ข้อกำหนด

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

ii. ไม่อนุญาต Data Type หรือ Module อื่น ๆ ที่ยังไม่สอนในบทเรียน ในการแก้ปัญหา (สามารถใช้เนื้อหาจากสัปดาห์ที่ 11 คือ **`set`** และ **`dict`** ได้)

File Header (กรณีไม่เขียน Header จะเสียคะแนน 5%)

```
#!/usr/bin/env python3
# ชื่อ (ไม่ต้องใส่นามสกุล)
# รหัสนศ
# Sec00x
```

1) **100 คะแนน** (`Q3P1_6XXXXXXXX.py`) คุณกี้เป็นนักวิ่งหน้าใหม่ประจำหมู่บ้านที่สวมบทออนไลน์เป็น anti-fan หมีเนยแบบลับๆ คุณกี้จึงมีเส้นทางวิ่งในหมู่บ้านที่จำกัดรวมแล้วเป็นระยะทางแค่ 2 กิโลเมตร เพราะไม่อยากวิ่งผ่านบ้านป้าๆ มัมหมี คุณกี้บันทึกเวลาที่ใช้ในการวิ่งในแต่ละวันอย่างสม่ำเสมอ แต่หากวันไหนฝนตกเธอจะงดวิ่ง และในบันทึกเวลาของวันนั้นจะถูกแทนด้วยค่า None

คุณกี้สังเกตคร่าวๆ ว่า ช่วงไหนที่เธอทำ IF (Intermittent Fasting) ติดต่อกันหลายๆ วัน เธอจะวิ่งได้เร็วกว่าปกติ เธอจึงต้องการหาจำนวนวันติดต่อกันที่ยาวที่สุดที่ใช้เวลาวิ่งน้อยกว่าค่าเฉลี่ย มาเทียบกับตารางการทำ IF ของเธอ

หน้าที่ของคุณคือให้เขียนฟังก์ชันดังนี้

1. ฟังก์ชัน `average_time(records: list[float | None]) -> float` เพื่อคืนค่าเฉลี่ยของเวลาที่เธอใช้วิ่ง จากบันทึกการวิ่ง *records* ที่เป็น List ของจำนวนจริงบวกแทนเวลาที่เธอใช้วิ่งในแต่ละวัน (หน่วยเป็นนาที) และจะต้องไม่นำวันที่เธอไม่ได้วิ่ง (None) มาคำนวณ ทั้งนี้คำตอบที่ถูกต้อง จะต้องอยู่ใน epsilon=10^{-4}

   Hint:
   - คืนค่าโดยไม่มีการปัดเลขทศนิยมในขั้นตอนใดๆ
   - ฟังก์ชันทำงานแบบ Non-destructive

Input	**Output** (แสดงค่าโดยประมาณ)
`[12., 14., 18., None, 14., 13., 14., 22., 20., 15., 15., 15.]`	`15.636364`

2. ฟังก์ชัน `running_streaks(records: list[float | None]) -> list[int]` เพื่อคืนค่า List ของจำนวนวันติดต่อกันที่ยาวที่สุดที่คุณกี้ใช้เวลาวิ่งน้อยกว่าค่าเฉลี่ย หากมีหลายช่วงที่มีความยาวสูงสุดที่สุดเท่ากันให้คืน List ของจำนวนวัน เช่น หากมีวันที่วิ่งได้เร็วกว่าค่าเฉลี่ยยาวติดต่อกันสูงสุดเท่ากันใน 2 ช่วง แต่ละช่วงคือ 3 วัน ให้คืนค่า `[3, 3]`

   Hint:
   - ฟังก์ชันทำงานแบบ Non-destructive

Input	**Output**
`[15., 15.]`	`[]`
`[12., 14., 18., None, 14., 13., 14., 22., 20., 15., 15., 15.]`	`[3, 3]`
`[15., 16.]`	`[1]`
`[1., 16., 8.]`	`[1, 1]`

รหัสนักศึกษา..... Section.....

