2110101: การบ้านครั้งที่ 7: เรื่องช่วงและเซต

ช่วงในคณิตศาสตร์มีหลายประเภท ในการบ้านนี้เราจะนิยามช่วง 2 ประเภท ดังนี้

1. ช่วงปิด เขียนได้เป็น [a,b] = {x | a $\leq$ x $\leq$ b} โดยที่ x, a, b เป็นจำนวนเต็ม
2. ช่วงว่าง เขียนได้เป็น (0,0)

ช่วงใดๆที่พูดถึงในการบ้านนี้จะเป็นจำนวนเต็มสองจำนวนคั่นด้วยเครื่องหมาย , โดยไม่มีช่องว่าง

งานของนิสิต

จงเขียนฟังก์ชัน `create_interval`, `intersection`, `is_subset` และ `get_subsets` ตามรายละเอียดที่เขียนใน comment ของ code cell ข้างล่างนี้

การส่งงาน

- Save a copy in Drive
- เปลี่ยนชื่อ Notebook ให้เป็น HW7_xxxxxxxxxx.ipynb โดย xxxxxxxxxx เป็นเลขประจำตัวนิสิต
- File -> Download -> Download .ipynb
- ส่งไฟล์ที่ดาวน์โหลดได้ใน MyCourseVille ก่อนเวลา 23:59 น. วันที่ 2 มีนาคม 2565
- จะตรวจแฟ้ม .ipynb แฟ้มล่าสุดที่ส่งในระบบ MyCourseVille เท่านั้น

สำคัญ: อ่านตรงนี้ด้วย

- ห้ามเปลี่ยนบรรทัดแรกของ code cell
- ห้ามเปลี่ยนบรรทัด def ของฟังก์ชันที่ให้เขียน
- นิสิตสามารถเขียนฟังก์ชันเพิ่มได้แต่ให้เขียนภายใน code cell เดียวกันนี้
- ฟังก์ชันต้องคืนข้อมูลและประเภทข้อมูลตามที่กำหนดเท่านั้น
- ฟังก์ชันต้องไม่ใช้ข้อมูลในตัวแปรใด ๆ นอกฟังก์ชัน
- ฟังก์ชันต้องไม่เปลี่ยนแปลงข้อมูลภายในของพารามิเตอร์ที่ได้รับ (หากระบุไว้ในข้อกำหนด)
- นิสิตสามารถลบ doc_str (คอมเม้นที่ตามหลังฟังก์ชันที่ปิดหัวท้ายด้วย """) ที่โจทย์ระบุไว้ในแต่ละฟังก์ชันได้ แต่ถ้าไม่อยากลบ ก็ให้เก็บ doc_str ไว้ที่เดิมหลังบรรทัด def และถ้าจะเพิ่ม code ใหม่ในฟังก์ชัน ให้เยื้องให้อยู่ในแนวเดียวกับ doc_str
- ส่งแฟ้ม .ipynb กี่ครั้งก็ได้ แต่จะตรวจแฟ้ม .ipynb **แฟ้มล่าสุด** เท่านั้น
- โปรแกรมที่ทำงานไม่ได้ เกิด error ก็ไม่ได้คะแนน (แนะนำให้ compile อย่างน้อยหนึ่งครั้งก่อนส่งเพื่อเช็ค error)
- ห้าม import อะไรใด ๆ เพิ่มและห้ามใช้ set นอกนั้น อยากใช้คำสั่งอะไร ก็ใช้ได้ ถ้าทำงานได้ใน Python version 3.7
- ส่อทุจริต เช่น

- ส่งโปรแกรมที่ผู้ส่งไม่สามารถอธิบายได้ว่า ใช้หลักการและทำงานอย่างไร
- หรือ ส่งโปรแกรมที่คล้ายกับโปรแกรมของผู้อื่นมาก ๆ (ไม่ว่าจะเป็นผู้ให้หรือผู้รับ จะตั้งใจหรือไม่ก็ตาม)

- หากพบว่า งานที่ส่งส่อทุจริต จะได้คะแนน **ติดลบคะแนนรวมสะสมตั้งแต่การบ้านครั้งแรกสุด**

