Solution Review: Evaluate Postfix Expression Using a Stack

This review provides a detailed analysis of a solution to the 'Evaluate Postfix Expression Using Stack' challenge.

We'll cover the following

Solution: Numbers as Stack Elements

main.py

Stack.py

```
from Stack import MyStack

def evaluate_post_fix(exp):
    stack = MyStack()
    try:
        for char in exp:
            if char.isdigit():
                # Push numbers in stack
                stack.push(char)
            else:
                # use top two numbers and evaluate
                right = stack.pop()
                left = stack.pop()
```

```
13                  left = stack.pop()
14                  ''' Using Python's eval () method that          e
15                  evaluates it and returns an integer ''
16                  stack.push(str(eval(left + char + right)))
17          # final answer should be a number
18          return int(float(stack.pop()))
19      except TypeError:
20          return "Invalid Sequence"
21
22  if __name__ == "__main__" :
23      print("Result of expression (921*-8-4+) : " + str(evaluate_post_f
24      print("Result of expression (921*-8--4+) : " + str(evaluate_post_
```

We check each **character** of the string from left to right. If we find a `digit`, it is pushed into the stack.

If we find an operand, we `pop` two elements from the stack (there have to be at least two present or else this postfix expression is invalid) and solve the expression. The resulting value is pushed back into the stack.

The process continues until we reach the end of the string.

> **Note:** We have used a `try except` block. If the expression is correct, the result is returned and the `try` block is executed. Otherwise, it is caught in the `except` block.

Time Complexity#

Since we traverse the string of **n** characters once, the time complexity for this algorithm is *O(n)*.

Back

Challenge 6: Evaluate Postfix Expressi...

Next

Challenge 7: Next Greater Element Usi...

Completed

Report an Issue