stanCode

標準程式教育機構

Assignment 0

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

作業檔案下載

歡迎來到 SC101! 由於進階班需要非常扎實的基礎往上蓋出程式巨塔，這份作業將訓練各位同學程式撰寫最重要的基本邏輯：靈活使用變數 (Variables)、使用者互動之對畫框 (Console)、各式迴圈 (Loops)、並熟悉運算法則 (Expression)

上過 SC001 的同學一定會覺得很眼熟。沒錯！所有題目都出自我們之前的作業！但好的作業絕對值得一寫再寫，請同學務必確保開課前可以獨立完成每一題

如果作業卡關 **歡迎各位到社團提問**，也非常鼓勵同學們互相討論作業之 **概念**，**但請勿把 code 給任何人看**（也不要把程式碼貼在社團裡）分享妳/你的 code 會剝奪其他學生獨立思考的機會，也會因此讓其他學生的程式碼與你/妳的極度相似，讓防抄襲軟體認定有抄襲嫌疑

如果真的有 **code** 方面的問題，可以**私訊** Jerry 或 **stanCode 粉絲專頁**

Problem 1 - weather_master.py

中央氣象局請同學幫忙處理天氣資料，身為 stanCode 學生的我們，當然就要使用程式替我們工作囉（挺）

中央氣象局希望我們特別注意在所有輸入程式數據中的四個數值：**最高溫**是多少？**最低溫**是多少？**平均溫度**是多少？以及有幾天可以發佈「**低溫警報**」（小於 16 度但不包含 16 度）所以聰明的 妳/你 就想到，何不使用上課所教的程式概念來處理這個問題？

您的程式會反覆請使用者輸入一個整數 ；若使用者想要離開程式只要輸入 -100 即可。然而，身為一個好的 programmer，我們要將「使程式離開的值（-100）」存入一個位於 main( ) 上方的 **constant**，就好像終極密碼最終的數字一樣。您寫的程式應該要可以得出與下圖一模一樣的內容：

```
stanCode "Weather Master 4.0"!
Next Temperature: (or -100 to quit)? 20
Next Temperature: (or -100 to quit)? 16
Next Temperature: (or -100 to quit)? 8
Next Temperature: (or -100 to quit)? 13
Next Temperature: (or -100 to quit)? 19
Next Temperature: (or -100 to quit)? 24
Next Temperature: (or -100 to quit)? 33
Next Temperature: (or -100 to quit)? 31
Next Temperature: (or -100 to quit)? -100
Highest temperature = 33
Lowest temperature = 8
Average = 20.5
2 cold day(s)
```

眼尖的同學一定有注意到，在 print (“”) 括弧的雙引號內 **再填入雙引號** 是不可能的事。舉例來說，print("SC001 "Weather") 會讓電腦認為我們只需要 "SC001 "

為了解決這件事情，電腦工程師就把「**當 \“ 出現在雙引號（”“）內**」定義為 **顯示引號** 的方法。舉例來說，print("SC001 \" Weather") 就會印出 **SC001 "Weather**

如同題幹一開始的敘述，身為一個好的 programmer，我們要將「使程式離開的值（-100）」存入一個位於 weather( ) 上方的 constant。然而，如果我們**更改它的值**，從 -100 變成 -1 並重跑（recompile / run）程式，畫面就會變成：

```
stanCode "Weather Master 4.0"!
Next Temperature: (or -1 to quit)? 33
Next Temperature: (or -1 to quit)? 22
Next Temperature: (or -1 to quit)? 27
Next Temperature: (or -1 to quit)? -1
Highest temperature = 33
Lowest temperature = 22
Average = 27.333333333333332
0 cold day(s)
```

請同學注意的地方是，使用者可以**只輸入一個數值**（如下圖，當我們只輸入 3 ）而不影響我們判別最高溫、最低溫、以及平均氣溫（應該都要是 3 ）

```
stanCode "Weather Master 4.0"!
Next Temperature: (or -1 to quit)? 3
Next Temperature: (or -1 to quit)? -1
Highest temperature = 3
Lowest temperature = 3
Average = 3.0
1 cold day(s)
```

最後，如果使用者一開始便輸入「使程式離開的值」，那我們就要印出

No temperature were entered. 的字樣，如下圖所示：

```
stanCode "Weather Master 4.0"!
Next Temperature: (or -1 to quit)? -1
No temperatures were entered.
```

