Clarusway

JS-CC-009 : Bracket Validator

Let's say:

- '(', '{', '[' are called "openers."
- ')', '}', ']' are called "closers."

Write an efficient function that tells us whether or not an input string's openers and closers are properly nested.

> **Examples:**
>
> - "`{ [ ] ( ) }`" should return true
> - "`{ [ ( ] ) }`" should return false
> - "`{ [ }`" should return false

- Simply making sure each opener has a corresponding closer is not enough—we must also confirm that they are correctly ordered.
- For example, "`{ [ ( ] ) }`" should return false, even though each opener can be matched to a closer.

Learning Outcomes

At the end of the this coding challenge, students will be able to;

- Analyze a problem, identify and apply programming knowledge for appropriate solution.
- Demonstrate their knowledge of algorithmic design principles by using JavaScript and Python effectively.

Problem Statement

- Write a function that takes series of brackets and returns `true` or `false` depending on the rules above.

☺ **Happy Coding** ✍

JavaScript

```
function isValid(code) {
    // write your code here

}

/* *** Tests *** */

let desc = "valid short code";
```

```
assertEqual(isValid("()"), true, desc);

desc = "valid longer code";
assertEqual(isValid("([]{[]})[]{{}()}"), true, desc);

desc = "mismatched opener and closer";
assertEqual(isValid("([][]}"), false, desc);

desc = "missing closer";
assertEqual(isValid("[[]()"), false, desc);

desc = "extra closer";
assertEqual(isValid("[[]]())"), false, desc);

desc = "empty string";
assertEqual(isValid(""), true, desc);

function assertEqual(a, b, desc) {
  if (a === b) {
    console.log(`${desc} ... PASS`);
  } else {
    console.log(`${desc} ... FAIL: ${a} != ${b}`);
  }
}
```

Python

```
import unittest

def is_valid(code):
    # write your code here
    pass

# ***  Tests ***

class Test(unittest.TestCase):
    def test_valid_short_code(self):
        result = is_valid("()")
        self.assertTrue(result)

    def test_valid_longer_code(self):
        result = is_valid("([]{[]})[]{{}()}")
        self.assertTrue(result)

    def test_interleaved_openers_and_closers(self):
        result = is_valid("([)]")
        self.assertFalse(result)

    def test_mismatched_opener_and_closer(self):
        result = is_valid("([][]}")
```

```
        self.assertFalse(result)

    def test_missing_closer(self):
        result = is_valid("[[](")
        self.assertFalse(result)

    def test_extra_closer(self):
        result = is_valid("[[]]())")
        self.assertFalse(result)

    def test_empty_string(self):
        result = is_valid("")
        self.assertTrue(result)

unittest.main(verbosity=2)
```