stanCode

標準程式教育機構

Assignment 6

This assignment is based on the Assignment 4 and 5 of CS106B at Stanford University
Image Credit: Algodaily.com

點此下載作業檔案

歡迎來到 SC101 的最後一份作業！

本次作業改編自各大公司軟體工程師面試題目！遞迴 (recursion) 的觀念在 LeetCode 上佔了 1/3 以上的篇幅，因此我們會透過第一題讓你更熟悉這個重點中的重點。而這也應該是你人生中第一道 LeetCode 的 Hard 題，挑戰！

除此之外，我們也引入了連結串列 (LinkedList) 面試題目，希望讓每位 SC101 結業的同學都能具備基本的 LeetCode 刷題能力。

本份作業請勿使用 global variables

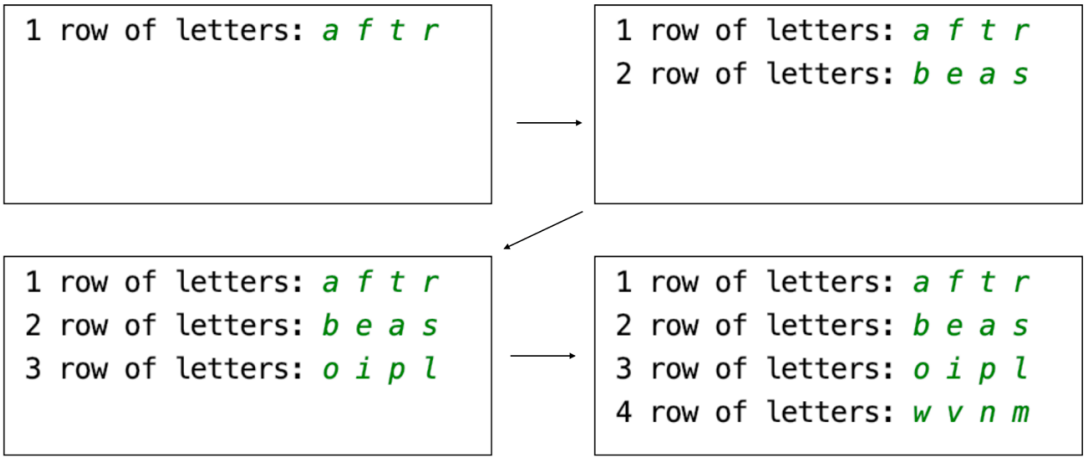
如果過程卡關歡迎各位向助教詢問！也非常鼓勵同學們互相討論作業之概念，但請勿直接把 code 分享給同學看，這很可能會剝奪他獨立思考的機會，並讓他的程式碼與你的極度相似，使防抄襲軟體認定有抄襲嫌疑。

LC212 - Word Search II - boggle.py

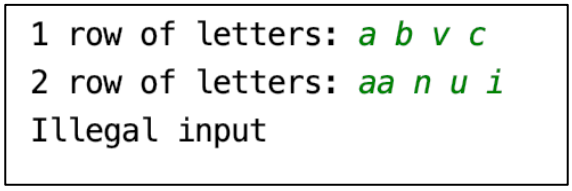
Boggle 是美國家喻戶曉的桌遊！遊戲一開始會先排出一個 4 x 4 的方形字母拼盤，接著玩家要去串連在字母盤上相連的字母，來找出所有存在於這個拼盤上的英文單字。概念上來說，這題有點類似前一份的 anagram.py，只是這次變成英文字母的排列變成平面 2D 了！我們一樣要使用 backtracking 的概念，來找到所有答案，並為 SC101 課程的遞迴部分劃下一個完美的句點。

