Validating Balanced Brackets with a Stack

To check if brackets are balanced in an expression like (2+3)*(4-1), we use a stack because it follows Last-In, First-Out (LIFO)—perfect for matching opening and closing brackets.

Step-by-Step Stack Validation For Expression (2+3)*(4-1):

- Initialize an empty stack.
- Scan each character:
- '(' → push to stack → stack: ['(']
- '2' → ignore
- '+' → ignore
- '3' → ignore
- ')' → pop from stack → stack: []
- '*' → ignore
- '(' → push to stack → stack: ['(']
- '4' → ignore
- '-' → ignore
- '1' → ignore
- ')' → pop from stack → stack: []
- Final check: stack is empty → brackets are balanced

Why it works:

- Every '(' is pushed.
- Every ')' pops the last '('.
- If the stack is empty at the end and no mismatches occurred, the brackets are balanced.

Why Stacks Power Undo Features:

Stacks are ideal for undo because they naturally reverse actions in the exact opposite order they occurred.

Comparison: Stack vs Queue

FEATURE	STACK (UNDO)	Queue (FIFO)
Order	Last-in, First-out	First-in, First-out
Use case	Undo last action	Process tasks in their arrival order
Example	Ctrl+Z in text editor	Print jobs in a printer queue

Why Stacks Win for Undo:

- You want to undo the most recent action first.

- Stacks let you pop the last change instantly.

- Queues would undo the oldest action first—not helpful when fixing a recent mistake.