กำหนดส่ง

Lab	
HW	
Until	

กระบวนวิชา 204111

การบ้านปฏิบัติการ 13

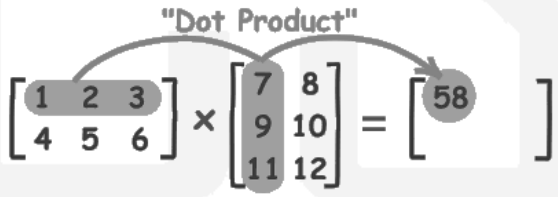
n-dimensional Lists and Nested Collections (20 คะแนน)

ข้อกำหนด

i. การเรียกใช้ฟังก์ชันเพื่อการทดสอบ ต้องอยู่ภายใต้เงื่อนไข `if __name__ == '__main__' :` เพื่อให้สามารถ `import` ไปเรียกใช้งานจาก Script อื่น ๆ ได้อย่างเป็นมาตรฐาน

ii. ทุกข้อต้องมีการสร้างฟังก์ชัน `my_id()` โดยให้คืนค่าสายอักขระแทนเลขประจำตัวนักศึกษา 9 หลัก

1) **4 คะแนน** (`Lab13_1_5XXXXXXXX.py`) **[Attachment]** ให้เขียนฟังก์ชัน `matrix_mult(`$m1$`, `$m2$`)` เพื่อทำการหาผลคูณของเมทริกซ์ $m1$ และ เมทริกซ์ $m2$ (wikipedia: https://goo.gl/S0DDZv) โดยฟังก์ชันจะทำงานแบบ Non-destructive กล่าวคือจะคืนค่าผลคูณที่ได้โดยไม่เปลี่ยนแปลงเมทริกซ์ $m1$ และ $m2$ ที่อยู่ในในรูปแบบ List สองมิติ ทั้งนี้หากไม่สามารถหาผลคูณได้ให้คืนค่า None



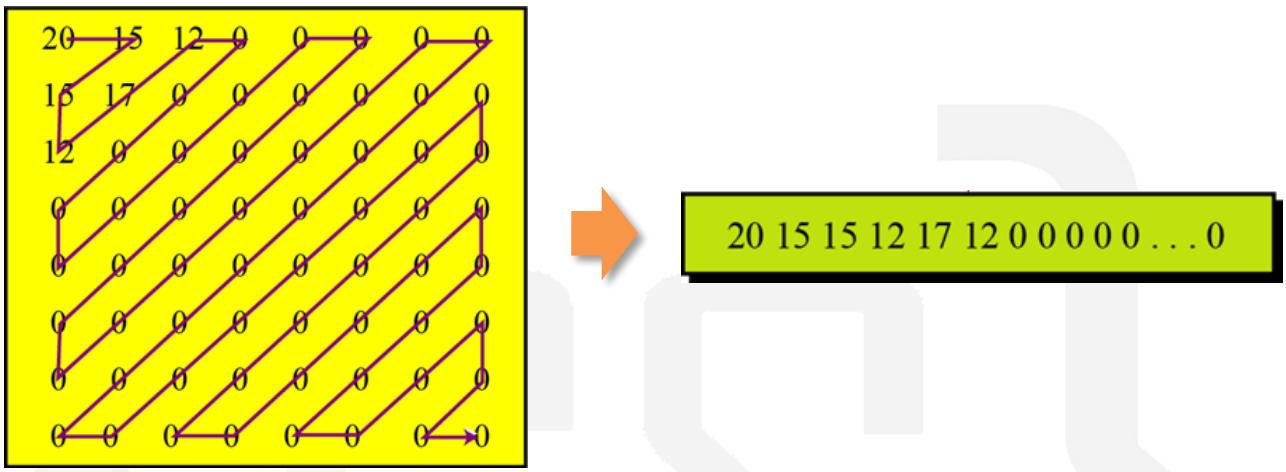
Input	Output

`[[1, 2, 3],`
` [4, 5, 6]]`
`[[7, 8],`
` [9, 10],`
` [11,12]]`
	`[[58, 64],`
	` [139, 154]]`

`[[1, 2, 3],`
` [4, 5, 6]]`
`[[7, 8, 5, 9, 3],`
` [9, 10, -3, 7, 13],`
` [11, 12, 6, 2, 9]]`
	`[[58, 64, 17, 29, 56],`
	` [139, 154, 41, 83, 131]]`

- การวิเคราะห์ปัญหา
  - Input: จำนวนข้อมูล_____ชนิดข้อมูล_____
  - Output: (แสดงค่า) จำนวนข้อมูล_____ชนิดข้อมูล_____
    (คืนค่า) จำนวนข้อมูล_____ชนิดข้อมูล_____

