UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 9

Queues

Submitted by:
Vasig, Yuan Hessed O.

Instructor:
Engr. Maria Rizette H. Sayo

October, 11, 2025

I. Objectives

Introduction

Another fundamental data structure is the queue. It is a close “the same” of the stack, as a queue is a collection of objects that are inserted and removed according to the first-in, first-out (FIFO) principle. That is, elements can be inserted at any time, but only the element that has been in the queue the longest can be next removed.

The Queue Abstract Data Type

Formally, the queue abstract data type defines a collection that keeps objects in a sequence, where element access and deletion are restricted to the first element in the queue, and element insertion is restricted to the back of the sequence. This restriction enforces the rule that items are inserted and deleted in a queue according to the first-in, first-out (FIFO) principle. The queue abstract data type (ADT) supports the following two fundamental methods for a queue Q:

Q.enqueue(e): Add element e to the back of queue Q.
Q.dequeue( ): Remove and return the first element from queue Q;
an error occurs if the queue is empty.

The queue ADT also includes the following supporting methods (with first being analogous to the stack’s top method):
Q.first(): Return a reference to the element at the front of queue Q, without removing it;
an error occurs if the queue is empty.
Q.is empty( ): Return True if queue Q does not contain any elements.
len(Q): Return the number of elements in queue Q; in Python, we implement this with the special method len .

This laboratory activity aims to implement the principles and techniques in:
- Writing Python program using Queues

Writing a Python program that will implement Queues operations

II. Methods

Instruction: Type the python codes below in your Colab. Reconstruct them by implementing Queues (FIFO) algorithm. Hint: You may use Array or Linked List

```
# Stack implementation in python

# Creating a stack
def create_stack():
    stack = []
    return stack
```

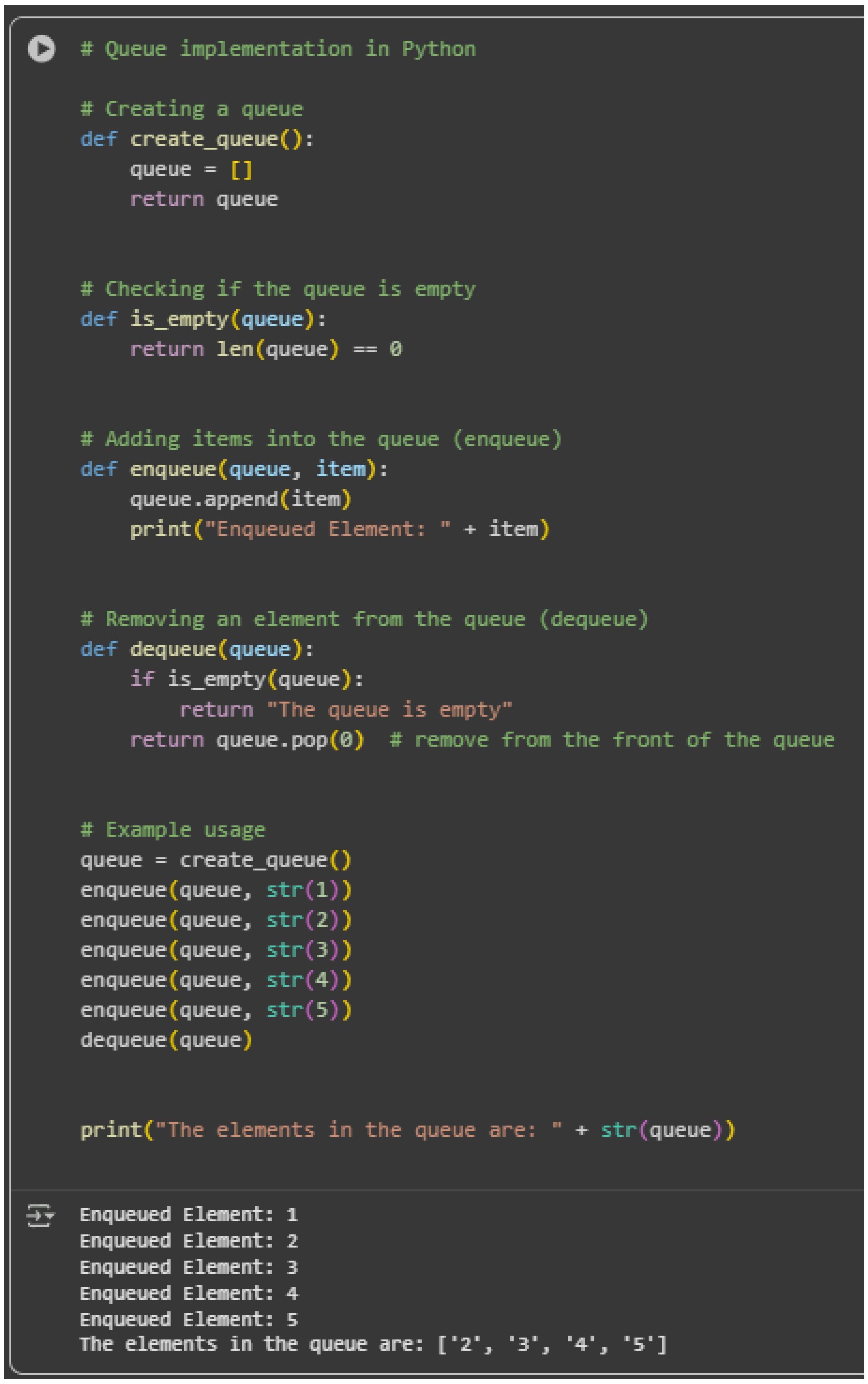
```
# Creating an empty stack
def is_empty(stack):
   return len(stack) == 0

# Adding items into the stack
def push(stack, item):
   stack.append(item)
   print("Pushed Element: " + item)

# Removing an element from the stack
def pop(stack):
   if (is_empty(stack)):
      return "The stack is empty"
   return stack.pop()

stack = create_stack()
push(stack, str(1))
push(stack, str(2))
push(stack, str(3))
push(stack, str(4))
push(stack, str(5))

print("The elements in the stack are:"+ str(stack))
```

