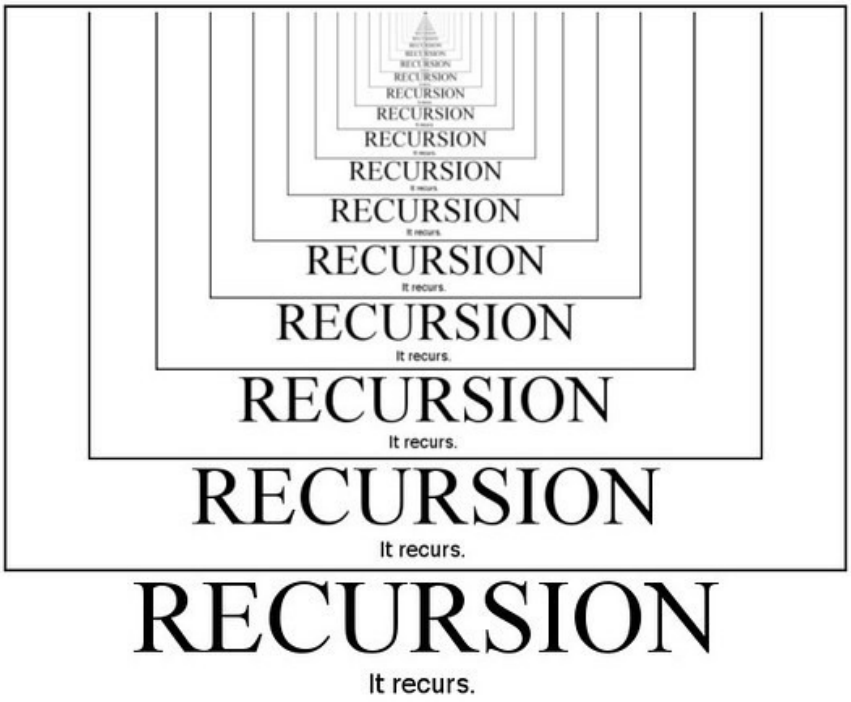
stanCode

標準程式教育機構

Assignment 5

This assignment is based on the Assignment 3 and 4 of CS106B at Stanford University
Image Credit:Algodaily.com



作業檔案下載

歡迎來到 **演算法** 的作業！改編至 Stanford 進階程式課程 CS106B，本次作業真的非常非常具有挑戰性。然而，若成功征服，您將可以開始準備面試各大公司軟體工程師職位，包含四大科技龍頭 FAGA – Facebook, Apple, Google, Amazon.

遞迴（recursion）是所有軟體工程師面試題最困難的部分。因此這份作業將提供同學們所有一切解題需要的基本觀念！讓同學之後在 LeetCode 刷題都可以一路過關斬將

