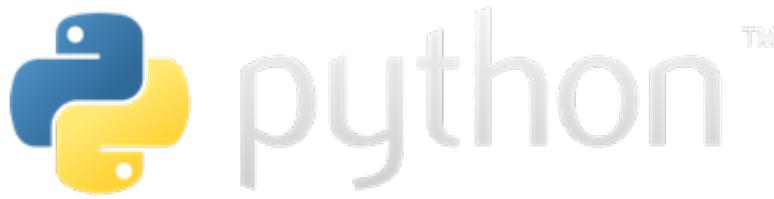




Assignment 0



This assignment is based on the Assignment 2, 3, and 4 of SC001 from stanCode

點此下載作業檔案

歡迎來到 SC101！進階班需要扎實基礎繼續往上蓋出程式巨塔，因此，這份作業將包含撰寫程式最重要的基本邏輯：靈活使用變數 (Variables)、使用者互動之對畫框 (Console)、各式迴圈 (Loops)及熟悉運算法則 (Expression)。

上過 SC001 的同學一定會覺得眼熟。沒錯！所有題目都改編自之前的工作/考試！本份作業的目的為提供同學一個基礎班重點的總複習，並讓您更了解基礎班內容的吸收程度，請盡量 **試著獨立完成** 每一題

如果卡關的話，請先試著從基礎班的上課範例 & 作業中複習並找到靈感。我們也知道，這份作業不容易，所以也非常歡迎在進階班開始前，來訊臉書粉專或來信 stancode.tw@gmail.com，會有**專人與您約時間線上討論**！任何問題都很歡迎，也請務必在開課前完成，會對進階班的課程很有幫助喔~

整份作業 請勿使用任何 Python 資料結構（如 list, dict, tuple, ...）

Problem 1 - class_reviews.py

請同學們幫忙 stanCode 寫出一個 SC001, SC101 期末評分計算程式！
(您可以假設使用者只會輸入 'SC001' 及 'SC101' 這兩種課程名稱
(string)，且大小寫對結果沒有影響)

首先，您的程式將先詢問課號（SC001 或是 SC101），接著詢問該課號所獲得之分數（一定是整數）。您的程式將分別計算每個課號所獲得之「最高分」、「最低分」以及「平均分數」。使用者將輸入「-1」來當作輸入結束的指示！若您程式撰寫正確，將可以重現下方截圖之內容：

```
Which class? Sc101
Score: 10
Which class? sc001
Score: 7
Which class? sC001
Score: 9
Which class? SC001
Score: 9
Which class? -1
=====SC001=====
Max (001): 9
Min (001): 7
Avg (001): 8.33333333333334
=====SC101=====
Max (101): 10
Min (101): 10
Avg (101): 10.0
```

眼尖的同學一定有注意到，SC101 在剛剛的例子中，只獲得了一筆評分（10 分）！這個時候，「最高分」、「最低分」以及「平均分數」都會是 10 分，惟平均分數的資料型態是 float，請同學們留意。
然而，若使用者輸入的課號只有 SC001，應在 SC101 的區間顯示「No score for SC101」，如下圖所示：

```
Which class? sc001
Score: 7
Which class? sc001
Score: 10
Which class? sc001
Score: 9
Which class? -1
=====SC001=====
Max (001): 10
Min (001): 7
Avg (001): 8.66666666666666
=====SC101=====
No score for SC101
```

同樣的，若使用者輸入的課號只有 SC101，會在 SC001 的區間顯示一樣的文字：

```
Which class? Sc101
Score: 10
Which class? sC101
Score: 9
Which class? SC101
Score: 9
Which class? sc101
Score: 10
Which class? -1
=====SC001=====
No score for SC001
=====SC101=====
Max (101): 10
Min (101): 9
Avg (101): 9.5
```

最後，如果使用者一開始便在輸入課號時輸入 -1，那我們就要印出「**No class scores were entered**」的字樣，如下圖所示：

```
Which class? -1
No class scores were entered
```

