Problem # 225 : Implement Stack using Queues

<https://leetcode.com/problems/implement-stack-using-queues/discuss/807894/Simple-Python-3-Solution->

Simple Python 3 Solution using queues -- Runtime 24ms beats 93.67%

1. Initialize myStack as deque object and with its size set to 0.
2. The push method appends the new integer x to myStack and increments the size.
3. For pop, we need to remove the last element. If myStack is empty, then print an error message and return. If myStack is not empty, popleft the stack except the last element, and save the elements in a tempQ which is a deque object as well. We track this using count which is initially the size of myStack.  
   Now there is only one element left in myStack. Get the top\_element from myStack by doing popleft. Note that myStack is now empty. Copy back the contents of tempQ into myStack. Decrement the size by 1.
4. For top, it is almost the same as pop except, we have to put back top\_element after coying the contents of tempQ into myStack. We are restoring myStack the way it was and so no change is made to the size.
5. For empty, check whether size is not zero.

from collections import deque

class MyStack:

def \_\_init\_\_(self):

"""

Initialize your data structure here.

"""

self.myStack = deque()

self.size = 0

def push(self, x: int) -> None:

"""

Push element x onto stack.

"""

self.myStack.append(x)

self.size += 1

def pop(self) -> int:

"""

Removes the element on top of the stack and returns that element.

"""

if not self.myStack:

print("Error! Trying to pop an empty stack!")

return

tempQ = deque()

count = self.size

while count > 1:

tempQ.append(self.myStack.popleft())

count -= 1

#Now myStack has only one element left and to get it do popleft

top\_element = self.myStack.popleft()

#Put back all the elements in the tempQ into the empty myStack

self.myStack = tempQ

self.size -= 1

return(top\_element)

def top(self) -> int:

"""

Get the top element.

"""

if not self.myStack:

print("Error! stack is empty!")

return

tempQ = deque()

count = self.size # For top, size does not change.

while count > 1:

tempQ.append(self.myStack.popleft())

count -= 1

#Now myStack has only one element left which was the topmost one

top\_element = self.myStack.popleft()

#Put back all the elements in the tempQ into the empty myStack

self.myStack = tempQ

# Put the top\_element in myStack since we want to restore it to its original state

self.myStack.append(top\_element)

return(top\_element)

def empty(self) -> bool:

"""

Returns whether the stack is empty.

"""

return(not self.size)

# Your MyStack object will be instantiated and called as such:

# obj = MyStack()

# obj.push(x)

# param\_2 = obj.pop()

# param\_3 = obj.top()

# param\_4 = obj.empty()