Min Stack

Difficulty	Medium
Category	Stack
Question	https://leetcode.com/problems/min-stack/
Solution	https://www.youtube.com/watch?v=qkLl7nAwDPo
Status	Done

Question

> Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.
>
> Implement the `MinStack` class:
>
> - `MinStack()` initializes the stack object.
> - `void push(int val)` pushes the element `val` onto the stack.
> - `void pop()` removes the element on the top of the stack.
> - `int top()` gets the top element of the stack.
> - `int getMin()` retrieves the minimum element in the stack.
>
> You must implement a solution with `O(1)` time complexity for each function.

Example

Example 1:

```
Input
["MinStack","push","push","push","getMin","pop","top","getMin"]
[[],[-2],[0],[-3],[],[],[],[]]

Output
[null,null,null,null,-3,null,0,-2]

Explanation
MinStack minStack = new MinStack();
```

```
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); // return -3
minStack.pop();
minStack.top();    // return 0
minStack.getMin(); // return -2
```

Idea

💡 Use two Stacks: One for the normal implementation, another one to hold the minimum of the stack at the top always

Solution

```
lass MinStack:

    def __init__(self):
        # Initialize two empty lists: stack and minStack
        self.stack = []     # This is the main stack that holds the elements.
        self.minStack = []  # This stack keeps track of the minimum elements.


    def push(self, val: int) -> None:
        self.stack.append(val)

        # Calculate the minimum of 'val' and the current minimum (if it exists) and push it onto minStack.
        val = min(val, self.minStack[-1] if self.minStack else val)
        self.minStack.append(val)


    def pop(self) -> None:
        # Pop the top element from both the main stack and minStack to maintain consistency.
        self.stack.pop()
        self.minStack.pop()


    def top(self) -> int:
        # Return the top element of the main stack without removing it.
        return self.stack[-1]

    def getMin(self) -> int:
        # Return the top element of minStack, which represents the current minimum element in the stack.
        return self.minStack[-1]
```

💡 The **Time complexity** of the provided code is **O(1).**

💡 The **Space complexity** of the code is **O(n)** as well.