然而，一定會有不少同學會寫這份作業寫到懷疑人生…

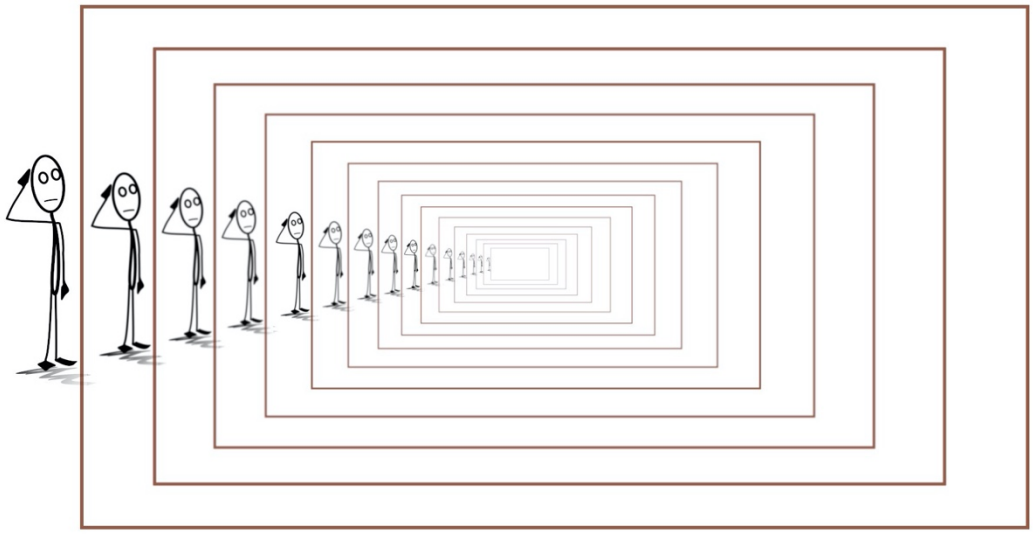


Image Credit: Kulbhushan Singhal

但請同學千萬不要放棄！教學團隊一定會陪各位同學完成不可能的任務，讓各位不管是邏輯思維還是程式能力都有爆炸性的突破

作業預計時間：**15** 小時

本份作業請勿使用global variables

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，也會讓其他學生的程式碼與你/妳的極度相似，使防抄襲軟體認定有抄襲嫌疑

Problem 1 – Tracing Recursive Code

第一題我們先來建立基本遞迴觀念，並了解遞迴程式執行順序。請模仿 Jerry 上課時在白板追蹤遞迴程式的過程追蹤下圖遞回程式碼，並在紙上寫下「每一個 **stack frames** 的變數數值與其執行順序」

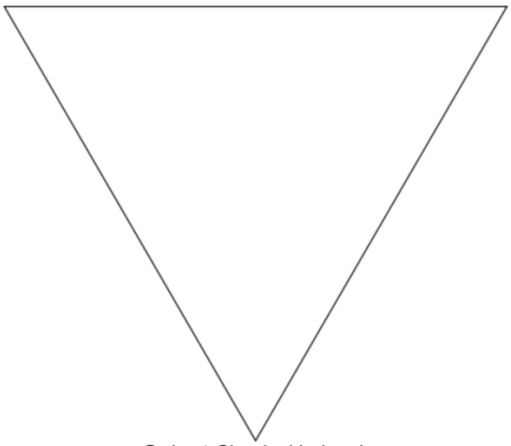
寫完之後，請拍下紙上的筆跡、將照片放到 Assignment5 作業資料夾 ”Problem 1” 中，並回答下列問題：
(1.) 請問執行這段程式碼會在 Console 上看到幾遍 “**Base Case**！”
(2.) 變數**num**數值是多少？

```
def recursion():
    num = b(5, 2)
    print(num)


def b(n, k):
    if k == 0 or k == n:
        print('Base Case!')
        return 2
    else:
        return b(n-1, k-1) + b(n-1, k)


if __name__ == '__main__':
    recursion()
```

Problem 2 – sierpinski.py

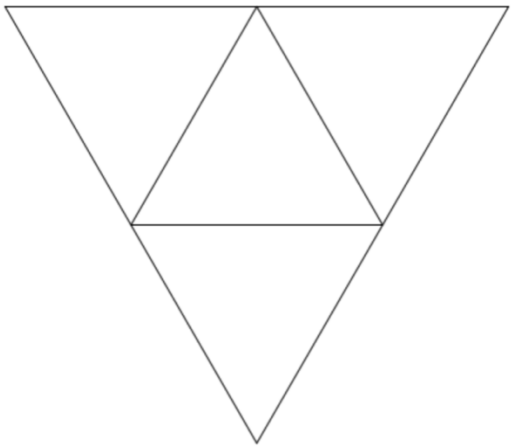
Fractal是用遞迴概念 – self similarity – 畫出來的圖形。其中一個最有名的例子，就是波蘭數學家 Waclaw Sierpinski 發明的三角形，Sierpinski Triangle

