4.12. Implementing a Queue in Python

It is again appropriate to create a new class for the implementation of the abstract data type queue. As before, we will use the power and simplicity of the list collection to build the internal representation of the queue.

We need to decide which end of the list to use as the rear and which to use as the front. The implementation shown in Listing 1 assumes that the rear is at position 0 in the list. This allows us to use the `insert` function on lists to add new elements to the rear of the queue. The `pop` operation can be used to remove the front element (the last element of the list). Recall that this also means that enqueue will be O(n) and dequeue will be O(1).

Listing 1

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

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

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

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

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

CodeLens 1 shows the `Queue` class in action as we perform the sequence of operations from Table 1 (TheQueueAbstractDataType.html#tbl-queueoperations).

(TheQueueAbstractDataType.html) (SimulationHotPo

```
1  class Queue:
2      def __init__(self):
3          self.items = []
4
5      def isEmpty(self):
6          return self.items == []
7
8      def enqueue(self, item):
9          self.items.insert(0,item)
10
11     def dequeue(self):
12         return self.items.pop()
13
14     def size(self):
15         return len(self.items)
16
17 q=Queue()
18
```

line that just executed

next line to execute

< Prev Next >

Step 1 of 21

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 Queue Operations (ququeuetest)

Further manipulation of this queue would give the following results:

(TheQueueAbstractDataType.html) (SimulationHotPo

```
>>> q.size()
3
>>> q.isEmpty()
False
>>> q.enqueue(8.4)
>>> q.dequeue()
4
>>> q.dequeue()
'dog'
>>> q.size()
2
```

Self Check

Q-2: Suppose you have the following series of queue operations.

```
q = Queue()
q.enqueue('hello')
q.enqueue('dog')
q.enqueue(3)
q.dequeue()
```

What items are left on the queue?

- **A. 'hello', 'dog'**
- **B. 'dog', 3**
- **C. 'hello', 3**
- **D. 'hello', 'dog', 3**

Check Me Compare me

Activity: 4.12.2 Multiple Choice (queue_1)

You have attempted 1 of 3 activities on this page

user not logged in

(TheQueueAbstractDataType.html) (SimulationHotPo