Programming Fundamentals I Lab.

7. การพัฒนาซอฟต์แวร์ขนาดใหญ่

ชื่อ_____ รหัสนิสิต_____

ในปฏิบัติการนี้ คุณจะได้ฝึกทักษะการพัฒนาซอฟต์แวร์ขนาดใหญ่ ที่มีความซับซ้อนมาก ผ่านการสร้างเกม 2048 ในรูปแบบที่เป็น text mode

คุณสามารถศึกษาวิธีการเล่นเกม 2048 ได้จาก `https://en.wikipedia.org/wiki/2048_(video_game)`

7.1 การทำงานหลัก

ในขั้นตอนแรก ให้คุณสร้างไฟล์ .py ขึ้นมาใหม่ เขียนโค้ดดังนี้

```
table = [[0, 0, 0, 0],
         [0, 0, 0, 0],
         [0, 0, 0, 0],
         [0, 0, 0, 0]]

random_add(table)
show_table(table)
while not game_end(table):
    d = input('Enter direction code:')
    merge(table,d)
    random_add(table)
    show_table(table)
```

เราจะใช้ `table` ซึ่งเป็น list ที่มีสมาชิก 4 ตัว โดยสมาชิกแต่ละตัวเป็น list ของจำนวนเต็ม 4 ตัว ในการเก็บข้อมูลของตารางเกมที่มีขนาด 4 แถว 4 หลัก โดยช่องที่ไม่มีตัวเลขจะให้มีค่าเป็น 0 ไปก่อน จงเขียนว่า จากโค้ดด้านบนนี้ มีฟังก์ชันใดบ้างที่เราจะต้องพัฒนาเพิ่มเติม หากฟังก์ชันใดต้องคืนค่า ให้บอกด้วยว่าต้องคืนค่าเป็นชนิดข้อมูลใด

จงอธิบายกระบวนการทำงานคร่าว ๆ ของเกม จากโค้ดหลักที่ได้เห็น

7.2 การแสดงตาราง การสุ่มเพิ่มตัวเลข และการตรวจสอบการจบเกม

ในส่วนนี้ เราจะให้คุณพัฒนาฟังก์ชันสามฟังก์ชันในเกม ได้แก่ `show_table()`, `random_add()`, และ `game_end()` ตามลำดับ ในขั้นตอนแรกให้คุณ comment โค้ดทั้งหมดไว้ก่อน เพื่อให้สามารถ run module และทดสอบหลังการสร้างแต่ละฟังก์ชันได้

1. จงสร้างฟังก์ชัน `show_table()` ไว้ในไฟล์เดียวกัน เหนือโค้ดหลักของโปรแกรม โดยให้ฟังก์ชันรับ list table ซึ่งมีลักษณะเป็น list ซ้อน list ที่ใช้เก็บข้อมูลตารางเกม และแสดงออกหน้าจอโดยมีขอบตารางตามตัวอย่างต่อไปนี้

```
>>> show_table([
                [0, 0, 16, 128],
                [0, 2, 8, 32],
                [4, 2, 32, 32],
                [0, 0, 0, 0]
                ])
+----+----+----+----+
|    |    |  16| 128|
+----+----+----+----+
|    |   2|   8|  32|
+----+----+----+----+
|   4|   2|  32|  32|
+----+----+----+----+
```

```
|    |    |    |    |
+----+----+----+----+
```

กล่าวคือ ให้จัดช่องว่างสำหรับตัวเลขแต่ละตัวให้ดี เพื่อให้เส้นขอบตารางเรียงกันเป็นแนวตรงสวยงาม

ข้อควรระวัง: ฟังก์ชันนี้ไม่ควรทำให้ตารางที่รับมามีการเปลี่ยนแปลง

2. จงสร้างฟังก์ชัน `random_add()` ไว้ใต้ฟังก์ชัน `show_table()` โดยให้ฟังก์ชันรับ list table มา และทำการสุ่มช่องที่มีค่าเป็น 0 มาช่องหนึ่ง และสุ่มเปลี่ยนตัวเลขในช่องนั้นให้เป็น 2 หรือ 4

คำแนะนำ: เรียกคำสั่ง `import random` เพื่อนำเข้าคำสั่งจากโมดูล `random` จากนั้นคุณสามารถใช้ฟังก์ชัน `random.choice(list)` ซึ่งจะสุ่มหยิบสมาชิกจาก `list` มาให้หนึ่งตัว