如果今天我們畫 order 1 的 Sierpinski Triangle，您會得到一個正三角形：

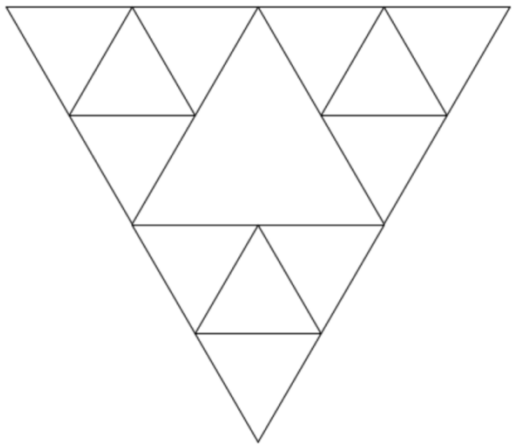


Order 1 Sierpinski triangle

為了完成一個 order K 的 Sierpinski Triangle，我們必須先畫三個邊長為原長一半的 order K-1 三角形，且這三個 order K-1 的三角形都在 order K 的三個頂點。
Order 2 與 order 3 的Sierpinski Triangle如下圖所示：

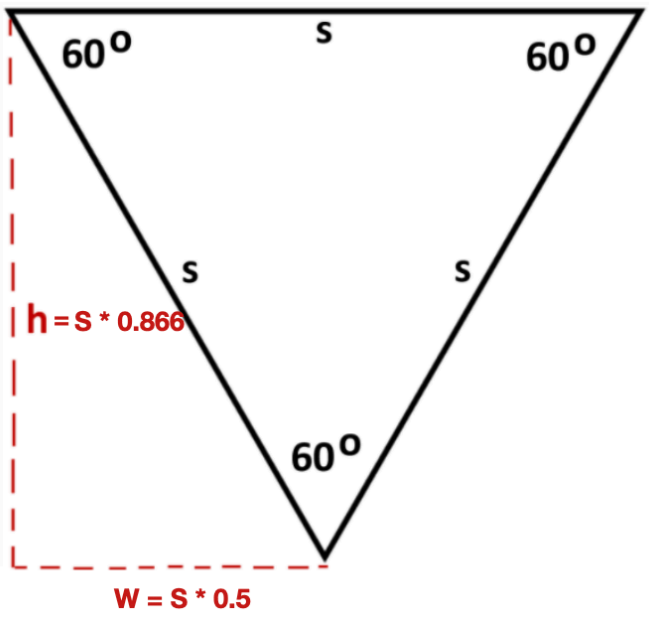


Order-2



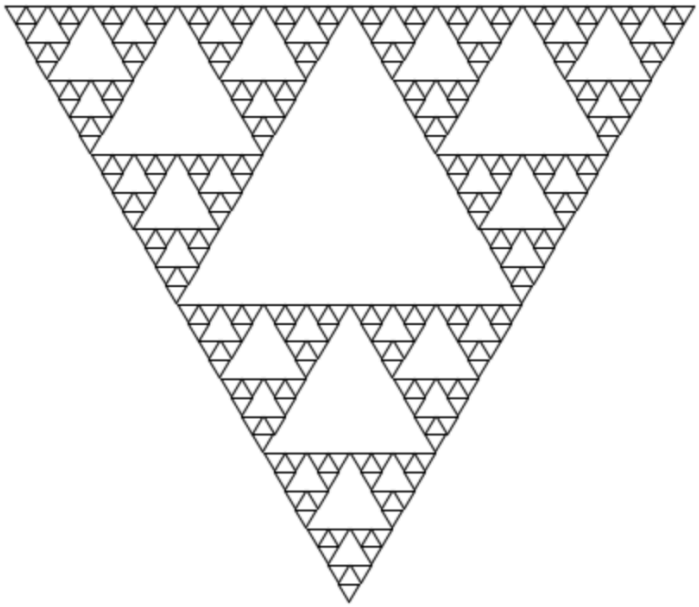
Order-3

請在 window 上畫出由常數 ORDER 控制的 Sierpinski Triangle。我們已經在 main( ) 呼叫了您即將要撰寫的 **def sierpinski_triangle(order, length, upper_left_x, upper_left_y)**. 這邊要注意的是，upper_left_x 與 upper_left_y 分別代表三角形左上方頂點的 x 座標與 y 座標，且可以假設 order 一定大於零
有些學生會認為：每一個 Sierpinski Triangle 都應該要畫出兩種不同的三角形：「往上指的三角形」還有 「往下指的三角形」。但這種想法是錯的，因為遞迴應該要尋找 self similarity。也就是說，在order 2 畫出的所有三角形都應該跟 order 1 畫出的三角形相似

