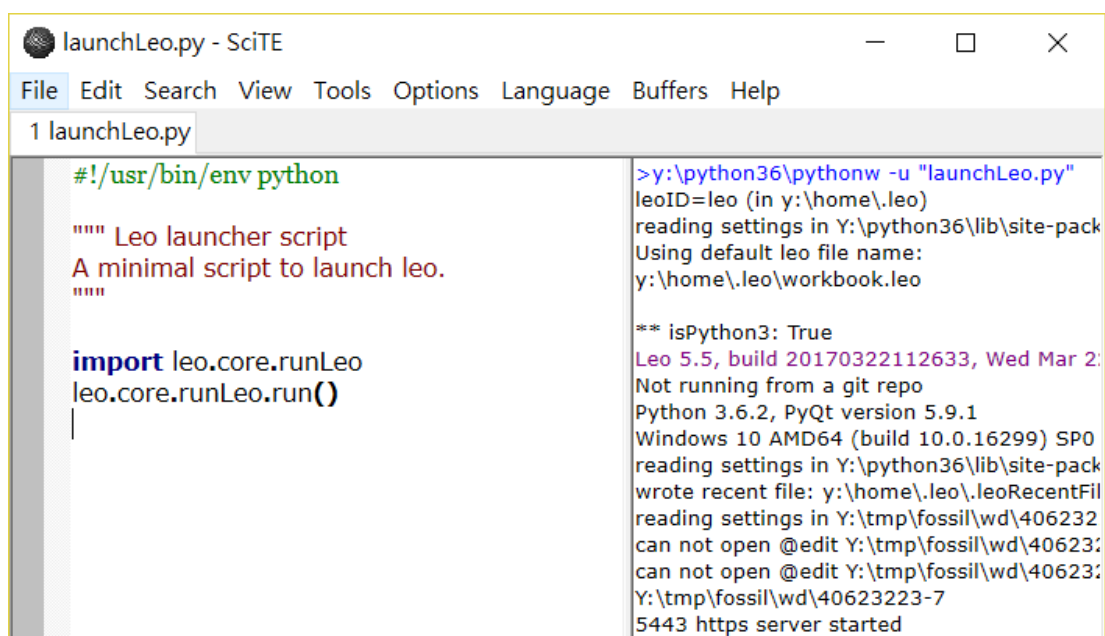
2. clone 完後興建一個 wd(工作目錄)以方便管理資料。

```
Y:\tmp\fossil>mkdir wd
```

3. 在 wd 下興建一個自行設定的子目錄，並於子目錄 open 相對於 fossil 目錄的封包檔。

```
Y:\tmp\fossil\wd\40623223>fossil open ../../40623223.fossil
```

