Queue

In [ ]:
```
--> Queue is also a linear Data Structure. which follows the principle ofFIFO(First in first out).
--> Whatever is coming first will be move out first.
--> In case of queue insertion is done from rear end and deletion is done from front end.
```

Terminologies and Operations of Queue

In [ ]:
```
Terminologies of Queue:

    front --> Front of the queue(Start Point)
    rear --> Last item or end of the point

Operations of Queue:
    1.Enqueue --> Insert an element into a queue --> rear
    2.Dequeue --> Delete an element from a queue --> front
    3.Front --> will return yoU first element of queue.

In case of stack Insertion and deletion will be done from same end(TOP).
In case of queue insertion is done from rear end and deletion is done from front end.
```

How to Implement a Queue

In [ ]:
```
There are Two ways to Implement a Queue:
    1. By Using Array
    2. By Using Linkedlist
```

Implementation of queue using Array

In [3]:
```
#Implementation of queue using Array
class Queue:
    def __init__(self):
        self.array=[]
        self.front=0

    def enqueue(self,data):
        self.array.append(data)
        return self.array

    def dequeue(self):
        if len(self.array)==0:
            return "Queue is empty"
        x = self.array.pop(0)
        return self.array

    def Front(self):
        if len(self.array)==0:
            return "Queue is empty"
        return self.array[self.front]
    def isEmpty(self):
        if len(self.array)==0:
            return True
        else:
            return False


q=Queue()
print(q.enqueue(1))
print(q.enqueue(2))
print(q.enqueue(3))
q.enqueue(3)
print(q.Front())
print(q.dequeue())
print(q.Front())
print(q.dequeue())
print(q.Front())
```

```
[1]
[1, 2]
[1, 2, 3]
1
[2, 3, 3]
2
[3, 3]
3
```

Implementation of Queue Using Linkedlist

In [4]:
```
#Implementation of Queue using Linkedlist:
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None


class Queue:
    def __init__(self):
        self.head=None
        self.count=0
    def enqueue(self,data):
        if self.head is None:
            newNode=Node(data)
            self.head=newNode
            self.count=self.count+1
            return self.head
        curr=self.head
        while curr.next is not None:
            curr=curr.next
        newNode=Node(data)
        self.count=self.count+1
        curr.next=newNode

    def dequeue(self):
        if self.count==0:
            return "Queue is empty"
        ele=self.head.data
        self.head=self.head.next
        self.count-=1
        return ele
    def front(self):
        if self.count==0:
            return "Queue is empty"
        return self.head.data
    def size(self):
        return self.count
q=Queue()
q.enqueue(10)
q.enqueue(20)
print(q.dequeue())
print(q.front())
```

```
10
20
```

Applications of Queue

In [ ]:
```
--> Semaphores
--> FCFS ( first come first serve) scheduling, example: FIFO queue
--> Spooling in printers
--> Buffer for devices like keyboard
--> CPU Scheduling
--> Memory management
--> Queues in routers/ switches
--> Mail Queues

Real Time Applications of Queue:

    --> Applied as waiting lists for a single shared resource like CPU, Disk, and Printer.
    --> Applied as buffers on MP3 players and portable CD players.
    --> Applied on Operating system to handle the interruption.
    --> Applied to add a song at the end or to play from the front.
    --> Applied on WhatsApp when we send messages to our friends and they don't have an internet connection then these
        messages are queued on the server of WhatsApp.
    --> Traffic software ( Each  light gets on one by one after every time of interval of time.)
```

Types/Variations of Queue

In [ ]:
```
There are four types of Queue:
    1. Simple Queue or Linear Queue --> Insertion is done from Rear and Deletion is done from Front
    2. Circular Queue --> rear is connected with front
    3. Priority Queue --> Each and Every element has its own prioerity and based on that insertion deletion will be done
    4. Double Ended Queue (or Deque) --> Insertion and Deletion both can be done from both rear and front ends.

    In Collection Module Deque is already inbuilt in python for Insertion and Deletion:
        Collections deque:
            1.Pop --> delete from rear
            2.popleft --> delete from front
            3.Append--> add at rear
            4.Appendleft --> add at front
```