Answer the following questions:

1 What is the main difference between the stack and queue implementations in terms of element removal?
2 What would happen if we try to dequeue from an empty queue, and how is this handled in the code?
3 If we modify the enqueue operation to add elements at the beginning instead of the end, how would that change the queue behavior?
4 What are the advantages and disadvantages of implementing a queue using linked lists versus arrays?
5 In real-world applications, what are some practical use cases where queues are preferred over stacks?

III. Results

Source Code from Stacks to Queues:



```
# Queue implementation in Python

# Creating a queue
def create_queue():
    queue = []
    return queue


# Checking if the queue is empty
def is_empty(queue):
    return len(queue) == 0


# Adding items into the queue (enqueue)
def enqueue(queue, item):
    queue.append(item)
    print("Enqueued Element: " + item)


# Removing an element from the queue (dequeue)
def dequeue(queue):
    if is_empty(queue):
        return "The queue is empty"
    return queue.pop(0)  # remove from the front of the queue


# Example usage
queue = create_queue()
enqueue(queue, str(1))
enqueue(queue, str(2))
enqueue(queue, str(3))
enqueue(queue, str(4))
enqueue(queue, str(5))
dequeue(queue)


print("The elements in the queue are: " + str(queue))
```

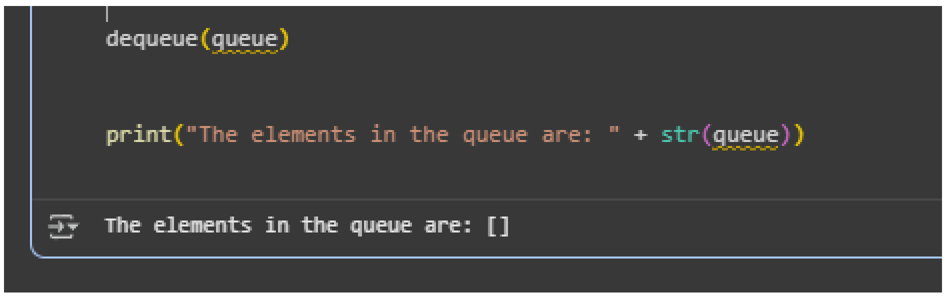
```
Enqueued Element: 1
Enqueued Element: 2
Enqueued Element: 3
Enqueued Element: 4
Enqueued Element: 5
The elements in the queue are: ['2', '3', '4', '5']
```

Figure 1: Source Code of Queues

Answer 1:

The main difference between the stack and queue implementation of removal is the functions. In stacks, if you want to remove an element, it's mainly called dequeue(queue) but inside the function is pop(0), removing the first element in the queue. Whilst, in stack pop() function is only used because it will remove the last element in the stack.

Answer 2:



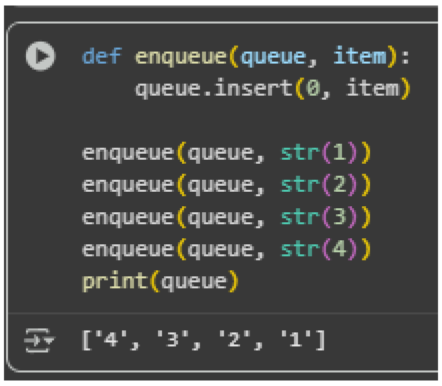
```
dequeue(queue)

print("The elements in the queue are: " + str(queue))
```

```
The elements in the queue are: []
```

Figure 2

Dequeueing an empty queue will result in outputting nothing in the code.

Answer 3:



```
def enqueue(queue, item):
    queue.insert(0, item)

enqueue(queue, str(1))
enqueue(queue, str(2))
enqueue(queue, str(3))
enqueue(queue, str(4))
print(queue)
```

```
['4', '3', '2', '1']
```

Figure 3

If you modify the enqueue instead of queue.append we use queue.insert(0). In this case the Elements come out in reverse order instead.

Answer 4:

The advantages of implementing a queue using linked lists versus array is first, it's easy to implement because of python's built-in function enqueue() and append(). Second, it's cache-friendly because arrays store elements contiguously in memory. Disadvantages of implementing said data structure is it has more wasteful memory in resizing.

Answer 5:

The example of application of queues over stack is, for example, undo/redo functions in editors, task/job scheduling and web request handling.

IV. Conclusion

In conclusion, each data structure whether it be stacks or queues have significant uses in programming, treat it as a tool in order to create an efficient program. Queues in data structure, have a significant use in certain scenarios and so as Stacks. Queues are more used in scenarios where we want a first inserted element to be removed first.

References

[1] Co Arthur O.. “University of Caloocan City Computer Engineering Department Honor Code,” UCC-CpE Departmental Policies, 2020.

[2]“Google Colab,” *Google*, [Online]. Available: https://colab.research.google.com/. Accessed: Oct. 11, 2025.