Data Structure and Algorithm

Laboratory Activity No. 8

Stacks

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Enverzo, Kyle Andrey D. | Engr. Maria Rizette H. Sayo |

October, 04, 2025

# 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)

This laboratory activity aims to implement the principles and techniques in:

* Writing Python program using Stack
* Writing a Python program that will implement Stack operations

# Methods

Instruction: Type the python codes below in your Colab. After running your codes, answer the questions below.

# 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. Upon typing the codes, what is the name of the abstract data type? How is it implemented?
2. What is the output of the codes?
3. If you want to type additional codes, what will be the statement to pop 3 elements from the top of the stack?
4. If you will revise the codes, what will be the statement to determine the length of the stack? (Note: You may add additional methods to count the no. of elements in the stack)

# Results

Answers in the Question:

1. The name of the code or program here is a stack. A stack adheres to the LIFO (Last In, First Out) principle, which means the most recent item that was added is the first one to be extracted. It's implemented in Python through a list (stack = []) and methods such as append() to add elements to the stack and pop() to extract them. The push() operation places an element at the end of the list (top of the stack), and the pop() operation takes away the end element from the list.
2. A white background with black text

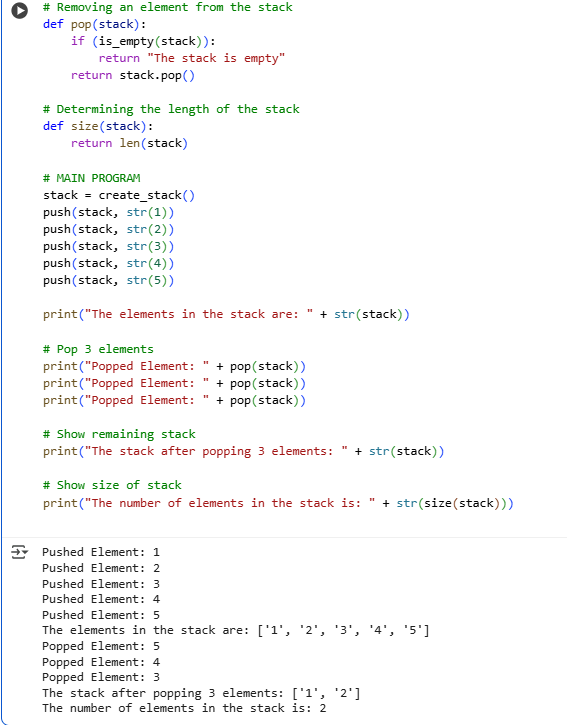
   AI-generated content may be incorrect.

This creates the effect of numbers being added one at a time to a stack. The stack maintains items in reverse so that the most recently added number will be the first to be popped out. In this case, numbers 1 through 5 were pushed in, and the program displays the stack as ['1', '2', '3', '4', '5'] with 1 at the bottom and 5 on top.

1. A screenshot of a computer

   AI-generated content may be incorrect.

Using the Last In, First Out (LIFO) principle, the operation eliminates the topmost items one at a time when we pop three elements from the stack. Popping three times will eliminate 5, 4, and 3 in that order because the stack initially contained [1, 2, 3, 4, 5]. Following this, [1, 2] will be the last elements in the stack.

1. 

To find out the size of the stack, we can make a function utilizing Python's len() function. It will give us the number of elements already saved in the stack. For instance, following pushing or popping of elements, the call to this function will give the number of items remaining in the stack. This keeps track of the size of the stack at any time during the program.

# Conclusion

In summary, the exercise illustrates how a stack operates as an abstract data type through the use of Python lists. By implementing the fundamental operations of push, pop, and size, we can easily see the Last In, First Out (LIFO) aspect of stacks. The code illustrates how elements are inserted, deleted, and enumerated so that one can easily comprehend how stacks process data in computer programming.

**References**

[1] “Stack (abstract data type),” \*Wikipedia\*. [Online]. Available: https://en.wikipedia.org/wiki/Stack\_(abstract\_data\_type) :contentReference[oaicite:0]{index=0}

[2] “Stack in Python,” \*GeeksforGeeks\*. [Online]. Available: https://www.geeksforgeeks.org/python/stack-in-python/ :contentReference[oaicite:1]{index=1}

[3] “How to Implement a Python Stack,” \*Real Python\*. [Online]. Available: https://realpython.com/how-to-implement-python-stack/ :contentReference[oaicite:2]{index=2}

[4] “4.4. The Stack Abstract Data Type,” \*Runestone Academy\*. [Online]. Available: https://runestone.academy/ns/books/published/pythonds/BasicDS/TheStackAbstractDataType.html :contentReference[oaicite:3]{index=3}

[5] “Stack Data Structure and Implementation in Python, Java and C/C++,” \*Programiz\*. [Online]. Available: https://www.programiz.com/dsa/stack :contentReference[oaicite:4]{index=4}