Queues

Lab 13

What is a "queue"?

- A *queue* is an ordered collection of items where the **addition** of new items happens at **one end** (the rear or back of the queue) and the **removal** of existing items always takes place at the **other end** (the front of the queue).
  - i.e. add new elements to back, remove existing elements from front
- First-in, first-out (FIFO) property
  - The first item placed in the queue will be the first item removed
- Example:
  - A queue of people in a bank

Queue operations

- Data in queue is ordered by the insertion time: front element is the first added element, tail element is the most recently added element.
- Access to the queue is limited to inserting at the tail and removing from the front. In addition information about size is available and front-most element can be looked at (peek).

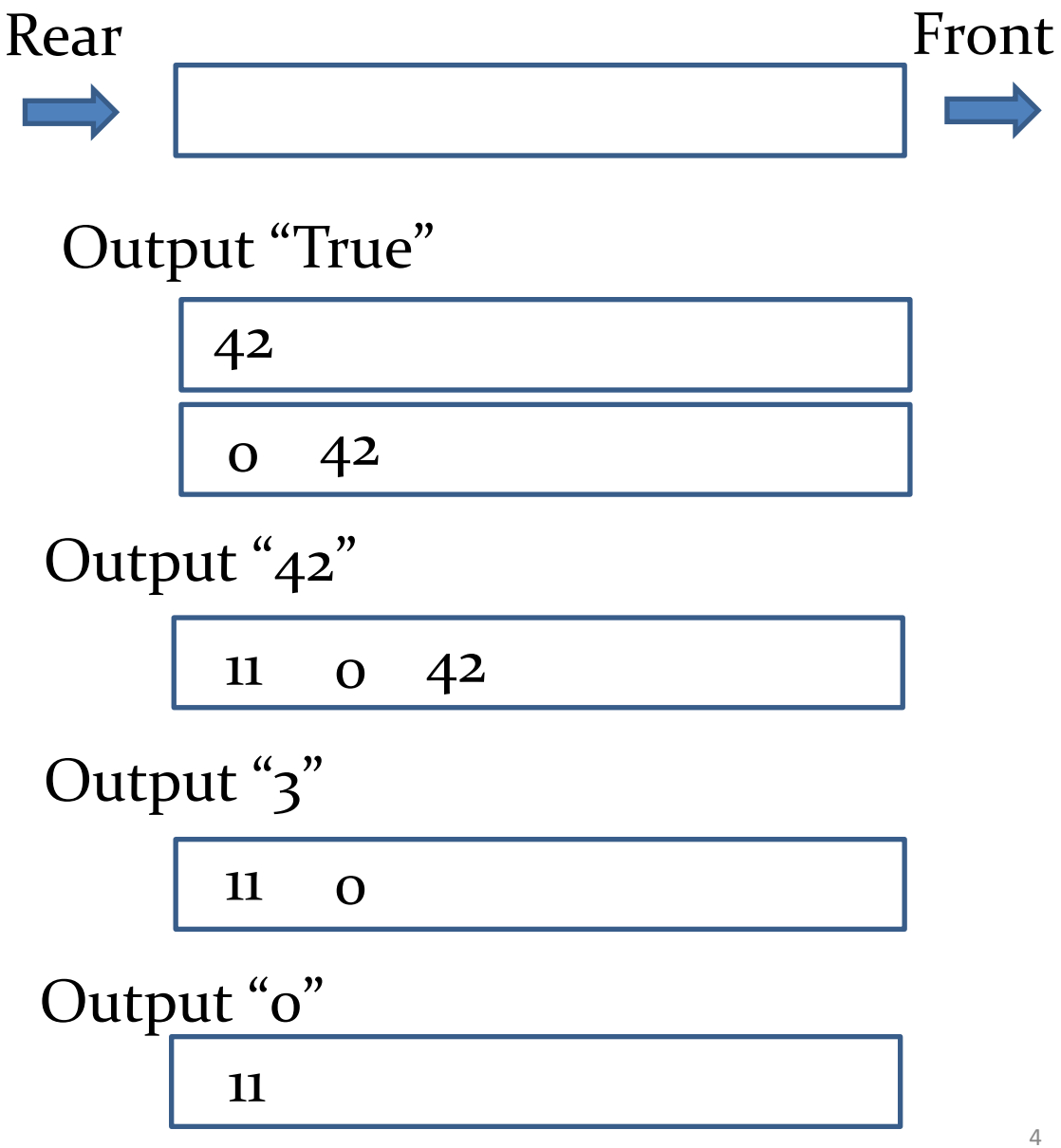
Operations:

- **create** a new empty queue (**Queue()**)
- determine whether a queue **is empty** (**is_empty()**)
- **add** a new item to the queue (**enqueue()**)
- **remove** the item added first to the queue (**dequeue()**)
- **look at** (but don't remove) the item added first (**peek()**)
- determine the **size** of a queue (how many elements) (**size()**)

The Example

- An example

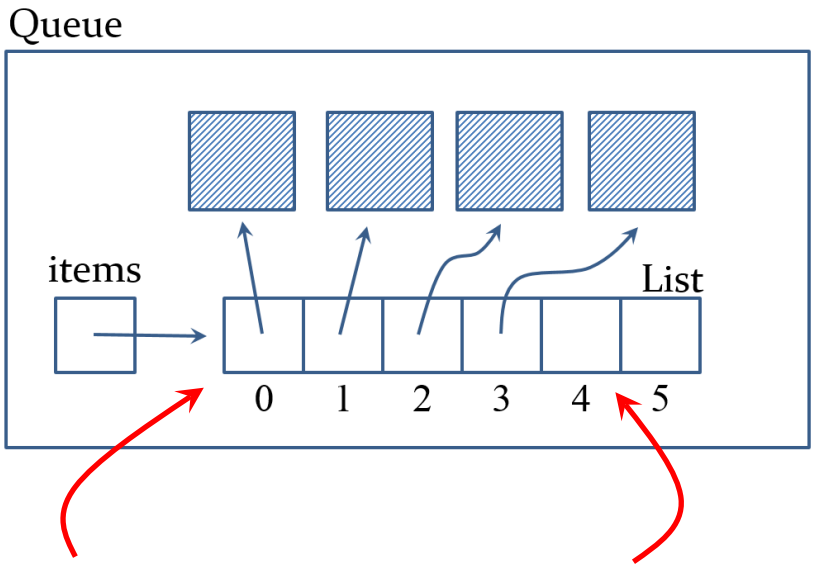
```
q = Queue()

print(q.is_empty())


q.enqueue(42)

q.enqueue(0)

print(q.peek())

q.enqueue(11)


print(q.size())

q.dequeue()


print(q.dequeue())
```



