CSC148 - Balancing Parentheses

We are writing client code and need a function (outside the class) to determine whether the parentheses in an expression are balanced: opening and closing parentheses match and are properly nested inside each other.

1. For four examples, we'll give you a string one character at a time. Your job is to determine whether the string has balanced parentheses or not. *Don't just write down every character without thinking!* Instead, use a stack to keep track of the minimum amount of information you need to solve the problem.

<table>
<thead>
<tr>
<th>Expression 1:</th>
<th>Expression 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack</td>
<td>Stack</td>
</tr>
<tr>
<td>Were the parentheses balanced?</td>
<td>Were the parentheses balanced?</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression 3:</th>
<th>Expression 4:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack</td>
<td>Stack</td>
</tr>
<tr>
<td>Were the parentheses balanced?</td>
<td>Were the parentheses balanced?</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
2. What general strategy will work in all cases:
   (a) What will you do with each character as you receive it?

   (b) At the end, how will you know whether the parentheses were balanced?

3. Now implement the function.

```python
def is_balanced(line: str) -> bool:
    '''Return whether <line> contains balanced parentheses.

    Ignore square and curly brackets.
    
    >>> is_balanced('a = (3 + b)')
    True
    >>> is_balanced('a = (3 + b)')  # Note that the two ']'s don't matter.
    False
    >>> is_balanced('1 + 2(x - y)')  # Note that the '(' doesn't matter.
    True
    >>> is_balanced('3 - (z')
    False
    '''
    ...for c in line:
        if c == '(':
            push onto stack
        elif c == ')':
            pop from stack  # if empty, extra closing paren
        check if stack is empty
        (after all characters processed)  # if not empty, extra open paren
```

4. How would you generalize this code to balance round, square, and curly brackets?