ABBOTTABAD UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

lab task 5

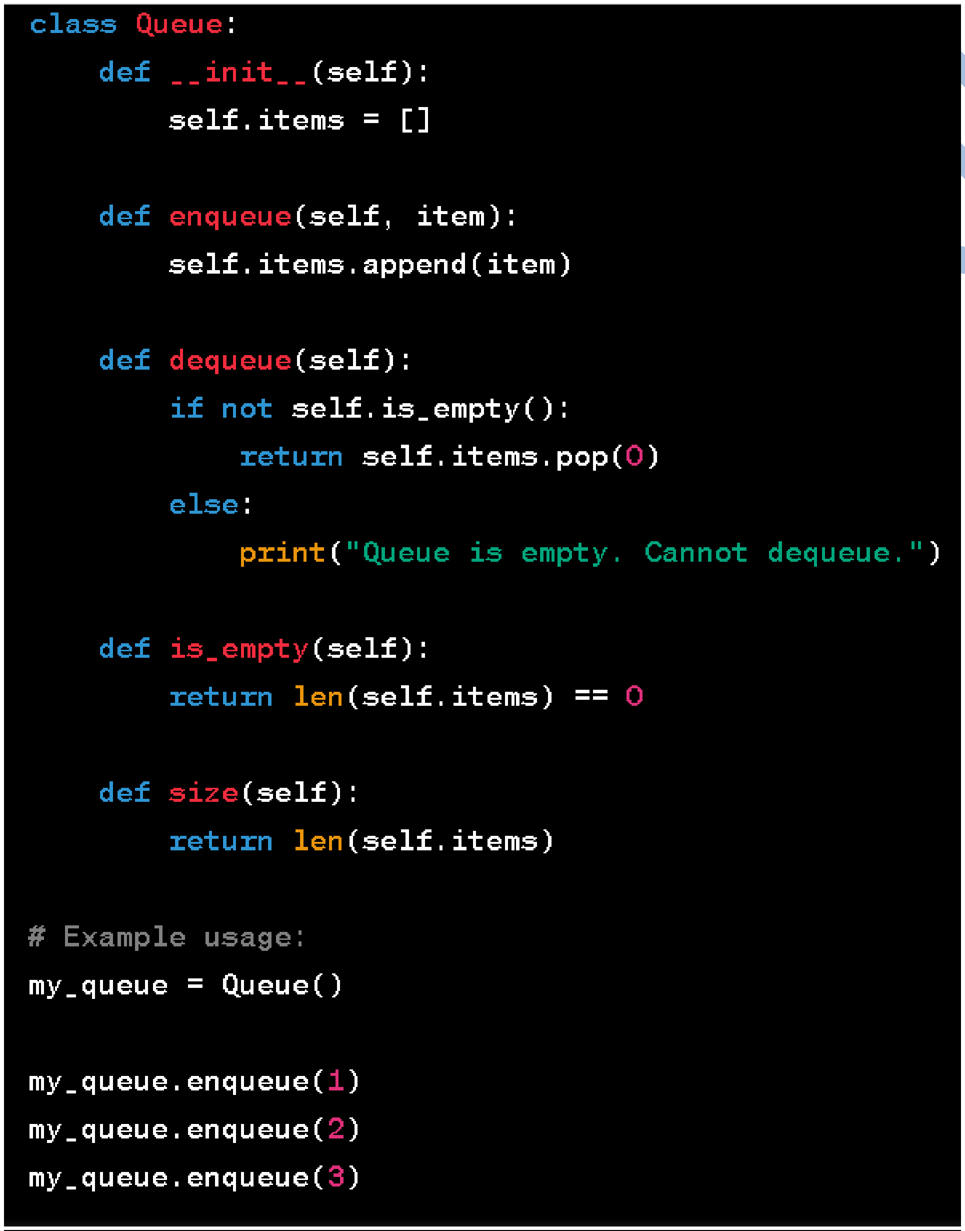
Data st..re & algorithim

Teacher	:	jamal abdul ahad
Created by	:	Tahir zubair
Roll no	:	12401
Class	:	BSSE 'C'