4. 在 sciTE 中開啟於 Y:目錄下的 launchLeo 啟動 leo。



The screenshot shows the SciTE editor window titled "launchLeo.py - SciTE". The menu bar includes File, Edit, Search, View, Tools, Options, Language, Buffers, and Help. The file list shows "1 launchLeo.py". The editor contains a Python script for launching Leo. The output pane on the right shows the execution results.

```
#!/usr/bin/env python

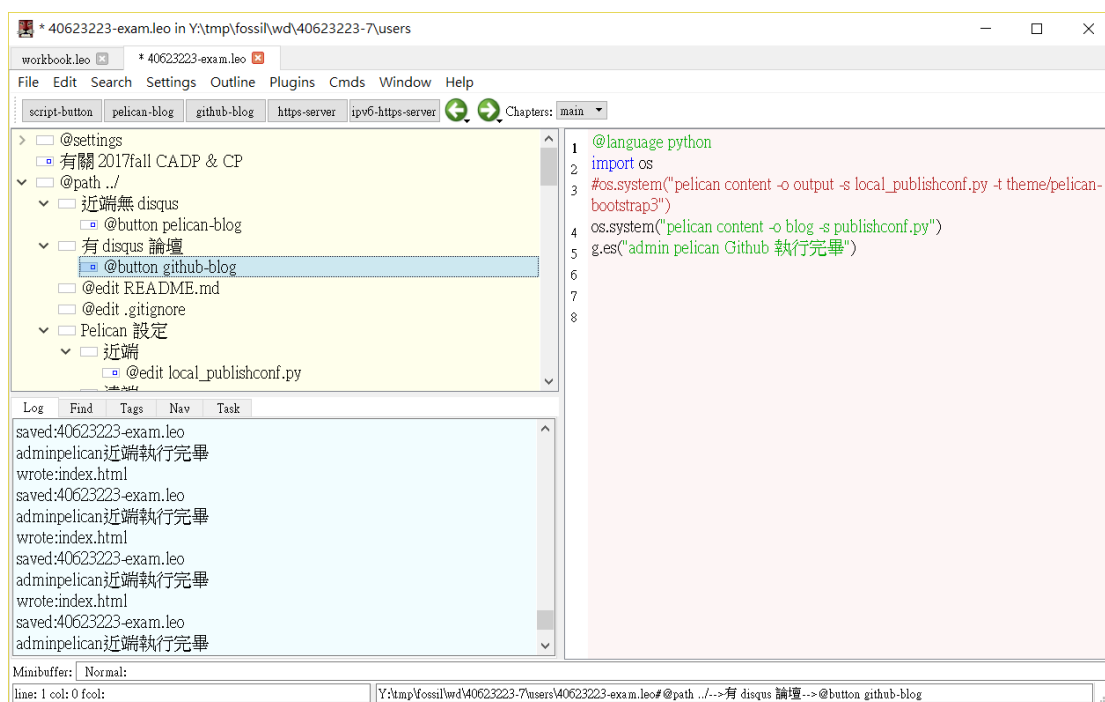
""" Leo launcher script
A minimal script to launch leo.
"""

import leo.core.runLeo
leo.core.runLeo.run()
```

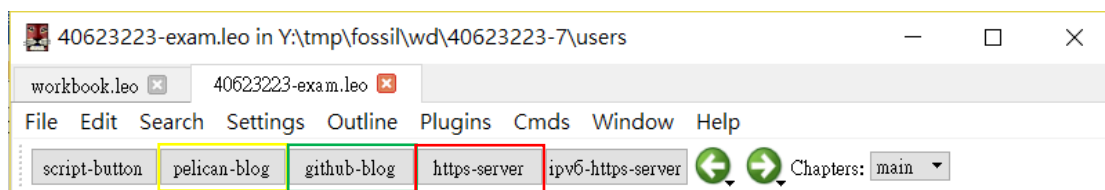
```
>y:\python36\pythonw -u "launchLeo.py"
leoID=leo (in y:\home\leo)
reading settings in Y:\python36\lib\site-pack
Using default leo file name:
y:\home\leo\workbook.leo

** isPython3: True
Leo 5.5, build 20170322112633, Wed Mar 2
Not running from a git repo
Python 3.6.2, PyQt version 5.9.1
Windows 10 AMD64 (build 10.0.16299) SP0
reading settings in Y:\python36\lib\site-pack
wrote recent file: y:\home\leo\leoRecentFil
reading settings in Y:\tmp\fossil\wd\406232
can not open @edit Y:\tmp\fossil\wd\40623:
can not open @edit Y:\tmp\fossil\wd\40623:
Y:\tmp\fossil\wd\40623223-7
5443 https server started
```

