



UNIVERSITY OF CALOOCAN CITY  
COMPUTER ENGINEERING DEPARTMENT



Data Structure and Algorithm

Laboratory Activity No. 8

---

# Stacks

---

*Submitted by:*  
Maringal, Czer Justine D.

*Instructor:*  
Engr. Maria Rizette H. Sayo

October 4, 2025

# 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)

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

# II. 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))
```

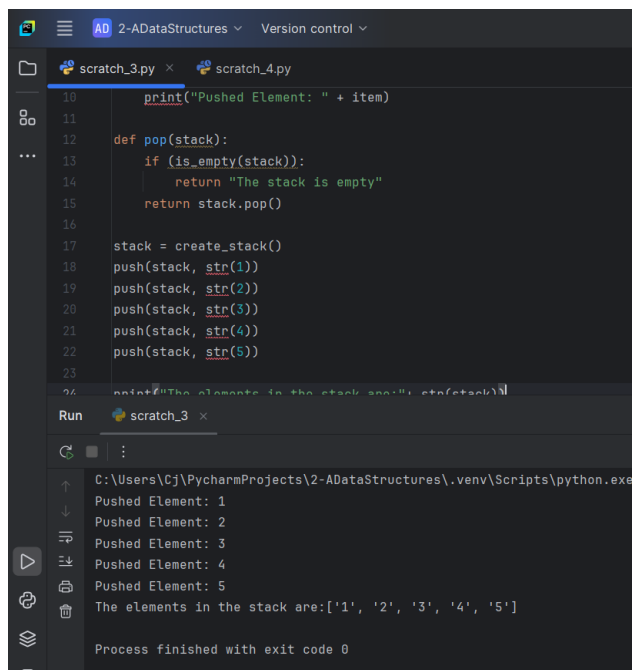
```
print("The elements in the stack are:" + str(stack))
```

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)

### III. Results

1. The name of the abstract data type is stack. It is implemented using a python list, push and pop are the operations defined for the stack in the code. Push is implemented using the append() method of the list, while pop is using the pop().



```
10 print("Pushed Element: " + item)
11
12 def pop(stack):
13     if (is_empty(stack)):
14         return "The stack is empty"
15     return stack.pop()
16
17 stack = create_stack()
18 push(stack, str(1))
19 push(stack, str(2))
20 push(stack, str(3))
21 push(stack, str(4))
22 push(stack, str(5))
23
24 print("The elements in the stack are: " + str(stack))
```

Run scratch\_3

```
C:\Users\Cj\PycharmProjects\2-ADDataStructures\.venv\Scripts\python.exe
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']
Process finished with exit code 0
```

2. Each push() call adds an item to the stack and prints the pushed element. At the end, the current state of the stack is printed.
3. To pop 3 elements from the top of the stack, I will write the code pop() function three times or maybe for short, I'm gonna using for loop to pop with the range of 3.

```
20 stack = create_stack()
21 push(stack, str(1))
22 push(stack, str(2))
23 push(stack, str(3))
24 push(stack, str(4))
25 push(stack, str(5))
26
27 print("The elements in the stack are: " + str(stack))
28 print("The length of the stack is: " + str(stack_size(stack)))
29
30 for i in range(3):
31     print("Popped Element: " + pop(stack))
32
33 print("The elements in the stack after popping 3 elements: " + str(stack))
34 print("The length of the stack is now: " + str(stack_size(stack)))
35
```

4. To determine the length, I'll use the len() function. Here's the revised code:

```
def create_stack():
```

```
    stack = []
```

```
    return stack
```

```
def is_empty(stack):
```

```
    return len(stack) == 0
```

```
def push(stack, item):
```

```
    stack.append(item)    print("Pushed Element: " + item)
```

```
def pop(stack):
```

```
    if is_empty(stack):
```

```
        return "The stack is empty"
```

```
    return stack.pop()
```

```
def stack_size(stack):
```

```
    return len(stack)
```

```
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))
```

```
print("The length of the stack is: " + str(stack_size(stack)))
```

```
for i in range(3):
```

```
    print("Popped Element: " + pop(stack))
```

```
print("The elements in the stack after popping 3 elements: " + str(stack))
```

```
print("The length of the stack is now: " + str(stack_size(stack)))
```

## IV. Conclusion

I learned how to create and use a stack in Python using a list. I practiced adding (push), removing (pop), checking if it's empty, and getting the size. I also learned how to use loops with `i` when the loop variable isn't needed. This activity helped me understand how stacks work and how to write simple Python functions.

## References

- [1] GeeksforGeeks. (2025, August 31). *Stack Data Structure*. GeeksforGeeks.  
<https://www.geeksforgeeks.org/dsa/stack-data-structure/>
- [2] W3Schools.com. (n.d.). [https://www.w3schools.com/dsa/dsa\\_data\\_stacks.php](https://www.w3schools.com/dsa/dsa_data_stacks.php)