



What is a Stack?

An introduction to the Stack data structure, its uses and functionality.

We'll cover the following



- Introduction
- What are Stacks Used for?
- How do Stacks work?

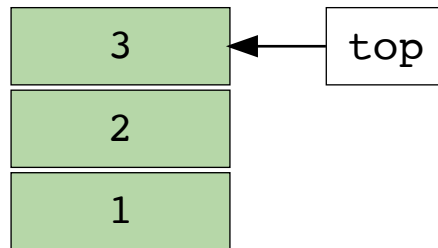
Introduction#

We are all familiar with the famous *Undo* option which exists in almost all popular applications. Ever wondered how that works? Well, you store the previous states of your work (which are limited to a specific number) in the memory in such an order that the last one appears first. You can't really do this with simple lists very efficiently for reasons we will explore in the coming chapters. So this is where the 'Stack' data structure comes in handy.

Stacks follow the *Last in First Out (LIFO)* ordering. This means that the last element added is the element on the top and the first element added is at the bottom.

A real-life example of Stack could be a stack of books. So, in order to get the book that's somewhere in the middle, you will have to remove all the books placed at the top of it. Also, the last book you added to the stack of books is at the top!





What are Stacks Used for?

Despite the simple implementation stacks can be used to solve very complex problems!

There are many famous algorithms such as *Depth First Search* and the *Expression Evaluation Algorithm*, which harness the functionality of Stacks. Stacks are used:




- To backtrack to the previous task/state, for example, in recursive code
- To store a partially completed task, for example, when you are exploring two different paths on a *Graph* from a point while figuring out the smallest path to the target.

How do Stacks work?#

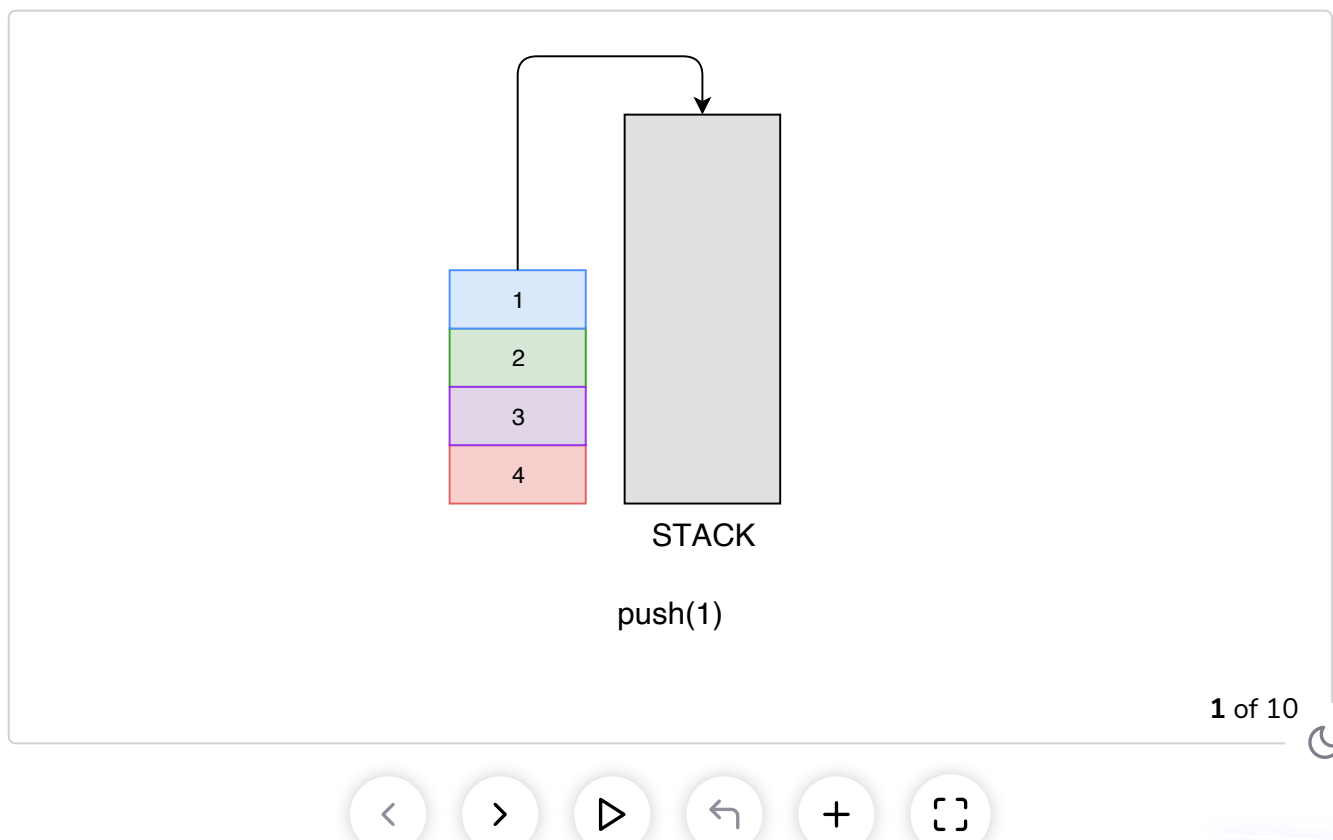
Stacks can be implemented in many ways, but a typical Stack must offer the following functionalities:

Function	What does it do?
<code>push(element)</code>	Inserts an element at the top



Function	What does   
<code>pop()</code>	Removes an element from the top and returns it
<code>peek()</code>	Returns the top element of the stack
<code>IsEmpty()</code>	Returns a boolean 1 if the stack is empty
<code>size()</code>	Returns the size of the stack

The following animation is a high-level demonstration of the `push(element)` and the `pop()` functions.





Now let's see how to implement them in Python. See you in the next lesson!

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. [See how](#) ⓘ



← Back

Next →

Linked Lists Quiz: Test your understand...

Stack (Implementation)

✓ Completed



Report an Issue

