

the last thing pushed in is always the first thing that comes back out

LIFO: Last In, First Out

- a stack is a linear structure where every insertion and every removal happens at the **same end**, called the top
- push adds a new element on top of the stack, pop removes whatever element is currently on top
- the element pushed most recently is always the one popped first, that ordering is what LIFO means
- peek looks at the top element without removing it, isEmpty checks whether the stack has anything left

Stack operations



one stack, four operations branching from it, every one of them working only at the top

Implementing a stack with a Python list

*

- a plain Python list already behaves like a stack if only the **last** index is ever touched
- stack.append(x) pushes x onto the top, it lands at the end of the list
- stack.pop() removes and returns the last element of the list, exactly the top of the stack
- stack[-1] peeks at the top without removing it, len(stack) equal to 0 means the stack is empty

Tracing three pushes and a pop

Q. stack = [] then push 5, push 8, push 2, then pop once. What does the stack hold, and what does pop return?

- push 5: stack becomes 5, one element, top is 5
- push 8: stack becomes 5 8, top moves to 8
- push 2: stack becomes 5 8 2, top moves to 2
- pop(): removes and returns 2, the element that was pushed last

5 8 2

the stack after three pushes, 5 then 8 then 2, the top highlighted at the right end

Answer

pop() returns 2, the stack is left holding 5 8, top is now 8

Overflow and underflow

- overflow is trying to push onto a stack that is already full, it only applies to a fixed-size, array-based stack
- a Python list grows on its own, so overflow there only shows up as running out of memory, an unusual case
- underflow is trying to pop from a stack that has nothing in it, that is the case worth checking for every time
- calling pop() on an empty Python list raises an error, so isEmpty should be checked before every pop