Balanced Parentheses Check - SOLUTION

Problem Statement

Given a string of opening and closing parentheses, check whether it’s balanced. We have 3 types of parentheses: round brackets: (), square brackets: [], and curly brackets: {}. Assume that the string doesn’t contain any other character than these, no spaces words or numbers. As a reminder, balanced parentheses require every opening parenthesis to be closed in the reverse order opened. For example ‘([])’ is balanced but ‘([)]’ is not.

You can assume the input string has no spaces.

Solution

This is a very common interview question and is one of the main ways to check your knowledge of using Stacks! We will start our solution logic as such:

First we will scan the string from left to right, and every time we see an opening parenthesis we push it to a stack, because we want the last opening parenthesis to be closed first. (Remember the FILO structure of a stack!)

Then, when we see a closing parenthesis we check whether the last opened one is the corresponding closing match, by popping an element from the stack. If it’s a valid match, then we proceed forward, if not return false.

Or if the stack is empty we also return false, because there’s no opening parenthesis associated with this closing one. In the end, we also check whether the stack is empty. If so, we return true, otherwise return false because there were some opened parenthesis that were not closed.

Here's an example solution:

In [1]:
```python
def balance_check(s):
    
    # Check is even number of brackets
    if len(s)%2 != 0:
        return False
    
    # Set of opening brackets
    opening = set('([{') 
    
    # Matching Pairs
    matches = set([ ('(',')'), ('[',']'), ('{','}') ]) 
    
    # Use a list as a "Stack"
    stack = []
    
    # Check every parenthesis in string
    for paren in s:
        
        # If its an opening, append it to list
        if paren in opening:
            stack.append(paren)

        else:

            # Check that there are parentheses in Stack
            if len(stack) == 0:
                return False

            # Check the last open parenthesis
            last_open = stack.pop()

            # Check if it has a closing match
            if (last_open,paren) not in matches:
                return False

    return len(stack) == 0
```

In [2]:
```
balance_check('[]')
```

Out[2]:
```
True
```

In [3]:
```
balance_check('[](){([[[]]])}')
```

Out[3]:
```
True
```

In [4]:
```
balance_check('()(){]}')
```

Out[4]:
```
False
```

Test Your Solution

In [5]:
```
"""
RUN THIS CELL TO TEST YOUR SOLUTION
"""
from nose.tools import assert_equal

class TestBalanceCheck(object):

    def test(self,sol):
        assert_equal(sol('[](){([[[]]])}('),False)
        assert_equal(sol('[{{{(())}}}]((()))'),True)
        assert_equal(sol('[[[]])]'),False)
        print('ALL TEST CASES PASSED')

# Run Tests

t = TestBalanceCheck()
t.test(balance_check)
```

```
ALL TEST CASES PASSED
```

Good Job!