stanCode

標準程式教育機構

Assignment 2

This assignment is based on the Assignment 2 of CS106A at Stanford University

作業檔案

這份作業將訓練各位同學運用 Python 程式最重要的基本技能：基本的數學運算式(Expressions)、 與使用者互動之對話框 (Console)、以及各式迴圈 (Loops)。

此份作業估計需要時間為 7 小時。

如果作業卡關 **歡迎與助教討論**，stanCode 也非常鼓勵同學們互相討論作業的概念，**但請不要把自己的 code 給任何人看**，分享您的 code 會剝奪其他學生獨立思考的機會，同時會導致其他學生的程式碼與您的 code 極度相似，使得防抄襲軟體認定有抄襲之嫌疑。

請注意：本份作業請勿使用字串處理(string　manipulation)或其他資料結構

Problem 1 - quadratic_solver.py

假設我們有一個二次函數如下圖方程式所示（**a 不等於零**）

$$ax^2+bx+c=0$$

只要使用者給定 a, b, c 三個數值，我們就可以依照下方公式計算出此方程式的根 roots（也就是下方 x 的值）

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

在上方公式的根號裡面的 $b^2 - 4ac$ 我們稱之為判別式（discriminant）

1. 如果 判別式 **大於零**，此方程式會有**兩個 real roots**（兩個 x 的值），同上方算式的兩組數值
2. 如果 判別式 **等於零**，我們只會有**一個 root**（一個 x 的值）
3. 如果 判別式 **小於零**，我們要告訴使用者 "**No real root**s"

請寫出一個可以完美重現下方三個圖例之程式（由左至右，分別代表 2 real roots, 1 root, 以及 No real roots）。 **請注意：要產生以下這三張圖，程式必須重新執行(re-run) 三次，同時請小心數學運算先乘除後加減的規則**。

首先，在 Console 顯示中，請告訴使用者此程式名稱（"**stanCode Quadratic Solver!**"）。然後請分別提供三行文字讓使用者能夠輸入 a, b, c 相對應之數值。

若您要計算開根號（√），請使用指令 **math.sqrt**。舉例來說，若我們要計算x值的開根號，並將這個開根號的數值存到 y，我們會使用以下指令:

y = math.sqrt (x)

請注意：**math.sqrt** 來自我們在程式最上方的 **import math**，同學們請直接使用

```
stanCode Quadratic Solver!
Enter a: 1
Enter b: -3
Enter c: -4
Two roots: 4.0, -1.0
```

```
stanCode Quadratic Solver!
Enter a: 1
Enter b: 6
Enter c: 9
One root: -3.0
```

```
stanCode Quadratic Solver!
Enter a: 2
Enter b: 4
Enter c: 6
No real roots
```

Problem 2 - hailstone.py

Douglas Hofstadter 獲得普立茲獎的得獎著作 《Gödel, Escher, Bach》裡面有許多有趣的數學謎題（很多問題都可以用電腦程式來計算）。

在書中，Hofstadter 提到，選一正整數 n ，重複以下指令，直到 n 變成 1：

- 如果 n 是奇數，把 n 乘 3 再加 1
- 如果 n 是偶數，對 n 除 2

例如，在 Hofstadter 著作中第 401 頁中提到，「若一開始選定的 n 為15」，其演算過程會如下呈現：

```
15  is odd, so I make 3n+1:  46
46  is even, so I take half:  23
23  is odd, so I make 3n+1:  70
70  is even, so I take half:  35
35  is odd, so I make 3n+1:  106
106  is even, so I take half:  53
53  is odd, so I make 3n+1:  160
160  is even, so I take half:  80
80  is even, so I take half:  40
40  is even, so I take half:  20
20  is even, so I take half:  10
10  is even, so I take half:  5
5  is odd, so I make 3n+1:  16
16  is even, so I take half:  8
8  is even, so I take half:  4
4  is even, so I take half:  2
2  is even, so I take half:  1
```

非常有趣的是，在目前數學家測試過的所有數字裡，不管過程中 n 的值上上下下起伏了幾次，最後一定會回到 1。這個概念就好像是一顆冰雹（hailstone）不斷被風盤旋，起起伏伏最後落至地面（希望沒有砸到任何東西）。因此，從 n 到 1 的這個數列又被稱為冰雹序列 Hailstone Sequence。