Queue Implementation

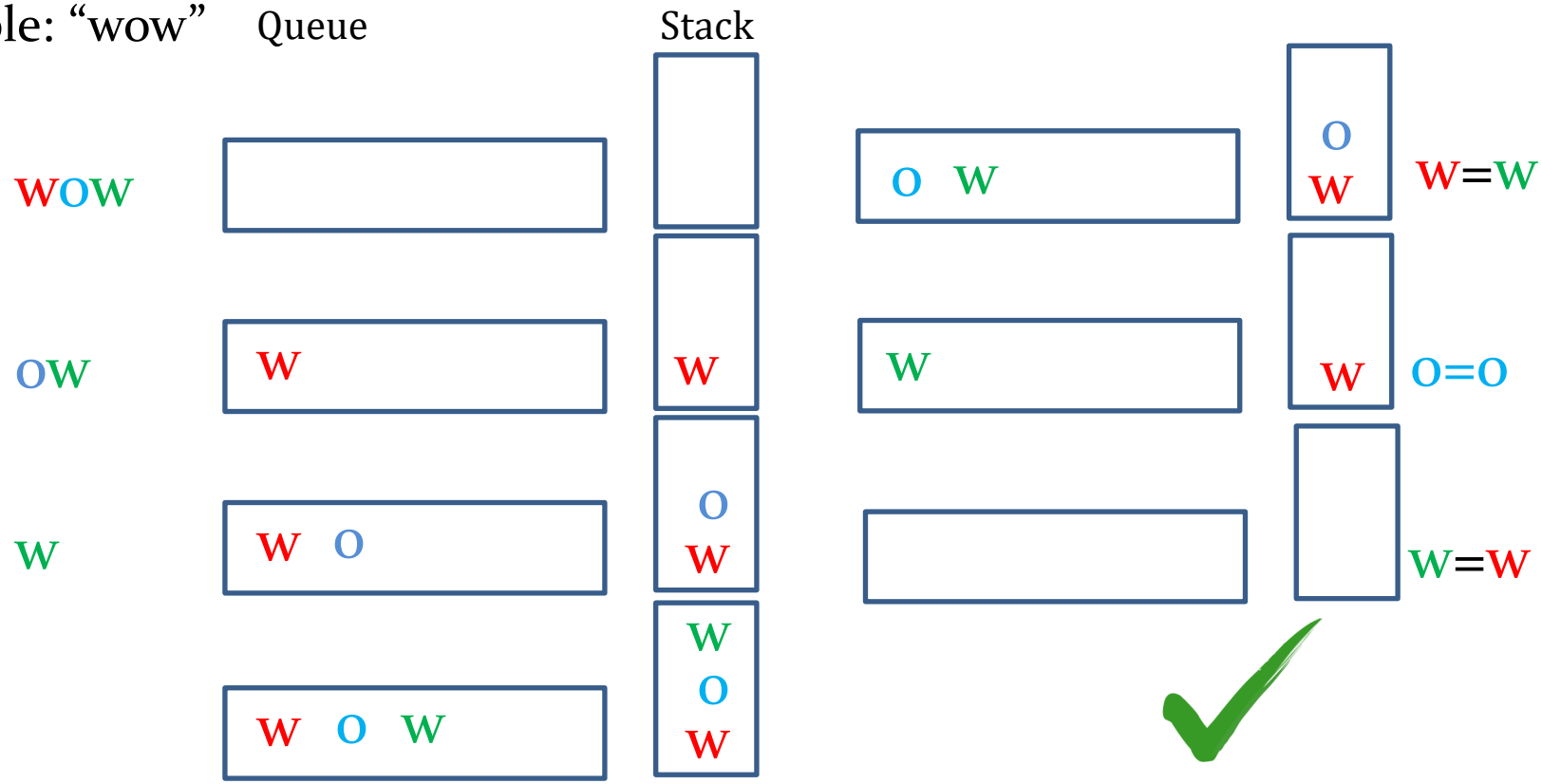
- We implement a queue using a Python list:
  - the addition of new items takes place at the beginning of the list (NOTE: Not efficient!)
  - The removal of existing items takes place at the end of the list

```
class Queue:

    def __init__(self):
        self.__items = []
    def is_empty(self):
        return self.__items == []
    def size(self):
        return len(self.__items)

    def enqueue(self, item):
        self.__items.insert(0,item)
    def dequeue(self):
        return self.__items.pop()
```



Application 2: Testing whether word is palindrome

- A word is a palindrome if it reads the same from the left and right, e.g.: “racecar”, “radar”, “wow”
- Idea: traverse word from left to right and put all characters into a stack and a queue. Then remove characters from stack and queue and check whether they are the same. If all pairs of characters are the same and both stack and queue are empty, word is a palindrome.
- Example: “wow”



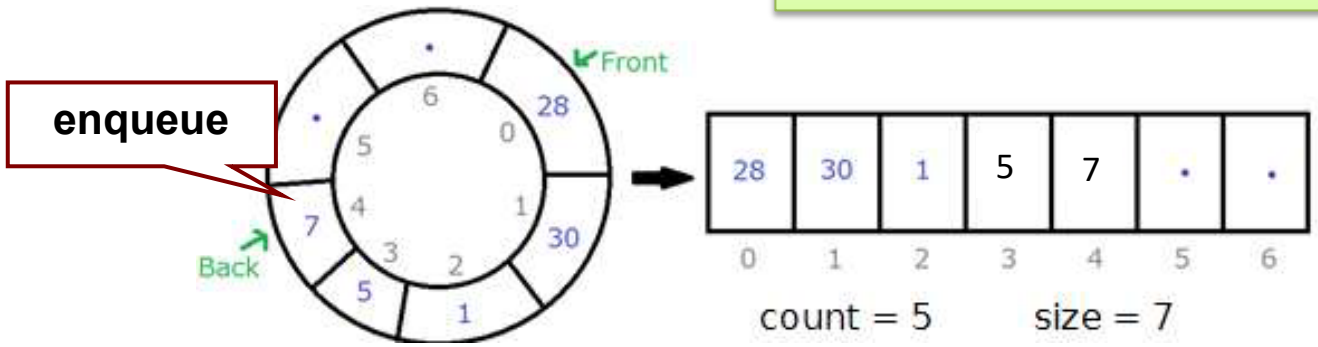
Circular Queue

- Uses a Python **list** data structure to store the items in the queue.
- The list has an initial capacity (all elements None)
- Keeps an index of the current **front** of the queue and of the current **back** of the queue.
    - set front to 0,
    - set back to MAX_QUEUE – 1,
    - set count to 0
- New items are **enqueued** at the **back** index position
- Items are **dequeued** at the **front** index position.
- A **count** of the queue items to detect queue-full and queue-empty conditions

