5. Longest Palindromic Substring

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

Example 1:

```
Input: s = "babad"
Output: "bab"
Note: "aba" is also a valid answer.
```

Example 2:

```
Input: s = "cbbd"
Output: "bb"
```

Example 3:

```
Input: s = "a"
Output: "a"
```

Example 4:

```
Input: s = "ac"
Output: "a"
```

Constraints:

- `1 <= s.length <= 1000`
- `s` consist of only digits and English letters.

```
def longestPalindrome(self, s: str) -> str:
        longest_palindrom = ''
        dp = [[0]*len(s) for _ in range(len(s))]
        for i in range(len(s)):
            dp[i][i] = True
            longest_palindrom = s[i]
        for i in range(len(s)-1,-1,-1):
            for j in range(i+1,len(s)):
                if s[i] == s[j]:
                    if j-i ==1 or dp[i+1][j-1] is True:
                        dp[i][j] = True
```

```
                        if len(longest_palindrom) < len(s[i:j+1]):
                            longest_palindrom = s[i:j+1]

        return longest_palindrom
```

Correct but failed TLE

```
p = set(s)
        if len(p)==1:
            return s
        dp = [[False]*len(s) for _ in range(len(s))]
        idx = 0
        idy = 0
        length = 0
        for gap in range(len(s)):
            i = 0
            j = gap
            while j<len(s):
                if gap==0 or (gap==1 and s[i]==s[j]):
                    dp[i][j]=True
                else:
                    if s[i]==s[j] and dp[i+1][j-1]:
                        dp[i][j] = True
                if dp[i][j]:
                    if j-i+1>=length:
                        idx = i
                        idy = j
                i = i+1
                j = j+1
        # return dp[0][-1]
        return s[idx:idy+1]
```