Problem 2 - coin_flip_runs.py

接下來的程式會讓使用者輸入一個正整數，代表當我們在玩用手指彈一枚硬幣的遊戲時，出現 **連續相同的正面(H)或反面(T)** 的次數。該程式會隨機選擇正、反面，並在達到該次數的時候終止程式，並將過程以一個串 string 呈現。

一開始會先出現一個歡迎標語：「**Let's flip a coin!**」。接著，使用者在看到 「**Number of runs:**」 的字樣後，輸入一個正整數（以下簡稱 num_run）。

```
Let's flip a coin!
Number of runs: 1
TT
```

若 num_run 的數值為 1，您的程式將在「連續出現 1 次的 'H' 或 'T' 時結束執行」，並將投擲過程印出！

請注意：整個過程是隨機的，每次產生的投擲過程很可能與我們的範例圖不一樣！

```
Let's flip a coin!
Number of runs: 1
HTHH
```

同樣的，若 num_run 的數值為 5，您的程式將在「連續出現 5 次的 'H' 或 'T' 時結束執行」，並將投擲過程印出！（下圖之紅線為解釋用標記，無需呈現在您的程式內）

```
Let's flip a coin!
Number of runs: 5
THHHHHHTTTHTHTHHTTTHTHTHH
  1 run   2 run   3 run   4 run   5 run
```

Problem 3 - blur.py

如果您尚未熟悉 simpleimage 的使用方法，請先觀看以下介紹再開始：

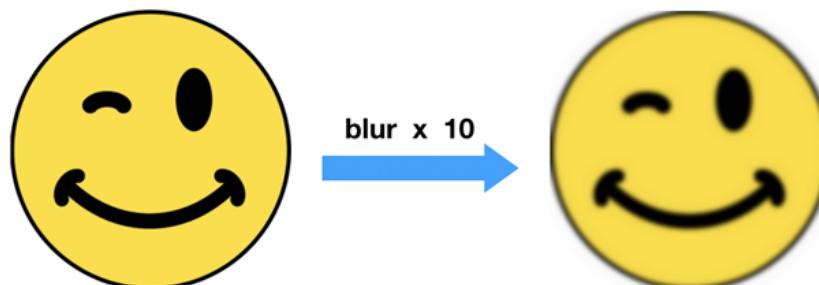
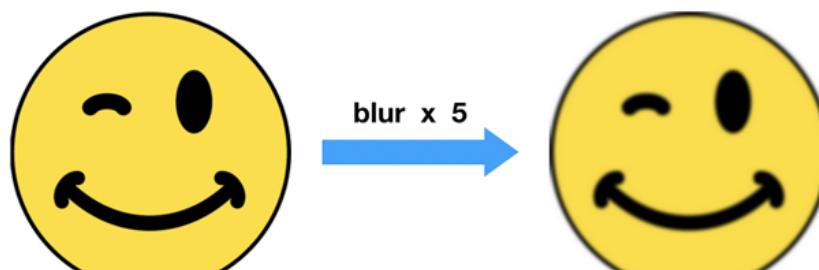
[銜接課程影片]

<https://youtu.be/SWT9biYKHqM>

[銜接課程講義]

<https://drive.google.com/file/d/1HM4udGkxSy5JXDxsWLU1U6wHn04tezJZ/view?usp=sharing>

第三題要請同學編輯 `def blur(img)` 並 `return` 一張將原圖 `img` 模糊處理的影像。我們使用的方法是將原本 `pixel` 數值改成此 `pixel` 與其身邊相鄰 `pixels` 之平均值。



	0	1	2	3	4
0	(14, 97, 63)	(84, 22, 99)	(74, 38, 69)	(16, 17, 18)	(85, 75, 75)
1	(21, 18, 45)	(66, 53, 88)	(32, 67, 12)	(95, 65, 35)	(6, 0, 2)
2	(37, 29, 61)	(28, 49, 31)	(47, 21, 94)	(31, 41, 51)	(246, 84, 13)
3	(82, 33, 90)	(42, 43, 44)	(15, 80, 50)	(60, 40, 12)	(188, 45, 1)

