UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 8

Stacks

Submitted by:
Adoracion, Jerick Dave D.

Instructor:
Engr. Maria Rizette H. Sayo

October 04, 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

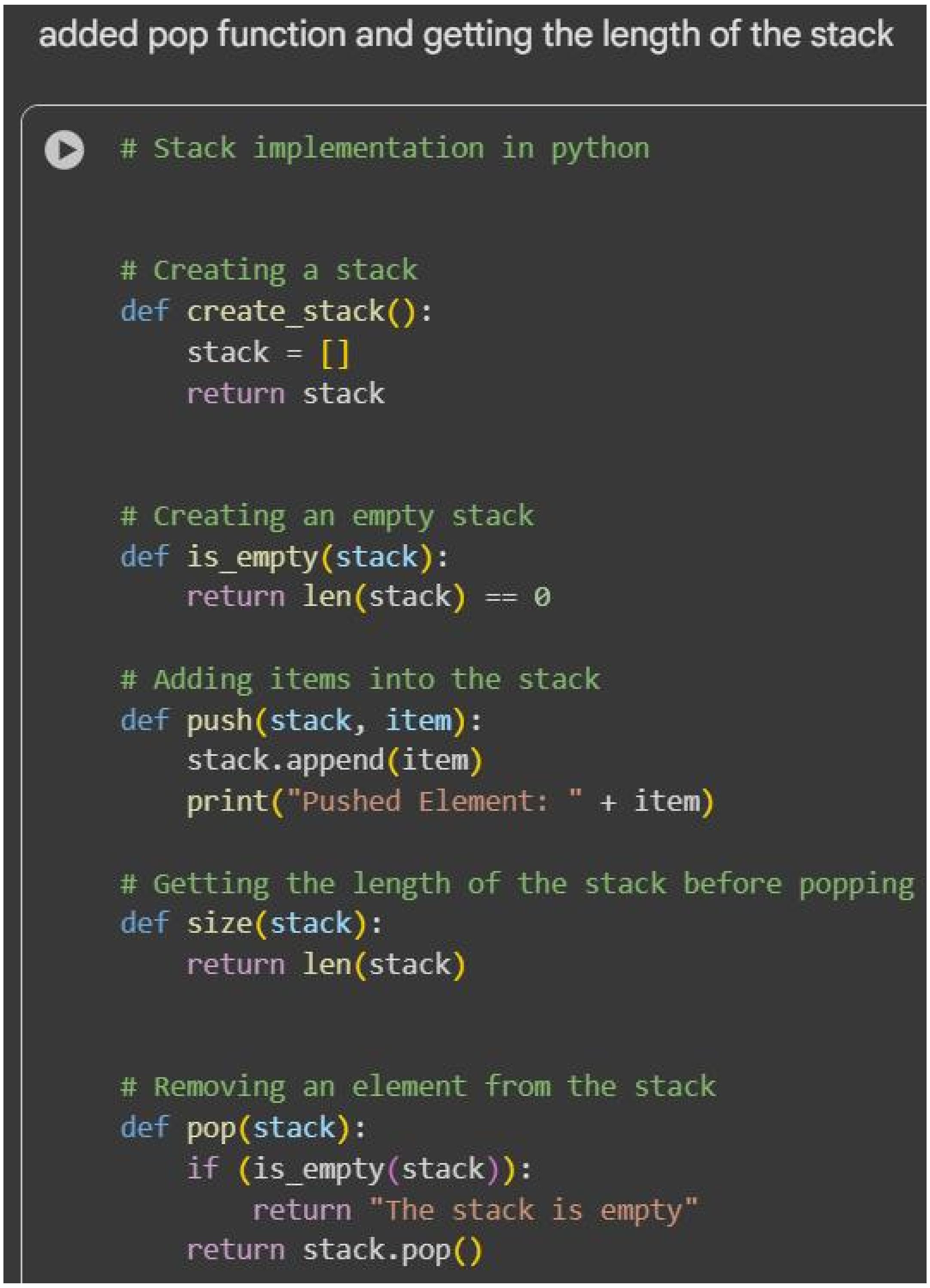
Instruction: Type the python codes below in your Colab. After running your codes, answer the questions below.

```
# 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))
```

