กำหนดส่ง

Lab	
HW	
Until	

กระบวนวิชา 204111

การบ้านปฏิบัติการ 11

Sets, Dictionaries and I/O Redirection (20 คะแนน)

ข้อกำหนด

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

1) **4 คะแนน** (`Lab11_1_6XXXXXXXX.py`) ให้เขียนฟังก์ชัน `word_count(text: str) -> dict[str, int]` เพื่อคืนค่า Dictionary ผลลัพธ์จากการนับจำนวนคำที่ปรากฏในสายอักขระ (String) *text* โดยฟังก์ชันจะ คืนค่าเป็น `dict` ที่มี *key* เป็นแต่ละคำที่ปรากฏใน *text* และมี *value* เป็นความถี่ ทั้งนี้ตัวอักษรที่อยู่ใน *key* จะต้องเป็นตัวอักษรพิมพ์เล็กเท่านั้น

ข้อกำหนด

- การนับความถี่จะเป็นแบบ Case Insensitive (`'ant'` และ `'Ant'` ถือเป็นคำเดียวกัน)
- ข้อความในไฟล์จะเป็นภาษาอังกฤษมาตรฐานในรูปแบบที่ถูกต้อง (well-formed English)
- ไม่พิจารณาเครื่องหมายวรรคตอนต่าง ๆ เฉพาะที่ล้อมรอบคำ เช่น **`!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~`**
- เนื่องจาก *key* มีคุณสมบัติเป็น **set** ลำดับในการแสดงผลใน output จึงไม่จำเป็นต้องเหมือนตัวอย่าง

Input	Output

`"He doesn't want to pay $40,000 for a new car, but his wife doesn't seem to care."`	`{'new': 1,`
	` 'but': 1,`
	` 'pay': 1,`
	` 'want': 1,`
	` 'seem': 1,`
	` 'care': 1,`
	` 'his': 1,`
	` '40,000': 1,`
	` 'wife': 1,`
	` 'a': 1,`
	` 'for': 1,`
	` 'car': 1,`
	` "doesn't": 2,`
	` 'to': 2,`
	` 'he': 1}`

2) **4 คะแนน** (Lab11_2_6XXXXXXXX.py) ให้เขียนฟังก์ชัน *matching_sum*(*t*: tuple[int], *target_value*: int) -> tuple[int] เพื่อคืนค่า List ของจำนวนเต็มสองจำนวนใน tuple *t* ที่มีผลรวมเท่ากับจำนวนเต็ม *target_value* โดยกำหนดให้สมาชิกใน tuple *t* เป็นจำนวนเต็มเท่านั้นและจะมีสมาชิกอย่างน้อย 1 ตัวเสมอ และหากไม่สามารถหาจำนวนสองจำนวนดังกล่าวได้ ให้คืนค่า list ว่าง ทั้งนี้ในกรณีที่มีสมาชิกใน *t* มากกว่า 1 คู่ ที่สามารถเป็นคำตอบได้ ให้คืนค่าเพียงคำตอบเดียวเท่านั้น

Hint: การจับคู่สมาชิกทั้งหมด แล้วนำมาตรวจสอบผลบวกทีละคู่ จะทำให้เวลาบน Grader เกิน ควรแก้ปัญหาโดยใช้ **collections** ประเภท set หรือ dict

Function Call	**Output**
matching_sum((1,), 1)	[]
matching_sum((5, 2), 7)	[5, 2]
	[2, 5]
matching_sum((10, -1, 1, -8, 3, 1), 2)	[10, -8]
	[-8, 10]
	[-1, 3]
	[1, 1]
matching_sum((10, -1, 1, -8, 3, 1), 10)	[]

3) **4 คะแนน** (HW11_1_6XXXXXXXX.py) ให้เขียนฟังก์ชัน *words_to_num*(*words*: str) -> int เพื่อทำการคืนค่าจำนวนเต็มบวกที่คำนวณได้จากคำอ่านของจำนวนเต็ม *words* โดยผลลัพธ์จะมีความยาวไม่เกิน 12 หลัก

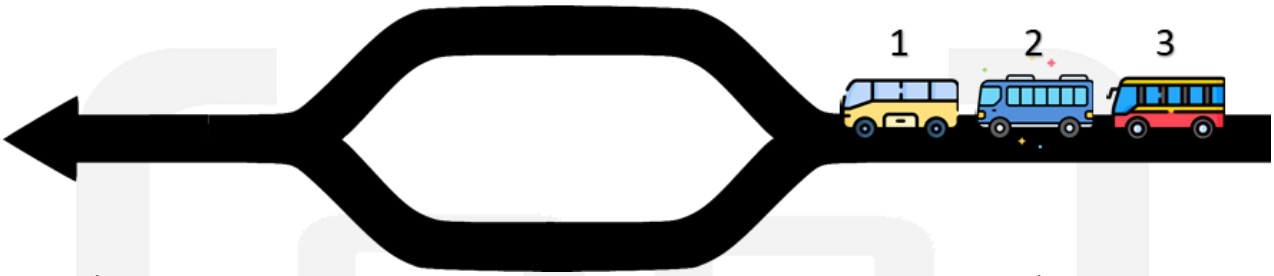
Hint:

- สามารถศึกษาการอ่านตัวเลขในภาษาอังกฤษได้จาก http://en.wikipedia.org/wiki/English_numerals
- สามารถใช้ฟังก์ชัน num_to_word() ที่เคยเขียนไว้ในสัปดาห์ก่อนๆ ในการทดสอบเช่น

```
assert word_to_num(num_to_word(5)) == 5
```

Input	**Output**
fourteen	14
two hundred forty-eight	248
one hundred eleven	111
forty-two billion six hundred forty-one million three hundred twenty-three thousand eight hundred sixty-two	42641323862

4) **4 คะแนน** (HW11_2_6XXXXXXXX.py) ท่ามกลางพายุฝนที่โหมกระหน่ำ ขบวนรถบัสจากบริษัทส้มพุ่งมาถึงทางแยก เพื่อกระจายความเสี่ยงที่จะพบปัญหาที่อาจเกิดขึ้นในเส้นทาง คนขับรถจึงตัดสินใจกระจายรถในขบวนให้วิ่งทั้งสองทาง กล่าวคือในแต่ละเส้นทางจะต้องมีอย่างน้อย 1 คันเสมอ ในจากการสำรวจเส้นทางล่วงหน้าพบว่าทางทั้งสองไปบรรจบกันที่ระยะ 50 เมตรก่อนถึงโรงแรมที่หมาย เพื่อความปลอดภัยรถทุกคันจะไม่แซงกัน และจะไม่ขับถอยหลัง



หน้าที่ของคุณคือให้เขียนฟังก์ชัน *split_and_merge*(*n*: int) -> list[str] เพื่อคืนค่า List ของ String แทนลำดับที่เป็นไปได้ทั้งหมดที่รถบัสของบริษัทส้ม n คันจะถึงโรงแรม ($0 < n \leq 11$) โดย String ของลำดับที่เป็น Output จะอยู่ในรูป เลขประจำรถ 1 ถึง n คั่นด้วย '>' (เครื่องหมายมากกว่า) ทั้งนี้ String ใน List ที่คืนค่าจะอยู่ในลำดับใดก็ได้ แต่จะต้องไม่มีลำดับการมาถึงที่ซ้ำกัน และกำหนดให้ถนนทุกเส้นเป็นถนนเลนเดียวและเดินรถได้ทางเดียว

Hint:

- พิจารณาการนำเรื่อง Superset มาใช้หาวิธีที่เป็นไปได้ทั้งหมดในการแยกรถเป็น 2 เส้นทาง
- พิจารณานำฟังก์ชัน arrival_sequnces() จาก HW10_2 มาเรียกใช้

Input	**Output**
3	['1>2>3', '1>3>2', '2>1>3', '2>3>1', '3>1>2']
4	['1>2>3>4', '1>2>4>3', '1>3>2>4', '1>3>4>2', '1>4>2>3', '2>1>3>4', '2>1>4>3', '2>3>1>4', '2>3>4>1', '2>4>1>3', '3>1>2>4', '3>1>4>2', '3>4>1>2', '4>1>2>3']

5) **4 คะแนน** (HW11_3_5XXXXXXXX.py) ให้เขียนฟังก์ชัน *runner_up*() -> None เพื่อรับค่าคะแนนของนักศึกษาในห้อง ตามจำนวนนักศึกษาที่ระบุในบรรทัดแรก แล้วแสดงผลคะแนนที่สูงเป็นอันดับที่ 1, อันดับที่ 2 และ ค่าเฉลี่ยคะแนน (ทศนิยม 2 ตำแหน่ง) โดยหากไม่มีตำแหน่งที่ 2 ให้ แสดงคำว่า **None**

ข้อกำหนด: ไม่อนุญาตให้ใช้ตัวแปรประเภท **iterables** เช่น **list**, **tuple**, **set** หรือ **dict** ในการเก็บข้อมูลที่ได้รับเพื่อการคำนวณ

Hint: ศึกษา Slide Input/Output Redirection เพื่อความสะดวกในการทดสอบข้อมูลนำเข้าจำนวนมาก

ตัวอย่างการ run 1

```
Total students: 3
Enter score:
13
12
13
---
Max score is: 13.00
Runner up is: 12.00
Average is: 12.67
```

ตัวอย่างการ run 2

```
Total students: 2
Enter score:
61
61
---
Max score is: 61.00
Runner up is: None
Average is: 61.00
```

ตัวอย่างการ run 3

```
Total students: 7
Enter score:
61
72
64
81
61
79
63
---
Max score is: 81.00
Runner up is: 79.00
Average is: 68.71
```

ตัวอย่างการ run 4

```
Total students: 1
Enter score:
13
---
Max score is: 13.00
Runner up is: None
Average is: 13.00
```

การส่งงาน

1. ลักษณะ/ลำดับข้อความของการรับค่า/แสดงผล จะ**ต้องเป็นไปตามที่ระบุ**ในตัวอย่างการ run
2. ไฟล์งานที่ส่ง จะต้องมีการแทรก comment ที่ต้นไฟล์ตามข้อกำหนดใน canvas รายวิชา
3. ไฟล์งานโปรแกรมที่ส่ง จะต้องมีการแทรก pseudocode เป็น comment ในแต่ละขั้นตอน
4. Upload ไฟล์ source code ตามที่ระบุในแต่ละข้อ ไปยังระบบตรวจให้คะแนนอัตโนมัติ https://cmu.to/gdr111