ITD62-124 Data Structure

Queue

Outline:

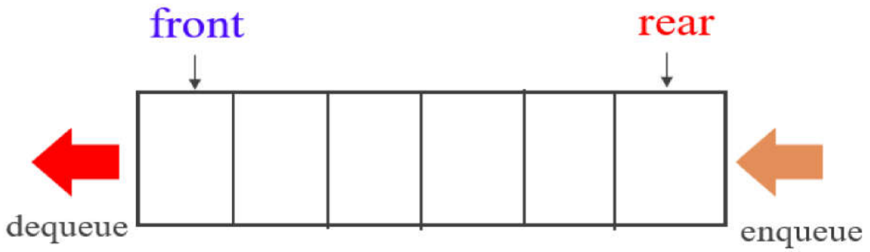
- Introduction
- Queue Operation
- Circular Queue

Introduction

- Queue is a data structure in which both ends are used:
    - ✓ one for adding new elements
    - ✓ one for removing them
- Example: a printer / job queue

Introduction

- FIFO structure: First in / First out
- the first element inserted is the first one to be removed



Queue Operation

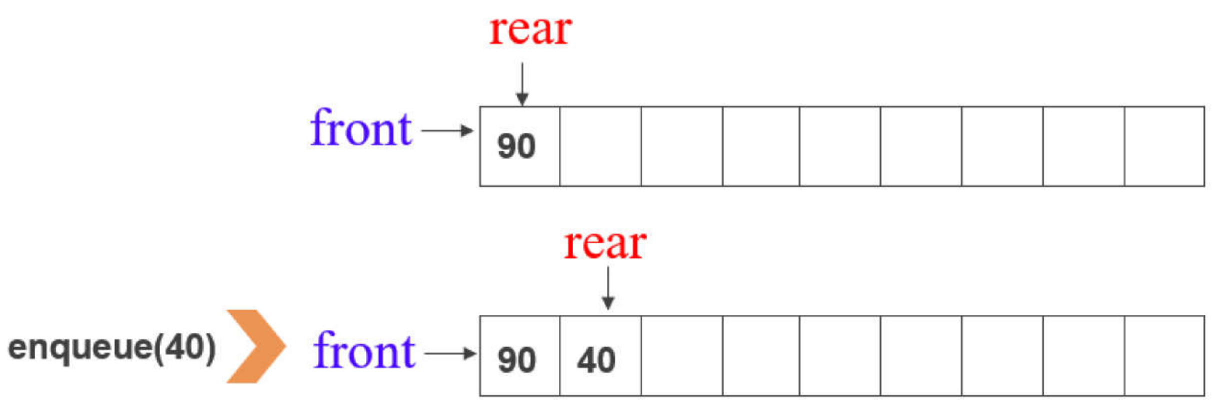
- the changes have to be monitored both at the beginning of the queue and at the end
- *enqueue* → add an item/element at the rear
- *dequeue* → remove an item/element from front

Queue Operation

- *create( )*: create a queue
- *enqueue(el)*: put the element *el* at the end of the queue
- *dequeue( )*: take the first element from the queue
- *length( )*: return the number of elements in the queue

