Practical No 6

Aim : Write A program To Implement Queue

1. Queue

```
class Queue:
    def __init__(self):
        self.queue = []

    def enque(self,data):
        self.queue.append(data)
        return True

    def deque(self):
        self.queue.pop(0)
        return True

    def isEmpty(self):
        return len(self.queue)==0

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

    def print(self):
        print(self.queue)

queue = Queue()

# Adding Elements
queue.enque(1)
queue.enque(2)
queue.enque(3)
queue.enque(4)
queue.enque(5)
queue.print()

# Removing Elements
queue.deque()
queue.deque()
queue.deque()
queue.print()

# Checking If Queue Is Empty And Length Of Queue
print(queue.isEmpty())
print(queue.length())
```

```
[1, 2, 3, 4, 5]
[4, 5]
False
2
```

2. Deque

```
from collections import deque

d = deque()

d.append(0)
d.append(2)
d.append(4)
d.append(6)
d.append(8)
d.append(10)

print("Before POP : ",d)
d.pop
print("After POP : ",d)

a = d.popleft()
print(d)
print(a)
```

```
Before POP :  deque([0, 2, 4, 6, 8, 10])
After POP :  deque([0, 2, 4, 6, 8, 10])
deque([2, 4, 6, 8, 10])
0
```

3. Priority Queue

```
import heapq

q = []
heapq.heappush(q,(2,"Write A Book"))
heapq.heappush(q,(1,"Lunch & Break"))
heapq.heappush(q,(3,"Relax & Sleep"))

while q:
    next_item = heapq.heappop(q)
    print(next_item)
```

```
(1, 'Lunch & Break')
(2, 'Write A Book')
(3, 'Relax & Sleep')
```