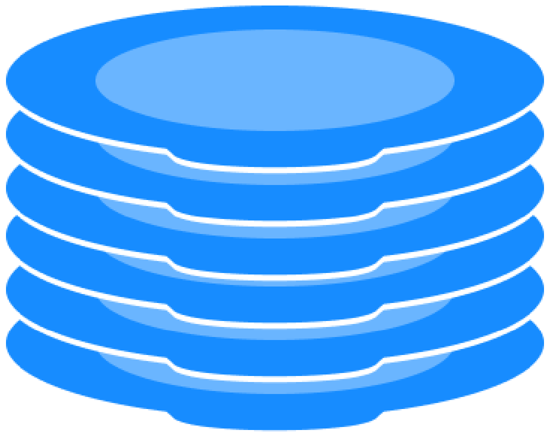
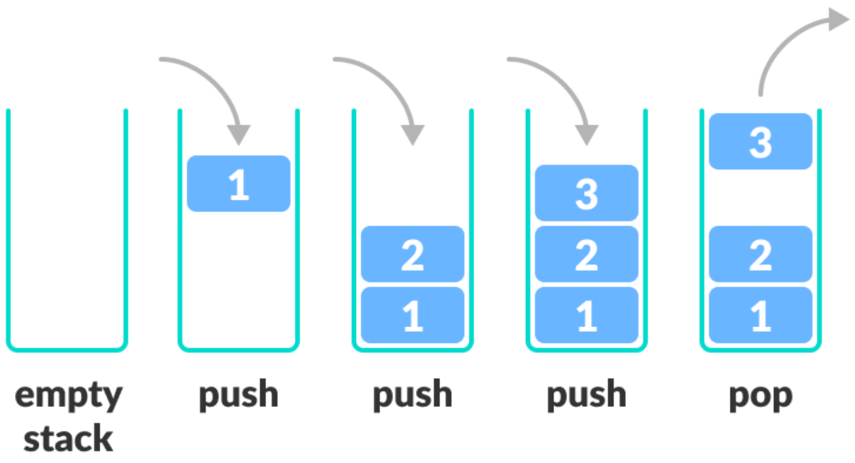
Stack Data Structure

A stack is a linear data structure that follows the principle of **Last In First Out (LIFO)**. This means the last element inserted inside the stack is removed first. You can think of the stack data structure as the pile of plates on top of another.



Stack representation similar to a pile of plate



Stack Push and Pop Operations

Basic Operations of Stack

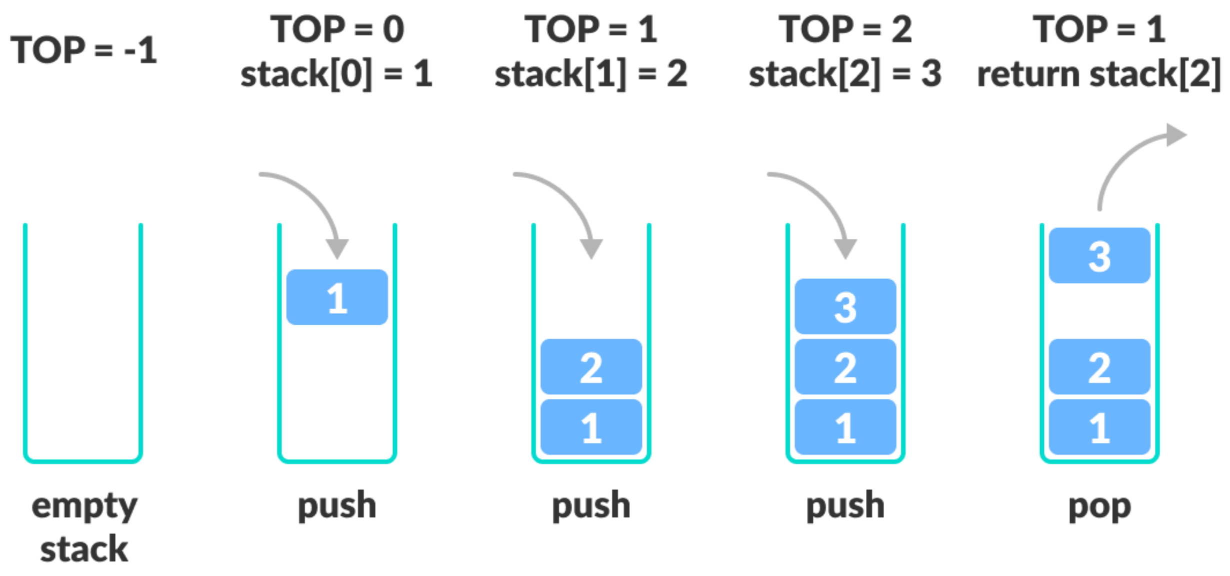
There are some basic operations that allow us to perform different actions on a stack.

