Problem #232 : Implement Queue using Stacks

<https://leetcode.com/problems/implement-queue-using-stacks/discuss/807029/Simple-Python-3-Solution-for-implementing-a-queue-using-stacks>

**Simple Python 3 Solution for implementing a queue using stacks -- Runtime 24ms, faster than 93.54%**

1. I have myStack which is a list containing the queue elements and it is my main stack. I am using tempStack as an additional stack for helping me with queue operations such as pop and peek.
2. I initialize both myStack and tempStack with empty lists in **init**() method.
3. For push operation, I just append the element x to myStack.
4. For pop operation, I need to get to the front of the queue which is element at index 0. In a stack we can only remove elements from the top of the stack i.e. from n - 1 where n = length of the stack. Here, I get the elements out of myStack and put them in the tempStack. Note that the tempStack has all the elements in the reverse order. So the top of the tempStack now contains the element wich was in the front of the queue that we want. To get the top\_element, pop the tempStack and save it. Now put the rest of the elements in the tempStack back into myStack. Finally, we return the top\_element.
5. For peek operation, it is almost the same as the pop operation, except after saving the top\_element, we have to push it back into myStack. In effect, myStack does not change at all. We get the top\_element value and put it back again. Finally, we return the top\_element.

class MyQueue:

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

"""

Initialize your data structure here.

"""

self.myStack, self.tempStack = [], []

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

"""

Push element x to the back of queue.

"""

self.myStack.append(x)

def pop(self) -> int:

"""

Removes the element from in front of queue and returns that element.

"""

if not self.tempStack:

while self.myStack:

self.tempStack.append(self.myStack.pop())

top\_element = self.tempStack.pop()

while self.tempStack:

self.myStack.append(self.tempStack.pop())

return(top\_element)

def peek(self) -> int:

"""

Get the front element.

"""

if not self.tempStack:

while self.myStack:

self.tempStack.append(self.myStack.pop())

top\_element = self.tempStack.pop()

self.myStack.append(top\_element)

while self.tempStack:

self.myStack.append(self.tempStack.pop())

return(top\_element)

def empty(self) -> bool:

"""

Returns whether the queue is empty.

"""

return(not self.myStack)

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

# obj = MyQueue()

# obj.push(x)

# param\_2 = obj.pop()

# param\_3 = obj.peek()

# param\_4 = obj.empty()