ตัวอย่างการทำงานเป็นดังนี้

```
>>> table = [[0, 0, 0, 0],
         [0, 0, 0, 0],
         [0, 0, 0, 0],
         [0, 0, 0, 0]]
>>> random_add(table)
>>> show_table(table)
+----+----+----+----+
|    |    |    |    |
+----+----+----+----+
|    |    |    |    |
+----+----+----+----+
|    |    |    |    |
+----+----+----+----+
|    |   4|    |    |
+----+----+----+----+
>>> random_add(table)
>>> show_table(table)
+----+----+----+----+
|    |    |    |    |
+----+----+----+----+
|    |    |    |   2|
+----+----+----+----+
|    |    |    |    |
+----+----+----+----+
|    |   4|    |    |
+----+----+----+----+
```

3. จงสร้างฟังก์ชัน `game_end()` ที่รับ list table ไป และคืนค่าเป็น `True` หากเกมสิ้นสุดแล้ว ไม่เช่นนั้นให้คืนค่าเป็น `False` โดยเกมจะสิ้นสุดถ้าเข้ากรณีใดกรณีหนึ่งในนี้

(a) มีช่องหนึ่งได้ 2048

(b) ไม่มีช่องว่างเหลืออยู่ และไม่สามารถรวมเลขต่อได้ (ไม่มีเลขเดียวกันอยู่ช่องติดกันอีกแล้ว)

ตัวอย่างการทำงาน เช่น

```
>>> game_end([
              [0, 0, 4, 16],
              [0, 2, 0, 0],
              [16, 32, 16, 8],
              [64, 32, 16, 8]
              ])
False
>>> game_end([
              [2, 8, 4, 16],
              [4, 2, 16, 128],
              [16, 64, 8, 2],
              [64, 32, 16, 8]
              ])
True
>>> game_end([
              [2, 8, 1024, 256],
              [4, 2, 0, 128],
              [16, 2048, 256, 2],
              [64, 0, 0, 0]
              ])
True
```

ข้อควรระวัง: ฟังก์ชันนี้ไม่ควรทำให้ตารางที่รับมามีการเปลี่ยนแปลง

7.3 การรวมเลข

มาถึงหัวข้อนี้ เราจะให้คุณสร้างฟังก์ชัน `merge()` ที่รับพารามิเตอร์สองตัวได้แก่ list table และสตริง `d` ที่ใช้แทนทิศทางในการรวมเลข โดยเราจะกำหนดดังนี้

- `'a'` หมายถึงให้รวมเลขไปด้านซ้ายของตาราง
- `'w'` หมายถึงให้รวมเลขไปด้านบนของตาราง
- `'d'` หมายถึงให้รวมเลขไปด้านขวาของตาราง
- `'s'` หมายถึงให้รวมเลขไปด้านล่างของตาราง

ให้ระวังขั้นตอนวิธีที่คุณใช้ในการรวมเลขให้ดี คุณอาจจะทดลองเล่นเกมดูก่อนเพื่อทำความเข้าใจกระบวนการ และในการสร้างฟังก์ชัน `merge()` นี้ คุณอาจแยกโค้ดบางส่วนออกเป็นฟังก์ชันย่อยเพื่อความสะดวกในการพัฒนาได้

ตัวอย่างการทำงาน เช่น

```
>>> table = [[0, 0, 4, 16],
             [0, 2, 0, 0],
             [16, 32, 16, 8],
             [64, 32, 16, 8]]
>>> merge(table, 's')
>>> show_table(table)
+----+----+----+----+
|    |    |    |    |
+----+----+----+----+
|    |    |    |    |
+----+----+----+----+
|  16|   2|   4|  16|
+----+----+----+----+
|  64|  64|  32|  16|
+----+----+----+----+
```

เมื่อสร้างและทดสอบการทำงานของฟังก์ชันทั้งหมดเสร็จเรียบร้อย ให้คุณทดลองเปิด comment โค้ดส่วนโปรแกรมหลัก และทดลองเกมโดยรวมทั้งหมดเพื่อทดสอบความถูกต้องอีกทีหนึ่ง โดยการเล่น ให้ผู้เล่นป้อนตัวอักษร a, w, d, s และกด enter เป็นการเลือกทิศทางในการรวมเลขในตาราง