5. Longest Palindromic Substring

Medium

Given a string s, return the longest palindromic substring in s.

```
from typing import List
class Solution:
   def longestPalindrome(self, s: str) -> str:
       result = ""
        for i in range(len(s)):
           1, r = i, i
            while l \ge 0 and r < len(s) and s[l] == s[r]:
                if (r - l + 1) > len(result):
                    result = s[1:r + 1]
                r += 1
           1, r = i, i + 1
           while l \ge 0 and r < len(s) and s[l] == s[r]:
                if (r - l + 1) > len(result):
                   result = s[l:r + 1]
                1 -= 1
               r += 1
       return result
```



Explanation: "aba" is also a valid answer.

Example 1:

Example 2:

Output: "bb"

Input: s = "cbbd"

Input: s = "babad" Output: "bab"

```
from typing import List

class Solution:
    def longestPalindrome(self, s: str) -> str:
        result = ""

    for i in range(len(s)):
        for l, r in ((i,i), (i,i+1)):
            while l >= 0 and r < len(s) and s[l] == s[r]:
            if (r - l + 1) > len(result):
                result = s[l:r + 1]
            l -= 1
            r += 1

    return result
```

"abbad" 5

"abbad" 5for i in range(len(s)): 0 1 2 3 4

first, check when l and r point to same index l, r = i, i

1 and r pointers are in bound, and s[l] and s[r] are palindromes

r += 1

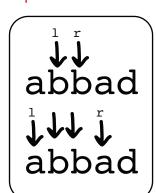
while $l \ge 0$ and r < len(s) and s[l] == s[r]:

abbad

next, check when 1 and r point to adjacent index 1, r = i, i + 1

1 and r pointers are in bound, and s[l] and s[r] are palindromes

```
while l \ge 0 and r < len(s) and s[l] == s[r]:
 (r-l+1) \text{ is the size of the substring,} 
 \text{if } (r-l+1) \text{ is greater than length of result,} 
 \text{then save the new result} 
 \text{if } (r-l+1) \ge len(result) \text{:} 
 \text{result} = s[l:r+1] 
 1 == 1
```



```
for i in range(len(s)):

((0,0),(0,1))
((1,1),(1,2))
((3,3),(3,4))
((2,2),(2,3))

es

for 1, r in ((i,i), (i,i+1)):

l and r pointers are in bound, and s[i] and s[r] are palindromes
while 1 >= 0 and r < len(s) and s[1] == s[r]:
(r-1+1) is the size of the substring,
if (r-1+1) is greater than length of result,
then save the new result

if (r - 1 + 1) > len(result):
result = s[1:r + 1]
```

checks for situation where

two adjacent letters are the same

1 -= 1 r += 1