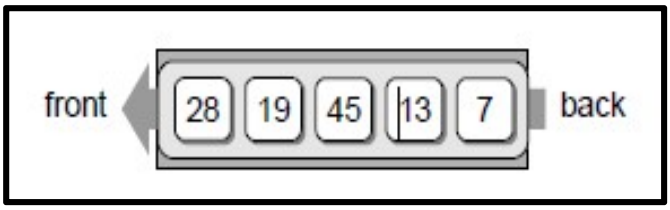
Week 10	Queue (First In First Out) Data Structures – Example: Media Player List, Keyboard Buffer Queue, Printer Queue etc. The Queue Implementation and its operations using Python Lists. Priority Queue and its implementation.

Queue

- **A Queue is a linear data structure, used to store group of data items such that the first item inserted is the first item removed.**
- **Queue data structure follows FIFO (First In First Out) principle for insertion-deletion operation.**
- **New items are inserted into a queue at the back/rear end while existing items are removed from the front end.**
- **Abstract View of Queue:**



Queue ADT

A Queue is a data structure that stores a linear collection of data items. New items are inserted at the back/rear end and existing items are removed from front end. An empty Queue is a queue containing no items.

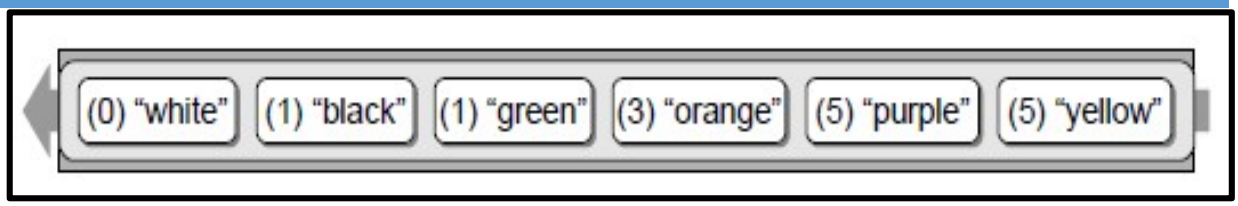
- **Queue( )**: Creates a new empty queue.
- **isEmpty( )**: Returns a Boolean value indicating whether the queue is empty or not.
- **length( )**: Returns the number of items in the queue.
- **enqueue(item)**: Adds the given item to the back/rear end of the queue.
- **dequeue()**: Removes and returns the front item of the queue, if the queue is not empty. Items cannot be dequeued from an empty queue.

Queue Implementation

```
class Queue:
def __init__(self):
        self.items = list()

def enqueue(self,value):
        self.items.append(value)

def  dequeue(self):
      if self.isEmpty() == True:
         print("Queue is empty, cannot remove")
      else:
         return self.items.pop(0)

def display(self):
      print("Queue items are")
      for i in self.items:
         print(i)

   def isEmpty(self):
      if len(self.items) == 0:
         return True
      else:
         return False

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

Priority Queue

- Priority Queue is a queue in which items are assigned a priority and the items with a higher priority are dequeued first.
- However, all items with the same priority still follow the FIFO principle. That is, if two items with the same priority are enqueued, the first in will be the first out.
- There are two basic types of priority queues: bounded and unbounded.
- The bounded priority queue assumes a small limited range of p priorities over the interval of integers [0 : : : p).
- The unbounded priority queue places no limit on the range of integer values that can be used as priorities.



Priority Queue ADT

A priority queue is a queue in which each item is assigned a priority and items with a higher priority are removed before those with a lower priority, irrespective of when they were added.

- **PriorityQueue():** Creates a new empty unbounded priority queue.
- **BPriorityQueue(numLevels):** Creates a new empty bounded priority queue with priority levels in the range from 0 to numLevels - 1.
- **isEmpty():** Returns a boolean value indicating whether the queue is empty.
- **length ():** Returns the number of items currently in the queue.
- **enqueue(item, priority):** Adds the given item to the queue by inserting itin the proper position based on the given priority. The priority value mustbe within the legal range when using a bounded priority queue.
- **dequeue():** Removes and returns the front item from the queue, which is theitem with the highest priority. The associated priority value is discarded. Iftwo items have the same priority, then those items are removed in a FIFOorder. An item cannot be dequeued from an empty queue.

Program #1: Python Program to implement Queue.

