Creating Queue using array

In [1]:

```
class QueueUsingArray:

    def __init__(self):
        self.__arr = []
        self.__count = 0
        self.__front = 0

    def enqueue(self,data):
        self.__arr.append(data)
        self.__count += 1

    def dequeue(self):
        if self.__count == 0:
            return -1

        element = self.__arr[self.__front]
        self.__front += 1
        self.__count -= 1
        return element

    def front(self):
        if self.__count == 0:
            return -1
        return self.__arr[self.__front]

    def size(self):
        return self.__count

    def isEmpty(self):
        return self.size() == 0


q = QueueUsingArray()
q.enqueue(1)
q.enqueue(2)
q.enqueue(3)
q.enqueue(4)

while (q.isEmpty() is False):
    print(q.front())
    q.dequeue()

print(q.dequeue())
```

```
1
2
3
4
-1
```

In [2]:

```
1  # Creating Queue using Linked List
```

In [16]:

```
class Node:

    def __init__(self,data):
        self.data = data
        self.next = next

class Queue:

    def __init__(self):
        self.__head = None
        self.__tail = None
        self.__count = 0

    def enqueue(self,element):

        newNode = Node(element)
        if self.__head is None:
            self.__head = newNode
        else:
            self.__tail.next = newNode

        self.__tail = newNode
        self.__count = self.__count + 1

    def dequeue(self):
        if self.__head is None:
            print("Hey! Queue  is Empty")
            return
        data = self.__head.data
        self.__head = self.__head.next
        self.__count = self.__count - 1
        return data

    def isEmpty(self):
        return self.size() == 0

    def size(self):
        return self.__count

    def front(self):
        if self.__head is None:
            print("Hey! Queue is Empty")
            return
        data = self.__head.data
        return data


q = Queue()
q.enqueue(1)
q.enqueue(5)
q.enqueue(3)
q.enqueue(4)

while (q.isEmpty() is False):
    print(q.dequeue())

print(q.isEmpty())
```

```
#print(q.front())
```

```
1
5
3
4
True
```

Inbuilt stack and queue

In [19]:

```
import queue

"""s = [1,2,3]
s.append(4)
s.append(5)

print(s.pop())
print(s.pop())"""

## Inbuilt Queue
q = queue.Queue()
q.put(1)
q.put(2)
q.put(3)
q.put(4)

while not q.empty():
    print(q.get())

## Inbuilt Stack
q = queue.LifoQueue()
q.put(1)
q.put(2)
q.put(3)

while not q.empty():
    print(q.get())
```

```
1
2
3
4
3
2
1
```

Queue using two stacks

In [2]:

```
class QueueUsingTwoStacks:

    def __init__(self):
        self.__s1 = []
        self.__s2 = []

    def enqueue(self,data):
        # O(n)
        while (len(self.__s1) != 0):
            self.__s2.append(self.__s1.pop())

        self.__s1.append(data)

        while (len(self.__s2) != 0):
            self.__s1.append(self.__s2.pop())

        return

    def dequeue(self):
        # O(1)
        if (len(self.__s1) == 0):
            return -1
        return self.__s1.pop()

    def front(self):
        if (len(self.__s1) == 0):
            return -1
        return self.__s1[-1]

    def size(self):
        return len(self.__s1)

    def isEmpty(self):
        return self.size() == 0


q = QueueUsingTwoStacks()
q.enqueue(1)
q.enqueue(2)
q.enqueue(3)
q.enqueue(4)

while (q.isEmpty() is False):
    print(q.front())
    q.dequeue()
```

```
1
2
3
4
```

Stack using two queue

In [4]:

```
class StackUsingTwoQueue:

    def __init__(self):
        self.__q1 = []
        self.__q2 = []

    def push(self,element):
        while (len(self.__q2) != 1):
            self.__q2.append(self.__q1.front())

        self.__q1.append(data)


    def pop(self):
        if (len(self.__q1) == 0):
            return -1
        return self.__q2.dequeue()

    def top(self):
        if (len(self.__q1) == 0):
            return -1
        return self.__q2[0]

    def size(self):
        return len(__q1)

    def isEmpty(self):
        return self.size() == 0


s = StackUsingTwoQueue()
s.push(1)
s.push(2)
s.push(3)
s.push(4)


while (s.isEmpty() is False):
    print(s.top())
    s.pop()
```

```
--------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
~\AppData\Local\Temp/ipykernel_3808/2785522368.py in <module>
     30
     31 s = StackUsingTwoQueue()
---> 32 s.push(1)
     33 s.push(2)
     34 s.push(3)

~\AppData\Local\Temp/ipykernel_3808/2785522368.py in push(self, element)
      7     def push(self,element):
      8         while (len(self.__q2) != 1):
----> 9             self.__q2.append(self.__q1.front())
     10
     11         self.__q1.append(data)

AttributeError: 'list' object has no attribute 'front'
```

In [ ]:

```
1
```