



Data Structure and Algorithm

Laboratory Activity No.8

Implementation of Stacks to Queues

Submitted by:

Manigbas, Jeus Miguel T. <Leader>

Barredo, Alwin P.

Bela, Lorenzo Miguel D.

Callorina, Robert Victor A.

Guzon, Kean Louiz I.

Instructor:

Engr. Maria Rizette H. Sayo

November, 29, 2023

I. Objectives

Introduction

A stack is a collection of objects that are inserted and removed according to the last-in, first-out (LIFO) principle. A user may insert objects into a stack at any time, but may only access or remove the most recently inserted object that remains (at the so-called “top” of the stack)

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))
```

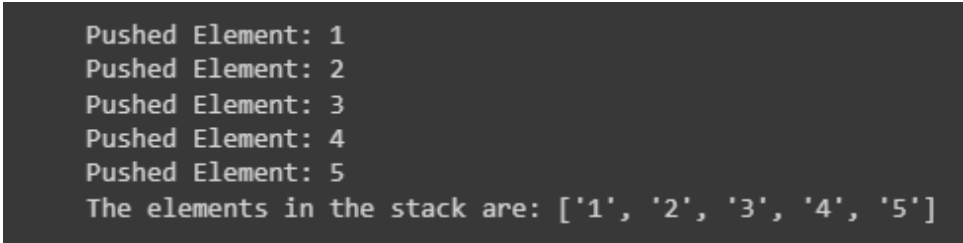
```
push(stack, str(3))
```

```
push(stack, str(4))
```

```
push(stack, str(5))
```

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

III. Results



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

Figure 1. Results of the Code

The provided code is an implementation of a stack data structure in Python. It defines four methods:

create_stack():

Functionality: Initializes and returns an empty stack.

Usage: Called to create an empty stack using a Python list.

is_empty(stack):

Functionality: Determines if the stack is empty.

Usage: Checks if the given stack (list) has zero elements; returns True if empty, False otherwise.

push(stack, item):

Functionality: Adds an item to the top of the stack.

Usage: Appends the provided item to the stack (list) and prints the item added.

pop(stack):

Functionality: Removes and returns the top element from the stack.

Usage: If the stack isn't empty, removes and returns the last element (top of the stack); otherwise, returns a message indicating an empty stack.

Functions:

create_stack(): Initializes an empty stack by creating an empty list.

is_empty(stack): Checks if the stack is empty based on the length of the provided stack (list).

push(stack, item): Adds the provided item to the top of the stack (list).

pop(stack): Removes and returns the top element from the stack (list).

The code creates a stack, pushes five items onto it, and then prints the elements in the stack

IV. Conclusion

The answer provides an implementation of a stack data structure in Python, which includes four methods: `create_stack()`, `is_empty(stack)`, `push(stack, item)`, and `pop(stack)`. The code creates a stack, pushes five items onto it, and then prints the elements in the stack. Additionally, the stack follows the Last-In, First-Out (LIFO) principle: elements pushed last are popped first. Managing exceptions during stack operations and considering trade-offs with list usage could enhance its functionality and performance.

References

- [1] “Stack in Python,” *GeeksforGeeks*, Oct. 09, 2019. <https://www.geeksforgeeks.org/stack-in-python/> (accessed Nov. 29, 2023).
- [2] “Queue in Python,” *GeeksforGeeks*, Oct. 10, 2019. <https://www.geeksforgeeks.org/queue-in-python/> (accessed Nov. 29, 2023).
- [3] “Stack and Queues in Python,” *GeeksforGeeks*, Sep. 17, 2017. <https://www.geeksforgeeks.org/stack-and-queues-in-python/> (accessed Nov. 29, 2023).
- [4] Real Python, “How to Implement a Python Stack,” *Realpython.com*, Jun. 05, 2019. <https://realpython.com/how-to-implement-python-stack/> (accessed Nov. 29, 2023).
- [5] “4.5. Implementing a Stack in Python — Problem Solving with Algorithms and Data Structures,” *Runestone.academy*, 2014. <https://runestone.academy/ns/books/published/pythonds/BasicDS/ImplementingaStackinPython.html> (accessed Nov. 29, 2023).
- [6] Great Learning, “Stack in Python: How To Implement Python Stack?,” *Great Learning Blog: Free Resources what Matters to shape your Career!*, Jul. 2022. <https://www.mygreatlearning.com/blog/python-stack/> (accessed Nov. 29, 2023).