Practice Final

測驗指示

請參考以下流程，並使用「作答模板」撰寫答案:

1. 新增一份自己的 Google Doc 空白文件
2. 打開作答模板連結，全選(Ctrl(Command) + A)整份文件內容
3. 按下(Ctrl(Command) + C)複製
4. 將複製的內容貼上自己的 Google Doc 空白文件，開始作答
5. 完成測驗後，請**開啟文件權限(查看)**，並將**共享連結**提交至**作答上傳表單**
   (模擬考無需提交，期末考時會提供上傳表單連結)

為了讓撰寫過程更順暢，請調整 Google Doc 的設定：
工具 -> 拼字與文法 -> 關掉「拼字建議」以及「文法修正」
(Tool -> Preferences -> Spelling and grammar -> Un-check every item)

考試可以使用任何筆記、書本或資料。這份考試**不要求同學寫註解**，只要達到題幹要求即可。若遇到不會的題目，請寫下您知道的所有程式碼來獲取部分分數，**千萬不要空白** 。作業皆通過且本測驗分數達 **50 分**(含) 以上者，得領取 1,000 元獎學金 ; 作業皆通過且本測驗分數達 **70 分(含)** 以上者，得領取 SC101 圖靈證書 。

測驗時間

您有 180 分鐘完成本次測驗。請先瀏覽題目、思考策略、再進行考試。請務必把握時間，不要在一題卡太久（先完成其他題目再回頭思考，有時會幫助您找到靈感）

Acknowledgement

請將底線文字在 Google doc 上方重填一遍

“我保證不會接受不該得到的幫助，並遵守上述考試規則”

	Score
Problem1	_____ / 20
Problem2	_____ / 15
Problem3	_____ / 15
Problem4	_____ / 10
Problem5	_____ / 15
Problem6	_____ / 15
Problem7	_____ / 10

Problem 1: Class Design (20 points)

臉書創辦人 Mark Zuckerberg 在哈佛大學大二時寫出了 Facebook 的原型。經過 stanCode 課程訓練的各位同學現在也可以寫出一個微型臉書（真可惜，如果我們早一點學，臉書創辦人就會是 stanCode 學生了！）Pamphlet 的中文翻譯為「小冊子」，因此我們要寫出一個名為 FacePamphlet 的程式，模擬「微型臉書」擁有的所有功能

FacePamphlet 的每個 instance 記錄著下列資訊：

- 該使用者用戶名稱 (str)
- 該使用者的朋友名單 (list[FacePamphlet])
- 該使用者動態歷史 (dict{int: str})

當使用者創造一個新帳戶時，必須對 FacePamphlet 輸入用戶名稱 (如下程式碼所示)：

jerry = FacePamphlet (‘YangHung Liao’)

（**jerry** 剛被製造出來時並沒有任何朋友或動態歷史）
FacePamphlet 這個 class 應該包含六個 instance methods（如下所示）

def post(self, message):

```
"""
Input:
    message (str): Add message to the dictionary that stores history of status.
                   Print out the username followed by ”posted \message\”
"""
```

def set_username(self, new_name):

```
"""
Input:
    new_name (str): Changes the old username to new_name
"""
```

def add_friend(self, profile):

```
"""
Input:
    profile (FacePamphlet): A FacePamphlet instance that needs to be added
                            as a friend with
"""
```

def get_username(self):

"""
Returns:
One of the instance variables (str) that stores the username
"""

def get_posts_history(self):

"""
Returns:
A dictionary containing keys of type int, values of type str. Keys indicate the order of posts; values hold each status. Let's say an instance had called **post('sleeping'), post('coding'), post('lifting weights')**, **get_posts_history( )** should return a dict which looks like:
{1: 'sleeping', 2: 'coding', 3: 'lifting weights'}
"""

def get_friends(self):

"""
Returns:
If an instance had friends in a list, [tom, jerry, bob], **get_friends( )** returns **['Tom' , 'Jerry', 'Bob']**
"""

為了讓同學更清楚地了解每個 instance methods, 我們以先前製造出的 **jerry** 來舉例

- **jerry.post('Coding...')** 會將 'Coding...' 加入儲存動態歷史的 Python dictionary; 並在 Console 印出 **YangHung Liao posted \ Coding... ** 的字樣 (" \ " 字母需要特別處理)

- **jerry.set_username('Magenta Liao')** 會將 **jerry** 的使用者名稱從先前的 'YangHung Liao' 更新成 'Magenta Liao' ，並在 Console 印出 **Name changed from "YangHung Liao" to "Magenta Liao"** 的字樣。請注意："YangHung Liao" 與 "Magenta Jerry" 的雙引號可能需要特別處理

```
def main():

    # post and set_username methods
    jerry = FacePamphlet('YangHung Liao')
    jerry.post('Coding...')
    jerry.set_username('Magenta Liao')
```

