

Creating Queue using array

In [1]:

```
1 class QueueUsingArray:
2
3     def __init__(self):
4         self.__arr = []
5         self.__count = 0
6         self.__front = 0
7
8     def enqueue(self,data):
9         self.__arr.append(data)
10        self.__count += 1
11
12    def dequeue(self):
13        if self.__count == 0:
14            return -1
15
16        element = self.__arr[self.__front]
17        self.__front += 1
18        self.__count -= 1
19        return element
20
21    def front(self):
22        if self.__count == 0:
23            return -1
24        return self.__arr[self.__front]
25
26    def size(self):
27        return self.__count
28
29    def isEmpty(self):
30        return self.size() == 0
31
32
33 q = QueueUsingArray()
34 q.enqueue(1)
35 q.enqueue(2)
36 q.enqueue(3)
37 q.enqueue(4)
38
39 while (q.isEmpty() is False):
40     print(q.front())
41     q.dequeue()
42
43 print(q.dequeue())
```

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

In [2]:

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


In [16]:

```
1 class Node:
2
3     def __init__(self,data):
4         self.data = data
5         self.next = next
6
7 class Queue:
8
9     def __init__(self):
10         self.__head = None
11         self.__tail = None
12         self.__count = 0
13
14     def enqueue(self,element):
15
16         newNode = Node(element)
17         if self.__head is None:
18             self.__head = newNode
19         else:
20             self.__tail.next = newNode
21
22         self.__tail = newNode
23         self.__count = self.__count + 1
24
25     def dequeue(self):
26         if self.__head is None:
27             print("Hey! Queue is Empty")
28             return
29         data = self.__head.data
30         self.__head = self.__head.next
31         self.__count = self.__count - 1
32         return data
33
34     def isEmpty(self):
35         return self.size() == 0
36
37     def size(self):
38         return self.__count
39
40     def front(self):
41         if self.__head is None:
42             print("Hey! Queue is Empty")
43             return
44         data = self.__head.data
45         return data
46
47
48 q = Queue()
49 q.enqueue(1)
50 q.enqueue(5)
51 q.enqueue(3)
52 q.enqueue(4)
53
54 while (q.isEmpty() is False):
55     print(q.dequeue())
56
57 print(q.isEmpty())
```

```
58 #print(q.front())
1
5
3
4
True
```

Inbuilt stack and queue

In [19]:

```
1 import queue
2
3 """s = [1,2,3]
4 s.append(4)
5 s.append(5)
6
7 print(s.pop())
8 print(s.pop())"""
9
10 ## Inbuilt Queue
11 q = queue.Queue()
12 q.put(1)
13 q.put(2)
14 q.put(3)
15 q.put(4)
16
17 while not q.empty():
18     print(q.get())
19
20 ## Inbuilt Stack
21 q = queue.LifoQueue()
22 q.put(1)
23 q.put(2)
24 q.put(3)
25
26 while not q.empty():
27     print(q.get())
```

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

Queue using two stacks

In [2]:

```
1 class QueueUsingTwoStacks:
2
3     def __init__(self):
4         self.__s1 = []
5         self.__s2 = []
6
7     def enqueue(self, data):
8         # O(n)
9         while (len(self.__s1) != 0):
10             self.__s2.append(self.__s1.pop())
11
12         self.__s1.append(data)
13
14         while (len(self.__s2) != 0):
15             self.__s1.append(self.__s2.pop())
16
17         return
18
19     def dequeue(self):
20         # O(1)
21         if (len(self.__s1) == 0):
22             return -1
23         return self.__s1.pop()
24
25     def front(self):
26         if (len(self.__s1) == 0):
27             return -1
28         return self.__s1[-1]
29
30     def size(self):
31         return len(self.__s1)
32
33     def isEmpty(self):
34         return self.size() == 0
35
36
37 q = QueueUsingTwoStacks()
38 q.enqueue(1)
39 q.enqueue(2)
40 q.enqueue(3)
41 q.enqueue(4)
42
43 while (q.isEmpty() is False):
44     print(q.front())
45     q.dequeue()
```

1
2
3
4

Stack using two queue

In [4]:

```
1 class StackUsingTwoQueue:
2
3     def __init__(self):
4         self.__q1 = []
5         self.__q2 = []
6
7     def push(self,element):
8         while (len(self.__q2) != 1):
9             self.__q2.append(self.__q1.front())
10
11         self.__q1.append(data)
12
13
14     def pop(self):
15         if (len(self.__q1) == 0):
16             return -1
17         return self.__q2.dequeue()
18
19     def top(self):
20         if (len(self.__q1) == 0):
21             return -1
22         return self.__q2[0]
23
24     def size(self):
25         return len(__q1)
26
27     def isEmpty(self):
28         return self.size() == 0
29
30
31 s = StackUsingTwoQueue()
32 s.push(1)
33 s.push(2)
34 s.push(3)
35 s.push(4)
36
37
38 while (s.isEmpty() is False):
39     print(s.top())
40     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	
---	--