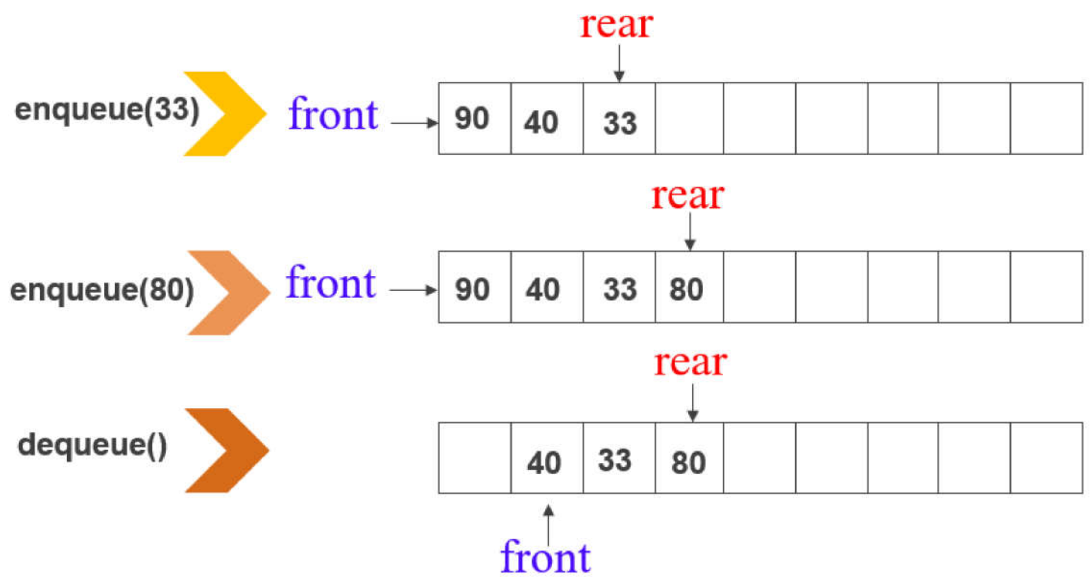
Queue Operation

- Example:



Queue Operation

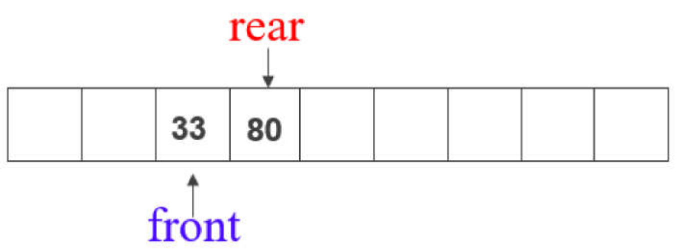
- Example:



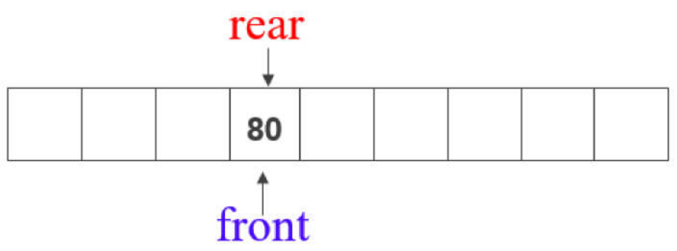
Queue Operation

- Example:

dequeue()

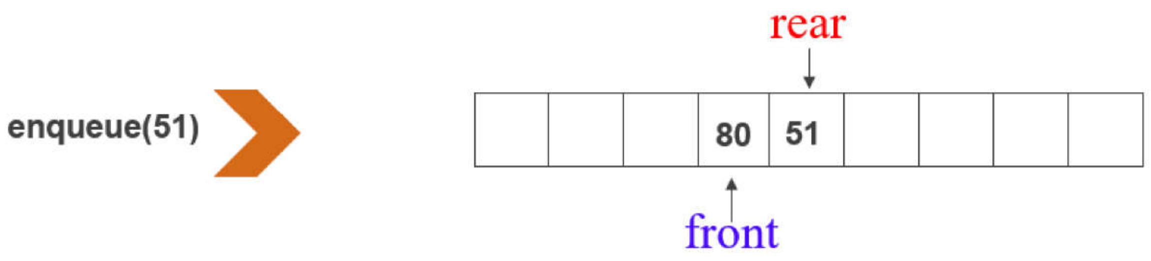


dequeue()



Queue Operation

- Example:



Queue Operation

- Example: create a queue

```
class Queue:
    def __init__(self, n):
        self.n = n
        self.queue = [ ] * n
        self.front = -1
        self.rear = -1



q = Queue(n)
```

Queue Operation

- Example: Add an element

```
def enqueue(self, item):
  if self.rear == self.n -1:
     print('Queue is full')
  else:
     if self.length() == 0:
         self.queue.append(item)
        self.front = 0
        self.rear = 0
     else:
        self.rear = self.rear + 1
        self.queue.append(item)
```