5. 開啟位於 user 的 .leo 檔以編寫 blog



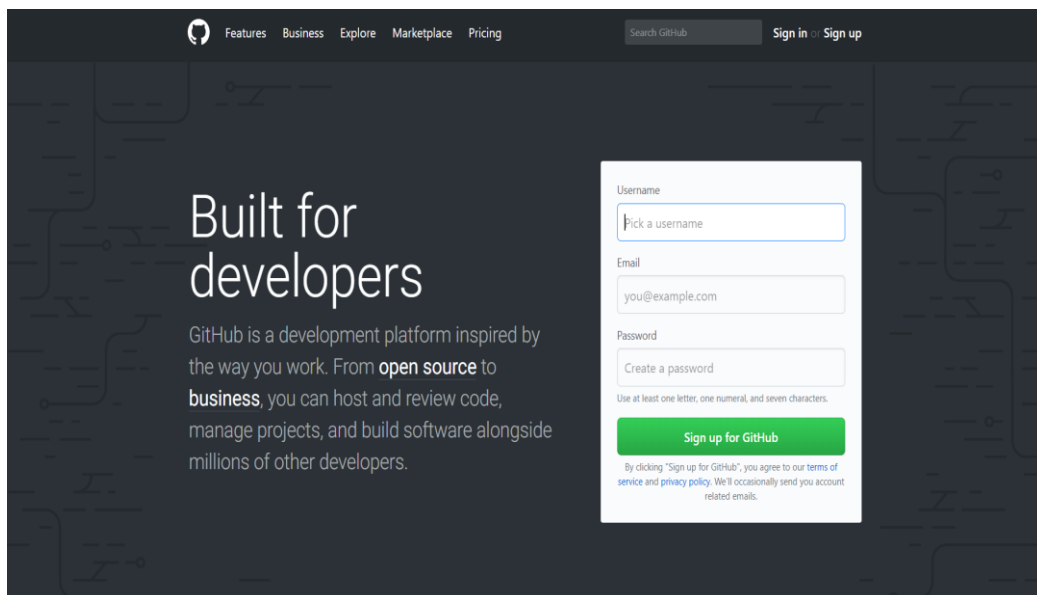
6. 利用 `https-server` 按鈕啟動近端檢視，並使用 `pelican-blog` 按鈕將所編寫之文字轉換成近端程式碼，檢查無誤後再用 `github-sever` 按鈕轉換成遠端格式



7. 最後再用 `fossil add .` 及 `fossil commit -m ""` 指令將完成編寫之 blog 推送。

```
Y:\tmp\fossil\wd\40623223>fossil add .  
Y:\tmp\fossil\wd\40623223>fossil commit -m ""
```

4-2 Github 協同



Github 為現在業界普遍的程式協同倉儲，課程中為學員練習協同計算機程式與報告，幫助各組同學熟悉未來開發程式或課程會遇到的作業方式。

以下為練習過程：

1. 建立一個新的專案並參照格式設定

Owner: 40623223 / Repository name: calculator

Great repository names are short and memorable. Need inspiration? How about **redesigned-rotary-phone**.

Description (optional)

☐ **Public**
Anyone can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **Python** | Add a license: **GNU General Public License v3.0**

Create repository

2. 於 Branch 分支中新增 **gh-pegas**

1 commit | 1 branch | 0 releases | 1 contributor | GPL-3.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

Switch branches/tags

gh-pegas

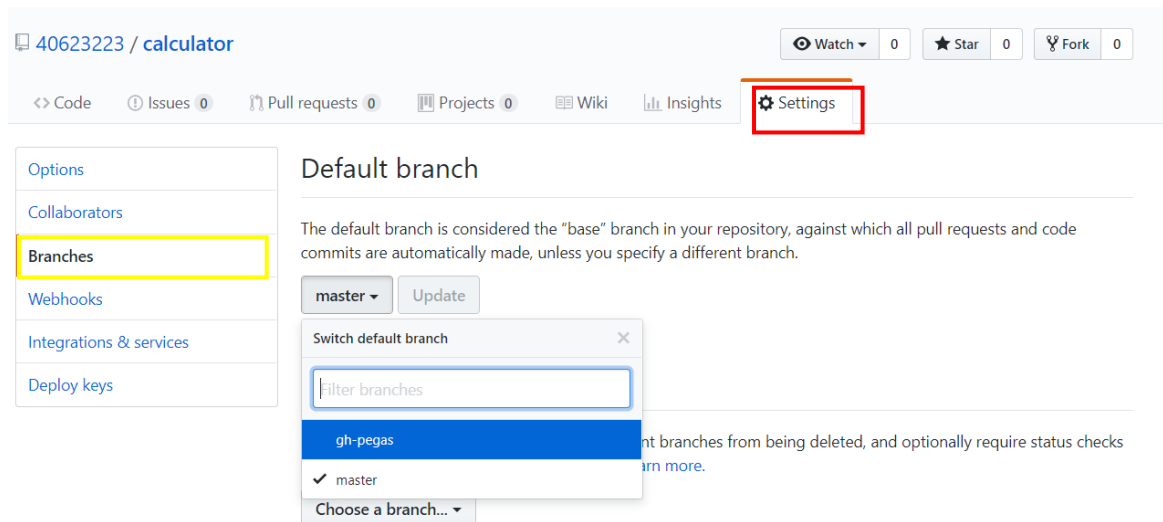
Branches | Tags

Create branch: gh-pegas
from 'master'

