Valid Parentheses

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 1:

Input: s = "()"

Output: true

Example 2:

Input: s = "()[]{}"

Output: true

Example 3:

Input: s = "(]"

Output: false

Example 4:

Input: s = "([])"

Output: true

In the first step, let's use what the question gives us to define the parenthesis we will use and define a stack.

```
parentheses = {'(':')', '{':'}','[':']'}
        stack = []
```

Pay attention to the split above “:” this operator this is defining that after “(” we want to see “)” etc.

Next we have to loop through the string and any we find a parethesis we will append it to our stack

```
for i in s:
            # check if the character is an opening bracket
            if i in parentheses:
                stack.append(i)
```

next we do another conditional block to check if the list is empty or too see if the last param is equal to the first param if the list is empty or the last param is not equal to the first one we return false.

```
# else return false if the stack is empty or the closing bracket does not match the
            elif len(stack) == 0 or parentheses[stack.pop()] != i:
                return False
        return len(stack) == 0
```

Play around with the different x, y and z values to understand the problem the interviewee is asking you to solve and to understand the code more

Complete solution

```
class Solution(object):
    def isValid(self, s):
        # define params and stack
        parentheses = {'(':')', '{':'}','[':']'}
        stack = []
        # iterate through the string
        for i in s:
            # check if the character is an opening bracket
            if i in parentheses:
                stack.append(i)
            # else return false if the stack is empty or the closing bracket does not ma
            elif len(stack) == 0 or parentheses[stack.pop()] != i:
                return False
        return len(stack) == 0

if __name__ == '__main__':
    s = Solution()

    x = "()"
    y = "()[]{}"
    z = "(]"

    print(s.isValid(x)) # True
    print(s.isValid(y)) # True
    print(s.isValid(z)) # False
```