現在請同學寫出一個程式，可以讓使用者輸入任意整數，並產生 Hailstone Sequence。如書中的圖例，所有在抵達 1 之前的數字都會被列舉出來。同時，這個序列中從 n 到 1 之間運算的次數（steps）也會被紀錄。因此，如下圖所示，您所完成的程式應該要能**完美重現下圖內所有文字與數字**：

```
This program computes Hailstone sequences.

Enter a number: 17
17 is odd, so I make 3n+1: 52
52 is even, so I take half: 26
26 is even, so I take half: 13
13 is odd, so I make 3n+1: 40
40 is even, so I take half: 20
20 is even, so I take half: 10
10 is even, so I take half: 5
5 is odd, so I make 3n+1: 16
16 is even, so I take half: 8
8 is even, so I take half: 4
4 is even, so I take half: 2
2 is even, so I take half: 1
It took 12 steps to reach 1.
```

若使用者 recompile 後直接輸入 1，程式會出現 0 steps 的情況（如下圖所示）。

```
This program computes Hailstone sequences.

Enter a number: 1
It took 0 steps to reach 1.
```

同學完成後，可以試試看 27 這個數字會花你們幾步，才能達到 1 呢？

Problem 3 - 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！

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：
(1) **最高溫**是多少？ (2) **最低溫**是多少？ (3) **平均溫度**是多少？ (4) 以及有幾天可以發佈「**低溫警報**」（小於16度但不包含16度）

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( ) 上方的常數 **constant**，就好像在課程中練習的終極密碼時輸入的答案數字一樣。您所完成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。例如 print("SC001 \" Weather") 就會印出 **SC001 "Weather** 。

如同本題目一開始的敘述，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 main( ) 上方的**常數 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 ）而這不會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 temperatures 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 4 - prime_checker.py

有一天 Jerry 在教弟弟寫功課，弟弟突然問：「到底什麼是質數啊？」Jerry 說：「就是一個大於等於 1 的正整數除了 1 跟那個數本身之外找不到其他數字可以整除它」。弟弟搔搔頭、似懂非懂地接著問：「那如果我這邊有一些數字，你可以請你的學生協助我判斷它們是不是質數嗎？」Jerry 說：「好！給我二十分鐘，我請我的學生來寫一個程式！」

為了讓Jerry的弟弟更了解質數，請同學完成 **prime_checker.py**，寫出一個讓使用者可以反覆輸入數字並判斷該數字是否為質數的程式。如上圖所示，您所完成的程式應該能完美重現每一行文字與數字。

```
Welcome to the prime checker!
n: 17
17 is a prime number.
n: 30
30 is not a prime number.
n: -100
Have a good one!
```

以下三點請同學注意:

1. 一開始請印出 'Welcome to the prime checker!'
2. 您可以假設使用者在 "n: " 後面輸入的數字一定是大於 1 的正整數，不需要做任何 error checking
3. 最後，請在 main function 上方，設定一個讓使用者可以輸入「使程式離開的值」，當使用者輸入這個EXIT常數時（如上圖中的 -100），那程式就要印出 "Have a good one!" 並結束。

如果同學完成以上 Assignment2 作業內容，歡迎撰寫以下 Extension 作業。
若同學完成全部 Extension 題目，就有機會獲得++喔！但請同學注意，所有Extension題目都**不能使用字串處理(string manipulation)或其他資料結構**。

Extension 1 - factorial.py

stanCode接獲神秘計算機協會的請求，想請各位同學協助完成一個可以計算階乘 (factorial) 的程式。請同學完成 **factorial.py**，寫出一個可以讓使用者反覆輸入數字(您可以假設這些數字一定為正整數)，並得出這些數字的階乘答案。如下圖所示，您所完成的程式應該要能**完美重現下圖內所有文字與數字**：

```
Welcome to stanCode factorial master!
Give me a number, and I will list the answer of factorial: 3
Answer: 6
Give me a number, and I will list the answer of factorial: 5
Answer: 120
Give me a number, and I will list the answer of factorial: 10
Answer: 3628800
Give me a number, and I will list the answer of factorial: 7
Answer: 5040
Give me a number, and I will list the answer of factorial: 2
Answer: 2
Give me a number, and I will list the answer of factorial: 15
Answer: 1307674368000
Give me a number, and I will list the answer of factorial: -100
- - - - - - See ya! - - - - - -
```

以下兩點請同學注意:

1. 一開始請印出 'Welcome to stanCode factorial master!'
2. 最後，請在 main function 上方，設定一個讓使用者可以輸入「使程式離開的值」，當使用者輸入這個 EXIT 常數時 (如上圖中的 -100) ，那程式就要印出 "- - - - - - See ya!----------" 並結束程式。

Extension 2 - number_checker.py

今天如果我們把一個整數所有的真因子(即除了自身以外的因數)全部加起來，

1. **真因子的和** 如果**等於**這一個**整數**，則這個整數就可以被稱為**"完全數"** **(Perfect number)**。
2. **真因子的和**如果**大於**這一個**整數**，則我們會將這個整數稱為**"盈數"** **(abundant number)**。
3. 最後，**真因子的和**如果**小於**這一個**整數**，則我們會將這個整數稱為**"虧數"** **(deficient number)**。