2) **4 คะแนน** (`Lab13_2_6XXXXXXXX.py`) **[Attachment]** ในการบีบอัดภาพใน Format JPEG หลังจากขั้นตอน Discrete Cosine Transform แล้วจะต้องอ่านข้อมูลที่ได้ใน Pixel Block ของจำนวนเต็ม ขนาด 8x8 ในรูปแบบสลับฟันปลาดังแสดงด้านล่าง แล้วจึงไปทำการบีบอัดแบบ Run-Length Compression อีกที



ให้เขียนฟังก์ชัน `read_pixel(`*pixel_block*`)` เพื่อคืนค่า List แทนตัวเลขในลำดับการอ่าน Matrix ในรูปแบบสลับฟันปลาดังอธิบายเมื่อ *pixel_block* เป็น Matrix จัตุรัส

Input	**Output**

`[[7, 12, 1],`
` [2, 13, 8],`
` [16, 3, 10]]`
	`[7, 12, 2, 16, 13, 1, 8, 3, 10]`

- การวิเคราะห์ปัญหา
  - Input: จำนวนข้อมูล_____ชนิดข้อมูล_____
  - Output: (แสดงค่า) จำนวนข้อมูล_____ชนิดข้อมูล_____
    (คืนค่า) จำนวนข้อมูล_____ชนิดข้อมูล_____

3) **4 คะแนน** (`HW13_1_6XXXXXXXX.py`) **[Attachment]** ให้เขียนฟังก์ชัน `square_matrix(`*list_x*`)` เพื่อทำให้ List 2 มิติ *list_x* ที่มีสมาชิกเป็นจำนวนเต็มกลายเป็น matrix จัตุรัสโดยเติม `0` เพื่อให้มีขนาด row และ column เท่ากัน โดยจะต้องคงทุก element ใน List เดิมไว้ และจำนวน `0` ที่เติมต้องเป็นจำนวนที่น้อยที่สุดที่เป็นไปได้ ทั้งนี้กำหนดให้ฟังก์ชันทำงานแบบ Destructive และแต่ละ element จะต้องไม่เป็น alias ซึ่งกันและกัน

Input	**Output**

`[[2, 3, 4],`
` [1, 2, 3]]`
	`[[2, 3, 4],`
	` [1, 2, 3],`
	` [0, 0, 0]]`

Input

```
[[1, 2],
 [1, 2, 3],
 [1, 2],
 [1, 2],
 [1]]
```

Output

```
[[1, 2, 0, 0, 0],
 [1, 2, 3, 0, 0],
 [1, 2, 0, 0, 0],
 [1, 2, 0, 0, 0],
 [1, 0, 0, 0, 0]]
```

- การวิเคราะห์ปัญหา
  - Input: จำนวนข้อมูล_____ชนิดข้อมูล_____
  - Output: (แสดงค่า) จำนวนข้อมูล_____ชนิดข้อมูล_____
    (คืนค่า) จำนวนข้อมูล_____ชนิดข้อมูล_____

4) **4 คะแนน** (`HW13_2_6XXXXXXXX.py`) ให้เขียนฟังก์ชัน `count_vote(`*pref_matrix*`)` เพื่อคืนค่าคะแนนโหวตของ Pokémon ที่ได้จากการลงคะแนนโหวต Twitter-wide Favorite Pokémon แบบจัดลำดับ (Ranked Voting)

การลงคะแนนแบบจัดลำดับ (อังกฤษ: ranked voting) หรือเรียกอีกอย่างว่า การลงคะแนนตามลำดับความชอบ (อังกฤษ: Ranked-choice Voting) หรือ การลงคะแนนตามความชอบ (อังกฤษ: Preferential Voting) เป็นระบบการลงคะแนนใดๆ ที่ผู้ลงคะแนนเสียงใช้การจัดลำดับผู้สมัคร (หรือลำดับความชอบ) ในบัตรลงคะแนนเพื่อเลือกผู้สมัครมากกว่าหนึ่งรายขึ้นไป และเพื่อเรียงลำดับตัวเลือกผู้สมัครทั้งหมดเป็นลำดับที่หนึ่ง สอง สาม ไปจนครบ (Wikipedia)

ในตัวแปร *pref_matrix* แต่ละ Row จะแทนการเลือกของ Voter แต่ละคน และ จำนวน Column ทั้งหมดแทนตัวเลือกที่เลือกได้ โดยการคำนวนคะแนนจะให้น้ำหนักคะแนนที่สูงที่สุดแก่ตัวเลือกอันดับแรกเช่น กรณีเลือกได้ 4 ตัวเลือก ตัวเลือกแรกจะได้น้ำหนักคะแนน 4 ตัวเลือกที่ 2 จะได้น้ำหนักคะแนน 3 ลดหลั่นกันไป จนตัวเลือกสุดท้ายจะมีน้ำหนักคะแนนเท่ากับ 1 ในกรณีที่เลือกได้ n ตัวเลือก อันดับที่ 1 ก็จะได้น้ำหนักคะแนนเท่ากับ n แทน เช่นในตัวอย่างด้านล่าง คะแนนของ Pikachu จะเท่ากับ 2 + 1 + 3 + 2 = 8