Problem 2 - hangman.py

第二題將請同學使用 Console 完成經典猜字遊戲 Hangman (吊死鬼)

程式的一開始會從字庫隨機選擇一個英文單字 (以下簡稱為 **answer**)，並將每一個字母用橫槓遮住（以下簡稱為 **dashed**）。玩家每次輸入一個大寫或寫小的字母（以下簡稱為 **input_ch**），如果 input_ch 存在於 answer 之中，程式就會更新 dashed 並把**所有 input_ch 所在的位置展示出來**。然而，如果 **input_ch** 不存在於 **answer** 之中，玩家就會損失一條命。若七條命扣完還沒猜出來，玩家挑戰失敗

真正的 Hangman 遊戲會在玩家猜錯時更新吊死鬼的圖樣（如下圖所示），當吊死鬼的頭部、身體、左手、右手、左腳、右腳、舌頭都被呈現出來時玩家級挑戰失敗

然而，**我們在這題並不需要做到圖樣的部分！**只要完成 Console 版的即可（或許圖樣版的可以當成各位的 Extensions ^^）

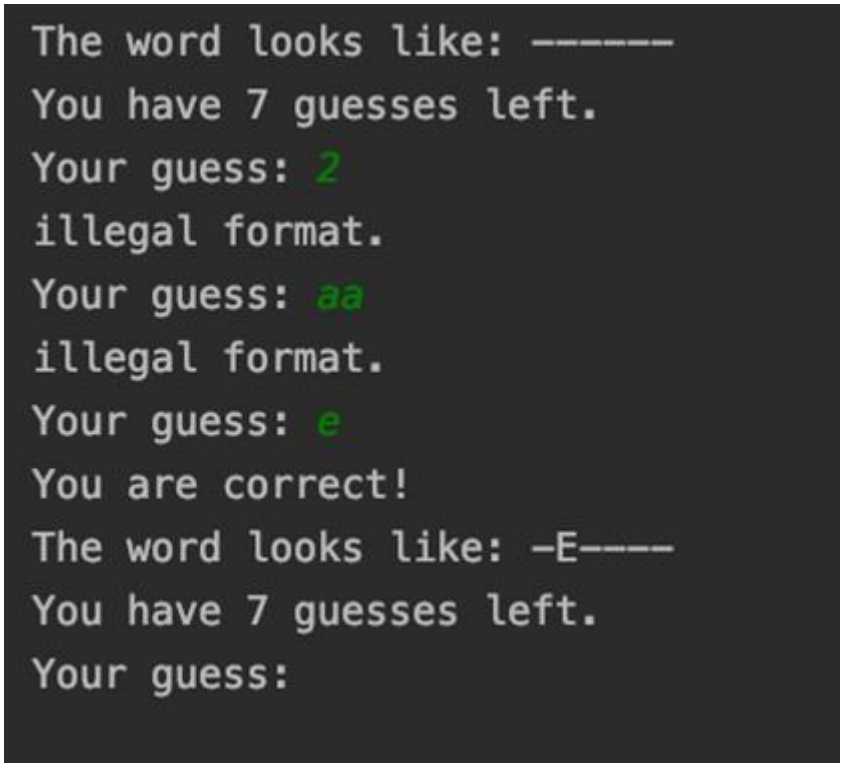
您的程式將可以完美呈現下圖的每一行文字與結果：

```
The word looks like: ------
You have 7 guesses left.
Your guess: k
There is no K's in the word.
The word looks like: ------
You have 6 guesses left.
Your guess: k
There is no K's in the word.
The word looks like: ------
You have 5 guesses left.
Your guess: j
There is no J's in the word.
The word looks like: ------
You have 4 guesses left.
Your guess: q
There is no Q's in the word.
The word looks like: ------
You have 3 guesses left.
Your guess: x
There is no X's in the word.
The word looks like: ------
You have 2 guesses left.
Your guess: b
There is no B's in the word.
The word looks like: ------
You have 1 guesses left.
Your guess: a
There is no A's in the word.
You are completely hung : (
The word was: REFUND
```

```
The word looks like: ----------
You have 7 guesses left.
Your guess: I
You are correct!
The word looks like: ----I---I--
You have 7 guesses left.
Your guess: i
You are correct!
The word looks like: ----I---I--
You have 7 guesses left.
Your guess: h
You are correct!
The word looks like: H---I---I--
You have 7 guesses left.
Your guess: o
You are correct!
The word looks like: HO--I---I--
You have 7 guesses left.
Your guess: P
You are correct!
The word looks like: HO-PI---I--
You have 7 guesses left.
Your guess: t
You are correct!
The word looks like: HO-PIT--IT-
You have 7 guesses left.
Your guess: a
You are correct!
The word looks like: HO-PITA-IT-
You have 7 guesses left.
Your guess: L
You are correct!
The word looks like: HO-PITALIT-
You have 7 guesses left.
Your guess: s
You are correct!
The word looks like: HOSPITALIT-
You have 7 guesses left.
Your guess: Y
You are correct!
You win!!
The word was: HOSPITALITY
```

