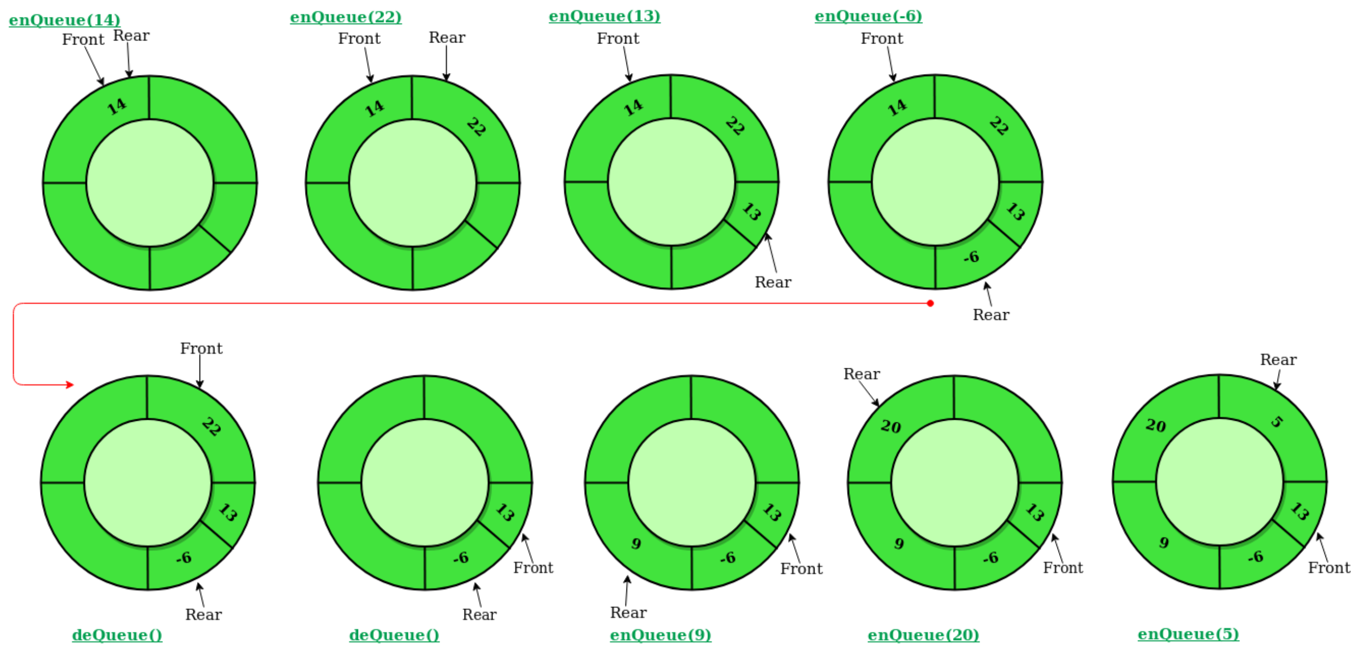
Circular Queue | Set 1 (Introduction and Array Implementation)

Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called **‘Ring Buffer’**.



```
class CircularQueue():

    # constructor
    def __init__(self, size): # initializing the class
        self.size = size

        # initializing queue with none
        self.queue = [None for i in range(size)]
        self.front = self.rear = -1

    def enqueue(self, data):

        # condition if queue is full
        if ((self.rear + 1) % self.size == self.front):
            print(" Queue is Full\n")

        # condition for empty queue
        elif (self.front == -1):
            self.front = 0
            self.rear = 0
            self.queue[self.rear] = data
        else:

            # next position of rear
            self.rear = (self.rear + 1) % self.size
            self.queue[self.rear] = data

    def dequeue(self):
        if (self.front == -1): # condition for empty queue
            print ("Queue is Empty\n")

        # condition for only one element
        elif (self.front == self.rear):
            temp=self.queue[self.front]
            self.front = -1
            self.rear = -1
            return temp
        else:
            temp = self.queue[self.front]
            self.front = (self.front + 1) % self.size
            return temp

    def display(self):

        # condition for empty queue
        if(self.front == -1):
            print ("Queue is Empty")

        elif (self.rear >= self.front):
            print("Elements in the circular queue are:",
                                                  end = " ")
            for i in range(self.front, self.rear + 1):
                print(self.queue[i], end = " ")
            print ()

        else:
            print ("Elements in Circular Queue are:",
                                               end = " ")
            for i in range(self.front, self.size):
                print(self.queue[i], end = " ")
            for i in range(0, self.rear + 1):
                print(self.queue[i], end = " ")
            print ()

        if ((self.rear + 1) % self.size == self.front):
            print("Queue is Full")

# Driver Code
ob = CircularQueue(5)
ob.enqueue(14)
ob.enqueue(22)
ob.enqueue(13)
ob.enqueue(-6)
ob.display()
print ("Deleted value = ", ob.dequeue())
print ("Deleted value = ", ob.dequeue())
ob.display()
ob.enqueue(9)
ob.enqueue(20)
ob.enqueue(5)
ob.display()
```

Applications:

1. **Memory Management:** The unused memory locations in the case of ordinary queues can be utilized in circular queues.
2. **Traffic system:** In computer controlled traffic system, circular queues are used to switch on the traffic lights one by one repeatedly as per the time set.
3. **CPU Scheduling:** Operating systems often maintain a queue of processes that are ready to execute or that are waiting for a particular event to occur.