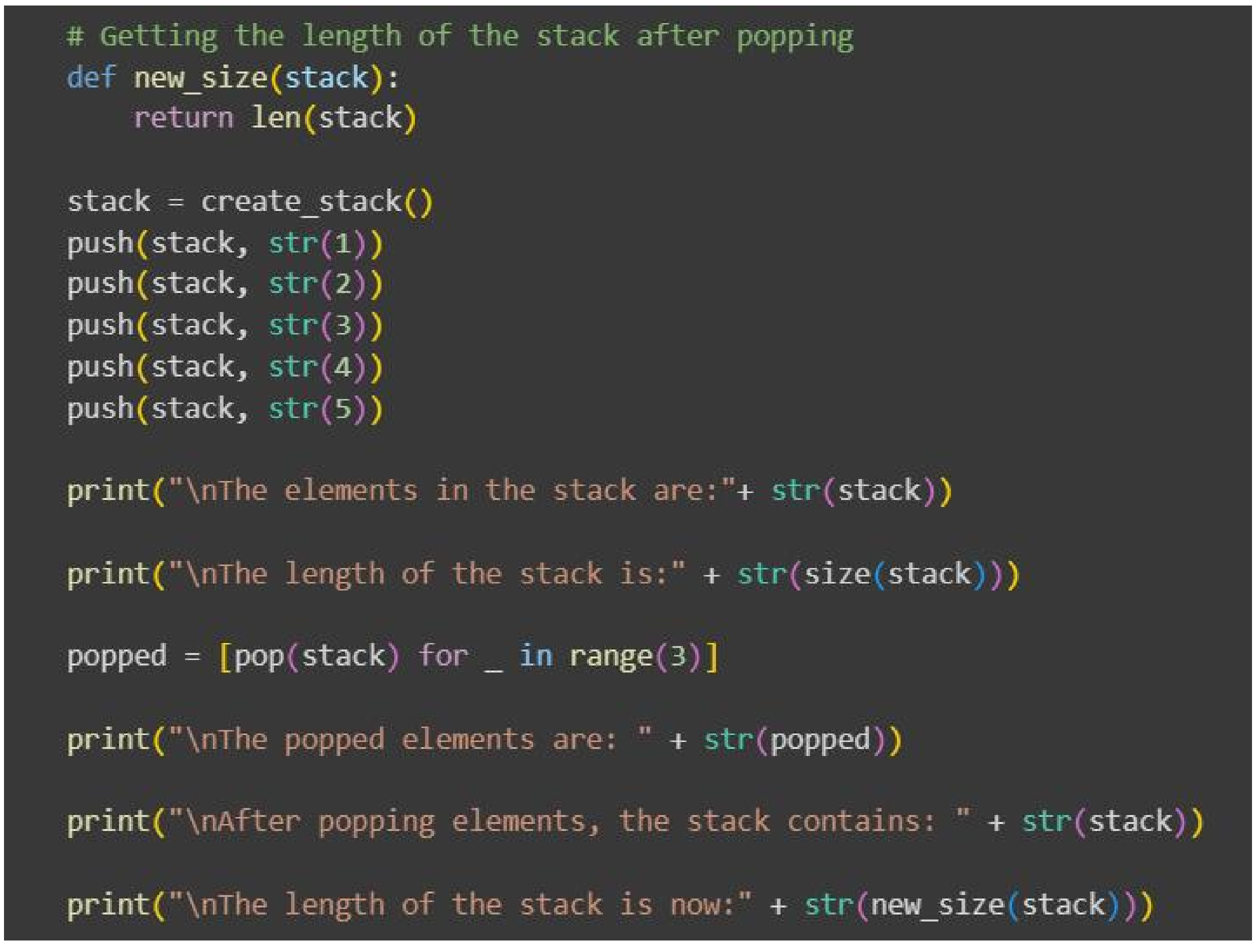
Answer the following questions:

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


added pop function and getting the length of the stack

```python
# 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)

# Getting the length of the stack before popping
def size(stack):
    return len(stack)


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



