กำหนดส่ง

Lab	
HW	
Until	

กระบวนวิชา 204111

การบ้านปฏิบัติการ 8

Recursion Part I (20 คะแนน)

ข้อกำหนด

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

ii. ไม่อนุญาตให้ใช้การทำซ้ำแบบ Iterations เช่น **`for`**, **`while`**, หรือ Data Type อื่น ๆ ที่ยังไม่สอนในบทเรียน เช่น **`set`** หรือ **`dict`** ในการแก้ปัญหา

iii. ระบบ grader จะไม่ตรวจให้คะแนนฟังก์ชัน `main()` และจะพิจารณาทดสอบเฉพาะฟังก์ชันที่ระบุชื่อในแต่ละโจทย์แต่ละข้อโดยตรง

iv. สามารถเพิ่ม **Default Argument** ใน Function Definition ที่โจทย์ระบุได้ตามความเหมาะสม

v. ควรสร้างฟังก์ชันทดสอบทุกข้อ และพิจารณาสร้างฟังก์ชันย่อยต่าง ๆ เพิ่มเติมได้ตามความเหมาะสม

vi. ในข้อที่ระบุว่ามี **[Attachments]** ให้ Download ไฟล์ Template จาก Grader ลงมา implement

1) **4 คะแนน** (`Lab08_1_6XXXXXXXX.py`) ให้เขียนฟังก์ชัน recursive ***`gcd`***`(`x`: int, `y`: int) -> int` เพื่อคืนค่าตัวหารร่วมมากของจำนวนเต็ม x ($x \neq 0$) และจำนวนเต็ม y ($y \neq 0$) ด้วยวิธีของ Euclid จากบทเรียนเรื่อง Numbers ทั้งนี้ไม่อนุญาตให้ใช้ฟังก์ชัน `math.gcd()` ในการแก้ปัญหา

Input	Output
19 71	1
-39 78	39

- การวิเคราะห์ปัญหา
  - Base Case:
  - D & C:
  - Combine

2) **4 คะแนน** (`Lab08_2_6XXXXXXXX.py`) ให้เขียนฟังก์ชัน recursive ***`reverse_digits`***`(`x`: int) -> int` เพื่อคืนค่าผลลัพธ์จากการกลับหลักจำนวนเต็ม x ใด ๆ ทั้งนี้ไม่อนุญาตให้ใช้ operation `reversed()` หรือ operation อื่น ๆ ที่ทำการกลับลำดับใน `str` หรือ `list` ในการแก้ปัญหา (e.g. `[::-1]`, `sorted(.., reverse=True)`)

Input	Output
1234	4321
1	1

- การวิเคราะห์ปัญหา
  - Base Case:
  - D & C:
  - Combine

3) **4 คะแนน** (HW08_1_6XXXXXXXX.py) **[Attachments]** ให้เขียนฟังก์ชัน recursive `pi(n: int) -> float` เพื่อคืนค่า π (pi) จากการประมาณจาก Series ผลบวกโดยมีความละเอียดตามจำนวนพจน์ที่ระบุด้วยจำนวนเต็ม n ดังนี้

$$\pi = 3 + \left(\frac{4}{2 \times 3 \times 4}\right) - \left(\frac{4}{4 \times 5 \times 6}\right) + \left(\frac{4}{6 \times 7 \times 8}\right) - \left(\frac{4}{8 \times 9 \times 10}\right) + \cdots$$

ค่า π จาก series ผลบวกดังกล่าวเป็นการประมาณค่าจากพจน์ที่ 0 - n ของ series โดยพจน์ที่ 0 จะเท่ากับ 3 ดังนั้น

`pi(0)` $= 3$ ≈ 3.000000

`pi(2)` $= 3 + \left(\frac{4}{2 \times 3 \times 4}\right) - \left(\frac{4}{4 \times 5 \times 6}\right)$ ≈ 3.133333

`pi(5)` $= 3 + \left(\frac{4}{2 \times 3 \times 4}\right) - \left(\frac{4}{4 \times 5 \times 6}\right) + \left(\frac{4}{6 \times 7 \times 8}\right) - \left(\frac{4}{8 \times 9 \times 10}\right) + \left(\frac{4}{10 \times 11 \times 12}\right)$ ≈ 3.142713

