Workshop #5: Deques

GitHub: abhijit-baruah

Implement the following for a Deque data structure:

- __len__
- __bool__
- __repr__ (“unambiguous representation of an object”)
- __str__
- __contains__

In [7]:

```
class Deque:

    def __init__(self):
        self.items = []

    def is_empty(self):
        return self.items == []

    def add_front(self, item):
        self.items.append(item)

    def add_rear(self, item):
        self.items.insert(0, item)

    def remove_front(self):
        return self.items.pop()

    def remove_rear(self):
        return self.items.pop(0)

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

    def __len__(self):
        return self.size()

    def __bool__(self):
        return not self.is_empty() # or return self.items != []

    def __repr__(self):
        return "Deque()"

    def __str__(self):
        q = "<Deque: {0}>".format(self.items)
        return q

    def __contains__(self, item):
        return item in self.items
```

In [8]:

```
d = Deque()
```

In [9]:

```
# Check addition of item in front of Deque
d.add_front(2)
d.add_front(1)
print(d) # also checks __str__
```

```
<Deque: [2, 1]>
```

In [10]:

```
# Check addition of items to the rear of the Deque
d.add_rear(3)
d.add_rear(4)
d.add_rear(5)
print(d)
```

```
<Deque: [5, 4, 3, 2, 1]>
```

In [11]:

```
#check _repr_
d
```

Out[11]:

```
Deque()
```

In [12]:

```
#check _len_
len(d)
```

Out[12]:

```
5
```

In [14]:

```
# check bool
d_empty = Deque()
print(bool(d_empty))
print(f'bool of Deque:d is {bool(d)}')
```

```
False
bool of Deque:d is True
```

In [15]:

```
# check __contains__
print(f'Is 6 in Deque:D? {6 in d}')
```

```
Is 6 in Deque:D? False
```

In [17]:

```
# finally check remove
d.remove_front()
d.remove_rear()
print(f'The deque after removing an item each from front and rear is: {d}')
```

```
The deque after removing an item each from front and rear is: <Deque: [4, 3,
2]>
```

In [ ]:

```

```