Algorithms & Data Structures

Lab 8 (Stack Implementation in Python)

```
# The Stack class simulates a stack
class Stack_class:
    # The _ _init_ _ method initializes the
    # stack data attribute
    def __init__(self, size):
        self.stack = []
        self.size = size

    # The push method add an item to the stack
    def push(self, item):
        if len(self.stack) == self.size:
            print("Stack is full!!")
        else:
            self.stack.append(item)
            print('stack size is: ', len(self.stack))

    # The pop method delete an item from the stack
    def pop(self):

        if self.stack == []:
            print("Stack is empty!")
            result = -1
        else:
            result = self.stack.pop()
        return result

    # The display method will state items
    # of the stack
    def display(self):
        if self.stack == []:
            print("Stack is empty!")
        else:
            print("Stack data:")
            for item in reversed(self.stack):
                print(item)
```

```
# Let the user set the size of the stack
s = int(input('Enter the size of the stack'))
exit = False

# Create an object from the Stack class.
# Stack (with a capital S) is the class name
# stack (with a small s) is the object
# created from the "Stack" class
stack = Stack_class(s)
while exit is False:
    # Choose which operation you want
    # to apply to the stack.
    print("1) Push")
    print("2) Pop")
    print("3) Display")
    print("Choose an operation: ")
    operation = input()


    def pushOp():
        number = input("Insert a number:")
        if number.isdigit():
            # we use the keyword "global" to force
            # the function to access
            # the main stack object that we created.
            global stack
            # call the function "push" that we
            # defined in the main "Stack" class
            stack.push(number)
        else:
            print("Invalid input...")


    def popOp():
        global stack
        n = stack.pop()
        if n != -1:
            print(f"Deleted data: {n}")
```

```
    def displayOp():
        global stack
        stack.display()


    def exitOp():
        global exit
        exit = True
        print("Exiting...")


    # Create a dictionary
    # Dictionaries are data structures special
    # to python they save values as a pair
    # all other data structures store values one
    # at a time.
    # dictionaries save values as (Key:Value)
    switch = {
        '1': pushOp,
        '2': popOp,
        '3': displayOp
    }
    # "get" is a special function related to
dictionaries.
    # When get() is called, Python checks if the
specified
    # key exists in the dict. If it does, then get()
returns
    # the value of that key. If the key does not
exist,
    # then get() returns the value specified in the
    # second argument
    switch.get(operation, exitOp)()
```