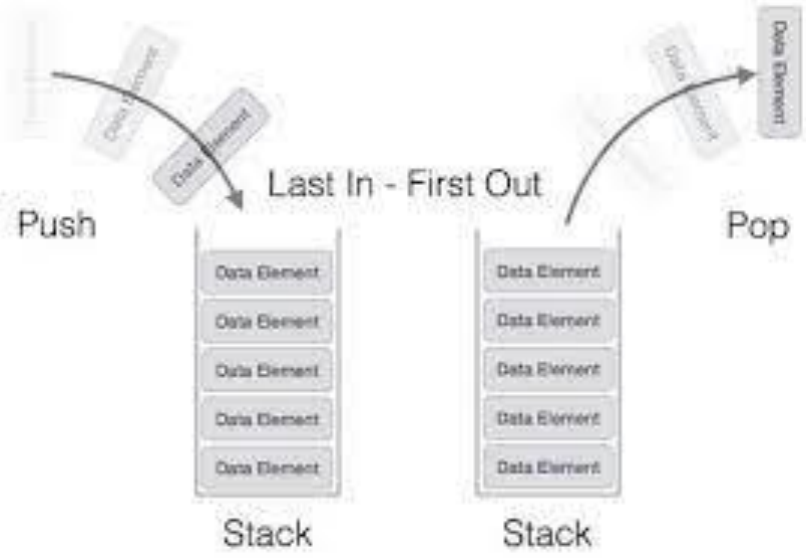
Stack Data Structure

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be **LIFO**(Last In First Out) or **FILO**(First In Last Out).

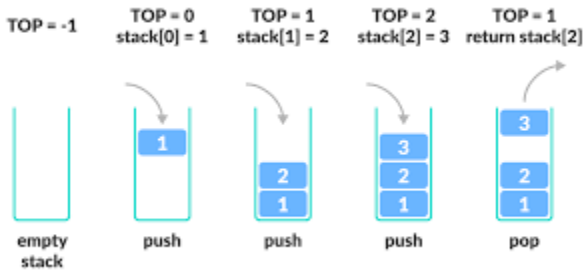


There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen. The plate which is at the **top** is the **first one to be removed**, i.e. the plate which has been placed at the **bottommost position** remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO(Last In First Out)/FILO(First In Last Out) order.

A stack may be **fixed** in size or may have **dynamic** implementation where the size is allowed to change. In the case of bounded capacity stacks, trying to add an element to an already full stack causes a **stack overflow** exception. Similarly, a condition where a pop operation tries to remove an element from an already empty stack is known as **stack underflow**.

Operations on stack

A stack represents a sequence of objects or elements in a linear data structure format. The stack consists of a **bounded bottom** and all the operations are carried out on the **top position**. Whenever an element is added to the stack by the **push** operation, the **top** value is **incremented** by one, and when an element is **popped** out from the stack, the **top** value is **decremented** by one. A pointer to the top



position of the stack is also known as the **stack pointer**.

A stack is considered to be a restricted data structure as only a limited number of operations are allowed. Besides the push and pop operations, certain implementations may allow for advanced operations such as:

- Peek — View the topmost item in the stack.
- Duplicate — Copy the top item's value into a variable and push it back into the stack.
- Swap — Swap the two topmost items in the stack.
- Rotate — Move the topmost elements in the stack as specified by a number or move in a rotating fashion.

Software implementations of the stack concept are done using **arrays** and **linked lists** where the **top position** is tracked using a **variable** or **header pointer** respectively. Many programming languages provide built-in features to support stack implementation.

Hardware stacks are implemented for the purpose of memory allocation and access using a fixed origin and size. Stack registers are used to store the value of the stack pointer.