Data Structures

Contents

- List as Stack
- List as Queue

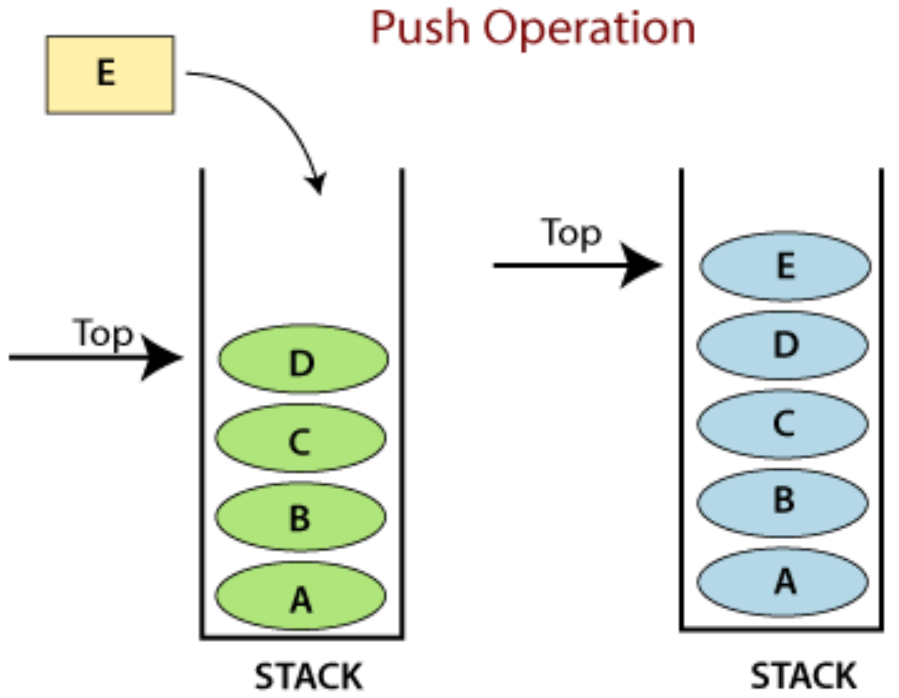
Data structure organizes the storage in computers so that we can easily access and change data.

Stack

A Stack is a data structure that follows the LIFO(Last In First Out) principle. To implement a stack, we need two simple operations:

push - It adds an element to the top of the stack.

pop - It removes an element from the top of the stack.

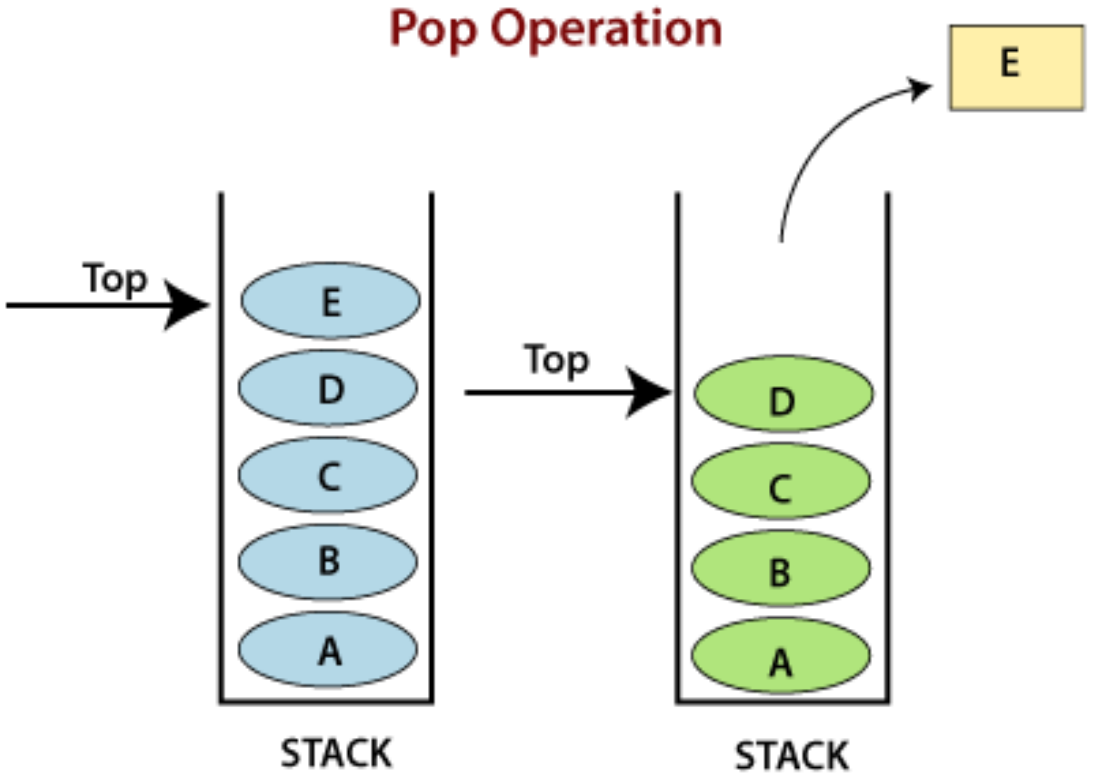


Operations:

•**Adding -** It adds the items in the stack and increases the stack size. The addition takes place at the top of the stack.

•**Deletion -** It consists of two conditions, first, if no element is present in the stack, then underflow occurs in the stack, and second, if a stack contains some elements, then the topmost element gets removed. It reduces the stack size.

•**Traversing -** It involves visiting each element of the stack.



```
# Code to demonstrate Implementation of
# stack using list
x = ["Python", "C", "Android"]
x.append("Java")
x.append("C++")
print(x)
print(x.pop())
print(x)
print(x.pop())
print(x)
```

Using Lists as Stacks

The list methods make it very easy to use a list as a stack, where the last element added is the first element retrieved ("last-in, first-out"). To add an item to the top of the stack, use append(). To retrieve an item from the top of the stack, use pop() without an explicit index. For example:

```
>>> stack = [3, 4, 5]
>>> stack.append(6)
>>> stack.append(7)
>>> stack
[3, 4, 5, 6, 7]
>>> stack.pop()
7
>>> stack
[3, 4, 5, 6]
>>> stack.pop()
6
>>> stack.pop()
5
>>> stack
[3, 4]
```

Using Lists as Queues

List is a Python's built-in data structure that can be used as a queue. append() and pop() functions are used.
However, lists are quite slow for this purpose because inserting or deleting an element at the beginning requires shifting all of the other elements by one.

```
# Python program to demonstrate queue implementation using list
# Initializing a queue
queue = []
 # Adding elements to the queue
queue.append('a')
queue.append('b')
queue.append('c')
print("Initial queue")
print(queue)
# Removing elements from the queue
print("\nElements dequeued from queue")
print(queue.pop(0))
print(queue.pop(0))
print(queue.pop(0))
print("\nQueue after removing elements")
print(queue)
```

Output:

```
Initial queue
['a', 'b', 'c']

Elements dequeued from queue
a
b
c

Queue after removing elements
[]
```