Solution Review: min( ) Function Using a Stack

This review provides a detailed analysis of a solution to the 'min() Function Using a Stack' challenge.

We'll cover the following

Solution: A Two Stack Class

main.py

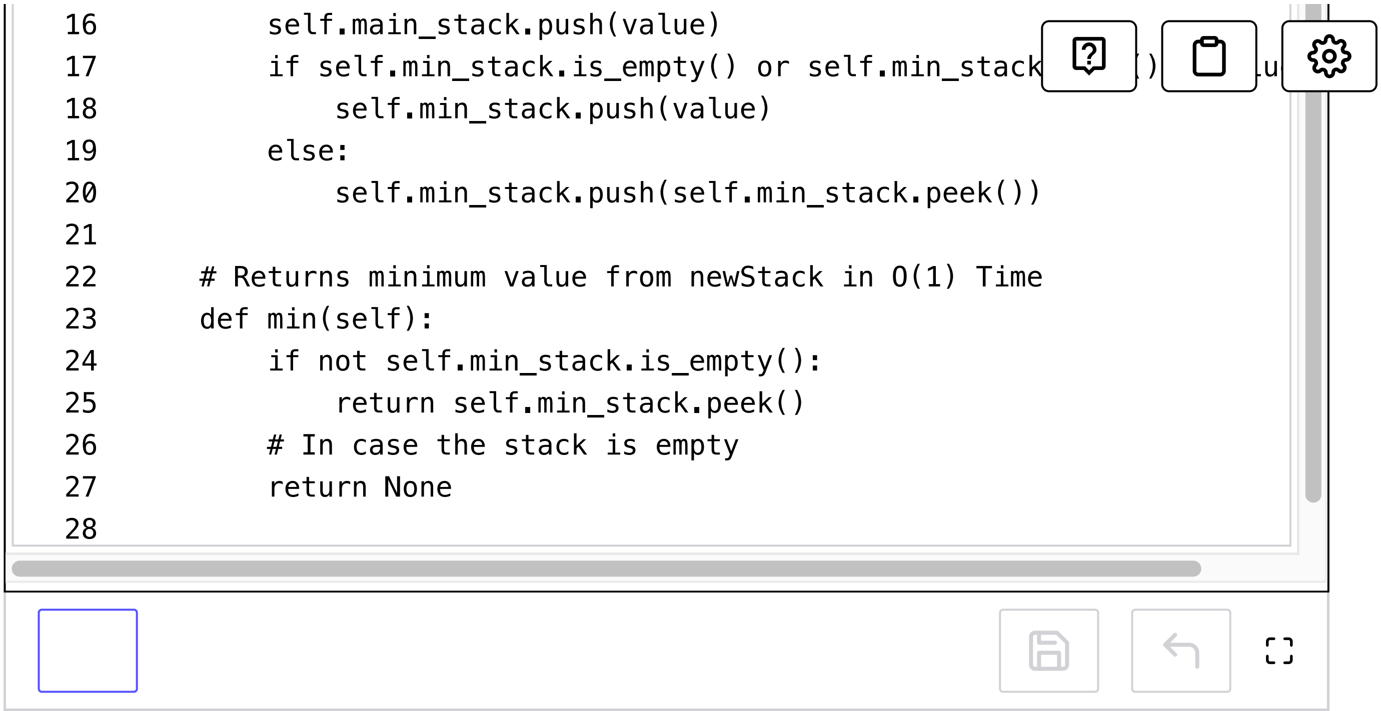
Stack.py

```
from Stack import MyStack

class MinStack:
    # Constructor
    def __init__(self):
        self.min_stack = MyStack()
        self.main_stack = MyStack()

    # Removes and returns value from min_stack
    def pop(self):
        self.min_stack.pop()
        return self.main_stack.pop()

    # Pushes values into min_stack
    def push(self, value):
```



```
        self.main_stack.push(value)
        if self.min_stack.is_empty() or self.min_stack
            self.min_stack.push(value)
        else:
            self.min_stack.push(self.min_stack.peek())

    # Returns minimum value from newStack in O(1) Time
    def min(self):
        if not self.min_stack.is_empty():
            return self.min_stack.peek()
        # In case the stack is empty
        return None

```

This is a smart solution for obtaining the minimum value in a stack, yet it isn’t a very tricky one.

The whole implementation relies on the existence of two stacks, `min_stack` and `main_stack`.

`main_stack` holds the actual stack with all the elements, whereas `min_stack` is a stack whose **top** always contains the current minimum value in the stack.

How does it do this? The answer is in the `push()` function. Whenever `push()` is called, `main_stack` simply inserts it at the top. However, `min_stack` checks the value being pushed. If `min_stack` is empty, this value is pushed into it and becomes the current minimum. If `min_stack` already has elements in it, the value is compared with the stack’s top element. The element is inserted if it is smaller than the top element; otherwise, we insert the top element again.

The `pop()` function pops off from the `main_stack` and `min_stack` as usual.

Due to all these safeguards we’ve put in place, the `min()` fun o ee to return the value at the top of `min_stack`.

Time Complexity#

Our goal was to create a stack that returns the minimum value in **constant** time. As we can see in the algorithm above, the `min()` function truly works in *O(1)*.

← Back

Challenge 9: min( ) Function Using a S...

Next →

Stack/Queue Quiz: Test your understa...

☑ Mark as Completed

Report an Issue