下方將舉一些例子，讓我們更了解這個遊戲。

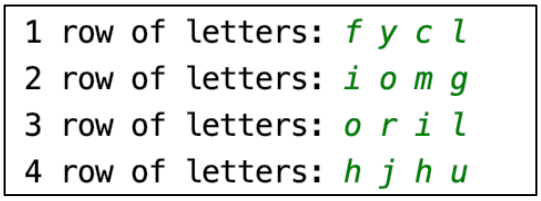
遊戲一開始玩家會被要求輸入 4 排以空白隔開的 4 個英文字母（如下圖所示）



若使用者的輸入不符合規定，程式就會立刻終止執行。換句話說，我們強迫使用者輸入 16 個英文字母且每一排的字母間一定要用空白隔開。



本次的題目，使用者要正確輸入如下圖所示的 16 個字母：



若題目輸入正確，程式就會開始搜尋這個 4x4 字母盤上所有可以被串連起來的英文單字。

而串連須遵守以下規則：

在串連字母的過程中，我們只能將相近的字母塊串起來，也就是字母塊當時八個方向的「鄰居」： 上、下、左、右、左上、左下、右上、右下 的字母塊。

並且，在同一次串連的過程中，被串過的字母塊，不能重複被選擇。

若程式正確，我們應該能在 console 上看到如下圖中的每一行文字：

```
1 row of letters: f y c l
2 row of letters: i o m g
3 row of letters: o r i l
4 row of letters: h j h u
Found "firm"
Found "form"
Found "foil"
Found "coif"
Found "coir"
Found "corm"
Found "coil"
Found "moor"
Found "moil"
Found "miri"
Found "giro"
Found "glim"
Found "roil"
Found "roof"
Found "room"
Found "roomy"
Found "rimy"
Found "iglu"
Found "limy"
Found "limo"
Found "hoof"
There are 21 words in total.
```

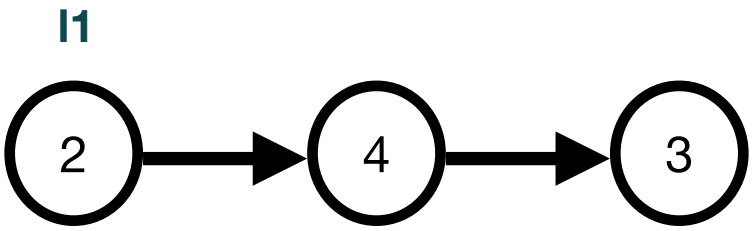
以下八個重點提醒：

1. 起始字母可以任意選擇。
2. 輸入的題目應該為 **case-insensitive**。
3. 程式只搜尋 字母長度 >= **4** 的英文單字。
4. 編輯名為 **def read_dictionary( )** 的函式，讀取整本英文字典
5. 編輯名為 **def has_prefix(sub_s)** 的函式，檢查是否存在以 sub_s 為開頭的英文字
6. 當您撰寫的遞迴程式找到一個單字時，請印出 Found 並緊接著印出「找到的單字」
7. 這題最困難的地方莫過於上圖中 “room” 與 “roomy” ！該如何在找到一個解答後，繼續讓我們的遞迴程式往下搜尋更長的單字呢？
8. 最後，請印出在該 4 x 4 字母盤上找到了幾個單字的總數量

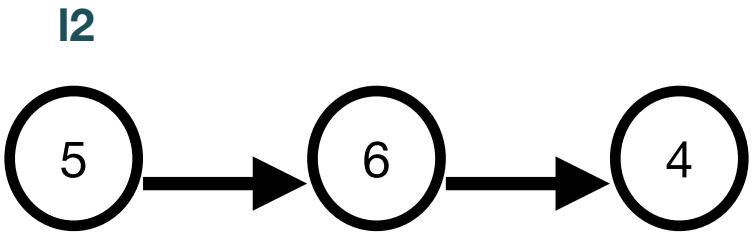
LC2 - Add Two Numbers - add2.py

兩條非 None 的 linked lists 代表了兩個正整數。然而，數值是「反向儲存」的。舉例來說，圖一的 l1（2 -> 4 -> 3）所代表的正整數是 342；而圖二的 l2（5 -> 6 -> 4）所代表的正整數是 465。請將這兩個正整數相加，將結果反向儲存成一條新的 linked list，並 return！

