Newton School of Technology

21

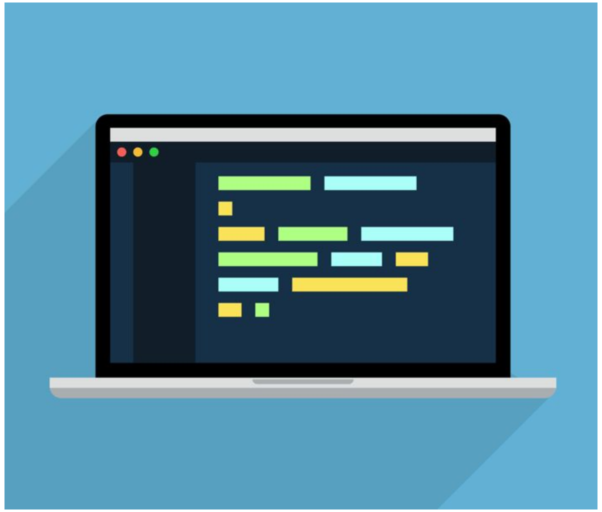
# Queue

by Gladden Rumao

CSA 221 : DSA

## Quick Recap :

- Stack
    - LIFO
    - Push, pop, peek/top.
    - Array implementation
    - Linked list implementation

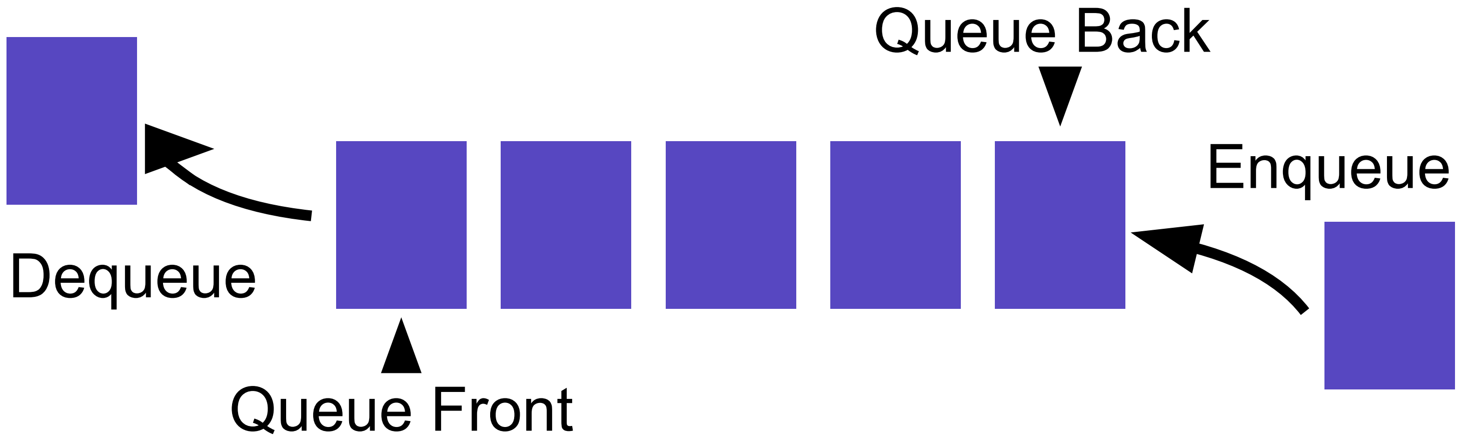


Newton School of Technology

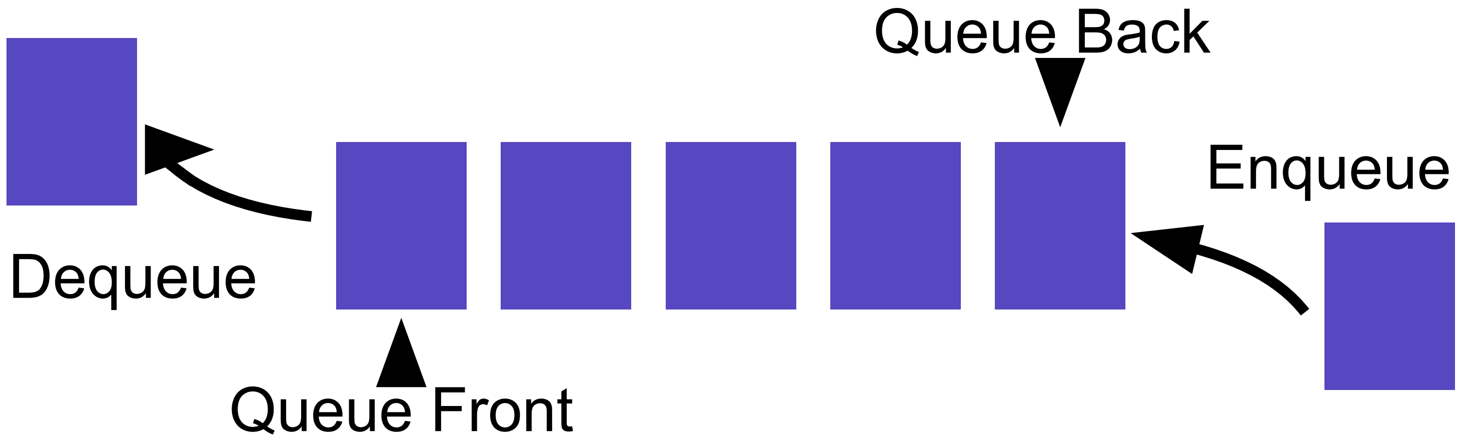
## Introduction to Queue

## What is a Queue?