- **Push**: Add an element to the top of a stack
- **Pop**: Remove an element from the top of a stack
- **IsEmpty**: Check if the stack is empty
- **IsFull**: Check if the stack is full
- **Peek**: Get the value of the top element without removing it
- **Size**: the number of elements in stack
- **Traverse** : loop through all elements of stack
- **Clear** : remove all elements from stack

Working of Stack Data Structure

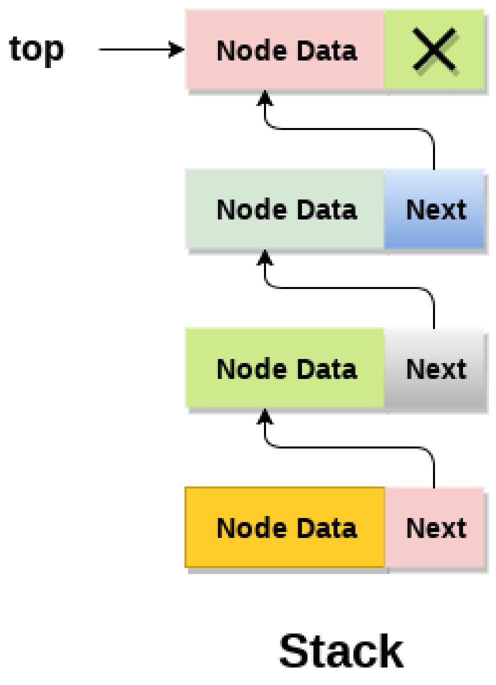
The operations work as follows:

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

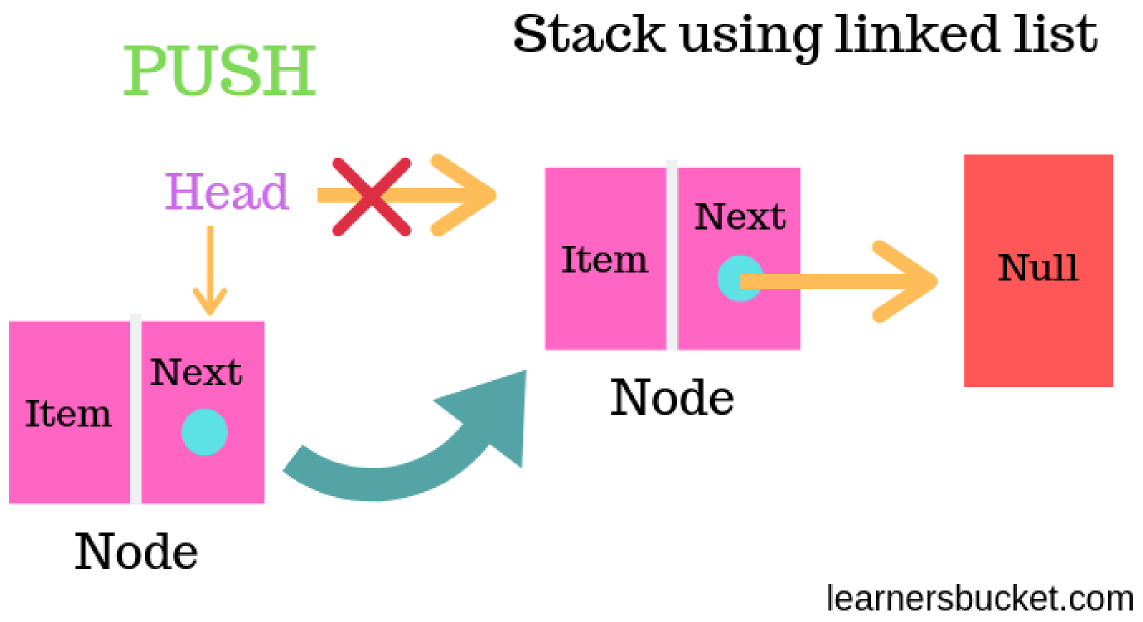


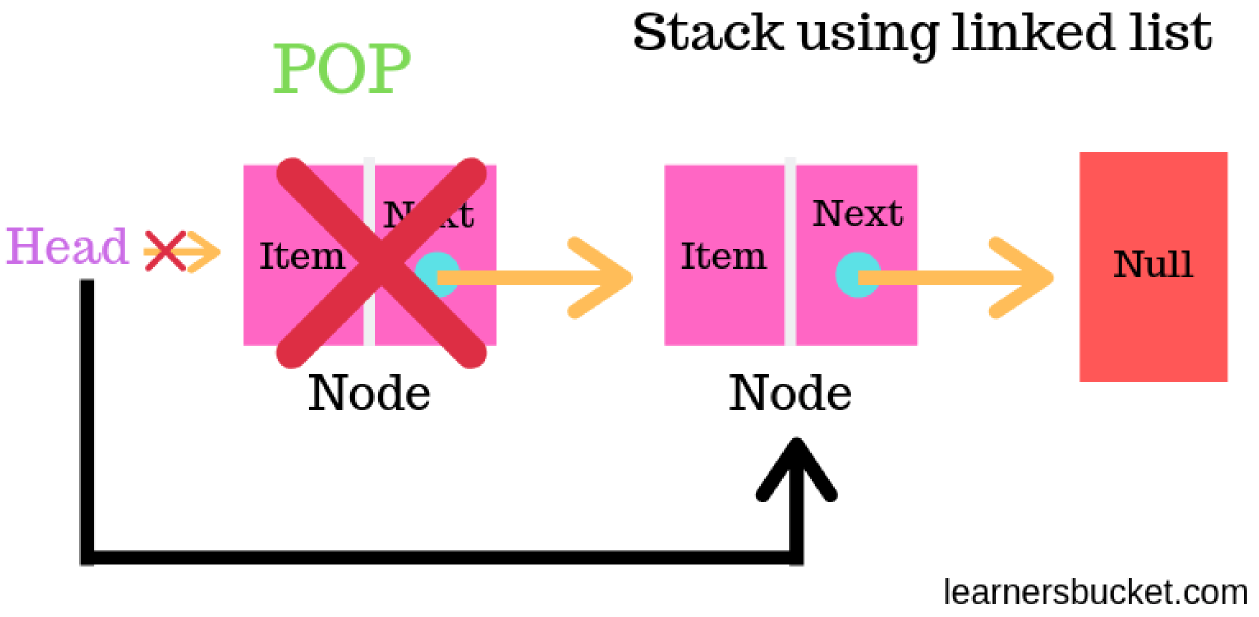
Working of Stack Data Structure

Stack with linked list



linked list stack


Stack using linked list
PUSH
Head
Next
Item
Node
Item
Next
Null
Node
learnersbucket.com


Stack using linked list
POP
Head
Item
Next
Node
Item
Next
Null
Node
learnersbucket.com