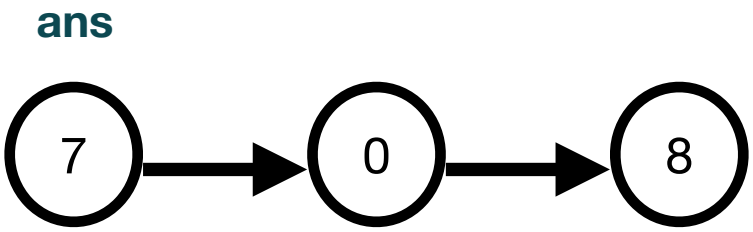
如圖三的 7 -> 0 -> 8 ，代表了 342 與 465 的總和，807。



圖一、開頭為 l1 的第一條 linkedlist，代表正整數 342

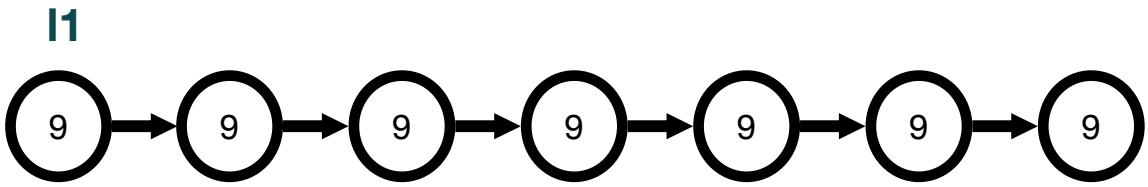


圖二、開頭為 l2 的第二條 linkedlist，代表正整數 465

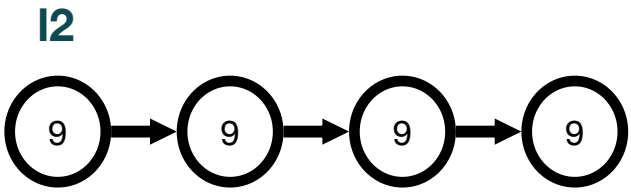


圖三、為第一條 linked list 與第二條 linked list 的總和，代表正整數 807

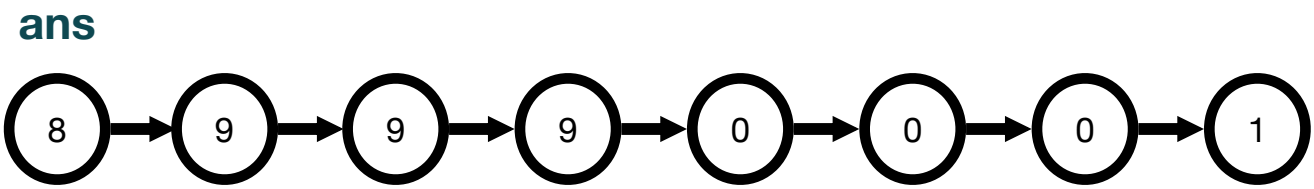
將兩條 linked lists 相加時，每一條 linked list 的長度是可以不一樣的。
圖四的 9->9->9->9->9->9->9 所代表的正整數是 9999999；圖五的 9->9->9->9 所代表的正整數是 9999 。要 return 的 linked list 為 8->9->9->9->0->0->0->1，代表他們的總和， 10009998。



圖四、開頭為 l1 的第一條 linked list，代表正整數 9999999

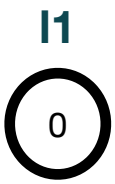


圖五、開頭為 l2 的第二條 linked list，代表正整數 9999



圖六、為第一條 linked list 與第二條 linked list 的總和，代表正整數 10009998

您可以假設 node 的數量一定介於 1~100 顆；node 的值一定介於 0~9；數值一定不會有多餘的零在開頭（例如， 2 -> 1 -> 0 -> 0 是不會出現的）。但 l1 及 l2 可以都只存 0。如下圖七所代表的正整數是 0 ; 圖八所代表的正整數也是 0 ; 而您要 return 的 linked list 如圖九所示，也是 0。