舉例來說，

完全數 (Perfect number) 的例子有 **6** 跟 **28**：

- 數字 **6** 的因數有 **1、2、3、6**，扣除 **6** 本身，**1+2+3 == 6**，因此 6 為完美數。
- 數字 **28** 的因數有 **1、2、4、7、14、28**，除去數字 **28** 本身外，其餘5個數相加，**1+2+4+7+14 == 28**，因此 28 為完美數。

盈數 (abundant number) 的例子有 **12** 跟 **20**：

- 數字 **12** 的因數有 **1、2、3、4、6、12**，扣除 **12** 本身，**1+2+3+4+6 == 16**，16 大於 12，因此 12 為盈數。
- 數字 **20** 的因數有 **1、2、4、5、10、20**，將除去數字 **20** 自己以外的五個數字加起來，**1+2+4+5+10 == 22**。22 大於 20，因此 20 為盈數。

虧數 (deficient number) 的例子有 **15** 跟 **27**：

- 數字 **15** 的因數有 **1、3、5、15**，扣除 **15** 本身，**1+3+5 == 9**，9 小於 15，因此 15 為虧數。
- 數字 **27** 的因數有 **1、3、9、27**，將除去數字 **27** 自己以外的三個數字相加，**1+3+9 == 13**。13 小於 27，因此 27 為虧數。

請同學完成 **number_checker.py**，寫出一個可以讓使用者反覆輸入數字(您可以假設數字一定為正整數)，並判斷該數字為完全數、盈數還是虧數的程式。

如下圖所示，您所完成的程式應該能完美重現圖中的每一行文字與數字。

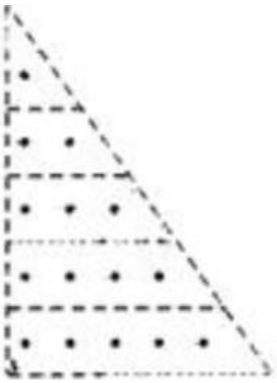
```
Welcome to the number checker!
n: 6
6 is a perfect number
n: 15
15 is a deficient number
n: 8128
8128 is a perfect number
n: 12
12 is an abundant number
n: 79
79 is a deficient number
n: 945
945 is an abundant number
n: -100
Have a good one!
```

以下三點請同學注意:

1. 一開始請印出 'Welcome to the number checker!'
2. **您可以假設使用者在 "n: " 後面輸入的數字一定是大於 1 的正整數**，不需要做任何 error checking
3. 最後，請在 main function 上方，設定一個讓使用者可以輸入「使程式離開的值」，當使用者輸入這個 EXIT 常數時（如上圖中的 -100），那程式就要印出 "Have a good one!" 並結束。

Extension 3 - triangular_checker.py

如果現在有一個正整數，我們拿跟這個整數一樣多的石頭排在地上，若是能在等距離的排列下可以形成一個等邊三角形(如下圖)，則這個數就可以被視為 **“三角形數” (triangular number)**。



$1 = 1$

$1 + 2 = 3$

$1 + 2 + 3 = 6$

$1 + 2 + 3 + 4 = 10$

$1 + 2 + 3 + 4 + 5 = 15$

我們可以透過**下列公式**來找到三角形數， n代表是第n個三角形數。

$$Tn = n(n + 1)/2$$

舉例來說，

- **3** 是第 **2** 個三角形數，因為他符合：2(2+1) / 2 == 3
- **15** 是第 **5** 個三角形數，因為他符合：5(5+1) / 2 == 15
- **28** 是 **7** 個三角形數，因為他符合：7(7+1) / 2 == 28

請同學完成 **triangular_checker.py**，寫出一個讓使用者可以反覆輸入數字並判斷該數字是否為三角形數的程式。

如下圖所示，您所完成的程式應該能完美重現圖中的每一行文字與數字。

```
Welcome to the triangular number checker!
n: 3
3 is a triangular number
n: 17
17 is not a triangular number
n: 171
171 is a triangular number
n: 12
12 is not a triangular number
n: 21
21 is a triangular number
n: 28
28 is a triangular number
n: 40
40 is not a triangular number
n: 371
371 is not a triangular number
n: -100
Have a good one!
```

以下兩點請同學注意:

1. 一開始請印出 'Welcome to the triangular number checker!'
2. 最後，請在 main function 上方，設定一個讓使用者可以輸入「使程式離開的值」，當使用者輸入這個 EXIT 常數時（如上圖中的 -100），那程式就要印出 "Have a good one!" 並結束。