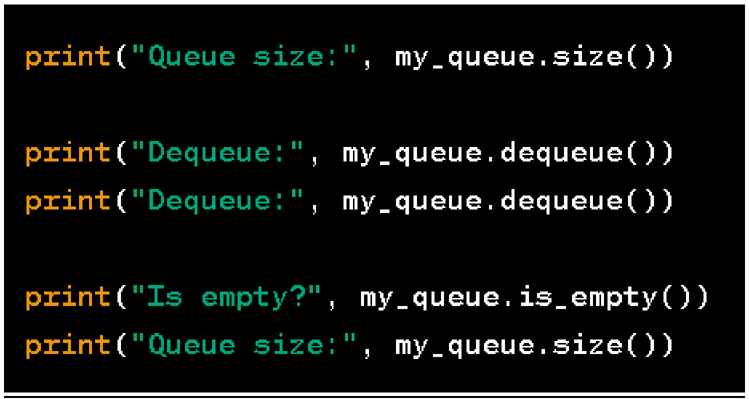
Question no 01

Implement a basic queue using an array in Python. Include methods for enqueue, dequeue, checking if the queue is empty, and finding the size of the queue.

Solution

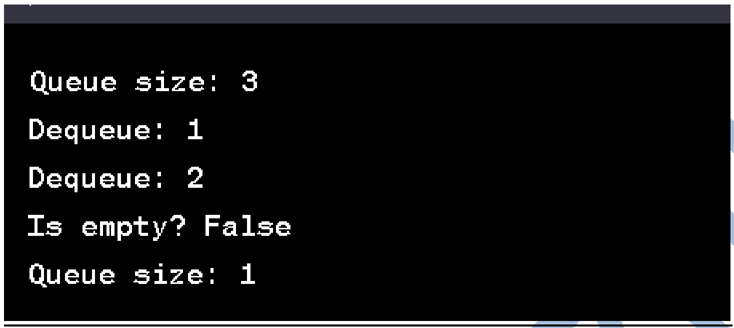


```python
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        if not self.is_empty():
            return self.items.pop(0)
        else:
            print("Queue is empty. Cannot dequeue.")

    def is_empty(self):
        return len(self.items) == 0

    def size(self):
        return len(self.items)

# Example usage:
my_queue = Queue()

my_queue.enqueue(1)
my_queue.enqueue(2)
my_queue.enqueue(3)
```



```python
print("Queue size:", my_queue.size())

print("Dequeue:", my_queue.dequeue())
print("Dequeue:", my_queue.dequeue())

print("Is empty?", my_queue.is_empty())
print("Queue size:", my_queue.size())
```

Output :