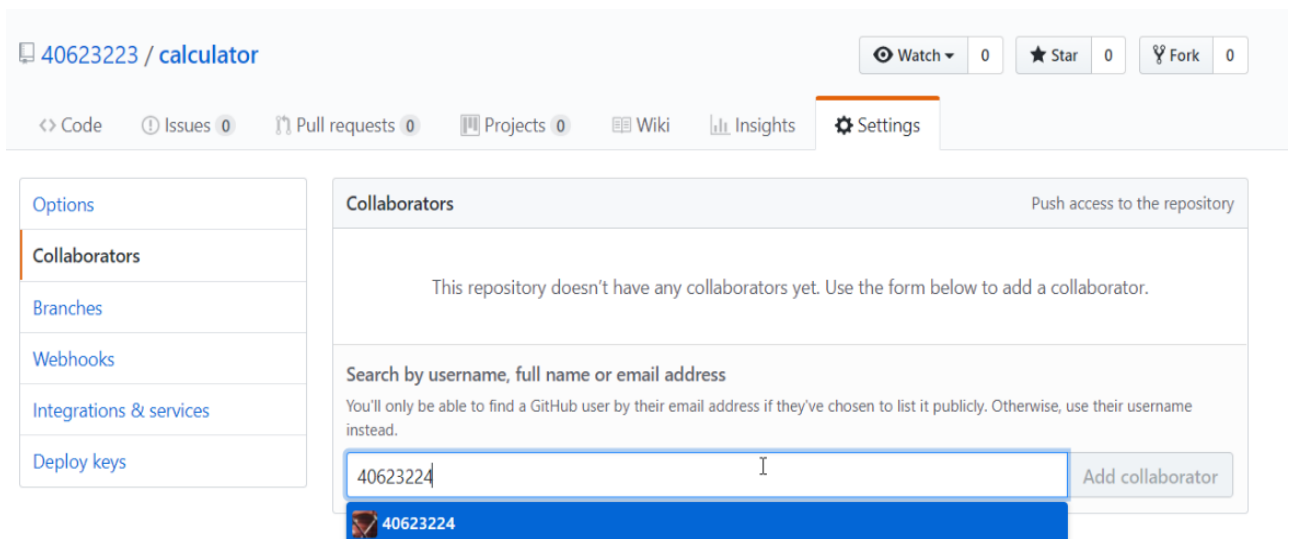
README.md

Latest commit e7a5803 just now	
Initial commit	just now
Initial commit	just now
Initial commit	just now

3. 進入 **Settings** 中的 **Branches** 將 master 修改成 gh-pegas



4. 進入 Collaborators 邀請協同之學員



5. 完成分配之作業如要推送需先設定帳號 email

```
y:\tmp\github>git config --global user.name "40623223"  
y:\tmp\github>git config --global user.email "40623223@gm.nfu.edu.tw"
```

6. 依序使用以下指令確認無誤後方可完成推送

```
y:\tmp\github>git add .  
  
y:\tmp\github>git commit -m ""  
  
y:\tmp\github>git push
```