A queue is a linear data structure which models real world queues by having two primary operations, namely **enqueue** and **dequeue**.



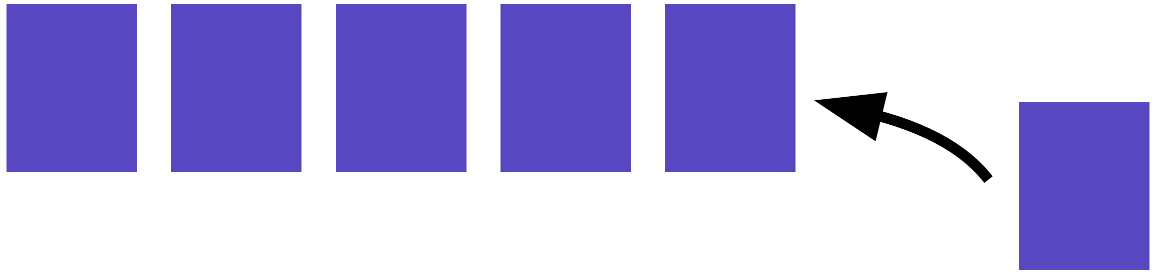
# Queue Terminology



## Queue Terminology

There does not seem to be consistent terminology for inserting and removing elements from queues.

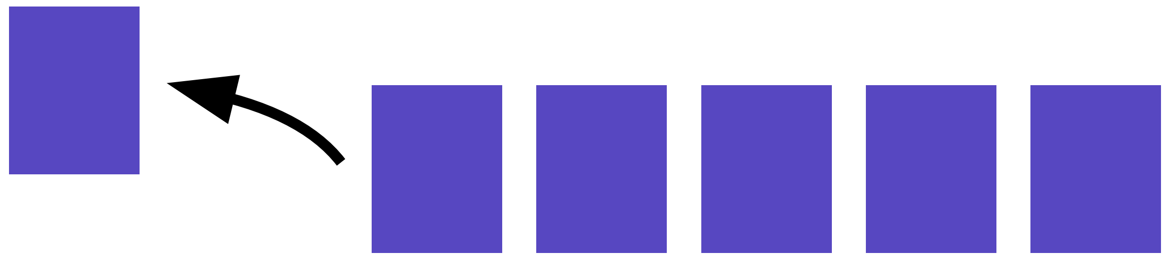
**Enqueue = Adding = Offering**



## Queue Terminology

There does not seem to be consistent terminology for inserting and removing elements from queues.