圖五、使用 post 與 set_username 之示意圖

```
YangHung Liao posted \ Coding... \
Name changed from "YangHung Liao" to "Magenta Liao"
```

圖六、執行圖五在 Console 得到的結果

- 假設用戶創建了新帳戶，**bob = FacePamphlet('Robert')**，Console 上不會有任何文字。但若使用者呼叫 **print(bob.get_username( ))** 這時 Console 會出現 **Robert** 的字樣

- 當使用者呼叫 **jerry.add_friend(bob)**，**bob** 會被加入 **jerry** 朋友名單，使用者也會在 Console 看到 **Magenta Liao added: Robert** 的字樣

```
def main():

    # post and set_username methods
    jerry = FacePamphlet('YangHung Liao')
    jerry.post('Coding...')
    jerry.set_username('Magenta Liao')

    # get_username and add_friend methods
    bob = FacePamphlet('Robert')
    print(bob.get_username())
    jerry.add_friend(bob)
```

圖七、接續使用 get_username 與 add_friend 之示意圖

```
YangHung Liao posted \ Coding... \
Name changed from "YangHung Liao" to "Magenta Liao"
Robert
Magenta Liao added Robert
```

圖八、執行圖七在 Console 得到的結果

- 假設使用者呼叫了 **jerry.post('Still Coding!!')**，Console 會印出 **Magenta Liao posted \ Still Coding!! ** 的字樣。如果之後再呼叫 **posts_dict = jerry.get_posts_history( )**，這時使用者會到一個內容為 **{1: 'Coding…', 2: 'Still Coding!!'}** 的 **posts_dict** (如圖十所示)

- 假設使用者創建了新帳戶，**bill = FacePamphlet('William')**，這時如果先呼叫 **jerry.add_friend(bill)**，Console 會先出現 **Magenta Liao added: William**。若我們接著呼叫 **print(jerry.get_friends( ))**，Console 就會出現 **[Robert, William]** 的字樣 **(請小心處理，這邊有陷阱!)**

```
def main():

    # post and set_username methods
    jerry = FacePamphlet('YangHung Liao')
    jerry.post('Coding...')
    jerry.set_username('Magenta Liao')

    # get_username and add_friend methods
    bob = FacePamphlet('Robert')
    print(bob.get_username())
    jerry.add_friend(bob)

    # get_status_history and get_friends
    jerry.post('Still Coding!!')
    posts_dict = jerry.get_posts_history()
    print(posts_dict)

    bill = FacePamphlet('William')
    jerry.add_friend(bill)
    print(jerry.get_friends())
```

圖九、接續使用 get_posts_history( ) 與 get_friends( ) 之示意圖

```
YangHung Liao posted \ Coding... \
Name changed from "YangHung Liao" to "Magenta Liao"
Robert
Magenta Liao added Robert
Magenta Liao posted \ Still Coding!! \
{1: 'Coding...', 2: 'Still Coding!!'}
Magenta Liao added William
[Robert, William]
```

圖十、執行圖九在 Console 得到的結果

您將於第 6 與第 7 頁完成 FacePamphlet，並讓這個 class 製造出來的 instances 都能順利達成上述工作 (撰寫上述 instance methods 時**不需寫註解**)。請注意：使用者輸入 constructor 的使用者姓名應該是 **private instance variable**；也就是說，使用者若要更改或得到該帳戶姓名只能透由 setter - **set_username(...)** - 或 getter - **get_username( )**

***class* FacePamphlet:**

```
    # Your Code Here
```

Problem 2: Nested Data Structures (15 points)

A. 一篇文章出現最多次的文字通常可以透露該文章主題。因此，史丹叩德日報想雇用各位做大數據分析！但有一件事情需要各位資料科學家注意：一篇文章中的 [‘and’, ‘a’, ‘the’, ‘or’, ...] 並沒有辦法提供任何資訊。因此，我們已將這些我們不感興趣的單字存入一個名為 **common_words** 的 Python list 裡

請在第 10 頁完成 **def get_word_frequencies(filename, common_words)** 將 filename 檔案裡的每一個單字出現次數整理成一個 Python dictionary 並 return 出來

以下 4 個重點提醒：

- **filename** (str): 該文章之檔名。該檔案裡的文字都**沒有任何標點符號** (如圖二)
- **common_words** (list[str]): 存入我們不感興趣的單字，如 ‘and’, ‘a’, ‘the’, …
- **returns** (dict({str, int})): 儲存每一個單字出現的次數。Key (str): 文章中的單字

  Value (int): 該單字出現的次數
- Key 應該為 **case-insensitive:** 舉例來說，**Science** 跟 **science** 應該視為一樣的