ตัวอย่างทดสอบแต่ละฟังก์ชัน

การเรียกใช้ฟังก์ชัน	ผลลัพธ์ที่ได้จากฟัง
`create_interval(3, 5)`	`'[3,5]'`
`create_interval(-2, 10)`	`'[-2,10]'`
`create_interval(0, 0)`	`'[0,0]'`
`create_interval(7, 5)`	`'(0,0)'`
`intersection('(0,0)', '[-15,5]')`	`'(0,0)'`
`intersection('[2,3]', '(0,0)')`	`'(0,0)'`
`intersection('[-4,2]', '[5,10]')`	`'(0,0)'`
`intersection('[3,4]', '[-2,0]')`	`'(0,0)'`
`intersection('[2,4]', '[4,8]')`	`'[4,4]'`
`intersection('[2,4]', '[3,5]')`	`'[3,4]'`
`intersection('[2,9]', '[3,5]')`	`'[3,5]'`
`intersection('[2,5]', '[3,5]')`	`'[3,5]'`
`intersection('[5,6]', '[-2,5]')`	`'[5,5]'`
`intersection('[-4,12]', '[-15,10]')`	`'[-4,10]'`
`intersection('[7,8]', '[3,9]')`	`'[7,8]'`
`intersection('[7,10]', '[-4,10]')`	`'[7,10]'`
`intersection('[1,5]', '[1,4]')`	`'[1,4]'`
`intersection('[1,5]', '[1,6]')`	`'[1,5]'`
`intersection('[1,5]', '[1,5]')`	`'[1,5]'`
`is_subset('[2,4]', '[3,5]')`	`False`
`is_subset('[4,6]', '[3,5]')`	`False`
`is_subset('[5,6]', '[-2,6]')`	`True`
`is_subset('[3,7]', '[3,7]')`	`True`
`is_subset('[-1,-1]', '[-1,-1]')`	`True`
`is_subset('[5,7]', '(0,0)')`	`False`
`is_subset('(0,0)', '[3,5]')`	`True`
`is_subset('[0,0]', '[1,6]')`	`False`
`is_subset('(0,0)', '(0,0)')`	`True`
`get_subsets(['[-2,8]'], ['[5,17]'])`	`[]`
`get_subsets(['[4,6]'], [])`	`[]`
`get_subsets([], ['[1,7]'])`	`[]`
`get_subsets(['[-2,3]','[4,6]', '[8,9]', '[10,10]', '[15,18]'], ['[5,17]'])`	`['[8,9]', '[10,10]']`

