Minimum Stack

My initial code was

```
class MinStack:

    def __init__(self):
        self.min_stack = []
        self.stack = []

    def push(self, val: int) -> None:
        if self.min_stack == []:
           self.stack.append(val)
           self.min_stack.append(val)


        elif val < self.min_stack[-1]:
         self.stack.append(val)
         self.min_stack.append(val)

        else:
           self.stack.append(val)





    def pop(self) -> None:
        if self.stack == []:
           return []

        if self.stack[-1] == self.min_stack[-1]:
            self.min_stack.pop()
        self.stack.pop()

    def top(self) -> int:
        if self.stack == []:
           return []
        else:

         return self.stack[-1]

    def getMin(self) -> int:
        if self.min_stack == []:
           return []
        else:
         return self.min_stack[-1]
```

This was the code that worked

```
class MinStack:

    def __init__(self):
        self.stack = []      # Main stack to store all values
        self.min_stack = []  # Auxiliary stack to store the min

    def push(self, val: int) -> None:
        # Push value onto the main stack
        self.stack.append(val)
        # Push value onto the min_stack if it's the first value
        if not self.min_stack or val <= self.min_stack[-1]:
            self.min_stack.append(val)

    def pop(self) -> None:
        # Check if the stack is empty before attempting to pop
        if not self.stack:
            return
        # Pop from min_stack only if the value being popped from
        if self.stack[-1] == self.min_stack[-1]:
            self.min_stack.pop()
        self.stack.pop()

    def top(self) -> int:
        # Return the top of the stack if not empty, otherwise re
        if not self.stack:
            return 0
        return self.stack[-1]

    def getMin(self) -> int:
        # Return the top of the min_stack if not empty, otherwis
        if not self.min_stack:
            return 0
        return self.min_stack[-1]
```

The main issue was that it did not handle duplicate minimums, so a ≤ is added to account for the minimum stack.append