圖二為名為 **‘cs.txt’** 的檔案（若同學把檔案打開看就會發現文字都沒有包含任何標點符號）；而圖三為名為 **common_words** 的 Python list，儲存無法提供文章主題資訊的英文單字

```
1 Computer Science
2 is a fun science
```

圖二、檔案 ‘cs.txt’ 的文字內容

```
common_words = [‘is’, ‘a’, ‘the’]
```

圖三、common_words 的 list 內容

若將上圖二方匡中的文字檔與 **common_words** 丟入：
get_word_frequencies (‘cs.txt’, common_words) 應該會 return 下圖 dictionary:

{‘computer’ : 1, ‘science’: 2, ‘fun’: 1}

B. 完成上方題目 A. 的程式後，我們將某神秘文字檔 **'stanCode.txt'** 與 **common_words** 輸入 **get_word_frequencies** ，並得到了名為 **word_counts** 的 Python dictionary
（如下圖四所示）：

```
word_counts = get_word_frequencies('stanCode.txt', common_words)
# word_counts == { 'stancode': 100, 'fun': 8, 'jerry': 3, 'coding': 1 }

n_most_list = get_n_most_frequent(word_counts, 2)
# n_most_list == ['stancode', 'fun']
```

圖四、get_n_most_frequent 輸入與輸出內容示意圖

請各位完成第 11 頁的 **def get_n_most_frequent(word_counts, n)** ，將 **word_counts** 中前 **n** 高頻的單字裝到一個 Python list 並 return 出來

以下三點請注意：

- 您可以假設文章單字數目遠遠大於 n
- 您可以假設前 n 高頻的單字**出現次數不重複**。也就是說，前 n 高頻單字的出現次數都是獨一無二的 (如圖四所示：前 2 高頻的字出現次數為 100 以及 8)
- word_counts 一定是對的！也就是您 **Problem 2-A** 是否答對完全不影響這題（但相信大家一定都會寫對的！）

請將答案寫在第 11 頁

```
def get_word_frequencies(filename, common_words):
    """

    Input:
        filename (str): name of the file to be processed
        common_words (list[str]): list of words to be ignored
    Returns:
        word_counts (dict{str: int}): the # of times each word appears

    """

    # Your Code Here
```

```
def get_n_most_frequent(word_counts, n):
    """

    Input:
        word_counts (dict{str: int}): the # of times each word appears
        n (int): the # of top frequent words of interest
    Returns:
        n_most_list (list[str]): a list containing top-n words

    """

    # Your Code Here
```

Problem 3: Python Code Trace (15 points)

A. 請問下圖程式碼在 Console 印出的文字與數字為何？請符合格式，將所有執行此程式後會看到的一切文字與數字寫在此頁右方的框框中：

```
def code_tracing( ):
    a=[0]
    b=101
    c={123: 456, 789: 101}
    if a[0]:
        print('Answer1:', b+c[123])
    else:
        print('Answer2:', b+c[789])

    mystery(b, a, a[0], c)

    if a[0]:
        print('Answer3:', c)
    else:
        print('Answer4:', c)

def mystery(b, a_0, a, c):
    b += 1
    a_0[0] += 1
    a -= 1
    c[123] = 101
    print('Answer5:', b, a_0, a)

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

```
# Your Answer Starts Here
```

B. 為了出期末考題的 Jerry 翻出塵封已久的史丹佛上課筆記。然而，筆記不僅少了一行， Jerry 還忘記當時為何在筆記上方寫了 "**Bubble sort**" 的註解。已知 sort 的中文為「排序」，因此，**lst** 這個 Python list 經過 **bubble_sort(lst)** 之後應該會由小到大排列。有了這些基本知識，再來就要請各位同學解密 Bubble sort 這個著名的演算法

```
#3. TODO Explain why it is called "Bubble sort"
def bubble_sort(lst):

    n = len(lst)

    for i in range(n):

        #1. TODO Explain why its n-i-1
        for j in range(0, n-i-1):

            if lst[j] > lst[j+1]:
                [                    ]
                #2. TODO fill in the blank above
```

 # 1. 請問 for j in range(0, n-i-1) 使用 n-i-1 為上限的意義為何？

_____

 # 2. 若此演算法可以將 lst 元素由小到大排列，檔案最後的空格內應該要填入什麼程式碼？

_____

 # 3. 就您的了解，為何此演算法叫做 "Bubble sort"？

_____

以下三點請同學注意：

- 每一題答案請勿超過 25 字。只要把演算法概念重點清楚點出即可
- 請回答該程式碼在整個函示中扮演的角色，而非該程式碼單獨存在的功能為何。如 **for j in range(0, n-i-1)** 不能回答「將 j 變數值從 0 換到 n-i-1」
- ***HINT:*** 如果不知道怎麼下手，可以試著丟入 lst = [1, 4, 2, 7, 3] 進去 mystery 看看會發生什麼事情

Problem 4: Recursion (10 points)

請寫出一個遞迴函式 **def balanced_brackets(s)**，判斷一個文字的括弧是否成雙成對？舉例來說: 若 s1 = “( ) ( ) (( )) ( ) ( )“，則 **balanced_brackets(s1)** 會 return **True** ; 若 s2 = “(( )) ( ) ( ))” 則 **balanced_brackets(s2)** 會 return **False** ; 若 s3 = “( )(( )) (” 則 **balanced_brackets(s3)** 會 return **False** (如下圖所示)