圖七、開頭為 l1 的第一條 linked list，代表整數 0

圖八、開頭為 l2 的第二條 linkedl ist，代表整數 0

圖九、為第一條 linked list 與第二條 linked list 的總和，代表整數 0

請編輯名為 **def add_2_numbers(l1: ListNode, l2: ListNode) -> ListNode:** 的函式。請注意這種新的 Python 註解寫法：l1, l2 是變數名稱，冒號後面的 ListNode 是 data type；而最後的 -> ListNode 代表 return 的 data type 是一個 ListNode。若程式撰寫正確，當你在 Terminal/cmd 執行 python3 add2.py test1 時，會看到與下圖一樣的輸出結果：

```
jerryliao@Jerrys-MacBook-Pro SC101_Assignment6 % python3 add2.py test1
---------test1---------
l1: 2->4->3
l2: 5->6->4
ans: 7->0->8
-----------------------
```

執行 python3 add2.py test2 時，會看到與下圖一樣的輸出結果：

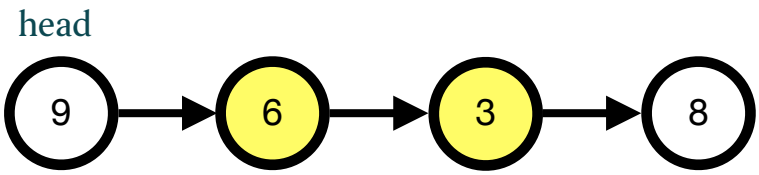
```
jerryliao@Jerrys-MacBook-Pro SC101_Assignment6 % python3 add2.py test2
---------test2---------
l1: 9->9->9->9->9->9->9
l2: 9->9->9->9
ans: 8->9->9->9->0->0->0->1
-----------------------
```

執行 python3 add2.py test3 時，會看到與下圖一樣的輸出結果：

```
jerryliao@Jerrys-MacBook-Pro SC101_Assignment6 % python3 add2.py test3
---------test3---------
l1: 0
l2: 0
ans: 0
-----------------------
```

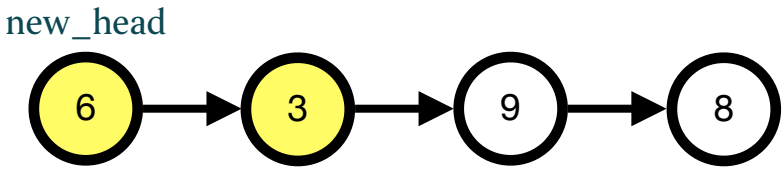
LC86 - Microsoft, Facebook - new_head.py

請將開頭為 head 的 linked_list 數值小於 x 的 ListNode 依序放到前方！
舉例來說，假設 head 是 9->6->3->8 這條 linked_list 的開頭 (如下圖一所示)



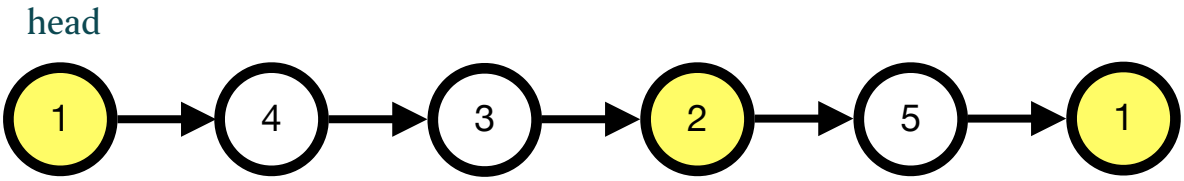
圖一、開頭為 head 的 linked_list

若使用者呼叫 new_head = newhead(head, 8)，則 new_head 會是 6，且所有小於 8 的 ListNode 都要需要依序往前放到開頭 (如下圖二所示)