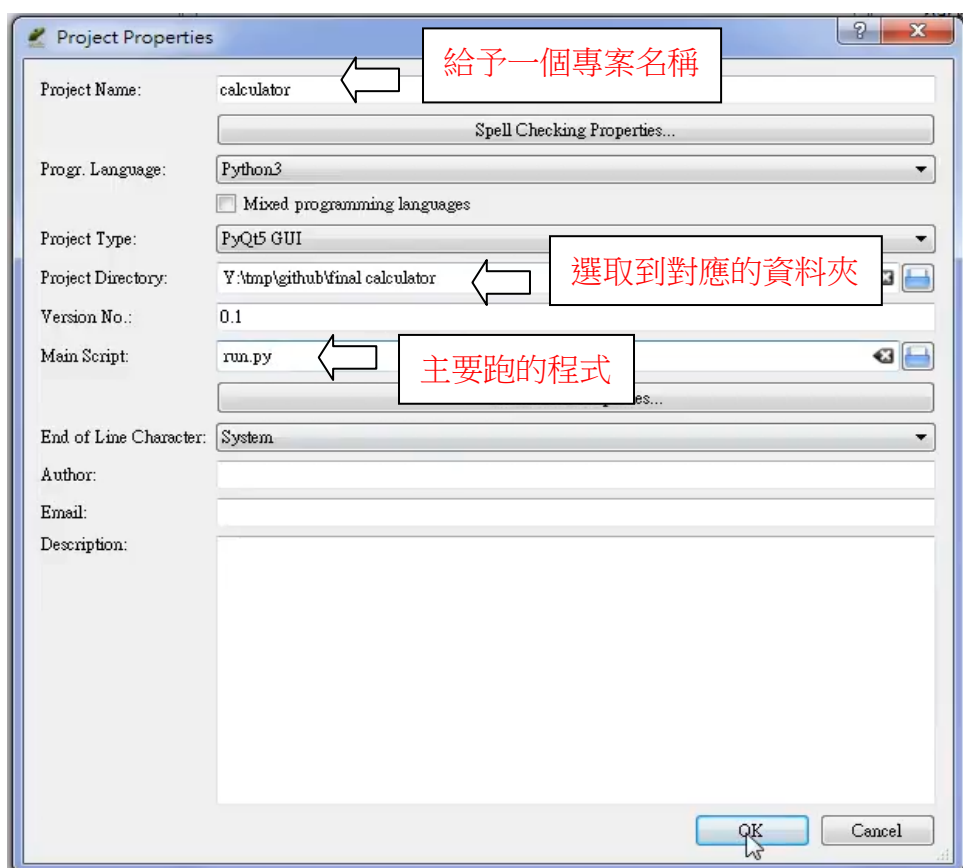
第五章 Calculator

5-1 計算機的製作

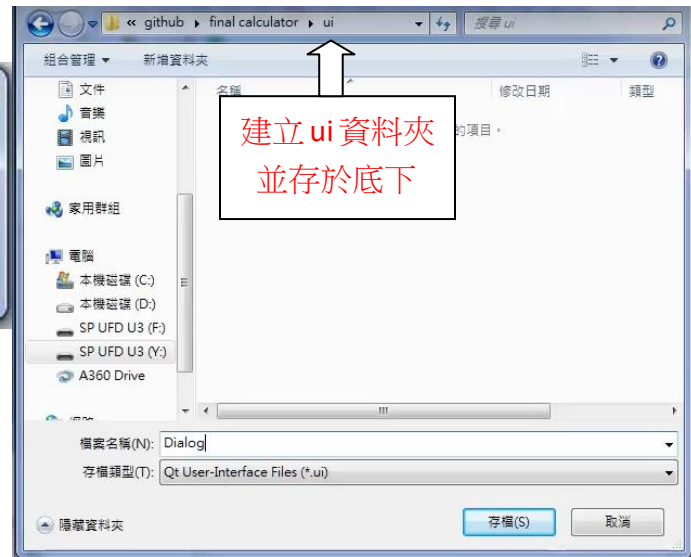
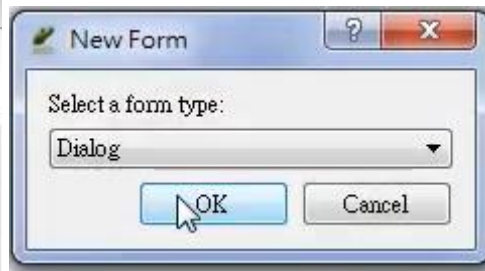
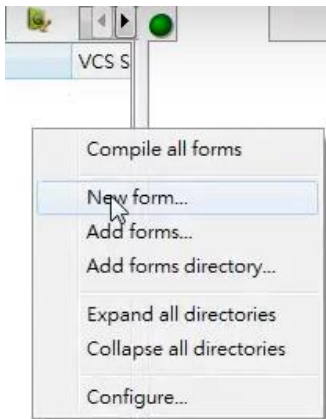
1. 先開啟 eric6