```
s1 = '()()(())()()'
print(balanced_brackets(s1))   # True
s2 = '(())()())'
print(balanced_brackets(s2))   # False
s3 = '()(()))('
print(balanced_brackets(s3))   # False
```

以下四點請同學注意：

1. 您可以假設 s 一定**不是空字串**，且 s 裡面只會包含 **‘)’** 或是 **‘(’**
2. 請務必**使用 recursion**！不然此題無法得分
3. 小心當 **‘)’** 落單的情況
4. 請將答案撰寫在 15 頁

```
def balanced_brackets(s: str) -> bool:
    """

    請注意這種新的 Python 註解寫法
    s 是變數名稱，冒號後面的 str 是 data type
    而最後的 -> bool 代表 return type 是一個 boolean

    """

    # Your Code Here
```

Problem 5: Backtracking (15 points)

身為 stanCode 大助教的 Dennis 無時無刻都在思考程式碼。有一天他站在他家一樓的樓梯準備爬上二樓時，突然靈光一閃💡想到一個 backtracking 的題目：「我一次的步伐可以跨 1 格、也可以跨 2 格，可不可以請學生在期末考的時候告訴我: 從一樓走到二樓總共有幾種走法呢 🤔 」

請完成第 17 頁名為 **def ways_to_climb(n)** 的遞迴程式，將所有可以剛好走 n 階台階的排列組合 print 出來。以下三點重點請同學注意：

- n 為大於 1 的正整數，代表一樓到二樓的所有台階數目
- Dennis 可以都走 1 格、也可以都走 2 格，但**一定要剛好**抵達 n 的台階數目
- 請將每一個可能的走法 print 在 Console。順序不是重點！我們只在意您的演算法有沒有辦法把所有排列組合都列出

若您呼叫 **ways_to_climb(4)** 您應該可以在 Console 看到下圖之文字＆數字：

```
[1, 1, 1, 1]
[1, 1, 2]
[1, 2, 1]
[2, 1, 1]
[2, 2]
```

若您呼叫 **ways_to_climb(3)** 您應該可以在 Console 看到下圖之文字＆數字：

```
[1, 1, 1]
[1, 2]
[2, 1]
```

def ways_to_climb(n):

```
    """
    :param n: int, the total number of stairs (always >= 1)
    """
```

Problem 6: LC203. Easy - Remove Linked List Elements (15 points)
Company : Google, Facebook, Uber

Remove all elements from a linked list of integers that have value ***val***.

Examples:

> **Input**: 1 -> 2 -> 6 -> 3 -> 4 -> 5 -> 6, **val** = 6
>
> **Output**: 1 -> 2 -> 3 -> 4 -> 5

> **Input**: 1 -> 1 -> 1, **val** = 1
>
> **Output**: **None**

Each node of the linked list **head** in the function **removeElements** is shown as follows:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
```

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next


def removeElements(head: ListNode, val: int) -> ListNode:

    """
    請注意這種新的 Python 註解寫法
    head 是變數名稱，冒號後面的 ListNode 是 data type
    而最後的 -> ListNode 代表 return type 是一個 ListNode
    """

    # Your Code Here
```

Problem 7: Short Answer Questions (10 points)

A. 請問該如何寫出一行 code 讓 **points = [(1, 3), (10, 0), (3, 2), (5, 4)]** 變成 **[(5, 4), (1, 3), (3, 2), (10, 0)]?**

_____

B. 請問什麼是 static method? 他跟 instance method 差在哪裡?（30 字以內）

_____

_____

C. 請問何為 stack overflow? (答案控制在15字內)

_____

D. Python list 可以表現成 Stack 也可以表現成 Queue。請問什麼是 Stack 的特色？什麼是 Queue 的特色？ 若要反轉輸出順序，會選哪一個？若要使用 Breadth-first search 演算法，又會選哪一個？(答案控制在20字內)

_____

_____

E. 請問 BFS 與 DFS 兩個演算法有什麼優劣？（30 字以內）

_____

_____

You have reached the end of the exam.