Experiment no 6

Stack Implementation

OBJECTIVES:

- To learn about the concept of Stack.
- To learn different ways for implementing a Stack.
- To implement Stack using class and functions.

1. STACK:

A stack is a linear data structure that stores items in a Last-In/First-Out (LIFO) or First-In/Last-Out (FILO) manner. In stack, a new element is added at one end and an element is removed from that end only. The insert and delete operations are often called push and pop.

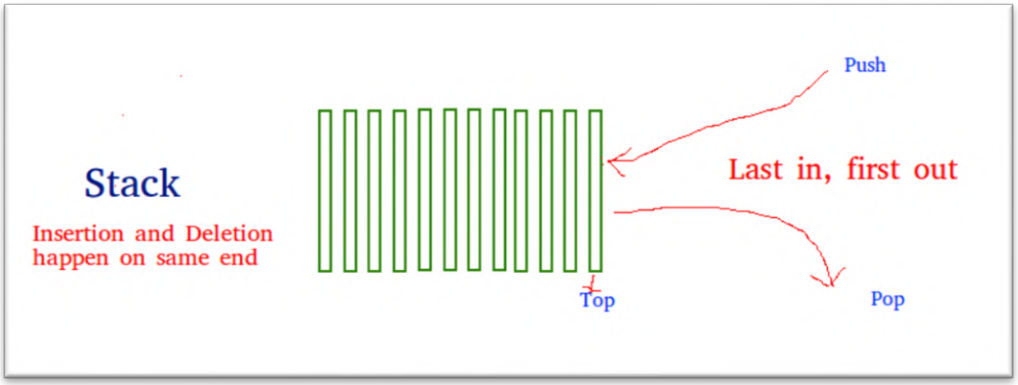


Figure 1: Stack Insertion and Deletion

The functions associated with stack are:

- empty() – Returns whether the stack is empty – Time Complexity: O(1)
- size() – Returns the size of the stack – Time Complexity: O(1)
- top() / peek() – Returns a reference to the topmost element of the stack – Time Complexity: O(1)
- push(a) – Inserts the element 'a' at the top of the stack – Time Complexity: O(1)
- pop() – Deletes the topmost element of the stack – Time Complexity: O(1)

There are various ways from which a stack can be implemented in Python. This article covers the implementation of a stack using data structures and modules from the Python library.
Stack in Python can be implemented using the following ways:

- list
- Collections.deque
- queue.LifoQueue

2. IMPLEMENTATION USING LIST:

Python's built-in data structure list can be used as a stack. Instead of push(), append() is used to add elements to the top of the stack while pop() removes the element in LIFO order.

```
# Python program to
# demonstrate stack implementation
# using list

stack = []

# append() function to push
# element in the stack
stack.append('a')
stack.append('b')
stack.append('c')

print('Initial stack')
print(stack)

# pop() function to pop
# element from stack in
# LIFO order
print('\nElements popped from stack:')
print(stack.pop())
print(stack.pop())
print(stack.pop())

print('\nStack after elements are popped:')
print(stack)

# uncommenting print(stack.pop())
# will cause an IndexError
# as the stack is now empty
```

Figure 2: Implementation using Lists

3. IMPLEMENTATION USING collections.deque:

Python stack can be implemented using the deque class from the collections module. Deque is preferred over the list in the cases where we need quicker append and pop operations from both the ends of the container, as deque provides an O(1) time complexity for append and pop operations as compared to list which provides O(n) time complexity.

The same methods on deque as seen in the list are used, append() and pop().

```
# Python program to
# demonstrate stack implementation
# using collections.deque

from collections import deque

stack = deque()

# append() function to push
# element in the stack
stack.append('a')
stack.append('b')
stack.append('c')

print('Initial stack:')
print(stack)

# pop() function to pop
# element from stack in
# LIFO order
print('\nElements popped from stack:')
print(stack.pop())
print(stack.pop())
print(stack.pop())

print('\nStack after elements are popped:')
print(stack)

# uncommenting print(stack.pop())
# will cause an IndexError
# as the stack is now empty
```

Figure 3: Implementation using deque

4. IMPLEMENTATION USING QUEUE MODULE:

Queue module also has a LIFO Queue, which is basically a Stack. Data is inserted into Queue using the put() function and get() takes data out from the Queue.

