Python 程式設計: Test 01, 2019

- 1. 除了電腦之外, 不可使用 手機, 平板 或智慧手錶 等裝置
- 2. 在電腦上可使用 email 繳交檔案, 但不可使用其他通訊軟體。
- 3. 需按每一題題目指定檔名存檔。
- 4. 違反規定者,以零分計算。

繳交: 所有.py 檔壓縮成 一個 zip or rar 檔

檔名: 你的學號-test01.zip (例如 s1061234-test01.zip)

寄送: yzu.python@gmail.com

主旨: 你的學號-test01 檔案 (例如 s1061234-test01 檔案)

同時 CC 給你 自己

題目 1(40%)

試撰寫一個程式,可以根據輸入的正整數 n,產生 2n 層沙漏圖形 (參照範例)。

(Hint: 觀察 空格與*的個數)

存檔: test01_Q1.py

輸出範例:

Please enter a number: 6

*

*

題目 2 (50%)

試撰寫一個程式,計算同學的修課平均成績。 credit_list 是課程學分數參考表, grade_dict 是學生科目成績表範例。平均成績 = (sum of 各科成績×各科學分數)/修課總學分數

請用亂數產生器 模擬 20 位同學的 grade_dict。他們學號範圍在 s1071201 - s1071299, 不可重複, 而且不是連號。 每位同學有六科成績,每科成績在 60 - 99 (含 99), 成績可重複出現。

注意: 由於學號與成績是用亂數產生, 每次程式執行的結果都會不太一樣。

- (1) 印出所有(20位)同學的修課成績與平均成績
- (2) 列出 Top3 同學的學號與平均成績

存檔: test01_Q2.py

開課程學分數

 $credit_list = [3, 3, 3, 2, 2, 3]$

學生科目成績表 (範例)

輸出範例:

Student Scores and Average:

#s1071202: [68, 69, 87, 64, 68, 77], 72.94 #s1071203: [96, 94, 95, 94, 81, 93], 92.75 #s1071206: [99, 60, 63, 63, 69, 61], 69.56 #s1071213: [62, 83, 95, 63, 87, 63], 75.56 #s1071214: [62, 83, 98, 72, 61, 70], 75.31

```
#$1071228: [87, 61, 64, 75, 80, 78], 73.75

#$1071231: [98, 99, 91, 81, 66, 97], 90.56

#$1071232: [91, 97, 77, 98, 70, 98], 89.06

#$1071237: [93, 74, 74, 76, 96, 93], 84.12

#$1071246: [98, 66, 98, 61, 64, 71], 78.06

#$1071260: [78, 87, 98, 71, 78, 86], 84.06
```

...

Top 3 performed students

#s1071203: 92.75 #s1071231: 90.56 #s1071232: 89.06

問題 3:(20%)

下列拼字遊戲程式範例 (Python 學運算思維 Chapter 19), 如果我們把字母牌 變數 (tiles) 的內容改成 "hijklmnopcrsatg", 這個程式還是找不到 innk, crosshatching 這類字母有重複的單字。

請修改程式, 讓它可以輸出 ('innk', 'ink', 'crosshatching') 這三個字。

存檔: test01_Q3.py

words = """art hue ink innk

crosshatching

" " "

tiles = "hijklmnopcrsatg"

all_valid_words =()

start = 0

end = 0

found_words =()

for char in words:

```
if char == "\n":
          all_valid_words = all_valid_words +(words[start:end], )
          start = end + 1
          end = end + 1
     else:
          end = end + 1
for word in all_valid_words:
     tiles_left = tiles
     for letter in word:
          if letter not in tiles_left:
               break
          else:
               index = tiles_left.find(letter)
               tiles_left = tiles_left[:index]+tiles_left[index+1:]
          if len(word)== len(tiles)-len(tiles_left):
               found_words = found_words +(word, )
print(found_words)
```