ฟังก์ชันจะคืนค่า `list` ของ `tuple` ที่ประกอบด้วยชื่อ Pokémon ทั้งหมดที่มีผู้ vote ให้ และคะแนนที่ได้ เรียงตามลำดับคะแนนจากมากไปน้อย และลำดับตัวอักษรในพจนานุกรมภาษาอังกฤษกรณีที่คะแนนเท่ากัน ทั้งนี้กำหนดให้ทุก row มีความยาวเท่ากัน

Input

```
[['Mewtwo', 'Pikachu', 'Suicune'],
 ['Mewtwo', 'Suicune', 'Pikachu'],
 ['Pikachu', 'Rayquaza', 'Charizard'],
 ['Suicune', 'Pikachu', 'Charizard']]
```

Output:

```
[('Pikachu', 8),
 ('Mewtwo', 6),
 ('Suicune', 6),
 ('Charizard', 2),
 ('Rayquaza', 2)]
```

- การวิเคราะห์ปัญหา
  - Input: จำนวนข้อมูล_____ชนิดข้อมูล_____
  - Output: (แสดงค่า) จำนวนข้อมูล_____ชนิดข้อมูล_____
    (คืนค่า) จำนวนข้อมูล_____ชนิดข้อมูล_____

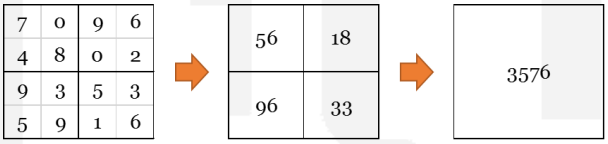
5) **4 คะแนน** (HW13_3_6XXXXXXXX.py) **[Attachment]** ให้เขียนฟังก์ชัน sum_d_product(m) เพื่อคืนค่าผลบวกของผลคูณทแยงใน matrix m ที่มีขนาด $n \times n$ เมื่อ n สามารถเขียนในรูปของ 2^x (x เป็นจำนวนเต็มบวก)

โดยกรณี matrix m ขนาด 2×2 เช่น $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ สามารถหาผลลัพธ์ได้จากสูตร $a \times d + c \times b$

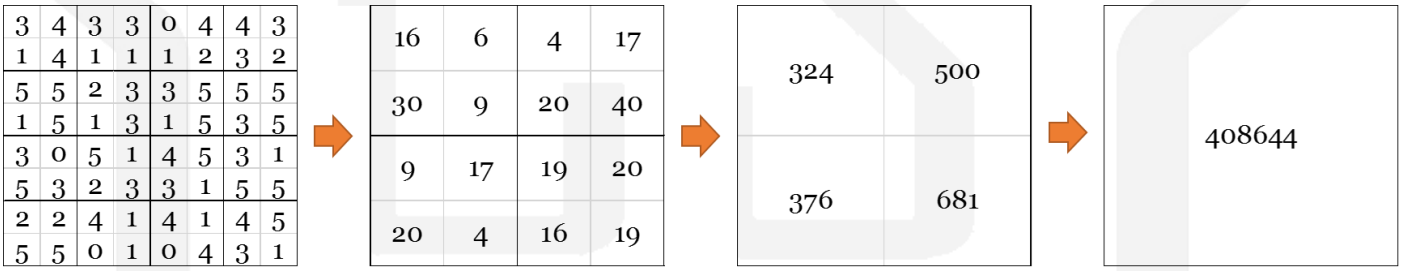
ดังนั้น matrix $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ จะมี sum_d_product() $= (1 \times 4) + (3 \times 2) = 10$

กรณีต้องการหา sum_d_product() ของ matrix ขนาดใหญ่กว่า 2 × 2 ทำได้โดยการหา sum_d_product() ของ matrix ย่อย ขนาด 2 × 2 ก่อน แล้วหา sum_d_product() ของ matrix ผลลัพธ์อีกที

เช่นกรณี matrix ขนาด 4 × 4 จะมีขั้นตอนดังนี้



หรือกรณี matrix 8 × 8



Input	**Output**
[[3, 3, 3, 2], [2, 0, 3, 1], [2, 1, 2, 3], [1, 0, 2, -1]	33
[[1, 1, 5, -1], [12, 2, -2, 0], [4, 8, 8, 12], [4, 12, 12, 15]]	3856
[[0, -1, -1, 3, 2, 3, -1, 3], [3, -1, -1, 2, 0, -1, 2, 1], [3, 0, 1, 2, 3, 1, 3, 1], [2, 2, 1, -1, -1, 2, 0, 3], [1, 3, 2, 1, 3, 2, 2, 1], [1, 2, 2, 1, 3, 3, 1, 3], [2, 2, 2, 2, 2, 2, 3, 3], [1, 3, 2, 3, 1, 1, 2, 2]]	-6290