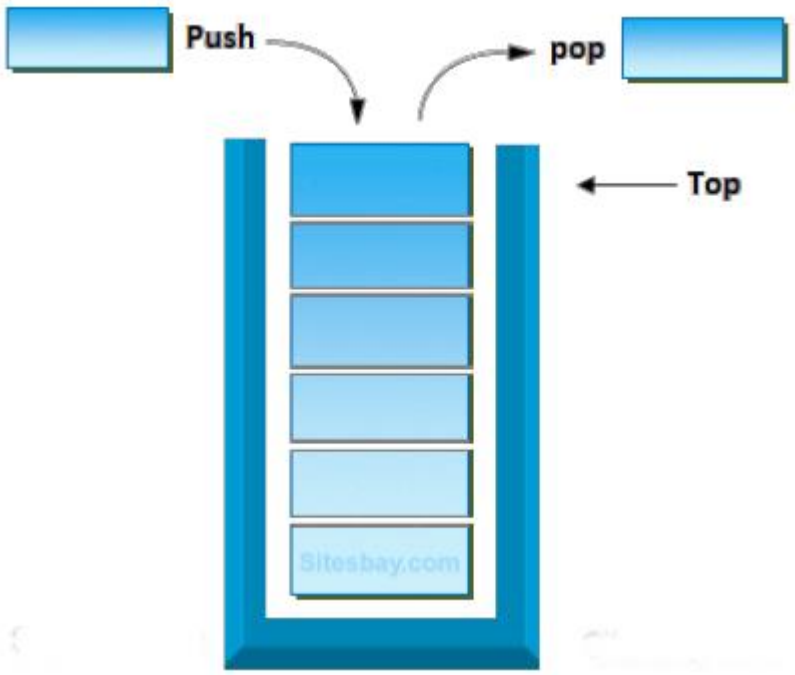
Stack ADT

Anoop S Babu
Faculty Associate
Dept. of Computer Science & Engineering
bsanoop@am.amrita.edu

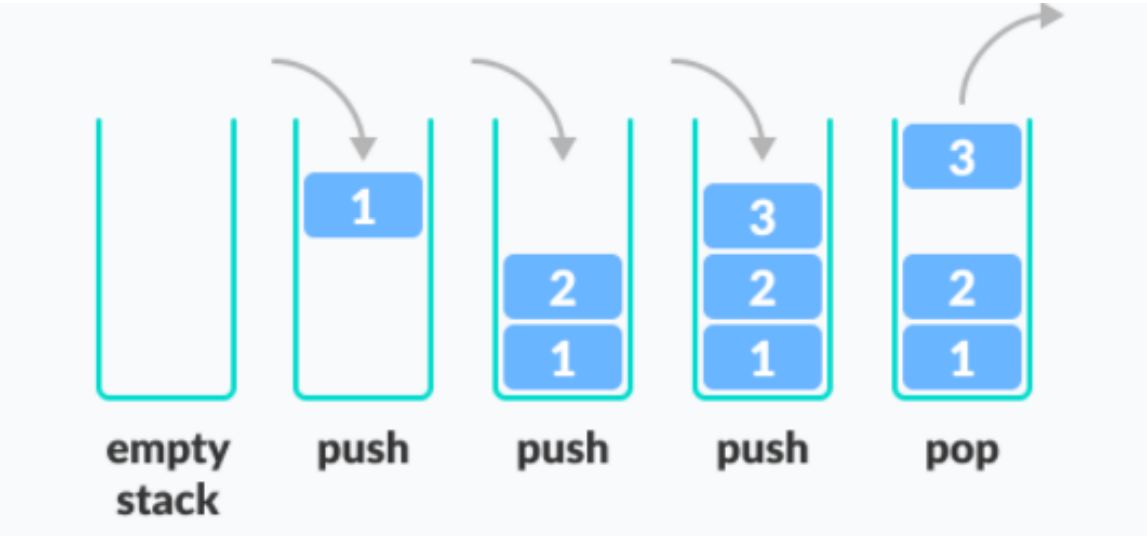
Stack

- A stack is a data structure that **stores a linear collection of items** with **access limited to a last-in first-out (LIFO)** order.
- Adding and removing items is **restricted to one end** known as the **top** of the stack.



LIFO Principle of Stack

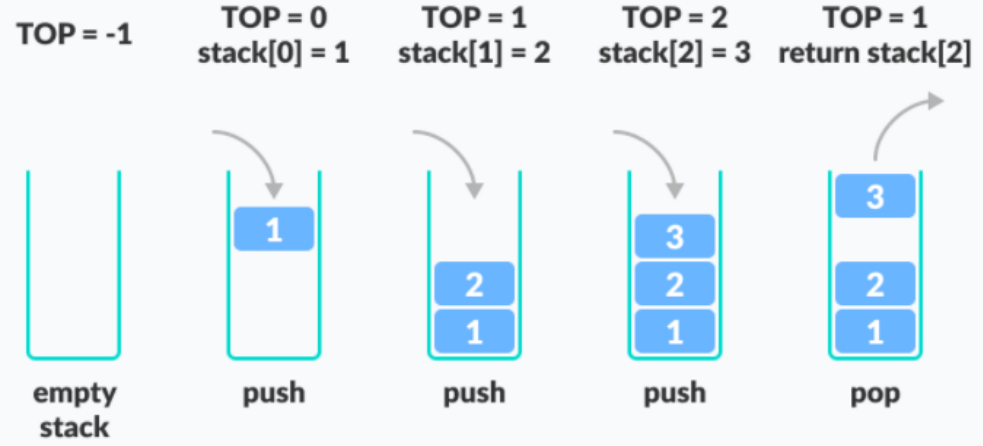
- Putting an item on top of the stack is called **push.**
- Removing an item is called **pop**.



