

Queue Python

วัตถุประสงค์

1. ศึกษาเรื่อง queue และ การเขียน class บน Python
2. เขียน class queue
3. ใช้ Python เขียน application เกี่ยวกับ queue

1. ทฤษฎี : การสร้าง Class, Queue & Queue Implementation

- | | |
|----------------------------------|---|
| 1.1. Queue | 2 |
| 1.2. Queue Data Implementation | 2 |
| 1.3. Queue Method Implementation | 3 |

2. การทดลองที่ 1 : Implement Queue Other Methods

- | | |
|---|---|
| 2.1. สร้าง class Queue, method enqueue() พร้อมทดสอบ | 2 |
| 2.2. เขียน method : dequeue() isEmpty() และ size() พร้อมทดสอบ | 4 |

3. การทดลองที่ 2 : Caesar Cipher 5

1. ทฤษฎี : การสร้าง Class, Stack & Stack Implementation

1.1. Queue



Queue แถวคอย คือแถว ที่เราเอาของเข้าทางด้านท้าย (rear / tail) และ เอาของออกทางด้านหัว (front / (head) เรียกว่า enQueue และ deQueue ตามลำดับ จะเห็นว่า ของที่เอาเข้าก่อนถูกเอาออกมาก่อน First in First out (FIFO)

นอกจากนี้ยังมี operation อื่นๆ อีก ดังแสดงใน ADT (abstract data type) ข้างล่าง

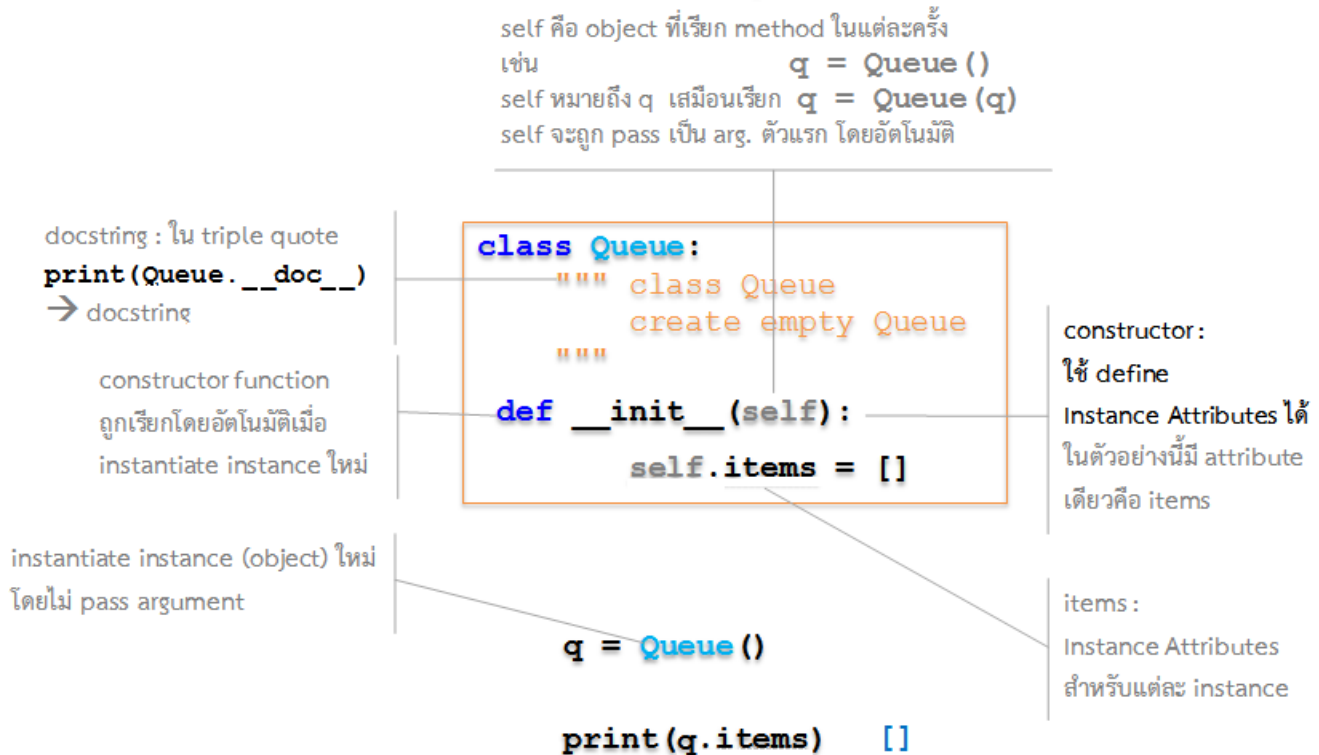
Data : แถว ของที่มีลำดับ มีปลายด้านหัว และ ท้าย	
Methods :	
init()	init empty queue
enQueue(i)	insert i ที่ top
i = deQueue()	return + เอาของที่ top ออก
b = isEmpty()	empty ?
b = isFull()	full ?
i = size()	return จำนวนของใน queue