2. 新增一個專案，完成後確認



3. 建立一個表單



4. 用 Push Button 以及 Line Edit 完成計算機表單並修改主詞和圖片大小



修改主詞

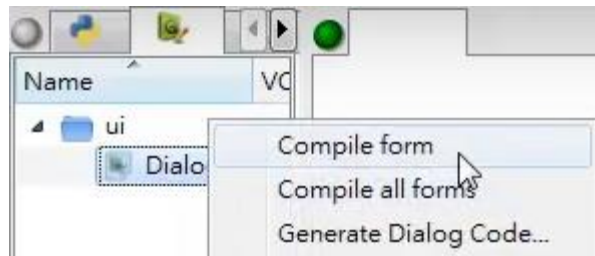
表單名稱為 Dialog

按鈕的長寬尺寸變更

minimumSize	84 x 54
Width	84
Height	54



5. 完成後把圖片的東西編寫上去



6. 在 run.py 的部分把程式撰寫上去



5-2 計算機的命名

1. 數字按鍵以 zero、one、two、three、four、five、six、seven、eight、nine 命名、顯示幕以 display 命名、等號以 equalButton 命名。

5-3 計算機邏輯

1. 與 MS, M+, 或 MC 按鍵相關的計算機記憶體數值, 存入 sumInMemory 變數對應的記憶空間, 以 sumSoFar 儲存累計數值, 使用者按下等號後, sumSoFar 重新計算結果, 並顯示在 display 幕, Clear All 按鍵則重置 sumSoFar 為 0。

以 factorSoFar 儲存乘或除運算子運算過程所得的暫存數值; pendingAdditiveOperator 儲存使用者最後點按的加或減運算子字串; pendingMultiplicativeOperator 儲存使用者最後點按的乘或除運算子字串; waitingForOperand 界定使用者是否處理運算數輸入階段, 若 waitingForOperand 為 True, 表示計算機正在等待使用者"開始"輸入運算數, waitingForOperand 起始值為 True, 只有重新進入 digitClicked 方法, display 才會 clear(), 否則在顯

示幕非為 0 的情況下，將堆疊數字字串。

因為考量先乘除後加減運算順序，將加減與乘除處理方法分開，若使用者輸入加減運算子後，緊接乘除運算子，計算機會先乘除運算後再加減；直接運算元可以在使用者按下按鍵後，直接對 display 中的數值進行處理，無需其他暫存需求；pendingAdditiveOperator、pendingMultiplicativeOperator、sumSoFar、factorSoFar 與 waitingForOperand，在 Dialog 類別建構子中設定起始值。

2. 數字按鍵點按處理：

使用者點按數字按鍵，將會送出該按鍵 clicked() 訊號，按鍵的 clicked() 訊號將會根據設定，觸發 digitClicked() 方法槽，由於 PyQt5 的 QPushButton 以 Qt5 中的 QObject::sender() 送出訊號，此函式會傳回 sender 作為 QObject 的指標，因為此一與 Push Button

配合的 sender 為 Button 物件，因此可以在 digitClicked() 函式中，利 sender().text() 取得按鍵的 text 字串，假如使用者點按 0，display 顯示字串 0，但是若一開始輸入兩個以上的 0，digitClicked() 應該仍只顯示 0 字串，但是若計算機處於等待新運算數輸入時（以 waitingForOperand 判定），新數字在顯示前，display 應該要清除先前所顯示的數字，最後，除了在顯示幕為 0 之後的 0 按鍵輸入，digitClicked() 方法槽不會繼續判定是否清除顯示幕或堆疊數字字串外，所按的數字將會堆疊顯示。

3. 直接運算按鍵處理：

Sqrt, x^2 與 $1/x$ 等按鍵的處理方法為 unaryOperatorClicked()，與數字按鍵的點按回應相同，透過 sender().text() 取得按鍵上的 text 字串；unaryOperatorClicked() 方法隨後根據 text 判定運算