Stack: Basic Operations

- **Push (*item*)**: Add an given item to the top of a stack.
- **Pop()**: Removes and returns the top item of the stack, if the stack is not empty. The next item on the stack becomes the new top item.
- **Stack():** Creates a new empty stack.
- **IsEmpty()**: Check if the stack is empty or not. Return a boolean value.
- **IsFull()**: Check if the stack is full. Return a boolean value.
- **Peek()**: Get the value of the top element without removing it.
- **Length():** Returns the number of items in the stack.

Working of Stack Data Structure



1. A pointer called TOP is used to keep track of the top element in the stack.
2. When initializing the stack, we set its value to -1 so that we can check if the stack is empty by comparing TOP == -1.
3. On pushing an element, we increase the value of TOP and place the new element in the position pointed to by TOP.
4. On popping an element, we return the element pointed to by TOP and reduce its value.
5. Before pushing, we check if the stack is already full
6. Before popping, we check if the stack is already empty

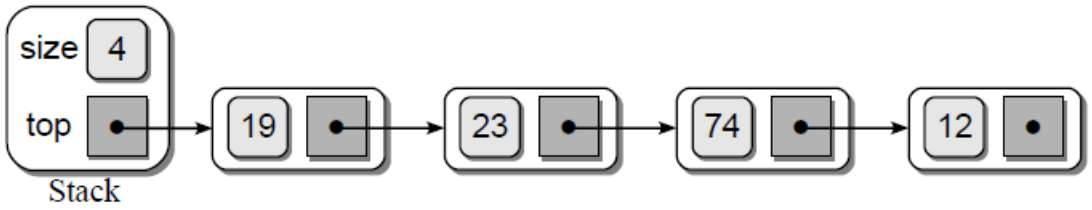
Example: Stack Operations

Operation	Return Value	Stack Contents
S.push(5)	–	[5]
S.push(3)	–	[5, 3]
len(S)	2	[5, 3]
S.pop( )	3	[5]
S.is_empty( )	False	[5]
S.pop( )	5	[ ]
S.is_empty( )	True	[ ]
S.pop( )	"error"	[ ]
S.push(7)	–	[7]
S.push(9)	–	[7, 9]
S.top( )	9	[7, 9]
S.push(4)	–	[7, 9, 4]
len(S)	3	[7, 9, 4]
S.pop( )	4	[7, 9]
S.push(6)	–	[7, 9, 6]
S.push(8)	–	[7, 9, 6, 8]
S.pop( )	8	[7, 9, 6]

Implementing the Stack: Using Python List

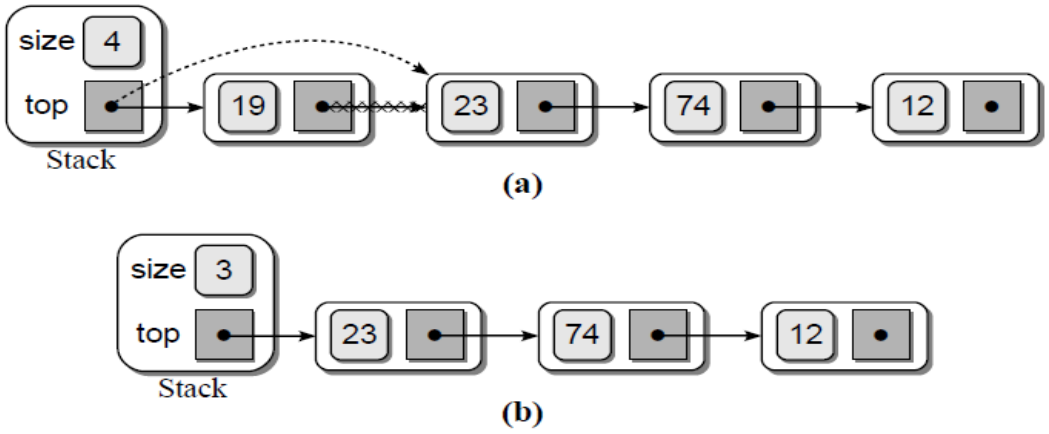
```
# Stack using Array - Implementation using python list

# Creating an empty stack
def stack():
    stack = []
    return stack
# Creating the stack is empty or not
def isEmpty(stack):
    return len(stack) == 0
# Returning the number of items in the stack
def length(stack):
    return len(stack)
# Adding items into the stack
def push(stack, item):
    stack.append(item)
    print("pushed item: " + item)
# Removing an element from the stack
def pop(stack):
    if (isEmpty(stack)):
        return "stack is empty"
    return stack.pop()
```

Implementing the Stack: Using Single Linked List



Sample object of the Stack ADT implemented as a linked list.

```
# Creates an empty stack.
class Stack :
    def __init__( self ):
        self._top = None
        self._size = 0
```



Popping an item from the stack: (a) the required link modifications, and b) the result after popping the top item.

Applications of Stack

- String reversal.
- Expression evaluation/conversion.
- Recursion.
- DFS(Depth First Search).
- UNDO/REDO.
- Backtracking.