1.2. Queue Data Implementation

เราต้องหา primitive data type ของ Python ซึ่งเก็บของได้หลายอัน และ เอาของเข้าออกที่ปลายด้านบนได้
--> Python List

Python List Type : สัญลักษณ์ []

1. เก็บของเรียงลำดับกัน
2. ใส่ของเข้าด้านท้าย append(i)
3. เอาของอันแรกออก i = pop(0)
4. i = len(L) returns จำนวนของใน list L



ข้างบนเป็นตัวอย่าง การ implement ส่วน data ของ class queue โดยใช้ Python List

ให้นักศึกษา implement โดยใช้ default argument เพื่อให้สามารถทำ code ข้างล่างได้

```

q1 = Queue(['A', 'B', 'C'])

print(q1.items)  ['A', 'B', 'C']
  
```

1.3. Queue Method Implementation

1.3.1. enQueue()

<pre> ใน class Queue: def enQueue(self, i): self.items.append(i) # insert i ที่ท้าย list </pre>	<pre> q = Queue() print(q.items) [] q.enQueue('A') print(q.items) ['A'] q.enQueue('B') print(q.items) ['A', 'B'] q.enQueue('C') print(q.items) ['A', 'B', 'C'] </pre>
--	---

2. การทดลองที่ 1 : Implement Queue Other Methods

2.1. สร้าง class Queue โดยอาจเขียนใหม่เอง หรือ ใช้ code ในข้อ 1

เขียน code ทดสอบความถูกต้อง instantiate Queue q เป็น empty queue วน loop enqueue() แต่ละ character ของ name = ชื่อนักศึกษา และ print items ของ q ทุก iteration

2.2. เขียน method ของ class Queue ในข้อ 1.2 จนครบ พร้อม code ทดสอบ ยกเว้น isFull() (เนื่องจาก Python ขยาย list เมื่อเต็ม)

2.2.1. size() return จำนวนของใน queue

```
def size(self):
```

```
    # YOUR CODE
```

code ทดสอบ : ใน loop ข้อ 2.1 print size() ของ Queue q ทุก iteration

2.2.2. isEmpty() return True ถ้า queue empty มิฉะนั้น return False

```
def isEmpty(self):
```


```
    # YOUR CODE
```

2.2.3. dequeue() return และ เอาของที่ front ออกจาก queue

```
def dequeue(self):
```

```
    # YOUR CODE
```

code ทดสอบ : loop dequeue() พร้อม print ของ ที่ dequeue ออกมา จน queue q empty (ใช้ isEmpty() ในข้อ 2.2.2)

3. การทดลองที่ 2 : Caesar Cipher**Decoded message**


I		l	o	v	e		P	y	t	h	o	n
2		5	6	1	8		3	2	5	6	1	8
K		q	u	w	m		S	a	y	n	p	v

Encoded message

Julius Caesar ได้ใช้การ encode ข่าวสารในการทหาร 50 ก่อนคริสตกาล โดยบวกค่า i เพื่อให้เอกสารเปลี่ยนไป ข้าศึกจะได้ไม่รู้ว่าความลับ เช่น หาก $i = 5$ แล้ว a จะเขียนแทนด้วย f และ b จะกลายเป็น g

Caesar Cipher ง่ายที่จะเดา หากเราเปลี่ยนให้การ encode ยากขึ้นโดยให้ค่า i มีหลายตัวขึ้นกับตำแหน่งของตัวอักษร เช่น ในตัวอย่าง ค่า i เป็น series ของเลข 2 5 6 1 8 3 โดยใช้ค่าเหล่านี้นวนไปเรื่อยๆ ดังรูป

จงเขียน Python functions encode และ decode เพื่อ return ข้อความที่ถูก encode และ decode input string ข้อความ และ i series รันทดสอบ ฟังก์ชันของท่าน

```
>>>ord('a')
97
>>>chr(97)
'a'
>>>ord('z')
122
>>>ord('Z')
90
>>>ord('A')
65
```