`[16., 8., 1.]`	`[2]`
`[1., None, 1., 10.]`	`[1, 1]`

ข้อกำหนด

- *records* จะไม่เป็น List ว่างและ มี Element ที่ไม่เป็น None อย่างน้อย 2 ตัวเสมอ

คำอธิบาย Test Case

มีทั้งหมด 10 Test Case คิดคะแนนเป็น 10 คะแนนต่อ Test Case

- Test Case ที่ 1 – 2: ทดสอบฟังก์ชัน `average_time()`
- Test Case ที่ 3 – 10: ทดสอบฟังก์ชัน `running_streaks()`
  - Test Case ที่ 3: เหมือนกรณีตัวอย่าง
  - Test Case 5, 7, 9: จะไม่มีวันที่ฝนตก
  - Test Case ที่ 4 – 7: มีช่วงวันที่วิ่งได้เร็วกว่าค่าเฉลี่ยยาวติดต่อกันสูงสุด 1 ช่วงเสมอ
  - Test Case ที่ 7 – 10: มีโอกาสที่ช่วงยาวที่สุดจะรวมวันสุดท้าย
  - Test Case ที่ 8 – 10: ทดสอบตามทุกข้อกำหนดของโจทย์

Python Tutor Visualizer: http://10.10.10.11/visualize.html

Grader: http://10.10.10.10

ข้อกำหนด

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

ii. ไม่อนุญาต Data Type หรือ Module อื่น ๆ ที่ยังไม่สอนในบทเรียน ในการแก้ปัญหา (สามารถใช้เนื้อหาจากสัปดาห์ที่ 11 คือ **`set`** และ **`dict`** ได้)

File Header (กรณีไม่เขียน Header จะเสียคะแนน 5%)

```
#!/usr/bin/env python3
# ชื่อ (ไม่ต้องใส่นามสกุล)
# รหัสนศ
# Sec00x
```

2) **100 คะแนน** (`Q3P2_6XXXXXXXX.py`) นายกิมหงต้องการหาเลขมงคลเพื่อนำมาใช้ในธุรกิจให้ราบรื่น โดยนำชุดตัวเลขจำนวนเต็มมาชุดหนึ่งแล้วทำการเลือกเลขนำโชคตามกติกาดังนี้

- ถ้าชุดตัวเลขมีตัวเลขเพียงตัวเดียว ตัวเลขนั้นคือเลขนำโชค
- ถ้าชุดตัวเลขมีตัวเลขตั้งแต่สองตัวขึ้นไป ให้แบ่งชุดตัวเลขเป็นครึ่งหน้าและครึ่งหลัง ถ้าจำนวนตัวเลขในชุดตัวเลขเป็นเลขคี่ ให้ครึ่งหลังมีจำนวนมากกว่าครึ่งหน้าอยู่หนึ่งตัว แล้วหาเลขนำโชคของชุดตัวเลขครึ่งหน้า (ให้ชื่อว่า ***a***) และชุดตัวเลขครึ่งหลัง (ให้ชื่อว่า ***b***) และเลือกเลขนำโชคจาก ***a*** และ ***b*** ดังนี้
  - ถ้า ***a*** + ***b*** เป็นจำนวน**คู่** ค่าที่**มากกว่า**คือเลขนำโชค
  - แต่ถ้า ***a*** + ***b*** เป็นจำนวน**คี่** ค่าที่**น้อยกว่า**คือเลขนำโชค

ตัวอย่าง ถ้าชุดตัวเลขเริ่มต้น คือ `[1, 3, 5, 6, 8, 7, 9, 2, 4]` จะมีขั้นตอนการหาเลขนำโชคดังนี้

- หาเลขนำโชคจาก `[1, 3, 5, 6]` และ `[8, 7, 9, 2, 4]`
- หาเลขนำโชคใน `[1, 3, 5, 6]`
  - หาเลขนำโชคจาก `[1, 3]` และ `[5, 6]`
  - หาเลขนำโชคใน `[1, 3]`
    - หาเลขนำโชคใน `[1]` ได้ 1 และ หาเลขนำโชคใน `[3]` ได้ 3

    `1 + 3` เป็นเลขคู่ ดังนั้นเลขนำโชคใน `[1, 3]` คือ 3
  - หาเลขนำโชคใน `[5, 6]`
    - หาเลขนำโชคใน `[5]` ได้ 5 และ หาเลขนำโชคใน `[6]` ได้ 6

    `5 + 6` เป็นเลขคี่ ดังนั้นเลขนำโชคใน `[5, 6]` คือ 5

  `3 + 5` เป็นเลขคู่ ดังนั้นเลขนำโชคใน `[1, 3, 5, 6]` คือ 5
- หาเลขนำโชคใน `[8, 7, 9, 2, 4]`
  - หาเลขนำโชคจาก `[8, 7]` และ `[9, 2, 4]`
  - หาเลขนำโชคใน `[8, 7]`
    - หาเลขนำโชคใน `[8]` ได้ 8 และ หาเลขนำโชคใน `[7]` ได้ 7

    `8 + 7` เป็นเลขคี่ ดังนั้นเลขนำโชคใน `[8, 7]` คือ 7
  - หาเลขนำโชคใน `[9, 2, 4]`
    - หาเลขนำโชคจาก `[9]` และ `[2, 4]`
      - หาเลขนำโชคใน `[9]` ได้ 9
      - หาเลขนำโชคใน `[2, 4]`

รหัสนักศึกษา..... Section.....