```python
# Getting the length of the stack after popping
def new_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("\nThe elements in the stack are:"+ str(stack))

print("\nThe length of the stack is:" + str(size(stack)))

popped = [pop(stack) for _ in range(3)]

print("\nThe popped elements are: " + str(popped))

print("\nAfter popping elements, the stack contains: " + str(stack))

print("\nThe length of the stack is now:" + str(new_size(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 code, it used **Stack** in which the elements are arranged in an LIFO (Last in, First Out) order. In the code, it used a **Push** function wherein the elements are pushed at the top of the previous one, arranging them in a LIFO order. ['1', '2', '3', '4', '5']

2. What is the output of the codes?

Pushed Element: 1

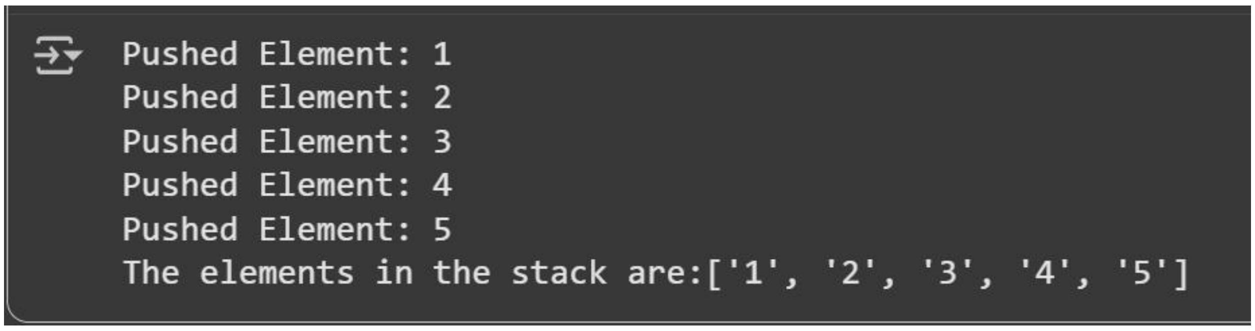
Pushed Element: 2

Pushed Element: 3

Pushed Element: 4

Pushed Element: 5

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



```
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']
```

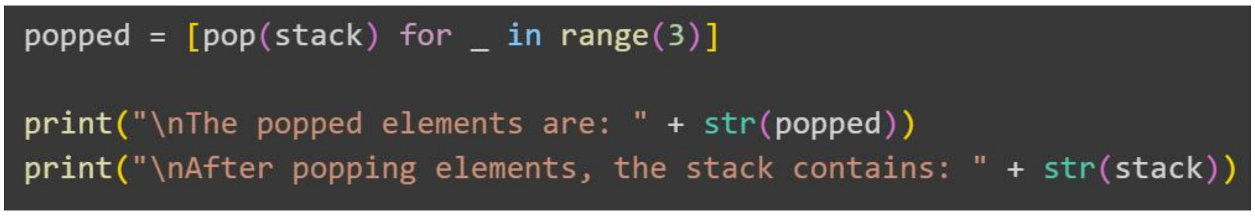
Figure 1: Output

3. If you want to type additional codes, what will be the statement to pop 3 elements from the top of the stack?

popped = [pop(stack) for _ in range(3)]