在畫出每一個三 

有幫助：

若您的程式撰寫正確，應該會在執行 sierpinski.py 出現與下圖一模一樣的圖樣：



... Order-6

Problem 3 – largest_digit.py

第三題要請同學用遞迴的方式找出任意整數裡最大的位數數值！我們在 main( ) 裡面放了五組不同的整數 (有正也有負) 在 **find_largest_digit** 函式。若您的程式撰寫正確無誤，執行 **largest_digit.py** 後應該會在 Console 看到 5, 8, 6, 1, 9 這五個數字（換行印） 如下圖所示：

```
def main():
    print(find_largest_digit(12345))      # 5
    print(find_largest_digit(281))        # 8
    print(find_largest_digit(6))          # 6
    print(find_largest_digit(-111))       # 1
    print(find_largest_digit(-9453))      # 9
```

為了通過我們的測試，您的程式不僅要可以正確印出 5, 8, 6, 1, 9 這五個數字，還要符合下列三個限制：

1. 請勿使用任何資料結構 (list, tuple, …) 或是文字 (str)
2. 您只能使用 recursion，禁止使用迴圈 (for, while, …)
3. 您的程式效率應該要在 O(N)，N是輸入整數的位數個數

Probleme 4 - anagram.py

Anagram 的中文解釋為「同字母異序字」。舉例來說，'stop' 這個單字總共有 6 個 Anagrams: ['stop', 'spot', 'tops', 'opts', 'post', 'pots']。我們可以發現，每一個字彙都只使用了四個字母： 'o', 'p', 's', 't'。有趣的是，並非所有 'o', 'p', 's', 't' 的排列組合出來的英文字彙都存在於英文字典裡！舉例來說，字典裡並不存在 opst 這個字、亦不存在 ptso。因此，我們將寫出一個程式，接收使用者輸入的單字，並使用 backtracking 找出所有的 anagrams！

程式一開始會先印出 Welcome to stanCode "Anagram Generator" (or -1 to quit) 的字樣。與 SC001 第二份作業的 weather 一樣，我們需要印出雙引號，也要使用一個常數 (-1) 控制離開的條件（如下圖所示）。現在回想兩個月前的 SC001，希望同學此刻有明顯地感受到自己的成長與蛻變！

```
Welcome to stanCode "Anagram Generator" (or -1 to quit)
Find anagrams for:
```

假設使用者輸入單字 ***stop*** 並按下鍵盤上的 Enter，您的程式將開始搜尋定義在常數 FILE裡的文字檔（就是一部字典），查看 'o', 'p', 's', 't' 這四個字母排列組合出來的文字是否為存在於字典裡。詳細程式執行畫面如下圖所示：

```
Welcome to stanCode "Anagram Generator" (or -1 to quit)
Find anagrams for: stop
Searching...
Found:  stop
Searching...
Found:  spot
Searching...
Found:  tops
Searching...
Found:  opts
Searching...
Found:  post
Searching...
Found:  pots
Searching...
6 anagrams:  ['stop', 'spot', 'tops', 'opts', 'post', 'pots']
Find anagrams for: -1

Process finished with exit code 0
```

以下七個重點提醒：

1. 請編輯名為**def read_dictionary( )** 的函示，讀取整本字典並將讀到的所有字彙（記得去除換行字元）儲存在 Python list（可以使用 Global Variable）

2. 請編輯名為 **def find_anagrams(s)** 的函示，讓 **s** 接收使用者輸入的單字，並使用 backtracking 搜尋所有的 anagrams