Queue Operation

- Example: Remove an element

```
def dequeue(self):
  if self.length() == 0:
     print('Queue is empty ')
  else:
     self.queue.pop(self.front)
     self.front = self.front + 1
```

Queue Operation

- Example: Display the number of elements in the queue

```
def length(self):
    return len(self.queue)
```

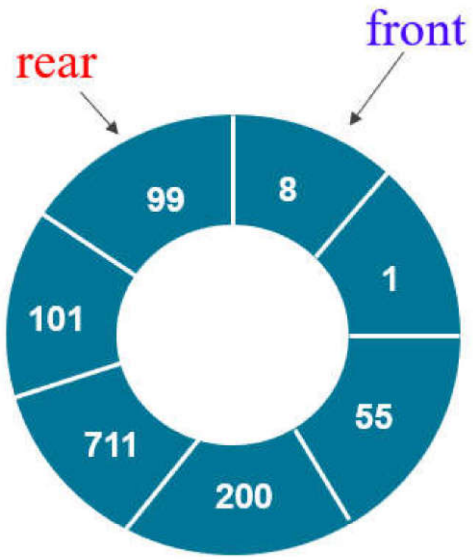
Queue Operation

- Example: Display the queue

```
def display(self):
    print('Queue = ', end = '  ')
    print(self.queue)
```

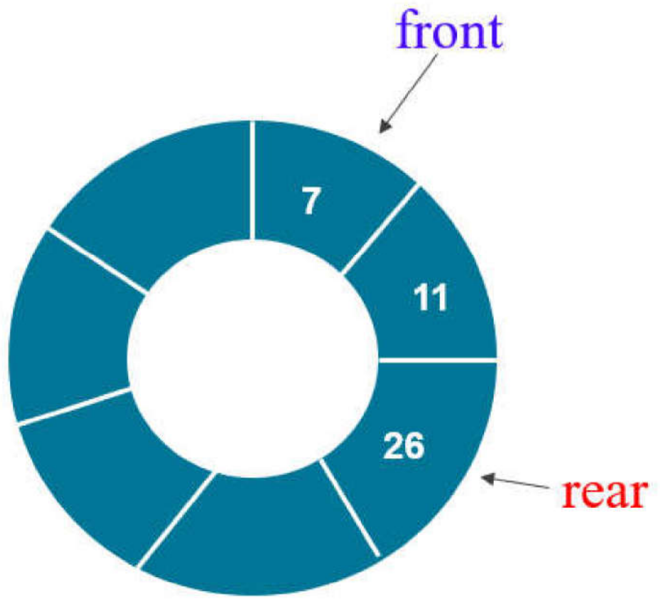
Circular Queue

- Example:



Circular Queue

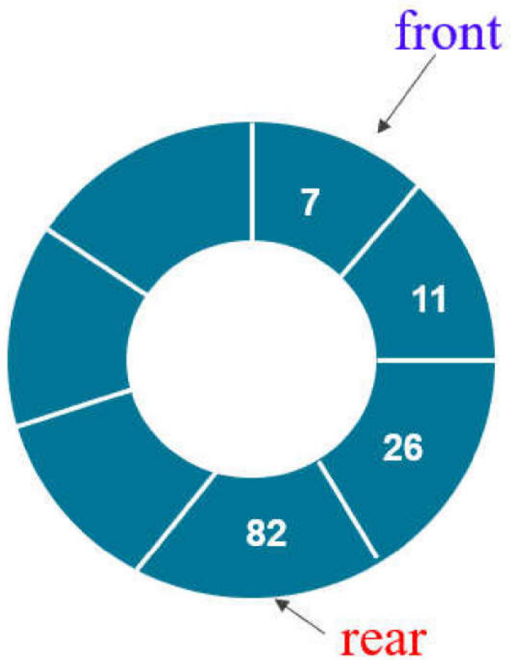
- Example:



Circular Queue

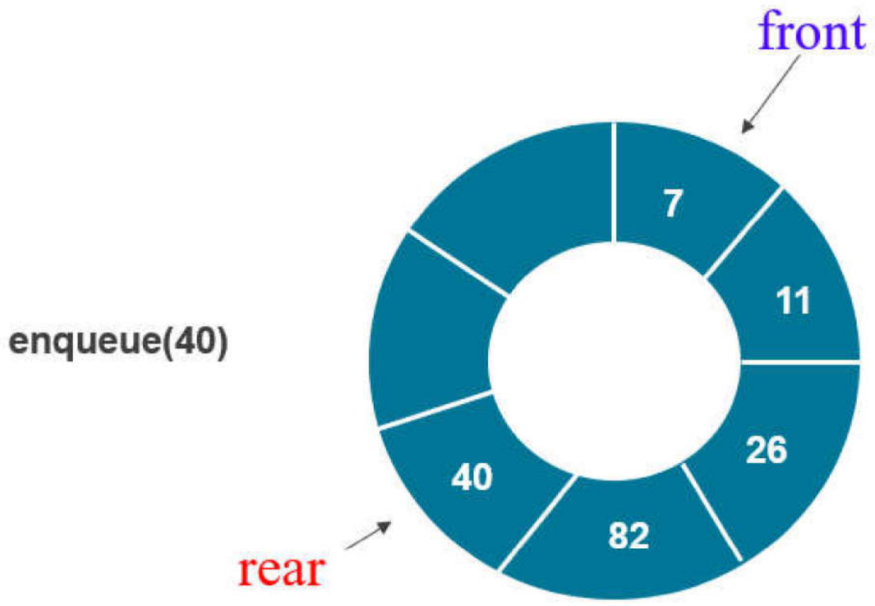
- Example:

enqueue(82)



Circular Queue

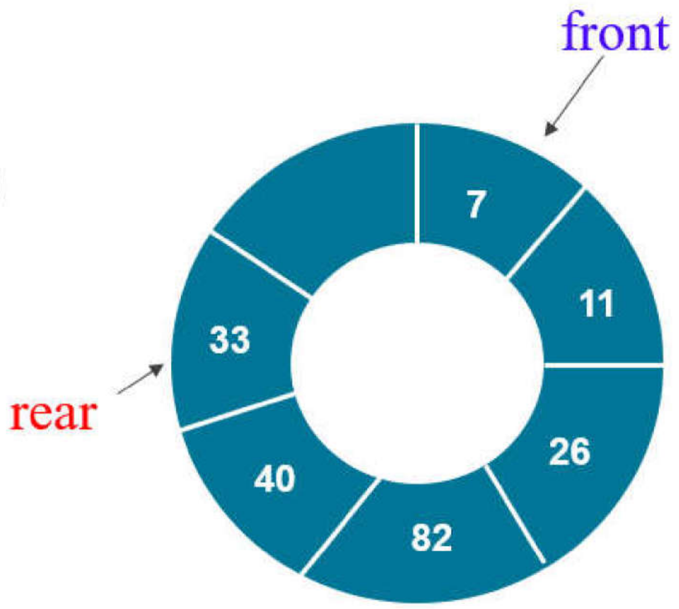
- Example:



Circular Queue

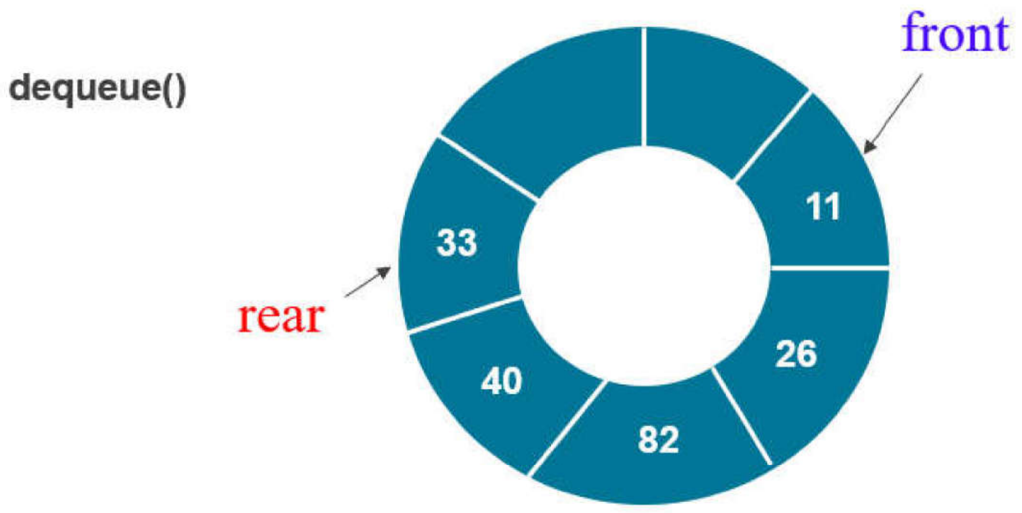
- Example:

enqueue(33)



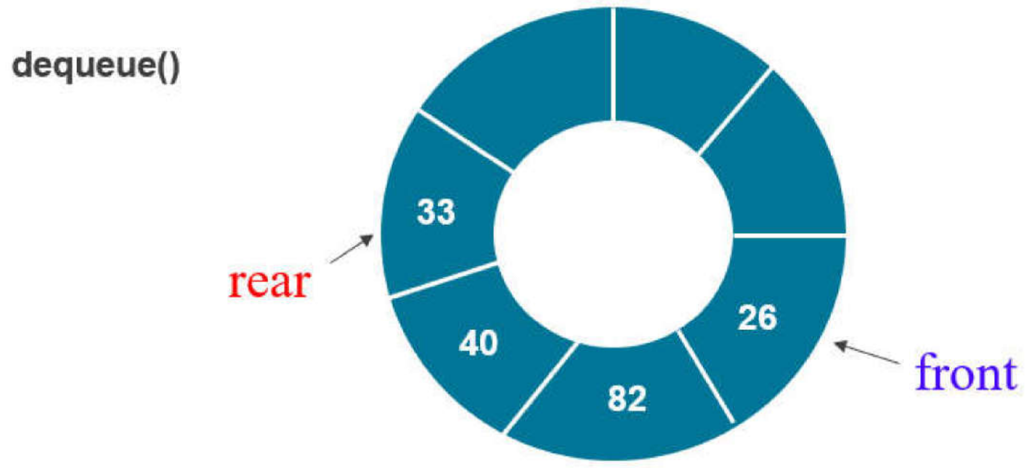
Circular Queue

- Example:



Circular Queue

- Example:



Circular Queue

- Example: create a circular queue

```
class CircularQueue:
    def __init__(self, n):
        self.n = n
        self.queue = [ ] * n
        self.front = -1
        self.rear = -1


q = CircularQueue(n)
```

Circular Queue

- Example: Add an element

```
def enqueue(self, data):
    if ((self.rear+1) % self.n == self.front):
        print('circular queue is full')
    elif (self.front == -1):
        self.front = 0
        self.rear = 0
        self.queue.append(data)
    else:
        self.rear = (self.rear + 1) % self.n
        self.queue.append(data)
```

Circular Queue

- Example: Remove an element

```
def dequeue(self):
   if (self.front == -1):
      print('circular queue is empty')
   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.n
      return temp
```

Circular Queue

- Example: Display the queue

```
def display(self):
    if(self.front == -1):
        print('No element in the circular queue')
    elif (self.rear >= self.front):
        for i in range(self.front, self.rear + 1):
            print(self.queue[i], end = "   ")
        print( )
    else:
        for i in range(self.front, self.n):
            print(self.queue[i], end= ' ')
        for i in range(0, self.rear + 1):
            print(self.queue[i], end= ' ')
        print( )
```

Class Activity

Q & A

Formative Assessment