คำถามท้ายการทดลอง

1. Queue ต่างจาก Stack อย่างไร?

Queue (คิว) และ Stack (สแต็ค) เป็นโครงสร้างข้อมูลที่ใช้ในการจัดเก็บและเข้าถึงข้อมูลแต่ทำงานในลักษณะที่แตกต่างกัน

Queue ใช้หลักการ FIFO (First In, First Out) หมายถึงข้อมูลที่ถูกเพิ่มเข้ามาก่อนจะถูกเอาออกก่อน เช่น ลูกค้าที่มายืนคิวก่อนก็จะได้บริการก่อน

Stack ใช้หลักการ LIFO (Last In, First Out) หมายถึงข้อมูลที่ถูกเพิ่มเข้ามาล่าสุดจะถูกเอาออกก่อน เช่น การวางหนังสือบนกองหนังสือใหม่จะถูกหยิบออกก่อน

2. เพราะเหตุใดการ dequeue จึงใช้ pop(0) แทนที่จะเป็น pop()?

การ dequeue (ลบข้อมูลจากคิว) ต้องการลบข้อมูลที่ถูกเพิ่มเข้ามาก่อน (ตามหลัก FIFO) ซึ่งใน Python สามารถทำได้โดยใช้ pop(0) เพราะมันจะลบข้อมูลที่อยู่ที่ตำแหน่งแรกในลิสต์

pop() ใช้ลบข้อมูลจากท้ายลิสต์ (Last element) ซึ่งเป็นการทำงานตามหลัก LIFO ของ Stack แต่สำหรับ Queue เราต้องการลบข้อมูลจากจุดเริ่มต้นของลิสต์ (First element) ซึ่งจึงต้องใช้ pop(0).

การใช้ pop(0) อาจจะช้ากว่า pop() ในบางภาษาโปรแกรม เพราะการย้ายข้อมูลในลิสต์ทั้งหมดหลังจากตำแหน่งที่ถูกลบออกมาจะต้องถูกย้ายไปข้างหน้า

3. ยกตัวอย่างการประยุกต์ใช้ Queue ในชีวิตประจำวันมา 3 ตัวอย่าง

1. คิวที่ธนาคาร:
2. คิวของเครื่องพิมพ์
3. คิวการส่งข้อความในแอปพลิเคชัน

4. หากต้องการทำ Priority Queue จะต้องปรับปรุงโค้ดอย่างไร? ตอบแบบเข้าใจง่ายๆ

Priority Queue คือการจัดการคิวที่ไม่เพียงแค่ลำดับการมาถึงของข้อมูลเท่านั้น แต่ยังคำนึงถึงลำดับความสำคัญของข้อมูลด้วยการปรับปรุงโค้ดเพื่อให้ทำงานได้แบบ Priority Queue จะต้องเพิ่มการจัดลำดับความสำคัญ (Priority) ให้กับแต่ละข้อมูลที่ถูกเพิ่มเข้าไปในคิว

ใช้ คู่อันดับ (tuple) ที่เก็บข้อมูลและความสำคัญไว้ เช่น (priority, data), โดยที่ priority คือค่าความสำคัญ (ตัวเลขที่มากกว่าจะมีความสำคัญมากกว่า)

คิวจะต้องถูกจัดเรียงตามลำดับความสำคัญ โดยข้อมูลที่มีความสำคัญสูงสุดจะถูกให้บริการก่อน

ใน Python สามารถใช้ heapq หรือ sorted เพื่อสร้าง Priority Queue ได้