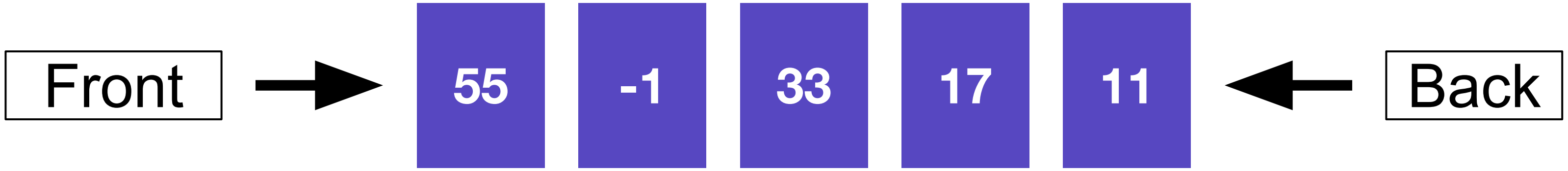
**Dequeue = Removing**



## Queue Example

**Instructions**:

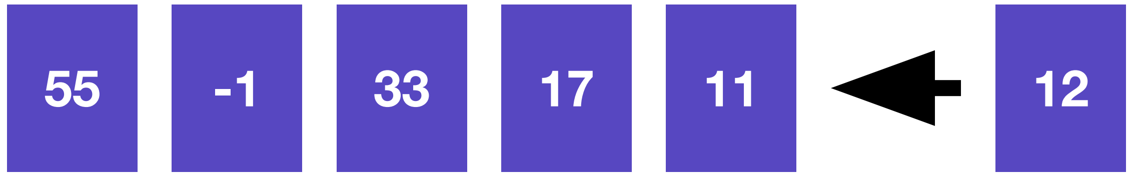
Enqueue(12)
Dequeue()
Dequeue()
Enqueue(7)
Dequeue()
Enqueue(-6)



## Queue Example

**Instructions**:

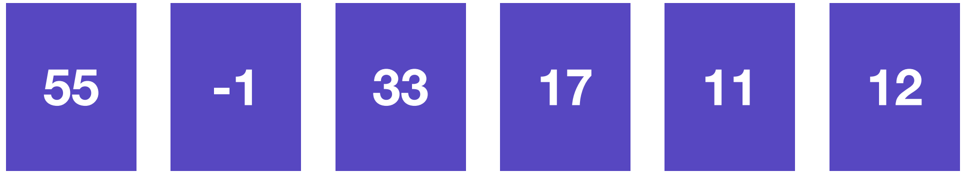
**Enqueue(12)**
Dequeue()
Dequeue()
Enqueue(7)
Dequeue()
Enqueue(-6)



## Queue Example

**<u>Instructions</u>**:

Enqueue(12)
Dequeue()
Dequeue()
Enqueue(7)
Dequeue()
Enqueue(-6)



## Queue Example

**Instructions**:

Enqueue(12)

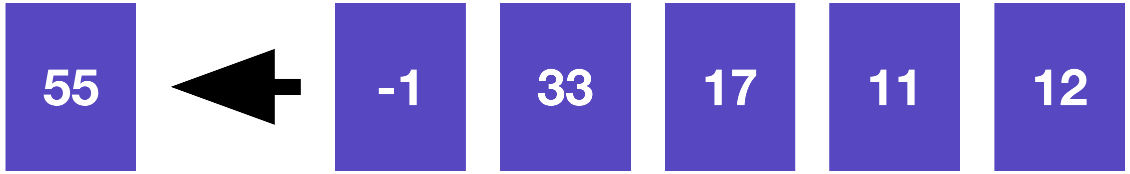
**Dequeue()**

Dequeue()