假設我們現在有一個座標為 (x, y) 的 pixel，它模糊後的 `new_r`, `new_g`, `new_b` 數值應該為 (x, y) 與其周圍八個點 $(x-1, y), (x+1, y), (x-1, y-1), (x, y-1), (x+1, y-1), (x-1, y+1), (x, y+1), (x+1, y+1)$ 的平均。舉例來說，下圖 $(2, 1)$ 點模糊後的新數值應該為 $(52, 41, 55)$

$$52 = (84+74+16+66+32+95+28+47+31) / 9$$

$$41 = (22+38+17+53+67+65+49+21+41) / 9$$

$$55 = (99+69+18+88+12+35+31+94+51) / 9$$

以下六點請注意：

1. 請務必將平均出來的值存在一個全新的 `new_img`，千萬不要用新得到平均數值來改變舊影像（請使用 `new_img = SimpleImage.blank(new_w, new_h)` 來製造空白的影像 `new_img`）
2. 位在角落的點，例如上圖之 $(0, 0)$ ，只會有三個鄰居 $(0, 1), (1, 0), (1, 1)$
3. 位在邊上的點，例如上圖之 $(2, 0)$ ，只會有五個鄰居 $(1, 0), (1, 1), (2, 1), (3, 0), (3, 1)$
4. 請注意 `def blur(img)` 接收的 `img` 已經是一張照片，並不是檔名
5. 在 `def main()` 裡我們使用 `for loop` 呼叫您要編輯的 `blur` 來達到多次的模糊效果（如下圖程式碼所示）
6. 此題的運算量極大，大約需要 一分鐘 的運算時間，請同學耐心等候

```
def main():
    old_img = SimpleImage("images/smiley-face.png")
    old_img.show()

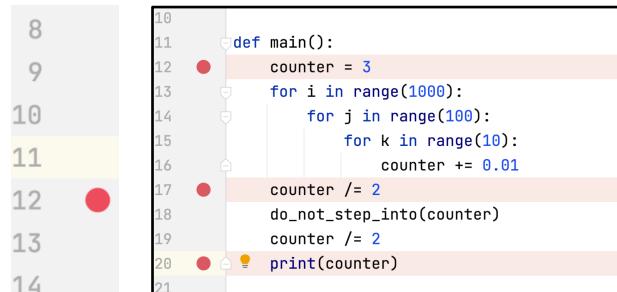
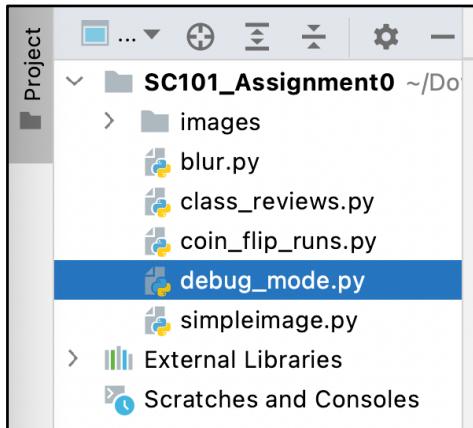
    blurred_img = blur(old_img)
    for i in range(4):
        blurred_img = blur(blurred_img)
    blurred_img.show()
```