```
class Queue:
    def __init__(self):
        self.items = list()

    def enqueue(self,value):
        self.items.append(value)

    def  dequeue(self):
        if self.isEmpty() == True:
            print("Queue is empty, cannot remove")
        else:
            return self.items.pop(0)

    def display(self):
        print("Queue items are")
        for i in self.items:
            print(i)

    def isEmpty(self):
        if len(self.items) == 0:
            return True
        else:
            return False

    def length(self):
        return len(self.items)

Q = Queue()
print("Queue Empty?:",Q.isEmpty())
Q.enqueue(10)
Q.enqueue(20)
Q.enqueue(30)
Q.display()
print("Queue Length:", Q.length())
print("Deleted Item:",Q.dequeue())
print("Deleted Item:",Q.dequeue())
Q.display()
print("Queue Length:", Q.length())S.pop()
```

Output #1:

```
Queue Empty?: True
Queue items are
10
20
30
Queue Length: 4
Deleted Item: 10
Deleted Item: 20
Queue items are
30
40
Queue Length: 2
```

Program #2: Python Program to implement Priority Queue.

```
class PQItem:
    def __init__(self,value,priority):
        self.value=value
        self.priority=priority

class PriorityQueue:
    def __init__(self):
        self.items = list()

    def enqueue(self,value,priority):
        item=PQItem(value,priority)
        self.items.append(item)

    def  dequeue(self):
        if self.isEmpty() == True:
            print("Queue is empty, cannot remove")
        else:
            highest = self.items[0].priority
```

```
            index = 0
            for i in range(1,len(self.items)):
                if self.items[i].priority < highest:
                    highest = self.items[i].priority
                    index = i
            item = self.items.pop(index)
            return item.value

    def display(self):
        if self.isEmpty() == True:
            print("Queue is empty")
        else:
            print("Queue items are")
            print("-----------------")
            print("Value \t Priority")
            print("-----------------")
            for i in self.items:
                print(i.value,"\t",i.priority)
                print("-----------------")

    def isEmpty(self):
        if len(self.items) == 0:
            return True
        else:
            return False

    def length(self):
        return len(self.items)

Q = PriorityQueue()
print("Queue Empty?:",Q.isEmpty())
Q.enqueue(10,2)
Q.enqueue(20,0)
Q.enqueue(30,1)
Q.enqueue(40,3)
Q.display()
print("Queue Length:", Q.length())
print("Deleted Item:",Q.dequeue())
print("Deleted Item:",Q.dequeue())
print("Deleted Item:",Q.dequeue())
```

```
print("Deleted Item:",Q.dequeue())
print("Deleted Item:",Q.dequeue())
Q.display()
print("Queue Length:", Q.length())
```

Output #1:

```
Queue Empty?: True
Queue items are
------------------
Value  Priority
------------------
10       2
------------------
20       0
------------------
30       1
------------------
40       3
------------------
Queue Length: 4
Deleted Item: 20
Deleted Item: 30
Deleted Item: 10
Deleted Item: 40
Queue is empty, cannot remove
Deleted Item: None
Queue is empty
Queue Length: 0
```

Activity #10

1. **Hand execute the following code segment and show the contents of the resulting Queue.**

```
values = Queue()
for i in range( 16 ) :
if i % 3 == 0 :
values.enqueue( i )
```

2. **Hand execute the following code segment and show the contents of the resulting Queue.**

```
values = Queue()
for i in range( 16 ) :
if i % 3 == 0 :
values.enqueue( i )
elif i % 4 == 0 :
values.dequeue()
```