Extension 4 - narcissistic_checker.py

在數學理論的世界中，如果一個**正整數(N位數)**的**所有位數**的**N次方**的**和**剛好**等於自己**，那他就會被稱為 **水仙花數 （ Narcissistic number ）** ！

水仙花數（Narcissistic number），又稱超完全數字不變數（pluperfect digital invariant, PPDI）、自戀數、自冪數、阿姆斯壯數（Armstrong number），之所以稱為阿姆斯壯數，是因為據說這個概念最早是由阿姆斯壯提出的！

舉例來說：

- **153 == 1^3 + 5^3 + 3^3** ， 因此 153 是水仙花數
- **1634 == 1^4 + 6^4 + 3^4 + 4^4** ， 因此 1634 是水仙花數
- **92727 == 9^5 + 2^5 + 7^5 + 2^5 + 7^5** ， 因此 92727 是水仙花數

請同學完成 **narcissistic_checker.py**，寫出一個讓使用者可以反覆輸入數字並判斷該數字是否為水仙花數的程式。如下圖所示，您所完成的程式應該要能**完美重現下圖內所有文字與數字**：

```
Welcome to the narcissistic number checker!
n: 153
153 is a narcissistic number
n: 1634
1634 is a narcissistic number
n: 92727
92727 is a narcissistic number
n: 120
120 is not a narcissistic number
n: 11
11 is not a narcissistic number
n: -100
Have a good one!
```

以下四點請同學注意:

1. 一開始請印出 'Welcome to the narcissistic number checker!'
2. 最後，請在 main function 上方，設定一個讓使用者可以輸入「使程式離開的值」，當使用者輸入這個 EXIT 常數時（如上圖中的 -100），那程式就要印出 "Have a good one!" 並結束。
3. 在 Python 的 n 次方 寫法為 **n , 舉例來說如果是 2 的 3 次方, 會是 2**3
4. 提示：這題建議使用 // 與 % ！

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、並確保程式並沒有卡在任何的無限迴圈（infinite loop）之中。

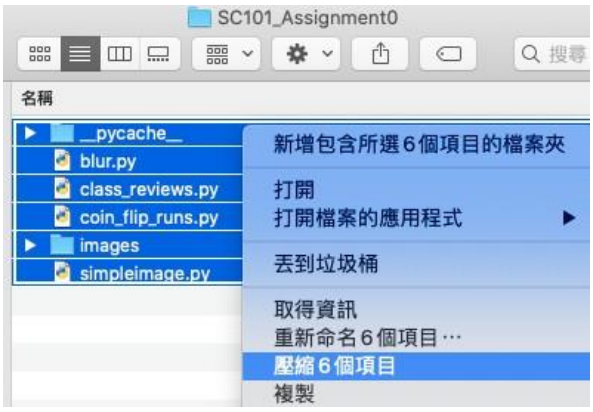
Style - 如同在課堂上所說，好的程式要有好的使用說明（comment），也要讓人一目瞭然，這樣全世界的人才能使用各位的 code 去建造更多更巨大更有趣的程式，因此請大家寫出**精簡扼要**的 main( ) 程式概要、function comments 和單行註解。

作業繳交

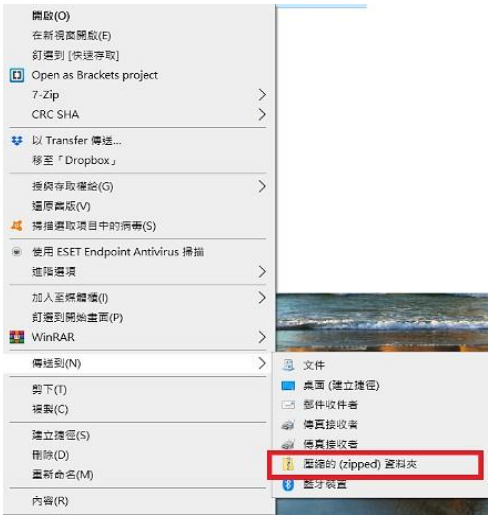
恭喜您完成 Assignment2！請同學於**作業繳交期限前**，依照下圖將您完成的作業的**下載連結**上傳至社團提供的**作業繳交表單**。

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。

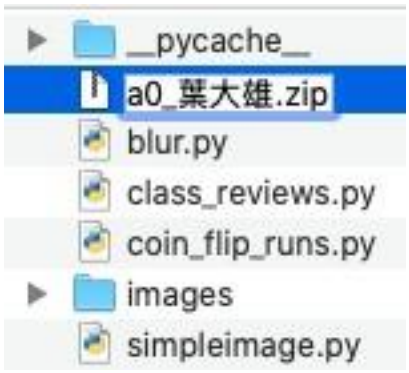
macOS：按右鍵選擇「**壓縮n個項目**」



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.