Enqueue(7)

Dequeue()

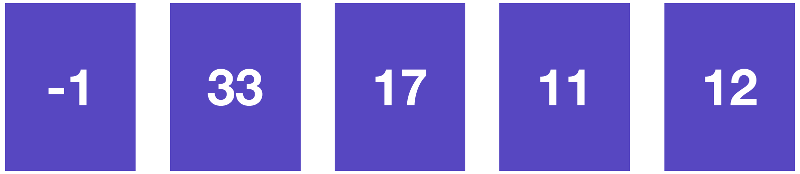
Enqueue(-6)



## Queue Example

**Instructions**:

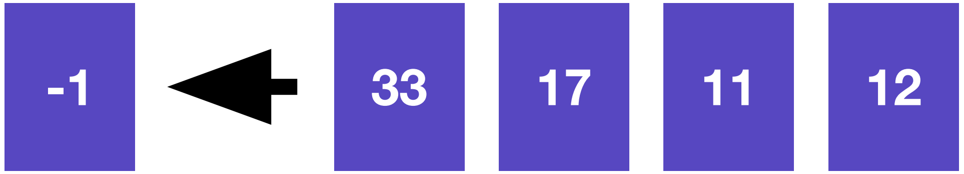
Enqueue(12)
Dequeue()
Dequeue()
Enqueue(7)
Dequeue()
Enqueue(-6)



## Queue Example

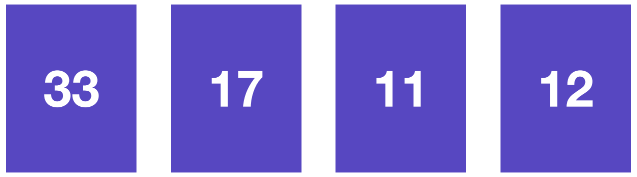
**Instructions**:

Enqueue(12)
Dequeue()
**Dequeue()**
Enqueue(7)
Dequeue()
Enqueue(-6)



## Queue Example

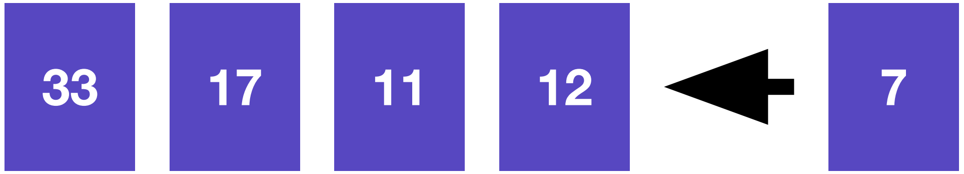
**<u>Instructions</u>**:
Enqueue(12)
Dequeue()
Dequeue()
Enqueue(7)
Dequeue()
Enqueue(-6)



## Queue Example

**Instructions**:

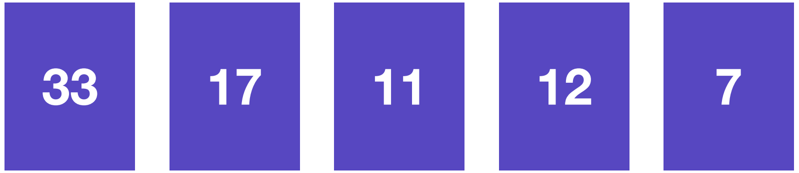
Enqueue(12)
Dequeue()
Dequeue()
**Enqueue(7)**
Dequeue()
Enqueue(-6)



## Queue Example

**Instructions**:

Enqueue(12)
Dequeue()
Dequeue()
Enqueue(7)
Dequeue()
Enqueue(-6)



## Queue Example

**Instructions**:

Enqueue(12)