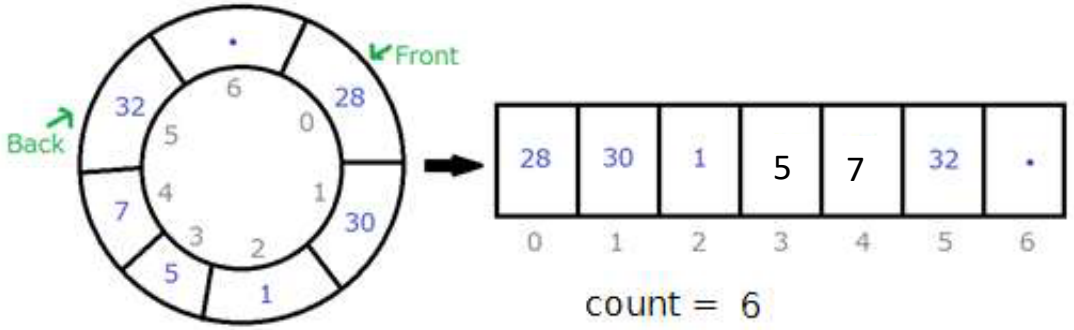
enqueue(32)

```
self.back = (self.back + 1) % self.MAX_QUEUE
self.items[self.back] = item
self.count += 1
```

- Before:



- After:

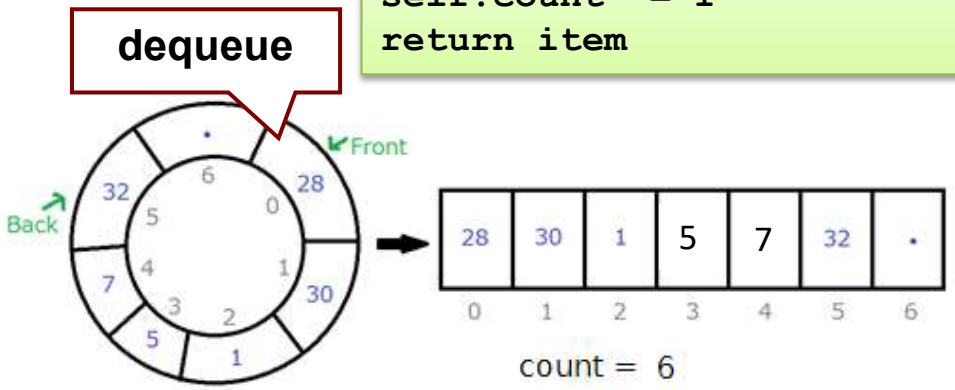


  - New item is inserted at the position following back
  - **back** is advanced by one position
  - **count** is incremented by 1

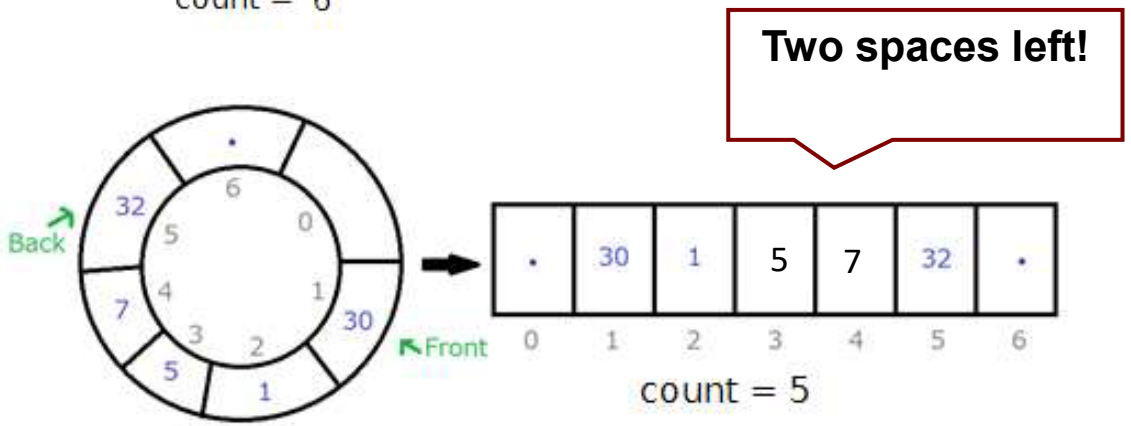
dequeue()

```
item = self.items[self.front]
self.front = (self.front + 1) % self.MAX_QUEUE
self.count -= 1
return item
```