3. 為了縮短搜尋時間，我們通常會提前將搜尋程序終止！為了做到這點，請同學編輯一個名為 **def has_prefix(sub_s)** 的函示，告訴使用者「**字典裡是否存在由 sub_s 開頭的字彙？**」

   Python 文字都有內建一個名叫 startswith(sub_s) 的 method，可以檢查任意文字是否以 sub_s 開頭。若我們使用 **'coding'.startswith('co')** 會得到 **True**；而若我們使用 **'standard'.startswith('stam')** 就會得到 **False**

   舉例來說，假設今天我們要搜尋橡皮擦 "eraser" 的所有 anagrams。若排列組合的過程中若碰到了 "rr"，程式就不應該繼續找下去了，因為 **has_prefix("rr")** 搜尋整本字典後發現沒有任何字彙是以 "rr" 開頭的。因此 **has_prefix("rr")** 會return False，告訴程式我們沒有繼續排列組合下去的必要。同理，**has_prefix("srr")**, **has_prefix("sre")** 都應該 return False。如此，我們程式的遞迴搜尋次數就可以大大減少、也大大加快我們程式的搜尋速度！

4. 這個程式在處理上會花很多時間。這時，適當在Console 印出文字就顯得格外重要！（不然使用者會以為電腦當機了）。因此，在您的遞迴開始搜尋前，請先印出**Searching…** 的字樣告訴使用者我們已經開始搜尋。除此之外，每當我們找到一個 anagram，請印出 **Found:** 的字樣，並將找到的字彙緊接在後印出

5. 最後，請將所有您搜尋到的 anagrams 儲存在一個 list，並在結束遞迴搜尋後印出該 list 的內容。舉例來說。若搜尋手臂的英文單字 'arm' 搜尋到的結果為 ['arm', 'ram', 'mar']，遞迴結束前的搜尋結果應該如下：
   3 anagrams: ['arm', 'ram', 'mar']

6. **def main( )** 裡面的 **start = time.time( )** 以及 **end = time.time( )** 可以計算出您演算法的速度（請將 **start** 的程式碼移到使用者輸入完後、請將 **end** 的程式碼移到所有字找到之後）。助教們會比較全班每一位同學的 anagram.py 看看誰的演算法速度最快！stanCode 也將頒發 500 NTD 給我們班的 演算法之王

```
def main():
    start = time.time()
    ####################
    #                  #
    #       TODO:      #
    #                  #
    ####################
    end = time.time()
    print('----------------------------------')
    print(f'The speed of your anagram algorithm: {end-start} seconds.')
```

7. 請勿重複加入字彙。答案展示出來的單字應該都是獨一無二的。同學們可以使用 **contains** 這個英文單字來測試。若程式這卻無誤，應該可以看到下圖的每一行文字：

```
Welcome to stanCode "Anagram Generator" (or -1 to quit)
Find anagrams for: contains
Searching...
Found:  contains
Searching...
Found:  canonist
Searching...
Found:  actinons
Searching...
Found:  sonantic
Searching...
Found:  sanction
Searching...
5 anagrams:  ['contains', 'canonist', 'actinons', 'sonantic', 'sanction']
Find anagrams for: -1

Process finished with exit code 0
```

評分標準

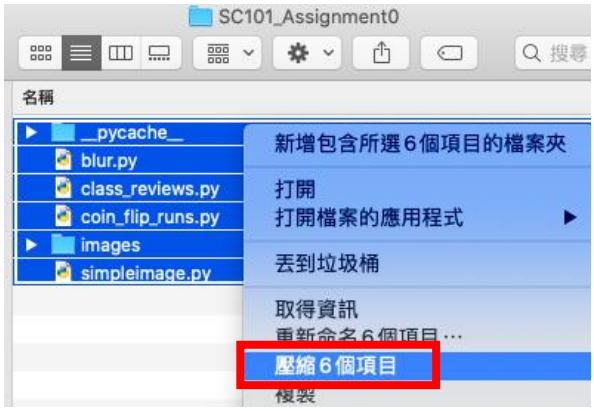
Functionality - 程式是否有通過我們的基本要求？程式必須沒有 bug 、能順利完成指定的任務、並確保程式沒有卡在任何的無限環圈（Infinite loop）之中

