Queue Implementation

February 5, 2021

[ ]:
```python
class Queue:

    # To initialize the object.
    def __init__(self, c):
        self.queue = []
        self.front = self.rear = 0
        self.capacity = c

    # Function to insert an element at the rear of the queue
    def Enqueue(self, data):
    # Check queue is full or not
        if(self.capacity == self.rear):
            print("\nQueue is full")
        # Insert element at the rear
        else:
            self.queue.append(data)
            self.rear += 1

    # Function to delete an element from the front of the queue

    def Dequeue(self):
        # If queue is empty
        if(self.front == self.rear):
            print("Queue is empty")

        else:                       # Pop the front element from list
            x = self.queue.pop(0)
            self.rear -= 1

    # Function to print queue elements
    def Display(self):
        if(self.front == self.rear):
            print("\nQueue is Empty")

    # Traverse front to rear to print elements
        for i in self.queue:  #use a for loop to print all the element in queue
            print(i, "<--", end = '')
    # Print front of queue
    def Front(self):
        if(self.front == self.rear):
            print("\nQueue is Empty")

        print("\nFront Element is:",
        self.queue[self.front])
```

[ ]:
```
q = Queue(4)
```

[ ]:
```
# Print queue elements
q.Display()
```

[ ]:
```
# Inserting elements in the queue
q.Enqueue(220)
q.Enqueue(350)
q.Enqueue(42)
q.Enqueue(59)
```

[ ]:
```
# Print queue elements
q.Display()
```

[ ]:
```
# Insert element in queue
q.Enqueue(60)
```

[ ]:
```
# Print queue elements
q.Display()
```

[ ]:
```
q.Dequeue()
q.Dequeue()
```

[ ]:
```
print("\n\nafter two dequeue\n")
# Print queue elements
q.Display()
```

[ ]:
```
# Print front of queue
q.Front()
```

0.1 Additonal Links

Python has an inbuilt synchronized queue class Queue module

[ ]:
```

```