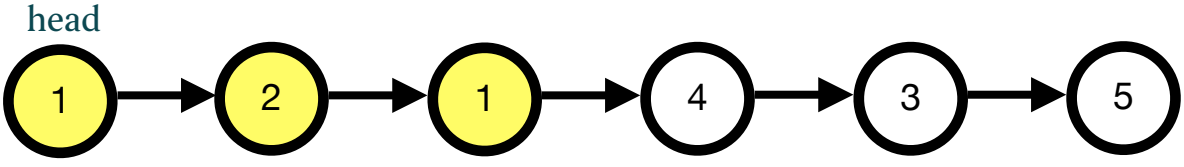
圖二、往前放置後的 linked_list 示意圖

再舉個例子！假設 head 是 1->4->3->2->5->1 這條 linked_list 的開頭 (如下圖三所示)



圖三、開頭為 head 的 linked_list

若使用者呼叫 new_head = newhead(head, 3)，則 new_head 會是 1，且所有小於 3 的 ListNode 都要需要依序往前放到開頭 (如下圖四所示)



圖四、往前放置後的 linked_list 示意圖

請編輯名為 **def new_head(head: ListNode, x: int) -> ListNode:**
的函式。若程式撰寫正確，當你在 Terminal/cmd 執行 python3 new_head.py test1 時，會看到與下圖一樣的輸出結果：

```
MacBook-Pro-5:SC101Assignment6 belindahsiao$ python3 newhead.py test1
---------test1---------
l1: 9->6->3->8
ans: 6->3->9->8
-----------------------
```

執行 python3 new_head.py test2 時，會看到與下圖一樣的輸出結果：

```
MacBook-Pro-5:SC101Assignment6 belindahsiao$ python3 newhead.py test2
---------test2---------
l1: 1->4->3->2->5->1
ans: 1->2->1->4->3->5
-----------------------
```

評分標準

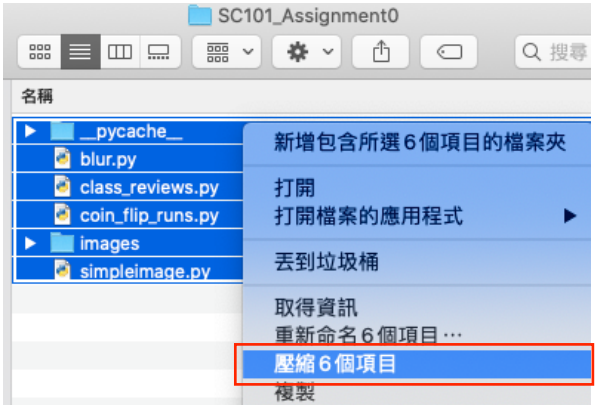
Functionality - 程式是否有通過我們的基本要求？程式必須沒有 bug 、能順利完成指定的任務、並確保程式沒有卡在任何的無限環圈（Infinite loop）之中。

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。

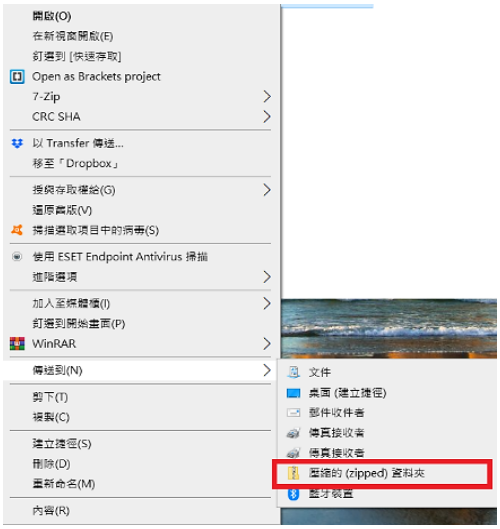
作業繳交

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。

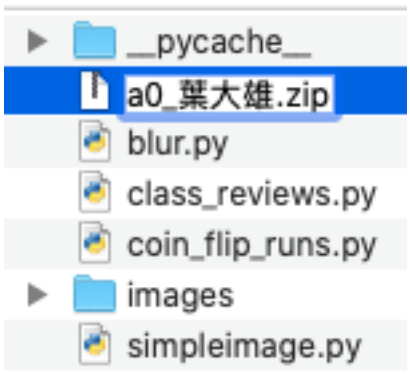
macOS：按右鍵選擇「壓縮n個項目」



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」