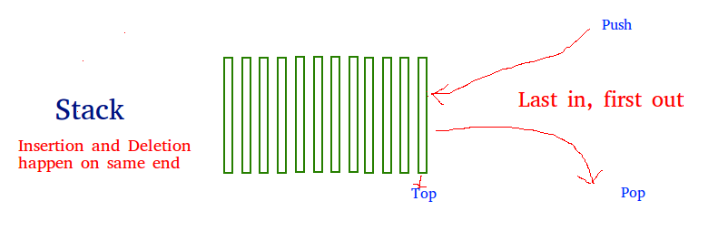
**Q - What is a Stack?**

A - 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**.

**The diagram below visualizes a Stack-**



**Q- What functions should a Stack have?**

**A- empty() –** Returns whether the stack is empty – Time Complexity : O(1)

* **size() –** Returns the size of the stack – Time Complexity : O(1)
* **peek() –** Returns a reference to the top most element of the stack – Time Complexity : O(1)
* **push(g) –** Adds the element ‘g’ at the top of the stack – Time Complexity : O(1)
* **pop() –** Deletes the top most element of the stack – Time Complexity : O(1)
* More functions can be added accordingly.

**Q- How to Implement a Stack?**

**A- There are three ways to implement a Stack.**

* list
* collections.deque
* queue.LifoQueue

**Q. Which implementation should we use?**

In general, you should use a **deque** if you’re not using **threading**. If you are using threading, then you should use a **LifoQueue** unless you’ve measured your performance and found that a **small boost** in speed for pushing and popping will make enough difference to warrant the maintenance risks.

**list may be familiar, but it should be avoided because it can potentially have memory reallocation issues.** The interfaces for deque and list are identical, and deque doesn’t have these issues, which makes **deque the best choice for your non-threaded Python stack.**

**#** We will be learning implementations using **deque and list** as, while implementation using list is the easiest which can be used for easier problems, for tougher problems deque implementations can be used.

**# Quick Facts - Deque is preferred over 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.**

**Implementation using deque -**

from collections import deque

**class Stack:**

**def \_\_init\_\_(self):**

**self.stack = deque()**

**def push(self,item):**

**self.stack.append(item)**

**def peek(self):**

**return self.stack[-1]**

**def pop(self):**

**return self.stack.pop()**

**def display(self):**

**print (self.stack)**

**def size(self):**

**print (len(self.stack))**

**def is\_empty(self):**

**if len(self.stack) == 0:**

**return True**

**else: return False**

**Implementation using list:**

**class Stack:**

**def \_\_init\_\_(self):**

**self.stack = []**

**def push(self,item):**

**self.stack.append(item)**

**def peek(self):**

**return self.stack[-1]**

**def pop(self):**

**return self.stack.pop()**

**def display(self):**

**print (self.stack)**

**def size(self):**

**print (len(self.stack))**

**def is\_empty(self):**

**if len(self.stack) == 0:**

**return True**

**else: return False**

**Useful links -**

[**https://www.geeksforgeeks.org/stack-in-python/**](https://www.geeksforgeeks.org/stack-in-python/)

**https://realpython.com/how-to-implement-python-stack/**