Solution Review: Check Balanced Parentheses using Stack

This lesson provides a detailed review of a solution to the 'Check Balanced Parentheses using a Stack' challenge.

We'll cover the following

Solution: A Stack of Characters

main.py

Stack.py

```
    ''' Iterate through the string exp.
    For each opening parentheses, push it into stack
    For every closing parentheses check
    for its opening parentheses in stack
    If you can't find the opening parentheses
    for any closing one then returns false.
    and after complete traversal of string exp,
    if there's any opening parentheses left
    in stack then also return false.
    At the end return true if you haven't
    encountered any of the above false conditions '''

from Stack import MyStack

def is_balanced(exp):
```

```
    closing = ['}', ')', ']']
    stack = MyStack()
    for character in exp:
        if character in closing:
            if stack.is_empty():
                return False
            top_element = stack.pop()
            if character is '}' and top_element is not '{':
                return False
            if character is ')' and top_element is not '(':
                return False
            if character is ']' and top_element is not '[':
                return False
```

This is a simple algorithm. We iterate over the string, one character at a time. Whenever we find a **closing** parenthesis, we can deduce that the string is unbalanced based on two conditions:

1. The stack is *empty*.
2. The top element in the stack is not an *opening* parenthesis of the same type.

If any of these conditions are `True`, we return `False`.

If a parenthesis in the string is an opening parenthesis, it is simply pushed into the stack. If all the parentheses are balanced, the stack should be empty by the end because we pop every opening parenthesis once its closing parenthesis is found. In the end, if stack is not empty, we return `False`.

Time Complexity#

We traverse the string `exp` once. So, the time complexity is *O(n)* where **n** is the length of the string.

Back

Challenge 8: Check Balanced Parenth...

Next

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

Completed

Report an Issue