Longest Repeating Subsequence

Given a string, find the length of the longest repeating subsequence such that the two subsequences don't have same string character at the same position, i.e., any i'th character in the two subsequences shouldn't have the same index in the original string.

Example 1:

```
Input: str = "axxxy"
Output: 2
Explanation: The longest repeating subsequenece
is "xx".
```

Example 2:

```
Input: str = "aab"
output: 1
Explanation: The longest reapting subsequenece
is "a".
```

Your Task:
You don't need to read or print anything. Your task is to complete the function **LongestRepeatingSubsequence()** which takes str as input parameter and returns the length of the longest repeating subsequnece.

Expected Time Complexity: $O(n^2)$

Expected Space Complexity: $O(n^2)$

Constraints:

$1 <= |str| <= 10^3$

```python
class Solution:
    def LongestRepeatingSubsequence(self, str):
        # Code here
        str2 = str
        dp = [[0]*(len(str)+1) for _ in range(len(str)+1)]
        for i in range(1,len(str)+1):
            for j in range(1,len(str)+1):
                if str[i-1]==str2[j-1] and i-1!=j-1:
                    dp[i][j] = 1 + dp[i-1][j-1]
                else:
```

```
                    dp[i][j] = max(dp[i-1][j],dp[i][j-1])
        return dp[-1][-1]
```