get_subsets(['[-2,3]','[4,6]', '[8,9]', '[10,10]', '[15,18]'], ['[0,10]', '[12,20]']) ['[4,6]', '[8,9]', '[10,10

```
# HW7_Function (ไม่ลบหรือแก้ไขบรรทัดนี้ หรือเพิ่มอะไรก่อนบรรทัดนี้ โดยเด็ดขาด)

# - เขียนในเซลล์นี้เท่านั้น
# - ถ้าต้องการเขียนฟังก์ชันเพิ่ม ก็เขียนในเซลล์นี้

def create_interval(x, y):
    """
    รับ x, y เก็บค่าจำนวนเต็ม

    หาก x มีค่าน้อยกว่าหรือเท่ากับ y
    ให้คืน สตริง แทนช่วงปิดตั้งแต่ x ถึง y (รวม x และ y ด้วย)
    เช่น create_interval(3, 5) คืน '[3,5]'
        create_interval(-2, 10) คืน '[-2,10]'
        create_interval(0, 0) คืน '[0,0]
    หาก x มีค่ามากกว่า y
    ให้คืน สตริง แทนช่วงว่าง เช่น create_interval(7, 5) คืน '(0,0)'

    โดยช่วงที่คืนจะต้องไม่มีช่องว่างกั้นระหว่างเครื่องหมาย "," กับตัวเลข
    ถ้าคืน '[3, 5]' หรือ '(0 ,0)' ถือว่าไม่ผ่าน
    """
    return ''

# ----------------------------------------------------
def intersection(a, b):
    """
    รับ a เก็บสตริงแทนช่วงของจำนวนแรก เช่น '[2,4]'
    รับ b เก็บสตริงแทนช่วงของจำนวนที่สอง เช่น '[3,5]'

    คืน สตริง ที่เก็บค่าช่วงที่มีทั้งใน a และ b
    เช่น intersection('[2,4]', '[3,5]') คืน '[3,4]'
        intersection('[5,6]', '[3,5]') คืน '[5,5]'
        intersection('[-4,12]', '[-15,10]') คืน '[-4,10]'
        intersection('[-4,2]', '[5,10]') คืน '(0,0)'
        intersection('(0,0)', '[-15,5]') คืน '(0,0)'

    โดย
    - ช่วงทั้งในค่า a และ b และตอนคืนค่า จะประกอบด้วยเลขจำนวนเต็มสองจำนวน
      ตัวแรกคือ x ตัวที่สองคือ y
    - x จะมีค่าน้อยกว่าหรือเท่ากับ y เสมอ
    - x กับ y จะคั่นด้วยเครื่องหมาย "," และไม่มีช่องว่าง (จะไม่มี '[3, 5]' หรือ '[2 ,4]')
    """
    return ''

# ----------------------------------------------------
def is_subset(a, b):
    """
    รับ a เก็บสตริงแทนช่วงของจำนวนแรก เช่น '[2,4]'
    รับ b เก็บสตริงแทนช่วงของจำนวนที่สอง เช่น '[3,5]'
    โดย
    - ช่วงจะประกอบด้วยเลขจำนวนเต็มสองจำนวนตัวแรกคือ x ตัวที่สองคือ y
```

```
    - x จะมีค่าน้อยกว่าหรือเท่ากับ y เสมอ
    - x กับ y จะคั่นด้วยเครื่องหมาย "," และไม่มีช่องว่าง (จะไม่มี '[3, 5]' หรือ '[2 ,4]')

    คืน บูลีน True ถ้า a เป็น subset ของ b
    คืน บูลีน False ถ้า a ไม่เป็น subset ของ b
    เช่น is_subset('[2,4]', '[3,5]') คืน False
        is_subset('[5,6]', '[-2,6]') คืน True
        is_subset('[3,7]', '[3,7]') คืน True
        is_subset('[-1,-1]', '[-1,-1]') คืน True
        is_subset('[5,7]', '(0,0)') คืน False
        is_subset('(0,0)', '[3,5]') คืน True
        is_subset('[0,0]', '[1,6]') คืน False
        is_subset('(0,0)', '(0,0)') คืน True
    """
    return ''

# ---------------------------------------------------
def get_subsets(list_a, list_b):
    """
    รับ list_a, list_b ที่ต่างก็เป็นลิสต์ที่มีสมาชิกแต่ละตัวเป็นสตริงที่เก็บค่าช่วง
      โดยจะไม่มีช่วงที่ซ้อนทับกัน ตัวอย่างของลิสต์ เช่น ['[2,4]', '[5,9]', '[10,12]']
      ซึ่งสมาชิกในลิสต์จะมีการเรียงมาให้เรียบร้อยแล้ว (จะไม่มี ['[5,9]', '[2,4]'])
      และจะไม่มีสมาชิกที่เป็นช่วงว่าง

    คืน ลิสต์ ที่มีสมาชิกช่วงเป็นสมาชิก list_a ที่เป็น subset ของสมาชิกหนึ่งใน list_b
    เช่น get_subsets(['[4,6]'], []) คืน []
        get_subsets([], ['[1,7]']) คืน []
        get_subsets(['[-2,8]'], ['[5,17]']) คืน []
        get_subsets(['[-2,3]','[4,6]', '[8,9]', '[10,10]', '[15,18]'], ['[5,17]']) คืน ['[
        get_subsets(['[-2,3]','[4,6]', '[8,9]', '[10,10]', '[15,18]'], ['[0,10]', '[12,20]
    โดยสมาชิกในลิสต์ที่คืนมาจะเรียงลำดับจากน้อยไปมาก (ถ้าคืน ['[10,10]', '[8,9]'] ถือว่าไม่ถูกต้อง)
    """
    return ''
```

Bonus (+10% ของคะแนนเต็ม)

สำหรับคนที่เขียน function `get_subsets` ที่สามารถหาคำตอบที่ถูกต้องในกรณีที่ `list_a` และ `list_b` มีขนาดใหญ่มากได้ภายใน 5 วินาที (`5000 <= len(list_a), len(list_b) <= 20000`)

เคล็ดลับ: ให้เขียน `get_subsets` โดยไม่ใช้ลูปซ้อนลูป (nested loop)

```
# ตัวอย่างทดสอบ (แน่นอนว่าของจริงจะไม่หน้าตาแบบนี้)
# จะใช้ตัวอย่างทดสอบนี้ได้ ต้องเขียน create_interval ข้างบนให้ถูกต้องก่อน
list_size = 6000
list_a = []
list_b = []
for i in range(list_size):

    list_a.append(create_interval(i*15+4, i*15+6))
    list_a.append(create_interval(i*15+9, i*15+12))
    list_b.append(create_interval(i*15+2, i*15+10))
```

```
get_subsets(list_a, list_b) # คำตอบของ get_subsets(list_a, list_b) ในที่นี้คือ list_a[::2]
```