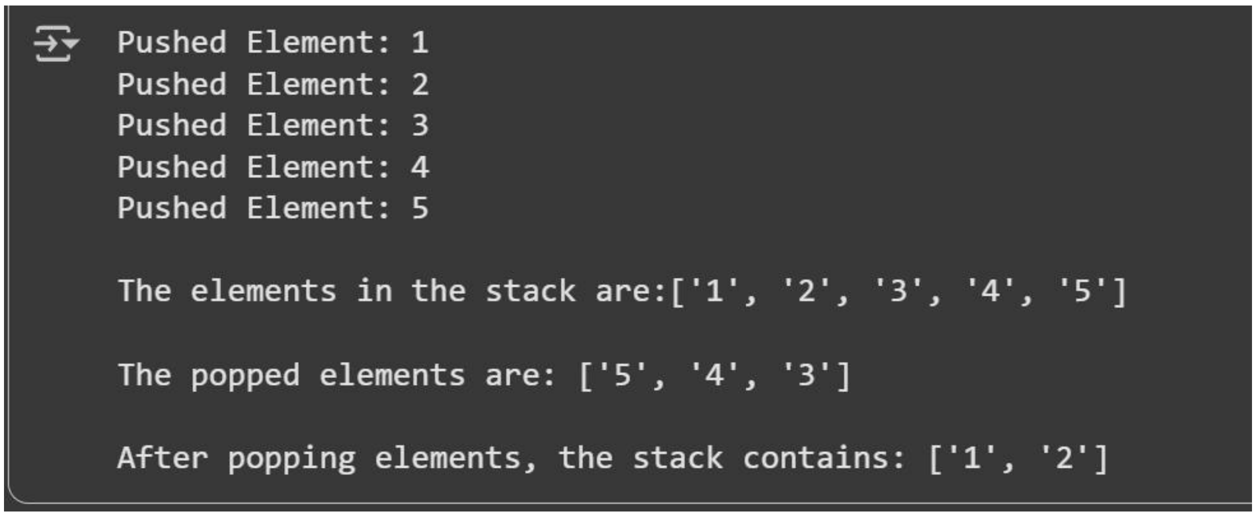
print("\nThe popped elements are: " + str(popped))

print("\nAfter popping elements, the stack contains: " + str(stack))



```
popped = [pop(stack) for _ in range(3)]

print("\nThe popped elements are: " + str(popped))
print("\nAfter popping elements, the stack contains: " + str(stack))
```

Figure 2: Additional code

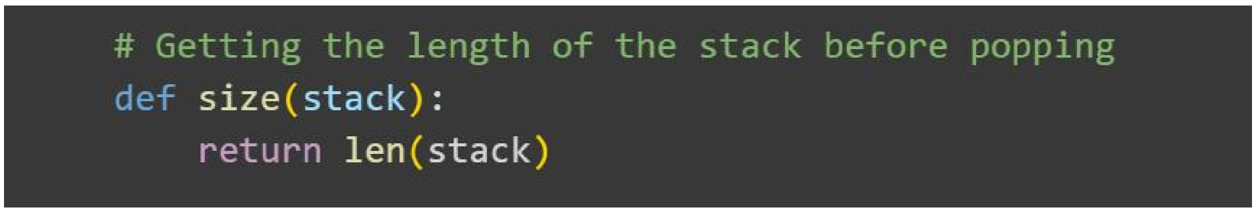


```
Pushed Element: 1
Pushed Element: 2
Pushed Element: 3
Pushed Element: 4
Pushed Element: 5

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

The popped elements are: ['5', '4', '3']

After popping elements, the stack contains: ['1', '2']
```

Figure 3: Output

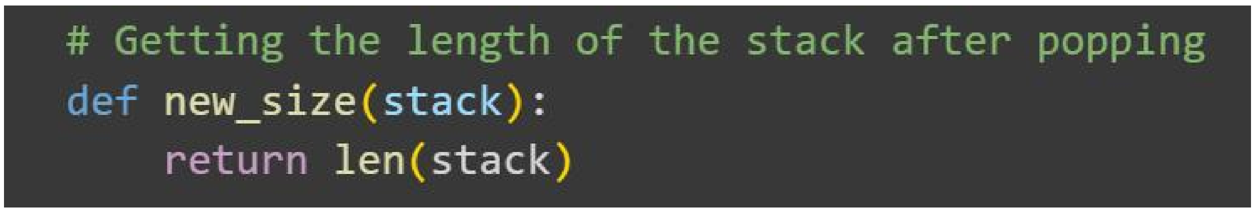
4. If you will revise the codes, what will be the statement to determine the length of the stack? (Note: You may add additional methods to count the no. of elements in the stack)

I used len() to measure the elements in the two stacks which are the before and after popping three elements.