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

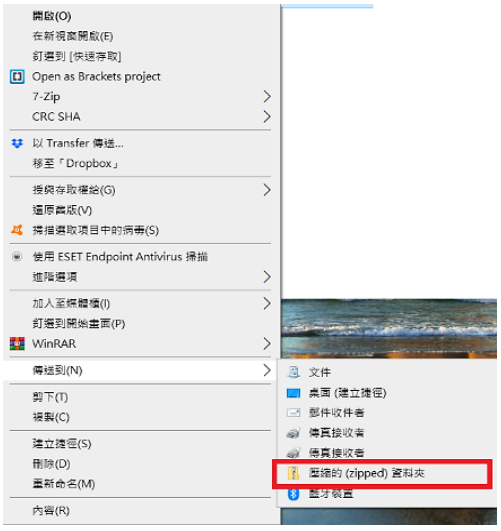
作業繳交

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。

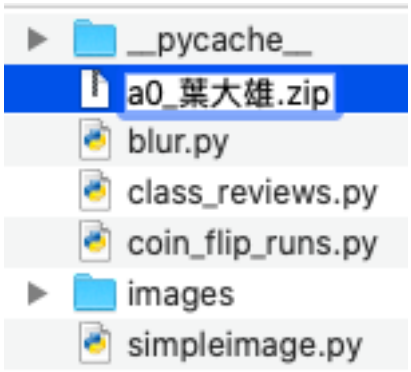
macOS：按右鍵選擇「**壓縮 n 個項目**」



Windows：按右鍵選擇「**傳送到**」→「**壓縮的(zipped)資料夾**」

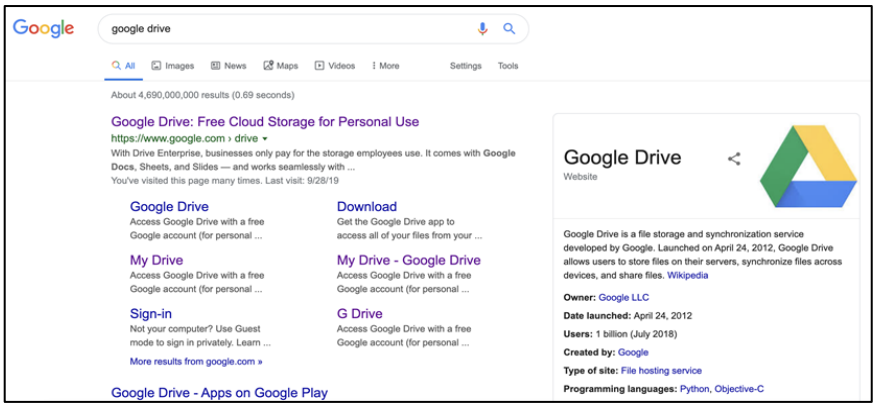


2. **將壓縮檔(.zip)重新命名為「a(n)_中文姓名」。如：**
   assignment 0 命名為 a0_中文姓名;
   assignment 1 命名為 a1_中文姓名; ...



3. **將命名好的壓縮檔(.zip)上傳至 Google Drive（或任何雲端空間）**

   1) 搜尋「google drive」



   2) 登入後，點選左上角「新增」→ 「檔案上傳」→ 選擇作業壓縮檔(.zip)

4. **開啟連結共用設定，並複製下載連結**

1) 對檔案按右鍵，點選「共用」

2) 點擊「**變更任何知道這個連結的使用者權限**」後，權限會變為「可檢視」

3) 點選「**複製連結**」

5. **待加入課程臉書社團後，將連結上傳至作業貼文提供的「作業提交表單」**

stanCode

Should you have any idea or questions, please feel free to contact.