- 如果原本就已經完成的同學，可以嘗試將程式的 行數控制在 40 行以內（使用 4 個 for loop 找鄰居），這個概念將在最後的演算法課程作業會派上用場！

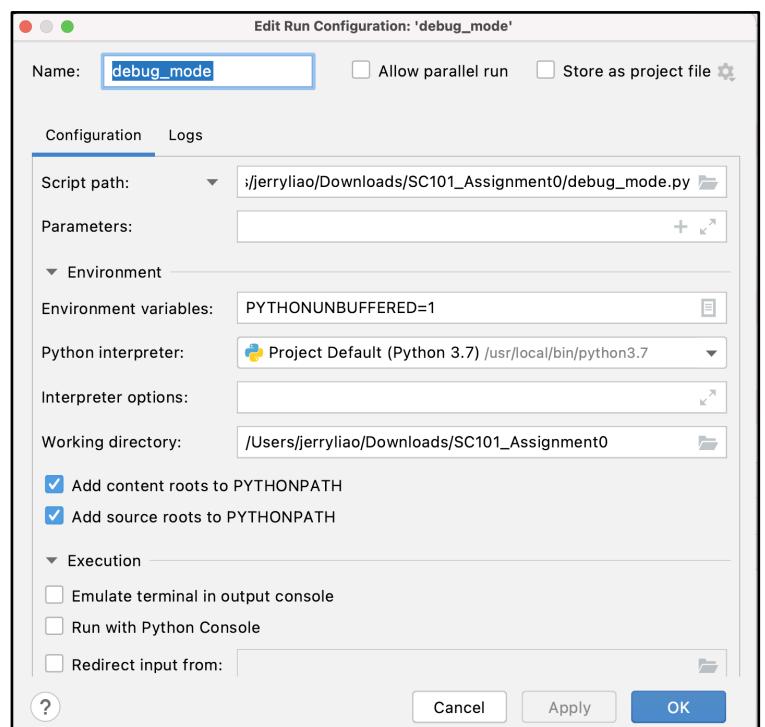
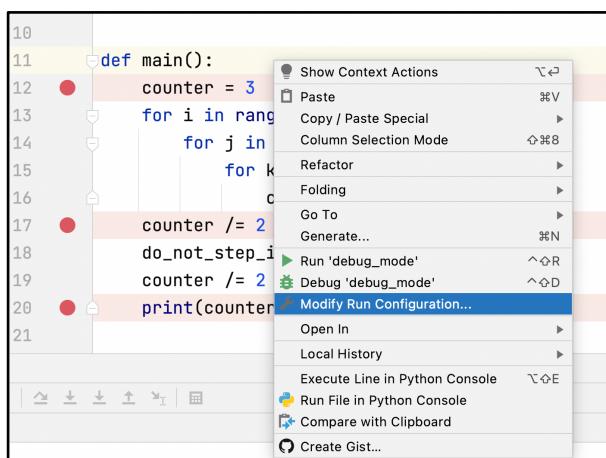
Problem 4 - debug_mode.py

最後一題要請大家學習一個強大的 debug 工具：**PyCharm Debug Mode**。
由於進階班程式相當複雜，請同學務必熟悉這個強大的 debug 工具！

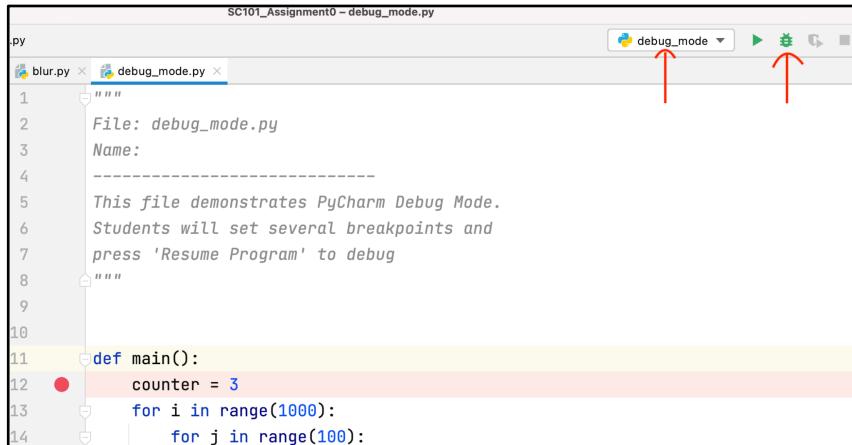
- 為了啟動 Debug Mode, 首先請同學先點選視窗左側  Project，
檢查匯入 PyCharm 的資料夾是否正確 (如下圖所示↓)