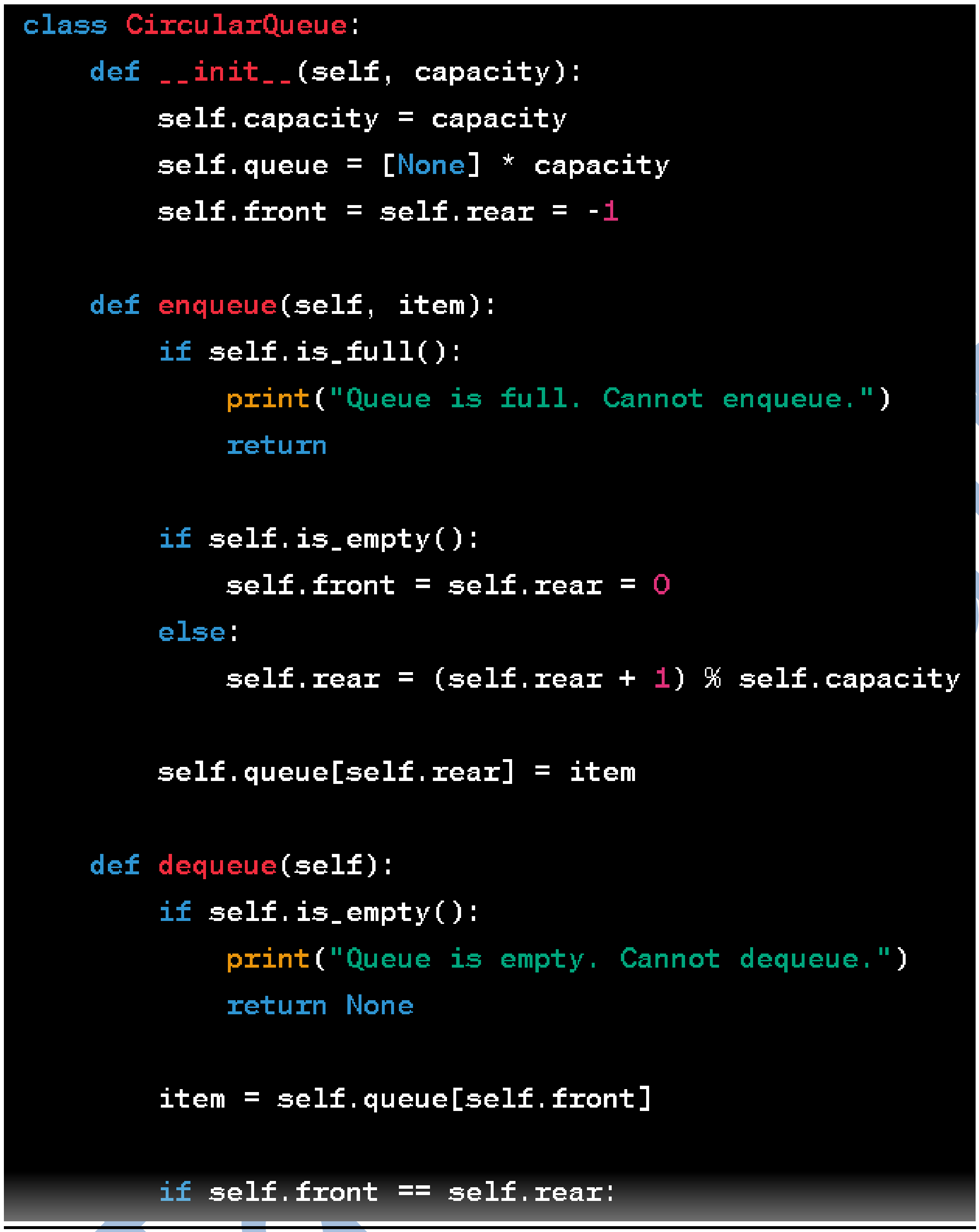


```
Queue size: 3
Dequeue: 1
Dequeue: 2
Is empty? False
Queue size: 1
```

Question no 02

Implement a circular queue in Python. Include methods for enqueue, dequeue, checking if the queue is empty, checking if the queue is full, and finding the size of the queue.

Solution



```python
class CircularQueue:
    def __init__(self, capacity):
        self.capacity = capacity
        self.queue = [None] * capacity
        self.front = self.rear = -1

    def enqueue(self, item):
        if self.is_full():
            print("Queue is full. Cannot enqueue.")
            return

        if self.is_empty():
            self.front = self.rear = 0
        else:
            self.rear = (self.rear + 1) % self.capacity

        self.queue[self.rear] = item

    def dequeue(self):
        if self.is_empty():
            print("Queue is empty. Cannot dequeue.")
            return None

        item = self.queue[self.front]

        if self.front == self.rear:
```

```python
        if self.front == self.rear:
            self.front = self.rear = -1
        else:
            self.front = (self.front + 1) % self.capacity

        return item

    def is_empty(self):
        return self.front == self.rear == -1

    def is_full(self):
        return (self.rear + 1) % self.capacity == self.front

    def size(self):
        if self.is_empty():
            return 0
        elif self.is_full():
            return self.capacity
        else:
            return (self.rear - self.front + self.capacity) % self.

# Example usage:
my_circular_queue = CircularQueue(5)
```

```python
my_circular_queue.enqueue(1)
my_circular_queue.enqueue(2)
my_circular_queue.enqueue(3)
my_circular_queue.enqueue(4)

print("Queue size:", my_circular_queue.size())

print("Dequeue:", my_circular_queue.dequeue())
print("Dequeue:", my_circular_queue.dequeue())

print("Is empty?", my_circular_queue.is_empty())
print("Is full?", my_circular_queue.is_full())
print("Queue size:", my_circular_queue.size())
```

<u>Output :</u>

```
Queue size: 4
Dequeue: 1
Dequeue: 2
Is empty? False
Is full? False
Queue size: 2
```