4.17. Implementing a Deque in Python

As we have done in previous sections, we will create a new class for the implementation of the abstract data type deque. Again, the Python list will provide a very nice set of methods upon which to build the details of the deque. Our implementation (Listing 1) will assume that the rear of the deque is at position 0 in the list.

Listing 1

```
 1 class Deque:
 2     def __init__(self):
 3         self.items = []
 4
 5     def isEmpty(self):
 6         return self.items == []
 7
 8     def addFront(self, item):
 9         self.items.append(item)
10
11     def addRear(self, item):
12         self.items.insert(0,item)
13
14     def removeFront(self):
15         return self.items.pop()
16
17     def removeRear(self):
18         return self.items.pop(0)
19
20     def size(self):
21         return len(self.items)
```

In `removeFront` we use the `pop` method to remove the last element from the list. However, in `removeRear`, the `pop(0)` method must remove the first element of the list. Likewise, we need to use the `insert` method (line 12) in `addRear` since the `append` method assumes the addition of a new element to the end of the list.

CodeLens 1 shows the `Deque` class in action as we perform the sequence of operations from Table 1 (TheDequeAbstractDataType.html#tbl-dequeoperations).

(TheDequeAbstractDataType.html)

(PalindromeChecl

```
class Deque:
    def __init__(self):
        self.items = []

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

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

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

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

    def removeRear(self):
        return self.items.pop(0)
```

line that just executed

next line to execute

< Prev Next >

Step 1 of 45

Python Tutor (http://pythontutor.com/) by Philip Guo (http://pgbovine.net/)

Customize visualization (NEW!)

Print output (drag lower right corner to resize)

Frames Objects

Activity: CodeLens Example Deque Operations (deqtest)

You can see many similarities to Python code already described for stacks and queues. You are also likely to observe that in this implementation adding and removing items from the front is O(1) whereas adding and removing from the rear is O(n). This is to be expected given the common operations that appear for adding and removing items. Again, the important thing is to be certain that we know where the front and rear are assigned in the implementation.

(TheDequeAbstractDataType.html) (PalindromeChecl

You have attempted 1 of 2 activities on this page user not logged in

(TheDequeAbstractDataType.html)

(PalindromeCheck