Data Structure and Algorithm

Laboratory Activity No. 8

Stacks

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Gabuyo, Ivan love Gabuyo | Engr. Maria Rizette H. Sayo |

OCT, 4, 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?  
     
   In this program, we use the abstract data type Stack. A stack is a collection that is organized based on the Last-In, First-Out (LIFO) principle. For this implementation, a stack is built using a Python list. To add items to the top of the stack (for the push operation), we use the append() method, and the pop() method is used to remove items from the top of the stack (for the pop operation).
2. What is the output of the codes?  
     
   When the program runs, it will display the following output:  
   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']  
This output shows that each element is successfully pushed onto the stack, and finally, the entire stack is printed as a list of elements.

1. If you want to type additional codes, what will be the statement to pop 3 elements from the top of the stack?  
     
   To remove three elements from the top of the stack, we need to call the pop() function three times. Each call will remove the most recently added element. The code for this is: print("Popped Element:", pop(stack))

print("Popped Element:", pop(stack))

print("Popped Element:", pop(stack))  
This means that the last three numbers pushed into the stack will be removed in reverse order of how they were added.

1. 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)   
     
   To determine the number of elements in the stack, we can create a new function called size() that simply returns the length of the stack using the built-in len() function. The code looks like this:

def size(stack):

return len(stack)

After adding this function, we can call it at any time to check how many items are currently stored in the stack. For example:  
  
 print("The number of elements in the stack is:", size(stack))  
  
This statement will display the exact count of elements in the stack.

# Results

My Google Colab program demonstrated the implementation of stack operations in Python. The software accurately showed the elements in the stack as it recorded them one by one. The push and pop operations were performed according to the last-in, first-out (LIFO) principle.

jus

A black background with white text

AI-generated content may be incorrect.

Figure 1 Screenshot of program  
  
Figure 2 Screenshot of program  
  
In the program illustrated, a stack is built using a Python list. With the push() function, elements are added, and with the pop() function, elements are removed. To add a new element to the top of the stack, the push() function uses the append() method. Then, using the pop() function, the most recently added element can be removed in accordance with the Last-In, First-Out (LIFO) principle.  
  
Figure 2 Screenshot of program  
  
  
  
The program was now able to print the proper number of elements in the stack after adding a function to determine the stack length. This was accomplished by adding a new function that returns the number of elements in the list which holds the stack via Python's len() function. The program can show the amount of items being stored at the time by calling this function after performing a few push or pop operations. This feature is beneficial in viewing how the stack dynamically adjusts as items are added and removed, and making sure that the stack operations are functioning as planned.  
  
  
Figure 3 Screenshot of program



A black screen with white text

AI-generated content may be incorrect.

If an image is taken from another literature or intellectual property, please cite them accordingly in the caption. Always keep in mind the Honor Code [1] of our course to prevent failure due to academic dishonesty.

# Conclusion

A sample program demonstrating stack operations illustrates how one can implement a stack using a list. It illustrates how to execute simple stack operations which are creating an empty stack, whether a stack is empty or not, `push()` adding data to the stack, and `pop()` removing data from the stack. It illustrates and confirms the Last-In, First-Out (LIFO) characteristic of a stack, which is the last one in is the first one out, by printing the pushed elements after pushing 1 through 5 into the stack.

This illustrates the effectiveness of using a stack for simple operations. It also indicates the stack as an Abstract Data Type and how simple data structures are implemented to perform the theoretical ideas. The code illustrates how the stack works and emphasizes the need to study the underlying data structure concepts in computer programming.

**References**

[1] Co Arthur O.. “University of Caloocan City Computer Engineering Department Honor Code,” UCC-CpE DepartGeeksforGeeks. (2023, September 18). *Implementing stack in Python*. GeeksforGeeks. https://www.geeksforgeeks.org/stack-in-python/

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2013). *Data structures and algorithms in Python*. Wiley.

Python Software Foundation. (2024). *Built-in types — list*. Python documentation. https://docs.python.org/3/tutorial/datastructures.html