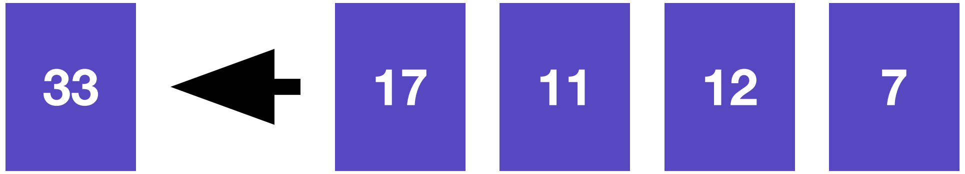
Dequeue()

Dequeue()

Enqueue(7)

**Dequeue()**

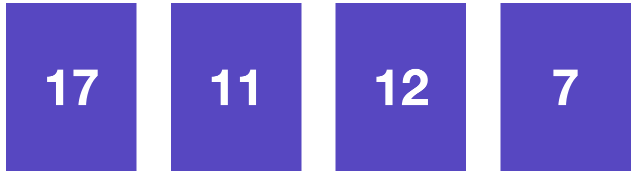
Enqueue(-6)



## Queue Example

**<u>Instructions</u>**:

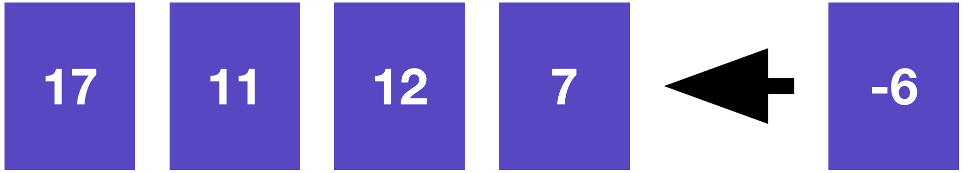
Enqueue(12)
Dequeue()
Dequeue()
Enqueue(7)
Dequeue()
Enqueue(-6)



## Queue Example

**Instructions**

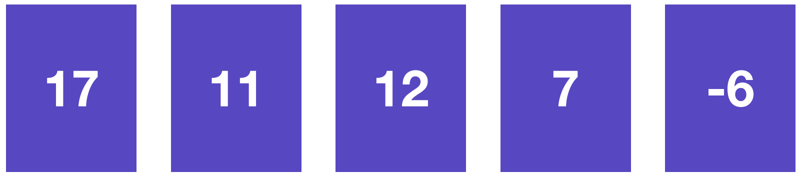
Enqueue(12)
Dequeue()
Dequeue()
Enqueue(7)
Dequeue()
**Enqueue(-6)**



## Queue Example

**Instructions**:

Enqueue(12)
Dequeue()
Dequeue()
Enqueue(7)
Dequeue()
Enqueue(-6)