  - หาเลขนำโชคใน `[2]` ได้ 2 และ หาเลขนำโชคใน `[4]` ได้ 4

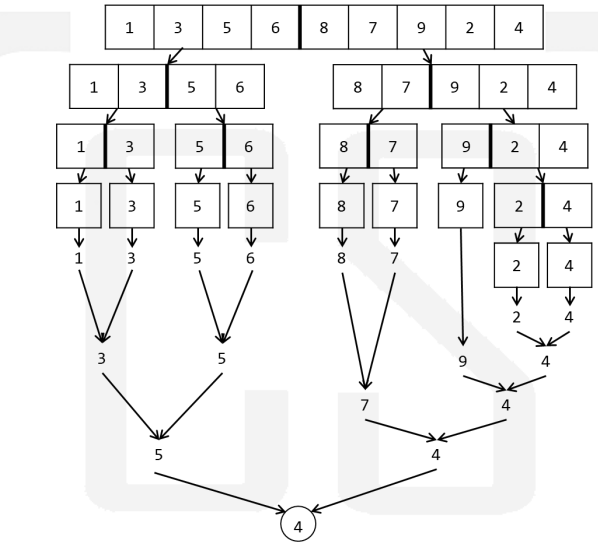
  `2 + 4` เป็นเลขคู่ ดังนั้นเลขนำโชคใน `[2, 4]` คือ 4

`4 + 9` เป็นเลขคี่ ดังนั้นเลขนำโชคใน `[9, 2, 4]` คือ 4

`7 + 4` เป็นเลขคี่ ดังนั้นเลขนำโชคใน `[8, 7, 9, 2, 4]` คือ 4

และ `5 + 4` เป็นเลขคี่ ดังนั้นเลขนำโชคของ `[1, 3, 5, 6, 8, 7, 9, 2, 4]` คือ 4

หรือตามแผนผังการแบ่งชุดตัวเลขและการเลือกตัวเลขดังนี้



หน้าที่ของคุณคือให้เขียนฟังก์ชัน *Lucky_number*(*list_num*: list[int]) -> int เพื่อ**คืนค่า** เลขนำโชคใน *list_num* ซึ่งเป็น List ของจำนวนเต็ม

Input	**Output**
`[1, 2, 3, 4, 5, 6, 7, 8]`	7
`[5, 2, 7, 4, 3, 6, 1, 8]`	3
`[1, 3, 5, 6, 8, 7, 9, 2, 4]`	4
`[1, 2, 3, 4, 5, 6, 7, 8, 9]`	7
`[9, 8, 7, 6, 5, 4, 3, 2, 1`	3
`[30, 20, 80, 60, 100]`	100

ข้อกำหนด

- *list_num* จะไม่เป็น List ว่าง

รหัสนักศึกษา..... Section.....

<u>คำอธิบาย Test Case</u>

มีทั้งหมด 10 Test Case คิดคะแนนเป็น 10 คะแนนต่อ Test Case

ตัวเลขในชุดตัวเลข จะมีค่าระหว่าง 1 และ 999 และอาจมีค่าเท่ากันได้

สำหรับ Test Case ที่ 1 – 5 จำนวนตัวเลขในชุด (n) จะอยู่ในรูป 2^k เมื่อ $k \geq 2$

- Test Case ที่ 1 ตามตัวอย่างการ Run ที่ 1 และ 2 แต่**<u>ต้อง</u>**<u>ใช้วิธีการทำงานตามกติกาที่ระบุไว้</u>
- Test Case ที่ 2 $n = 4$
- Test Case ที่ 3 $n \leq 16$ ตัวเลขเป็นจำนวนคู่ทั้งหมด
- Test Case ที่ 4 $n \leq 1024$ ตัวเลขเป็นจำนวนคี่ทั้งหมด
- Test Case ที่ 5 $n \leq 1024$

สำหรับ Test Case ที่ 6 – 10 n อาจเป็นค่าใดก็ได้ที่ $4 \leq n \leq 1024$

- Test Case ที่ 6 $n \leq 10$
- Test Case ที่ 7 ตัวเลขจะเรียงกันตามลำดับ (ได้ทั้งมากไปหาน้อย และน้อยไปหามาก)
- Test Case ที่ 8 – 10 ไม่มีเงื่อนไขเพิ่มเติม

Python Tutor Visualizer: **http://10.10.10.11/visualize.html**

Grader: **http://10.10.10.10**