ทั้งนี้ไม่อนุญาตให้ใช้ `list` หรือ `map` ในการแก้ปัญหา

Input	Output
0	3
1	3.1666666666666665
2	3.1333333333333333
5	3.1427128427128426

- การวิเคราะห์ปัญหา
  - Base Case:
  - D & C:
  - Combine

4) **4 คะแนน** (HW08_2_6XXXXXXXX.py) ให้เขียนฟังก์ชัน recursive ***skip_list***`(`*word*`: str) -> list[str]` เพื่อคืนค่า `list` ของ `str` ที่สร้างจากอักขระใน *word* เมื่อ *word* เป็น String ใดๆ และให้สร้าง List ผลลัพธ์ตามข้อกำหนดดังนี้

- สมาชิกตัวที่ n เกิดจากการพิจารณาอักขระใน *word* แบบข้ามทุกๆ n index
- ให้ n เริ่มจาก 0 และมีค่าไม่เกินความยาวของ *word*

เช่น เมื่อพิจารณา String `'ABCDEF'`

[0]	[1]	[2]	[3]	[4]	[5]
A	B	C	D	E	F

จะเห็นได้ว่า ผลลัพธ์คือ `['ABCDF', 'BDF', 'CF', 'D', 'E', 'F']` เนื่องจาก

- สมาชิกที่ 0 ของ List เป็น String `'ABCDEF'` จากการพิจารณาข้ามทีละ 0 index
- สมาชิกที่ 1 ของ List เป็น String `'BDF'` จากการพิจารณาข้ามทีละ 1 index
- สมาชิกที่ 2 ของ List เป็น String `'CF'` จากการพิจารณาข้ามทีละ 2 index
- สมาชิกที่ 3 ของ List เป็น String `'D'` จากการพิจารณาข้ามทีละ 3 index
- สมาชิกที่ 4 ของ List เป็น String `'E'` จากการพิจารณาข้ามทีละ 4 index
- สมาชิกที่ 5 ของ List เป็น String `'F'` จากการพิจารณาข้ามทีละ 5 index

Input	**Output**
ABCD	`['ABCD', 'BD', 'C', 'D']`
hello!	`['hello!', 'el!', 'l!', 'l', 'o', '!']`
a	`['a']`

- การวิเคราะห์ปัญหา
  - Base Case:
  - D & C:
  - Combine

5) **4 คะแนน** (HW08_3_6XXXXXXXX.py) ให้เขียนฟังก์ชัน recursive ***is_anagram***`(`*s1*`: str, `*s2*`: str) ->` `bool` เพื่อคืนค่าผลลัพธ์ `True` หรือ `False` (Boolean Function) จากการตรวจสอบการเป็นแอนาแกรม (anagram) ซึ่งกันและกันของ string *s1* และ *s2* เมื่อทั้ง *s1* และ *s2* เป็น non-empty string (string ที่ไม่ใช่ string ว่าง) ที่มีตัวอักษรภาษาอังกฤษอย่างน้อย 1 ตัวเสมอ โดยการพิจารณาจะเป็นแบบ case-**in**sensitive (ไม่แบ่งแยกระหว่างตัวอักษรพิมพ์ใหญ่และพิมพ์เล็ก) และ ไม่พิจารณาเครื่องหมายวรรคตอน สัญลักษณ์พิเศษ ตัวเลข หรือ อักขระว่างต่าง ๆ ทั้งนี้ไม่อนุญาตให้ใช้การเรียงลำดับจากฟังก์ชันหรือ method ต่าง ๆ เช่น ฟังก์ชัน **`sorted()`** และ/หรือ Module อื่น ๆ นอกเหนือจากเนื้อหาในบทเรียนในการแก้ปัญหา

คำสลับอักษร หรือ อะนาแกรม หรือ แอนาแกรม (อังกฤษ: anagram ; กรีก: anagramma) คือข้อความเกิดจากการนำตัวอักษรในอีกข้อความหนึ่งมาเรียงสลับที่กัน เช่น `'Eleven plus two'` เป็นอะนาแกรมของ `'Twelve plus one'`
Ref: https://th.wikipedia.org/wiki/คำสลับอักษร

Input	**Output**
`Tom Marvolo Riddle` `I am Lord Voldemort!!!`	`True`
`cat` `tab`	`False`
`Nissan` `Insane`	`False`

- การวิเคราะห์ปัญหา
  - Base Case:
  - D & C:
  - Combine

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