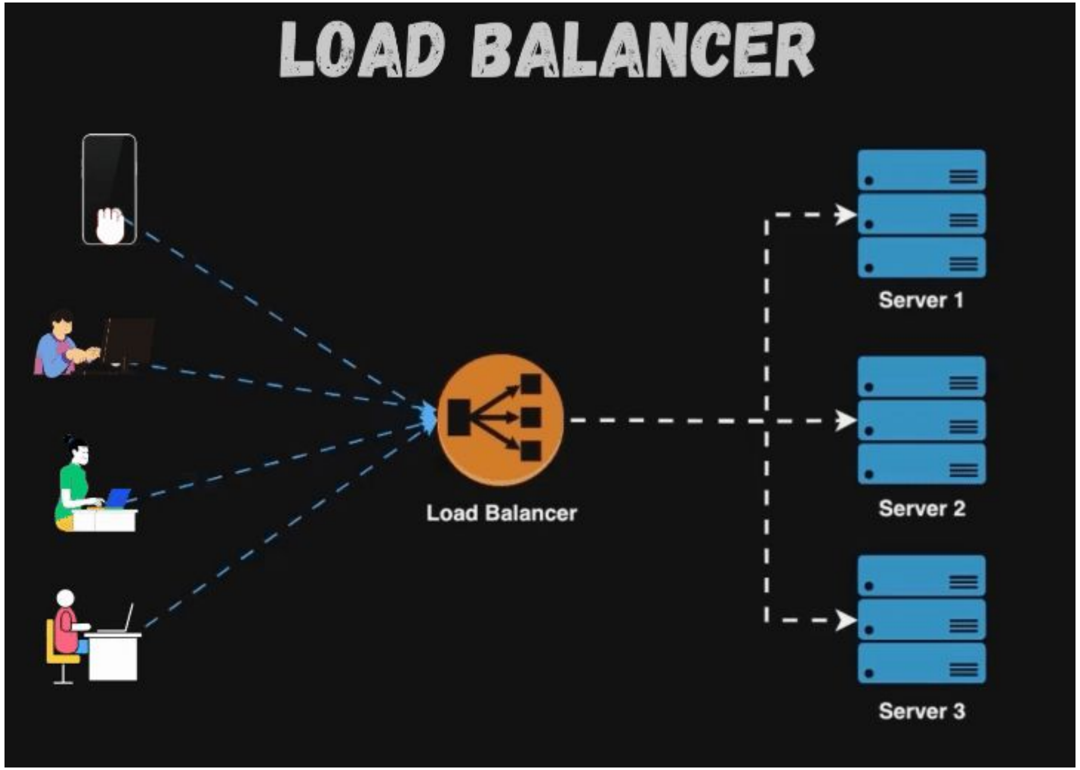


## When and where queue is used?

- Any waiting line models a queue, for example a lineup at a movie theatre.
- Can be used to efficiently keep track of the x most recently added elements.
- Web server request management where you want first come first serve.
- Breadth first search (BFS) graph traversal.

Newton School of Technology

## Queue Applications


LOAD BALANCER
Load Balancer
Server 1
Server 2
Server 3

Newton School of Technology

## Implementation of Queue

Newton School of Technology

## Queue Using Linked List

## Queue Using Doubly Linked List with head and tail

| Queue operation | Operation on Linked List |
|---|---|
| Enqueue | Insert at tail of the linked list. |
| Dequeue | Delete at head of the linked list. |
| Peek | Return the value at the head of the linked list. |
| Size | Return size of the linked list. |

## Implementation: Node & queue class

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class Queue:
    def __init__(self):
        self.front = None
        self.rear = None
        self.count = 0
```

## Implementation: enqueue and dequeue

```python
def enqueue(self, data):
    new_node = Node(data)
    if self.rear:
        self.rear.next = new_node
    self.rear = new_node
    if not self.front:
        self.front = new_node
    self.count += 1

def dequeue(self):
    if not self.front:
        raise IndexError("Queue is empty")
    data = self.front.data
    self.front = self.front.next
    if not self.front:
        self.rear = None
    self.count -= 1
    return data
```

## Implementation: peek and size

```python
    def peek(self):
        if not self.front:
            raise IndexError("Queue is empty")
        return self.front.data

