Problem # 933: Number of recent calls

<https://leetcode.com/problems/number-of-recent-calls/>

My Solution:

This solution was accepted.

*# NumberOfRecentCalls.py***class** RecentCounter:  
 **def** \_\_init\_\_(self):  
 self.requests = []  
 self.size = 0  
 self.count = 0  
  
 **def** ping(self, t: int) -> int:  
 self.requests.append(t)  
 self.size += 1  
 **if** self.requests[0] >= t - 3000:  
 **return** (self.size)  
  
 **for** i **in** range(self.count, self.size):  
 **if** self.requests[i] < t - 3000:  
 self.count += 1  
 **else**:  
 **break  
  
 return** (self.size - self.count)  
  
  
**def** main():  
 obj = RecentCounter()  
 **for** t **in** [1, 100, 3001, 3002]:  
 print(**"t = "**, t)  
 num\_requests = obj.ping(t)  
 print(**"num\_requests = "**, num\_requests)  
 print(**"--------------------------"**)  
  
**if** \_\_name\_\_==**'\_\_main\_\_'**:  
 main()

------------------------------------

I looked at the discussions and found a better solution using deque

<https://leetcode.com/problems/number-of-recent-calls/discuss/873333/Python-easy-solution-using-queue-explained>

My second solution based on deque – Runtime beats 99.45%

class RecentCounter:

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

self.requests = deque()

def ping(self, t: int) -> int:

self.requests.append(t)

while self.requests[0] < t - 3000:

self.requests.popleft()

return(len(self.requests))

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

# obj = RecentCounter()

# param\_1 = obj.ping(t)

--------------------------------------------------------------