**QUEUE**

queue is a linear data structure that stores items in a First In First Out (FIFO) manner. With a queue, the least recently added item is removed first.

* **Enqueue:** Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition – Time Complexity: O(1)
* **Dequeue:** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition – Time Complexity: O(1)
* **Front:** Get the front item from the queue – Time Complexity: O(1)
* **Rear:** Get the last item from the queue – Time Complexity: O(1)

**Basic Queue Operation:**

class Queue:  
 def \_\_init\_\_(self):  
 self.queue = []  
  
 def enqueue(self, item): #adds an element to the end of the queue happens at the rear end of the queue time complexity is O(1)  
 self.queue.append(item)  
  
 def dequeue(self):#removes the first element of the queue happens at front end of the queue time complexity is O(1) where n is the size of the queue  
 if self.is\_empty():  
 return None  
 return self.queue.pop(0)  
  
 def is\_empty(self):  
 return len(self.queue) == 0  
  
 def size(self):  
 return len(self.queue)  
  
 def \_\_str\_\_(self):  
 return str(self.queue)  
  
  
  
queue = Queue()  
queue.enqueue(1)  
queue.enqueue(2)  
queue.enqueue(3)  
print(queue)  
print(queue.dequeue())  
print(queue.is\_empty())

**characteristics of a queue**:  
queue can handle multiple data types  
we can access the elements of the queue both ends  
fast and flexible  
  
**Applications of a queue**:  
Task scheduling  
for synchronizing tasks

**Circular queue:**

 the last element points to the first element making a circular link.

better memory utilization

If the last position is full and the first position is empty, we can insert an element in the first position. This action is not possible in a simple queue.

The circular queue solves the major limitation of the normal queue. In a normal queue, after a bit of insertion and deletion, there will be non-usable space

**Priority Queue**

each element is associated with a priority and is served according to its priority. If elements with the same priority occur, they are served according to their order in the queue.

Insertion occurs based on the arrival of the values and removal occurs based on priority.

In a queue, the **first-in-first-out rule** is implemented whereas, in a priority queue, the values are removed **on the basis of priority**.

**Deque (Double-Ended Queue):**

In a double-ended queue, elements can be inserted and removed from the front or rear. Thus, it does not follow the FIFO (First In First Out) rule.

**Types of Deque**

* **Input Restricted Deque**  
  In this deque, input is restricted at a single end but allows deletion at both ends.
* **Output Restricted Deque**  
  In this deque, output is restricted at a single end but allows insertion at both ends.

My [GitHub](https://github.com/manojreddy24/people_tech) link contains documentation and practiced problems https://github.com/manojreddy24/people\_tech