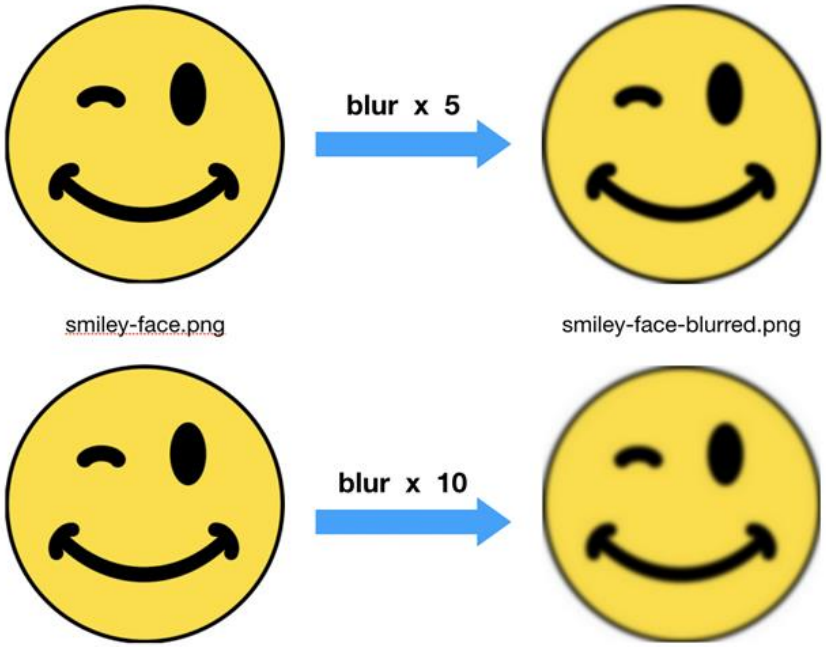
以下五點重點提醒

1. 請使用我們寫好的 **random_word( )** 來得到一個隨機的英文單字。同學不用了解此 function 的每一行 code，只要知道 random_word( ) 每一次會隨機 return 一個英文單字出來即可
2. 使用者的輸入為 **case-insensitive**，**upper case (大寫) lower case (小寫)** 都可以
3. 輸入兩次錯誤答案還是會少一條命（如左圖 k and k）
4. 輸入兩次正確答案還是得到一樣的結果（如右圖 I and i）
5. 當使用者輸入格式錯誤，例如輸入的不是英文字母(可以用 **str.isalpha( )** 判斷)或不只一個字母，程式應該要**印出 "Illegal format."** 並重複要求使用者輸入，直到格式正確（如下圖所示）



Problem 3 - blur.py

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 **(請思考怎麼使用 4 個 for loop 完成此題，進階班最後一份作業會用到)**



假設我們有一個座標為 **(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**）

	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)

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：

- 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（您應該會使用到 **new_img = SimpleImage.blank(new_w, new_h)** 來製造空白的影像 new_img）

- 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)**

- 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)**

- 您會發現使用 **4 個 for loop** 可以省去判斷角落、邊上、中間的時間。讓電腦自己判斷到底要取幾顆 pixel

- 在 def main( ) 裡我們使用 for loop 呼叫您要編輯的 blur 四次來達到更明顯的模糊效果（如下圖之程式碼所示）。然而，因為第一次模糊處理的對象為 old_img，因此，雖然 for loop 只反覆四次，但其實您的原圖已經被模糊處理了五次。（同學可以將 for loop 的次數改為 9 次，您的圖像就可以被糊糊十次了！）

- 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()
```

stanCode - 標準程式教育機構

Should you have any questions please feel free to contact us.