Valid Parentheses

Difficulty	Easy
Category	Stack
Question	https://leetcode.com/problems/valid-parentheses/
Solution	https://youtu.be/WTzjTskDFMg
Status	Done

Question

> Given a string `s` containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.
>
> An input string is valid if:
>
> 1. Open brackets must be closed by the same type of brackets.
> 2. Open brackets must be closed in the correct order.
> 3. Every close bracket has a corresponding open bracket of the same type.

Example

Example 1:

```
Input: s = "()"
Output: true
```

Example 2:

```
Input: s = "()[]{}"
Output: true
```

Example 3:

```
Input: s = "(]"
Output: false
```

Idea

💡 Use stack to keep track of pushed and popped char. If they match in order, it's valid

Solution

```
class Solution:
    def isValid(self, s: str) -> bool:
```

```
        # Define a mapping of closing brackets to their corresponding opening brackets.
        bracket_map = {")": "(", "]": "[", "}": "{"}

        # Create an empty stack to store opening brackets.
        stack = []

        # Iterate through each character in the input string.
        for char in s:
            # If the character is not in the bracket_map, it means it's an opening bracket.
            if char not in bracket_map:
                stack.append(char)
                continue

            # If the character is in the bracket_map, it's a closing bracket.
            # Check if the stack is empty or if the top of the stack does not match the current closing bracket.
            if not stack or stack[-1] != bracket_map[char]:
                return False

            # Pop the corresponding opening bracket from the stack.
            stack.pop()

        # After iterating through the string, check if the stack is empty.
        # If the stack is empty, it means all opening brackets have been matched with their corresponding closing brackets, and the string
        return not stack
```

💡 The **Time complexity** of the provided code is **O(n)**, where 'n' is the length of the input string 's'.

💡 The **Space complexity** of the code is **O(n)** as well. This is because, in the worst case, the entire input string might consist of only opening brackets, and they would all be pushed onto the stack