Challenge 9: min( ) Function Using a Stack

Using your knowledge, create an efficient min() function using a stack.

We'll cover the following

Problem Statement#

You have to implement the `MinStack` class which will have a `min()` function. Whenever `min()` is called, the minimum value of the stack is returned in *O(1)* time. The element is not popped from the stack. Its value is simply returned.

Input#

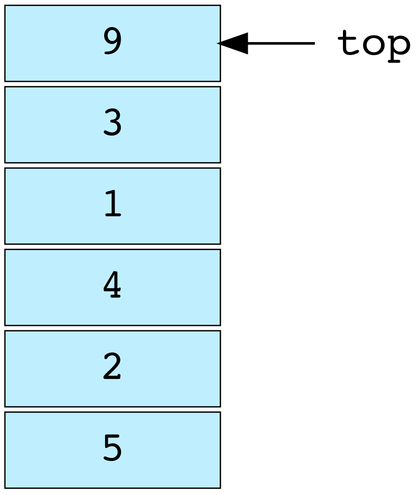
`min()` operates on an object of `MinStack` and doesn't take any input

Output#

Returns minimum number in `O(1)` time

Sample Output#

```
## min_stack = [9, 3, 1, 4, 2, 5]
## min_stack.min()
Result = 1
```



Minimum Value = 1

Coding Exercise#

Design a step-by-step algorithm first before jumping on to the implementation.

The original `MyStack` class and all its members are available to you. You can use it to create a stack in the constructor for `min_stack` if you want. On the other hand, you may write your own constructor and helper functions.

`pop()`, `push()`, and `min()` is what you really need to implement by yourself as their functionality will differ from the original `MyStack` class.

Keep in mind that the `min()` function should work in *O(1)* and should not pop the minimum element out of the stack. It simply returns the minimum value.

The default skeleton code written below will not run on any il ce is incomplete, but don’t let that scare you.

If you get stuck, you can always refer to the solution review. You’re given the basic structure of this problem. Try to solve the rest of the puzzle.

Good luck!

main.py

Stack.py

```
# Create Stack => stack = myStack(5); where 5 is size of stack
# Create Queue => queue = myQueue(5); where 5 is size of queue
# Stack Functions => isEmpty(), isFull(), top()
# Queue Functions => enqueue(int),dequeue(),isEmpty(),getSize()

from Stack import MyStack

class MinStack:
    # Constructor
    def __init__(self):
        # Write your code here
        self.stack = []
        self.min_val = float('inf')

    def pop(self):
        # Write your code here
        pop_val = self.stack.pop()
        if pop_val < self.min_val:
            self.min_val = min(self.stakc)
        return pop_val

    # Pushes value into new stack
    def push(self, value):
        # Write your code here
        self.stack.append(value)
        if value < self.min_val:
            self.min_val = value
        return True

    # Returns minimum value from new stack in constant time
    def min(self):
        # Write your code here
        return self.min_val
    # Write any helper functions here
```

Back

Solution Review: Check Balanced Pare...

Next

Solution Review: min( ) Function Usin...

Completed

Report an Issue