- Before:



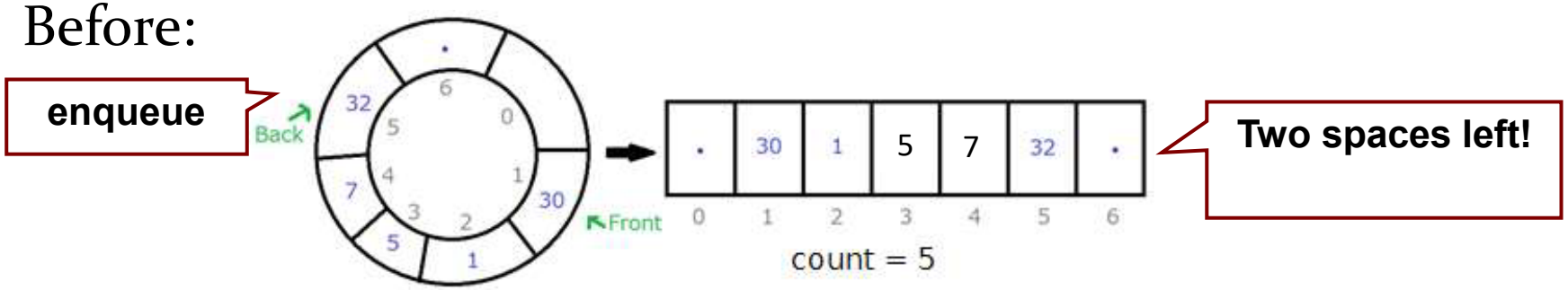
- After:



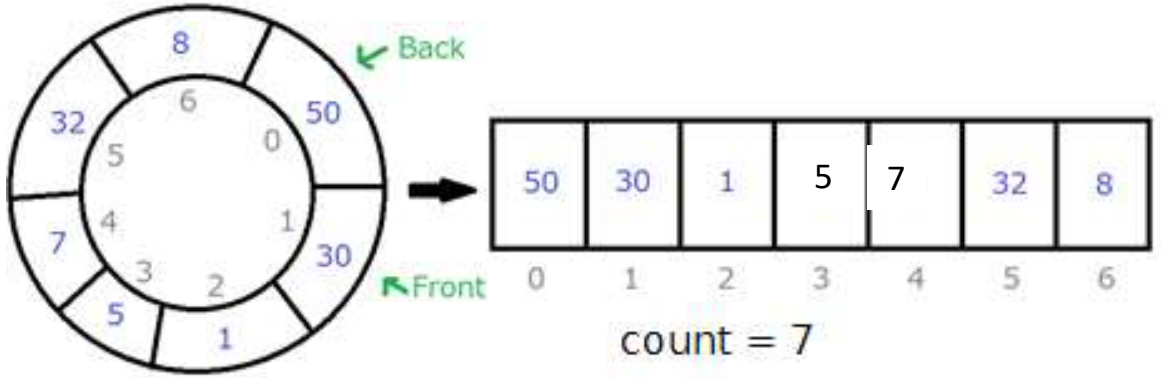
  - Value in front position is removed and returned
  - **front** is advanced by 1
  - **count** is decremented by 1

enqueue(8) and enqueue(50)

- Before:



- After:



  - After running the first enqueue, back = 6
  - After running the second enqueue, back = 0
    - as the "Back" is wrapped around the list

Coderunner

- Q1-4 – Making a Queue implementation
- Q5-6 – Using Queues
- Q7-10 – Making a circular Queue implementation

Coderunner Tips

- Q2 – Think about the order you need to display things for the __str__ method
- Q3 – List addition should be helpful here – Think about what order and orientation the lists need to be when you add them
- Q7 – The only parameters for the __init__ method should be self and capacity with a default value of 8 – other fields will need to be set within the method