## Expanding around the centres

The idea is to generate all even length and odd length palindromes and keep track of the longest palindrome seen so far.

- If the string length is less than or equal to 1 then return the string, as a single character is always palindrome.
- Run a loop where ‘i’ will be from 0 to ‘N’ - 1.
- To generate an odd length palindrome, fix ‘i’ as the center, and expand in both directions for getting possible longer palindromes. Odd length palindromes will have a character at the center.
- Similarly, for an even length palindrome, fix the center as ‘i’, ‘i’ + 1, and expand in both directions. An even palindrome will have a partition between ith char and (i+1)th char as the center.
- If the length of the current palindromic substring length is greater then update the starting length of the string and length of the palindromic substring.
- Return the substring of the string having a starting index as ‘START’ and of maximum length.

**For expanding :**

- For expanding around a center ‘i’ for odd length, initialize two variables ‘LEFT’ and ‘RIGHT’ to ‘i’ and go until ‘LEFT 'and ’RIGHT' are in range and STR[LEFT] == STR[RIGHT]. Decrement the ‘LEFT’ and increment the ‘RIGHT’.
- For expanding around a centre ‘i’ and ‘i’ + 1 for even length, initialise two variables ‘LEFT’ = ‘i’ and ‘RIGHT’ = ‘i’ + 1, and go until ‘LEFT’ and ‘RIGHT’ are in range and STR[LEFT] == STR[RIGHT]. Decrement the ‘LEFT’ and increment the ‘RIGHT’.
- Return the length of the palindromic substring.

```
/*
        This expands the interval left..right
        until the string str[left..right] remains a palindrome
*/
function expandAroundCenter(str, left, right)

        start = left
        end = right
        n = len(str)
```

```
        //   Expand the center.
        while (start >= 0 and end < n and str[start] == str[end])
                start -= 1
                end += 1

        return end - start - 1

/*
        Function to find the longest palindromic substring of the given string
*/
function longestPalinSubstring(str)
        n = len(str)
        if n < 1
                return ""

        start = 0
        end = 0
        for i in range(n)
                //   Longest odd length palindrome with center points as i.
                len1 = expandAroundCenter(str, i, i)

                //   Longest even length palindrome with center points as i and i + 1.
                len2 = expandAroundCenter(str, i, i+1)

                length = max(len1, len2)

                if (length > end - start + 1)
                        start = i - (length - 1) / 2
                        end = i + (length) / 2

        return str[start  end + 1]
```

**Time Complexity : O(N ^ 2)**, where N is the length of the given string.
As we are expanding twice for every index and expanding could take O(N) in the worst case.

**Space Complexity : O(1),** as we are not using any extra space.