There are various functions available in this module:

- maxsize – Number of items allowed in the queue.
- empty() – Return True if the queue is empty, False otherwise.
- full() – Return True if there are *maxsize* items in the queue. If the queue was initialized with maxsize=0 (the default), then full() never returns True.
- get() – Remove and return an item from the queue. If the queue is empty, wait until an item is available.
- get_nowait() – Return an item if one is immediately available, else raise QueueEmpty.
- put(item) – Put an item into the queue. If the queue is full, wait until a free slot is available before adding the item.
- put_nowait(item) – Put an item into the queue without blocking. If no free slot is immediately available, raise QueueFull.
- qsize() – Return the number of items in the queue.

```python
# Python program to
# demonstrate stack implementation
# using queue module

from queue import LifoQueue

# Initializing a stack
stack = LifoQueue(maxsize=3)

# qsize() show the number of elements
# in the stack
print(stack.qsize())

# put() function to push
# element in the stack
stack.put('a')
stack.put('b')
stack.put('c')

print("Full: ", stack.full())
print("Size: ", stack.qsize())

# get() function to pop
# element from stack in
# LIFO order
print('\nElements popped from the stack')
print(stack.get())
print(stack.get())
print(stack.get())

print("\nEmpty: ", stack.empty())
```

Figure 4: Implementation using queue module

5. ACTIVITIES:

A1) Write a code for stack implementation using lists for your registration number.

A2) Write a code for stack implementation using deque for your department name i.e "Electrical and Computer Engineering"

A3) Write a code for stack implementation using queue module for name of all the courses of your current semester. (Make a queue for the names of your courses of current semester).

A4) Write a python program to implement a stack using linked list, class of stack using functions of the methods used for stack.

IMPORTANT: All the activities` code should be attached to the manual before summary section.

Lab Exercise and Summary

Summary should cover Introduction, Procedure, Data Analysis and Evaluation.

Student's Signature:_____ **Date:**_____

LABORATORY SKILLS ASSESSMENT (Psychomotor)

Total Marks:100

Criteria (Max Marks)	Level 1 0% ≤ S < 50%	Level 2 50% ≤ S< 70%	Level 3 70% ≤ S< 90%	Level 4 90%≤ S ≤100%	Score (S)
Procedural Awareness (30)	Selects inappropriate skills and/or strategies Required by the task.	Selects and applies appropriate skills and/or strategies required by the task with major errors.	Selects and applies the appropriate strategies and/or skills specific to the task without significant errors.	Selects and applies appropriate strategies and/or skills specific to the task without any error.	
Practical Implementation (30)	Makes major critical errors in applying procedural knowledge related to python Stack Implementation.	Makes numerous critical errors in applying procedural knowledge related to python Stack Implementation.	Makes some non-critical errors in applying procedural knowledge related to python Stack Implementation.	Applies the procedural knowledge in optimized ways related to python Lists, Stack Implementation.	
Use of Tool/Equipment (30)	Uses tools, equipment and materials with limited competence.	Uses tools, equipment and materials with some competence.	Uses tools, equipment and materials with considerable competence.	Uses tools, equipment and materials with a high degree of competence.	
Safety (10)	Requires constant reminders to follow safety procedures.	Requires some reminders to follow safety procedures.	Follows safety procedures with only minimal reminders.	Routinely follows safety procedures.	
				Marks Obtained	

Instructor Name:_____ **Sign:** _____

LABORATORY SKILLS ASSESSMENT (Affective)

Total Marks: 40

Criteria (Max. Marks)	Level 1 0% ≤ S < 50%	Level 2 50% ≤ S < 70%	Level 3 70% ≤ S < 90%	Level 4 90% ≤ S ≤ 100%	Score
Introduction (5)	Very little background information provided or information is incorrect	Introduction is brief with some minor mistakes	Introduction is nearly complete, missing some minor points	Introduction complete and well-written; provides all necessary background principles for the experiment	
Procedure (5)	Many stages of the procedure are not entered on the lab report.	Many stages of the procedure are entered on the lab report.	The procedure could be more efficiently designed but most stages of the procedure are entered on the lab report.	The procedure is well designed and all stages of the procedure are entered on the lab report.	
Data Record (10)	Data is brief and missing significant pieces of information.	Data provides some significant information and has few critical mistakes.	Data is almost complete but has some minor mistakes.	Data is complete and relevant. Tables with units are provided. Graphs are labeled. All questions are answered correctly.	
Data Analysis (10)	Data is presented in very unclear manner.	Data is presented in ways that are not clear enough.	Data is presented in ways that can be understood and interpreted.	Data is presented in ways that best facilitate understanding and interpretation.	
Report Quality (10)	Report contains many errors.	Report is somewhat organized with some spelling or grammatical errors.	Report is well organized and cohesive but contains some grammatical errors.	Report is well organized and cohesive and contains no grammatical errors. Presentation seems polished.	
				Marks Obtained	

LABORATORY SKILLS ASSESSMENT (Cognitive)

Total Marks: 10

(If any) **Marks Obtained**	

Instructor's Signature:_____ **Date:**_____