# COMPUTER SCIENCE AND DA

## Data Structures through Python

## Queues and Hash Tables

**Lecture No. 2**


By- Kashif Sir

# RECAP

Types of Queues →
- Simple Queue
- Circular Queue

} Insertion - Rear End
Deletion - Front End

- Double Ended Queue } Insertion & Deletion possible at both end.
- Priority Queue } Insertion - Rear End
Deletion - Front End
But on the priority

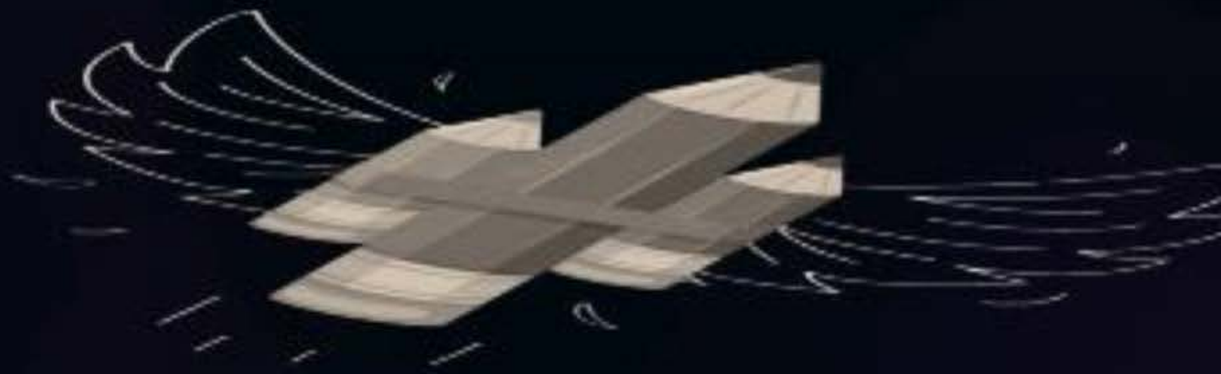
Enqueue( )

Dequeue( )

# TOPICS TO BE COVERED

Implementation

- Simple Queue
- Drawback of Simple Queue
- Circular Queue



# QUEUE

## Implementation of simple Queue using Brute force approach

Let Q be a simple Queue, with maxsize = 5

f = r = None

```
def Enqueue(value, Q):
    if rear == maxsize - 1:
        print("Queue is full")
        return
    if f is None and r is None:
        r = 0
        f = 0
    else
        r = r + 1
    Q[r] = value
```

Enqueue(10, Q)
↳ r = 0, f = 0
Q[0] = 10

Enqueue(20, Q) → r = 1
Q[1] = 20

Enqueue(30, Q)
r = 2
Q[2] = 30

Enqueue(40, Q)
r = 3
Q[3] = 40

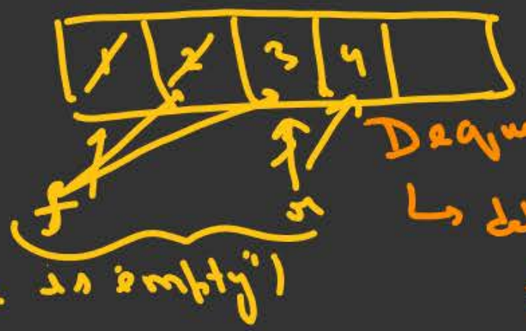
Enqueue(50, Q)
r = 4
Q[4] = 50

Enqueue(60, Q)
(Queue is full)

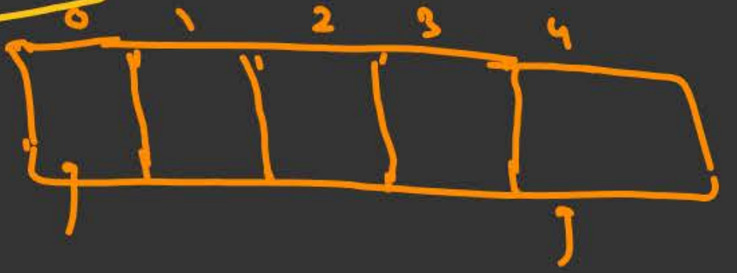
Q

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 50 |

```
def Dequeue(Q):
    if f == None:
        print("Queue is empty")
        return
    del_val = Q[f]
    if f == r:
        f = None
        r = None
    else
        f = f + 1
    return del_val
```




Dequeue(Q)
↳ del_val = Q[0] = 10
f = 1


Dequeue(Q)
↳ del_val = Q[1] = 20
f = 2

r = 4
f = 4

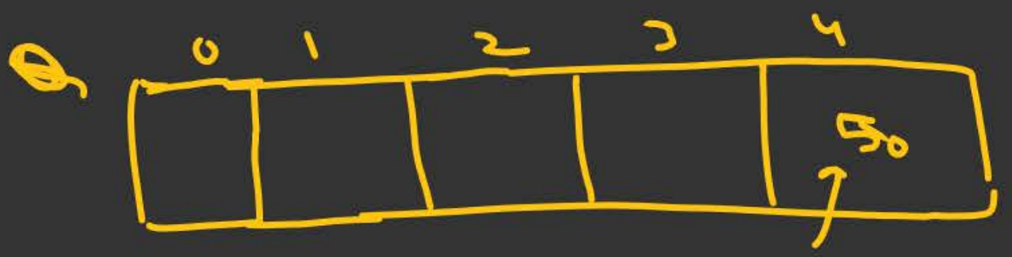
Dequeue(Q)
del_val = Q[2] = 30
f = 3

Dequeue(Q)
del_val = Q[3] = 40
f = 4

Dequeue(Q)
del_val = Q[4] = 50
f = None
r = None

Dequeue(Q)
Queue is empty

# Drawback of simple Queue





$r = 4$ None

$f = 0$ ~~X~~ ~~X~~ 4 None

Dequeue(Q)

Dequeue(Q)

Enqueue(60, Q) → Queue is full

Dequeue(Q)

Dequeue(Q)

Enqueue(60, Q) ⇒ Queue is full

Dequeue(Q) $r = 0$, $f = 0$

$r = r + 1$ (linearly)

## Solution

If $r$ moves in circular way (Circular Queue)

# THANK - YOU