    def size(self):
        return self.count
```

Newton School of Technology

## What is a deque?

## Deque confusion :

**Dequeue (Removing elements)**

**Deque (Double ended queue)**

## Queue vs Deque

| Data Structure | Operations Allowed |
|---|---|
| Queue | Enqueue at end, dequeue from start. |
| Deque | Enqueue at start or end (both), dequeue from start or end (both). |

Newton School of Technology

## Inbuilt Python Implementation

## Inbuilt Implementation: Deque

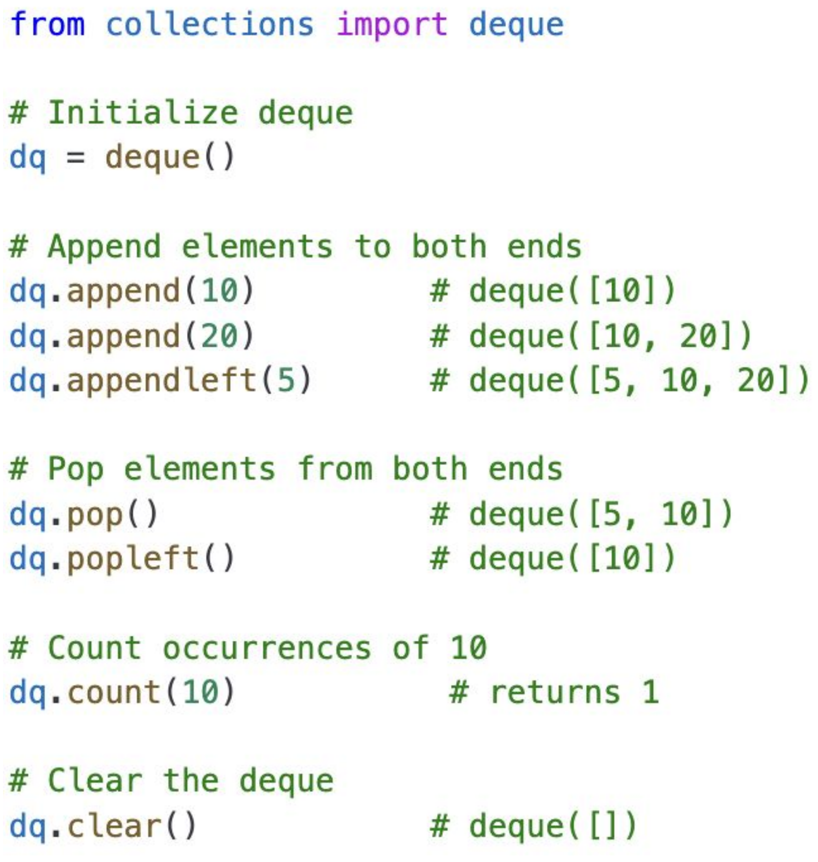
The **deque (double-ended queue)** from Python's `collections` module is a versatile and efficient data structure that allows fast appends and pops from **both ends**.

```
from collections import deque
```

## Inbuilt Functions: Deque

| Method | Description | Time Complexity |
|---|---|---|
| `append(x)` | Add element `x` to the **right end** (rear / end of queue) | O(1) |
| `appendleft(x)` | Add element `x` to the **left end** (front / start of queue) | O(1) |
| `pop()` | Remove and return element from the **right end** (rear) | O(1) |
| `popleft()` | Remove and return element from the **left end** (front) | O(1) |
| `clear()` | Remove **all elements** from the deque | O(n) |
| `count(x)` | Return the **number of occurrences** of element `x` | O(n) |
| `len(dq)` | Return the **number of elements** in the deque | O(1) |

# Inbuilt Implementation: Deque



```python
from collections import deque

# Initialize deque
dq = deque()

# Append elements to both ends
dq.append(10)         # deque([10])
dq.append(20)         # deque([10, 20])
dq.appendleft(5)      # deque([5, 10, 20])

# Pop elements from both ends
dq.pop()              # deque([5, 10])
dq.popleft()          # deque([10])

# Count occurrences of 10
dq.count(10)           # returns 1

# Clear the deque
dq.clear()            # deque([])
```

Newton School of Technology

## Complexity Analysis

## Time Complexity – Queue

| | |
|---|---|
| Enqueue at end | O(1) |
| Dequeue from start | O(1) |
| Peek / Accessing start | O(1) |
| Enqueue at start | ❌ |
| Dequeue from end | ❌ |

## Time Complexity - Deque

| | |
|---|---|
| Enqueue at end | O(1) |
| Dequeue from start | O(1) |
| Peek / Accessing start or end | O(1) |
| Enqueue at start | O(1) |
| Dequeue from end | O(1) |

Newton School of Technology

## Summary Quiz

Newton School of Technology

# END