子後，利用 display 上的運算數進行運算後，再將結果顯示在 display 顯示幕，若進行運算 Sqrt 求數值的平方根時，顯示幕中為負值，或 $1/x$ 運算時， x 為 0，都視為無法處理的情況，以呼叫 abortOperation() 處理；abortOperation() 方法則重置所有起始變數，並在 display 中顯示 "####"，直接運算子處理結束前，運算結果會顯示在 display 中，而且運算至此告一段落，計算機狀態應該要回復到等待新運算數的階段，因此 waitingForOperand 要重置為 True。

4. 加或減按鍵處理：

使用者按下加或減運算子按鍵時，程式設定以 additiveOperatorClicked() 處理，進入 additiveOperatorClicked() 後，必須先查是否有尚未運算的乘或除運算子，因為必須先乘除後才能加減，先處理乘與除運算後，再處理加或減運算後，將 sumSoFar 顯示

在 display 後，必須重置 sumSoFar 為 0，表示運算告一段落。

5. 乘或除按鍵處理：

使用者按下乘或除運算子按鍵時，程式設定以 multiplicativeOperatorClicked() 處理，進入 multiplicativeOperatorClicked() 後，無需檢查是否有尚未運算的加或減運算子，因為乘除運算有優先權，先處理乘與除運算後，再處理加或減運算，將 sumSoFar 顯示在 display 後，必須重置 sumSoFar 為 0，表示運算告一段落。

6. 小數點按鍵處理：

使用者按下小數點按鍵後，以 pointClicked() 方法處理，直接在 display 字串中加上 "." 字串。

7. 數值變號按鍵處理：

使用者按下變號按鍵後，由 `changeSignClicked()` 處理，若顯示幕上為正值，則在 `display` 字串最前面，疊上 "-" 字串，假如顯示幕上為負值，則設法移除 `display` 上字串最前方的 "-" 字元。

8. 退格按鍵處理：

使用者按下退格按鍵後，由 `backspaceClicked()` 處理，這時可以利用 Python 字串數列中的 `[:-1]`，保留除了最後一個字元的字串，離開 `backspaceClicked()` 前，將顯示幕中原有字串的 `[:-1]` 字串，顯示在 `display` 上，若退格後 `display` 上為空字串，則顯示 0，並且將 `waitingForOperand` 起始設為 `True`，表示等待新運算數中。

9. 清除按鍵處理：

使用者按下 Clear 按鍵後，以 `clear()` 方法處理，進入函式後，將現有的運算數重置為 0，離開 `clear()` 前，將 `waitingForOperand` 起始設為 `True`，表示等待新運算數中，ClearAll 按鍵，則將所有變數全部重置為起始狀態。

10. 記憶體按鍵處理：

`clearMemory()` 方法與 "MC" 按鍵對應，清除記憶體中所存 `sumInMemory` 設為 0；`readMemory()` 方法與 "MR" 按鍵對應，功能為讀取記憶體中的數值，因此將 `sumInMemory` 顯示在 `display`，作為運算數；`setMemory()` 方法則與 "MS" 按鍵對應，功能為設定記憶體中的數值，因此取 `display` 中的數字，存入 `sumInMemory`；`addToMemory()` 方法與 "M+" 按鍵對應，功能為加上記憶體中的數值，因此將 `sumInMemory` 加上 `display` 中的數

值，因為 `setMemory()` 與 `addToMemory()` 方法，都需要取用 `display` 上的數值，因此必須先呼叫 `equalClicked()`，以更新 `sumSoFar` 與 `display` 上的數值。

11. `calculate()` 方法：

`calculate()` 方法中的運算，以 `rightOperand` 為右運算數當執行加或減運算時，左運算數為 `sumSoFar`，當執行乘或除運算時，左運算數為 `factorSoFar`，若運算過程出現除與 0 時，將會回傳 `False`。

完成後，計算機就能使用了~

第六章 結論與建議

有了這次的協同設計練習，讓我們體會到協同設計的重要性，在職場上不可能所有的東西都是你一個人自己做，一定是協同設計，每個人都負責不同的地方一點一點的拼湊出來的，至於為甚麼要用協同設計，主要的原因是因為人不可能都不休息和有問題都自己解決，有了這個協同設計當你在休息的時候換別人依序你所做的，有問題也可以提出來一起討論，以利提升行事的效率。