Assignment 7

December 4, 2024

1 Assignment 7

1.0.1 Auriana Anderson

1.0.2 December 3, 2024

1.1 Question 1

A palindrome is a word, phrase, or sequence that is the same spelled forward as it is backwards. Write a function using a for-loop to determine if a string is a palindrome. Your function should only have one argument.

[8]:
```
def is_palindrome(string):
    for i in range(len(string)//2):
        if string[i] != string[-(i + 1)]:
            return False
        return True

print(is_palindrome("radar"))

print(is_palindrome("rabbit"))
```

```
True
False
```

1.2 Question 2

Write a function using a while-loop to determine if a string is a palindrome. Your function should only have one argument.

[12]:
```
def is_palindrome(string):
    left = 0
    right = len(string) - 1

    while left < right:
        if string[left] != string[right]:
            return False
        left += 1
        right -= 1
    return True
```

```
print(is_palindrome("radar"))

print(is_palindrome("rabbit"))
```

```
True
False
```

1.3 Question 3

Two Sum - Write a function named two_sum() Given a vector of integers nums and an integer target, return indices of the two numbers such that they add up to target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.Use defaultdict and hash maps/tables to complete this problem.

Example 1: Input: `nums = [2,7,11,15]`, target = 9 Output: `[0,1]` Explanation: Because `nums[0] + nums[1] == 9`, we return `[0, 1]`.

Example 2: Input: `nums = [3,2,4]`, `target = 6 Output: [1,2]`

Example 3: Input: `nums = [3,3]`, `target = 6 Output: [0,1]`

Constraints: `2 <= nums.length <= 104 -109 <= nums[i] <= 109 -109 <= target <= 109` Only one valid answer exists.

[24]:
```
from collections import defaultdict

def two_sums(nums, target):
    number_map = defaultdict(int)
    for i in range(len(nums)):
        num = nums[i]
        complement = target - num
        if complement in number_map:
            return [number_map[complement], i]
        number_map[num] = i

nums1 = [2,7,11,15]
target1 = 9
print(two_sums(nums1, target1))

nums2 = [3,2,4]
target2 = 6
print(two_sums(nums2, target2))

nums3 = [3,3]
target3 = 6
print(two_sums(nums3, target3))
```

```
[0, 1]
[1, 2]
[0, 1]
```

1.4 Question 4

How is a negative index used in Python? Show an example

A negative index in python is used to access the end of a string or list

[28]:
```
nums =  30, 7, 3000000, 78, 62, 34567890

print(nums[-2])

print(nums[-3])
```

```
62
78
```

1.5 Question 5

Check if two given strings are isomorphic to each other. Two strings str1 and str2 are called isomorphic if there is a one-to-one mapping possible for every character of str1 to every character of str2. And all occurrences of every character in 'str1' map to the same character in 'str2'.

```
Input:  str1 = "aab", str2 = "xxy"

Output: True

'a' is mapped to 'x' and 'b' is mapped to 'y'.

Input:  str1 = "aab", str2 = "xyz"

Output: False

One occurrence of 'a' in str1 has 'x' in str2 and other occurrence of 'a' has 'y'.
```

A Simple Solution is to consider every character of 'str1' and check if all occurrences of it map to the same character in 'str2'. The time complexity of this solution is O(n*n).

An Efficient Solution can solve this problem in O(n) time. The idea is to create an array to store mappings of processed characters.

[54]:
```
def is_isomorphic(str1,str2):
    if len(str1) != len(str2):
            return False
    map1 = {}
    map2 = {}

    for i in range(len(str1)):
        char1 = str1[i]
        char2 = str2[i]

        if char1 in map1:
            if map1[char1] != char2:
                return False
```

```
        else:
            map1[char1] = char2

        if char2 in map2:
            if map2[char2] != char1:
                return False

        else:
            map2[char2] = char1

    return True

print(is_isomorphic("aab", "xxy"))
print(is_isomorphic("aab", "xyz"))
```

```
True
False
```