edX (https://www.edx.org)

MITx: 6.00.1x Introduction to Computer Science and Programming Using Python

sandipan_dey (/dashboard) ▾

Courseware (/courses/MITx/6.00.1_4x/3T2014/courseware) Updates & News (/courses/MITx/6.00.1_4x/3T2014/info)

Calendar (/courses/MITx/6.00.1_4x/3T2014/89309559b0414f6d8cbef9e48ca19f4b/) Wiki (/courses/MITx/6.00.1_4x/3T2014/course_wiki)

Discussion (/courses/MITx/6.00.1_4x/3T2014/discussion/forum) Progress (/courses/MITx/6.00.1_4x/3T2014/progress)

L11 PROBLEM 6 (5/5 points)

For this exercise, you will be coding your very first class, a `Queue` class. Queues are a fundamental computer science data structure. A queue is basically like a line at Disneyland - you can add elements to a queue, and they maintain a specific order. When you want to get something off the end of a queue, you get the item that has been in there the longest (this is known as 'first-in-first-out', or FIFO). You can read up on queues at Wikipedia if you'd like to learn more.

In your `Queue` class, you will need three methods:

1. `__init__`: initialize your `Queue` (think: how will you store the queue's elements? You'll need to initialize an appropriate *object attribute* in this method)
2. `insert`: inserts one element in your `Queue`
3. `remove`: removes (or 'pops') one element from your `Queue` and returns it. If the queue is empty, raises a ValueError.

When you're done, you should test your implementation. Your results should look something like this:

```
>>> queue = Queue()
>>> queue.insert(5)
>>> queue.insert(6)
>>> queue.remove()
5
>>> queue.insert(7)
>>> queue.remove()
6
>>> queue.remove()
7
>>> queue.remove()
Traceback (most recent call last):
  File "<stdin>", line 26, in <module>
  File "queue.py", line 15, in remove
    raise ValueError()
ValueError
```

Be sure to handle that last case correctly - when popping from an empty `Queue`, throw the appropriate error.

```
class Queue(object):
    """ Queues are a fundamental computer science data structure.
    A queue is basically like a line at Disneyland - you can add
    elements to a queue, and they maintain a specific order.
    When you want to get something off the end of a queue,
    you get the item that has been in there the longest (this is
    known as 'first-in-first-out', or FIFO). """

    def __init__(self):
        """Create an empty queue of integers"""
        self.vals = []

    def insert(self, e):
        """Assumes e is an integer and inserts e into self"""
        if not e in self.vals:
            self.vals.append(e)
```

Correct

```
class Queue(object):

    def __init__(self):
        "Initializes 1 attribute: a list to keep track of the queue's elements"
        self.qlist = []

    def insert(self, element):
        "Adds an element to the attribute list using append"
        self.qlist.append(element)

    def remove(self):
        "Removes an element from the attribute list"
        # Check if the list is empty; if so then raise a ValueError
        if self.qlist == []:
            raise ValueError()
        else:
            # Otherwise we want to remove the first element from the list
            # and return that to the user.
            element = self.qlist[0]
            self.qlist.remove(element)
            return element
            # Could also do this in 1 line:
            # return self.qlist.pop(0)
```

Help

Test results

See full output

CORRECT

See full output

Check Hide Answer

Show Discussion New Post

EdX is a non-profit created by founding partners Harvard and MIT whose mission is to bring the best of higher education to students of all ages anywhere in the world, wherever there is Internet access. EdX's free online MOOCs are interactive and subjects include computer science, public health, and artificial intelligence.