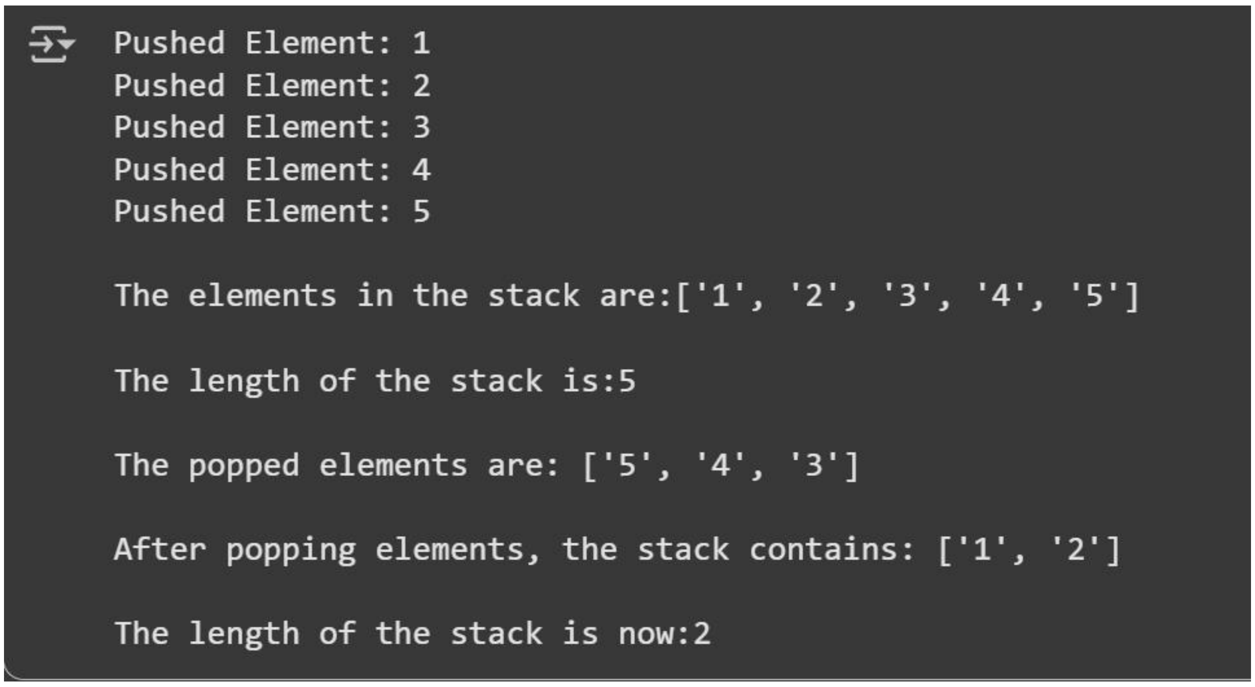
```python
# Getting the length of the stack before popping
def size(stack):
    return len(stack)
```

Figure 4: Getting the length before popping



```python
# Getting the length of the stack after popping
def new_size(stack):
    return len(stack)
```

Figure 5: Getting the length after popping



```
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']

The length of the stack is:5

The popped elements are: ['5', '4', '3']

After popping elements, the stack contains: ['1', '2']

The length of the stack is now:2
```

Figure 6: Final Output

IV. Conclusion

In this laboratory activity, I’ve recalled our lesson in the mid term which is stack. Stack uses a LIFO Order wherein the elements are arranged in a Last in, first out order. During the activity, I encountered an error at first wherein I can’t get the output that I want, but after analyzing and with the help of my references, I managed to identify the error, I forgot to that I am using string.

In conclusion, this is an easy activity as we don’t have to start from scratch because we are provided with a code reference to add different functions. I’ve learned again, and will seek for more.

References

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

[2] M. A. Weiss, *Data Structures and Algorithm Analysis in C++*, 4th ed. Boston, MA, USA: Pearson, 2014.

[3] D. M. Beazley, *Python Essential Reference*, 4th ed. Upper Saddle River, NJ, USA: Addison-Wesley, 2009.

[4] G. van Rossum and Python Software Foundation, “Python Language Reference,” Python.org, 2023. [Online]. Available: https://docs.python.org/3/reference/index.html

[5] S. Sedgewick and K. Wayne, *Algorithms*, 4th ed. Boston, MA, USA: Addison-Wesley, 2011.