3. Longest Substring Without Repeating Characters

Problem Statement

Given a string `s`, find the length of the **longest** **substring** without repeating characters.

Example 1:

```
Input: s = "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.
```

Example 2:

```
Input: s = "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.
```

Example 3:

```
Input: s = "pwwkew"
Output: 3
Explanation: The answer is "wke", with the length of 3.
Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.
```

Approach:

1. **Approach 1 - Set**

   1. We use a set (`charSet`) to keep track of unique characters in the current substring.
   2. We maintain two pointers, `left` and `right` , to represent the boundaries of the current substring.
   3. The `maxLength` variable keeps track of the length of the longest substring encountered so far.
   4. We iterate through the string using the `right` pointer.
   5. If the current character is not in the set (`charSet`), it means we have a new unique character.
   6. We insert the character into the set and update the `maxLength` if necessary.
   7. If the character is already present in the set, it indicates a repeating character within the current substring.
   8. In this case, we move the `left` pointer forward, removing characters from the set until the repeating character is no longer present.
   9. We insert the current character into the set and continue the iteration.
   10. Finally, we return the `maxLength` as the length of the longest substring without repeating characters.

Python Code:

1.
```python
class Solution:
    def lengthOfLongestSubstring(self, s: str) -> int:

# Input: s = "abcabcbb"
# Output: 3

        n = len(s)
        maxL = 0
        charS = set()
        left = 0

        for right in range(n):
            if s[right] not in charS:
                charS.add(s[right])
                maxL = max(maxL,right-left+1)

            else:
                while s[right] in charS:
                    charS.remove(s[left])
                    left += 1
                charS.add(s[right])

        return maxL
```