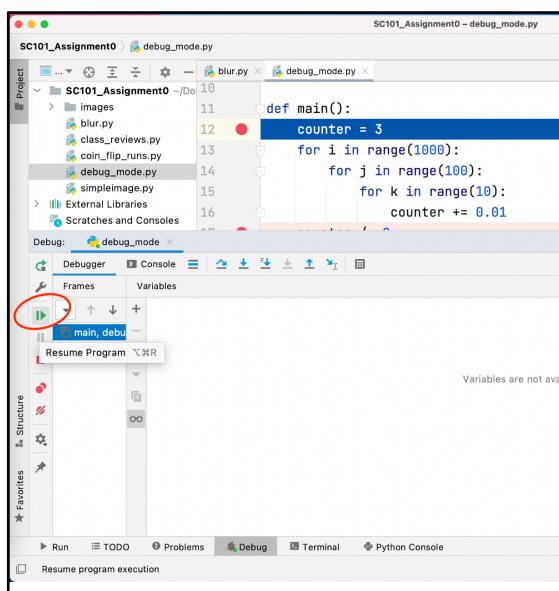
- 再來請於 **12、17 及 20 行** 右側點擊，
製造三顆 **breakpoints** (如右圖→)
- 對檔案任意處點擊右鍵，點選 「**Modify Run Configuration**」
(如下圖↓)，按下 **Ok**



- ④ 確認右上方下拉選單之檔名為「debug_mode」後，按下「綠色蟲蟲」(如下圖右上方箭頭所指)



- ⑤ 最後，請同學們找到螢幕左下一個名為「Resume Program」的按鈕，功能為「繼續執行直到下一個 breakpoint」(如下圖↓)



- ⑥ 請問：若您點按第一次 Resume Program 抵達第 17 行，counter 的數值在經過 13-16 行的迴圈後變多少？若您接著再按一次 Resume Program 抵達第 20 行，counter 數值變多少？請將答案寫在檔案第 17 行以及第 20 行的註解區(如下圖↓)

```
11 def main():
12     counter = 3
13     for i in range(1000):
14         for j in range(100):
15             for k in range(10):
16                 counter += 0.01
17     counter /= 2
18     do_not_step_into(counter)
19     counter /= 2
20     print(counter)

# Answer1: -----
# Answer2: -----
```

評分標準

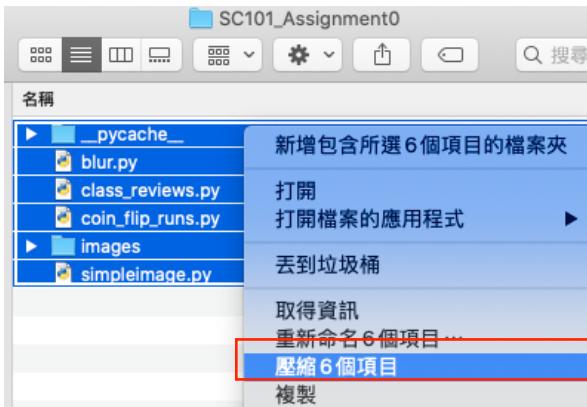
Functionality - 程式是否有通過我們的基本要求？程式必須沒有 bug、能順利完成指定的任務、並確保程式沒有卡在任何的無限環圈（Infinite loop）之中。

Style - 好的程式要有好的使用說明，也要讓人一目瞭然，這樣全世界的人才能使用各位的 code 去建造更多更巨大更有趣的程式。因此請大家寫 精簡扼要 的使用說明、function 敘述、單行註解。

作業繳交

1. 以滑鼠「全選」作業資料夾內的所有檔案，並壓縮檔案。請見下圖說明。

macOS : 按右鍵選擇「壓縮n個項目」



Windows : 按右鍵選擇「傳送到」→「壓縮的(zipped)資料夾」

