ChrisRodgers.module07lab01

April 25, 2024

1 Assignment 7

1.0.1 Christopher Rodgers

1.0.2 25 Apr 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.

[7]:
```
# your code here
def for_palindrome(str):
    reverse_list = []
    for i in range(len(str)-1, -1, -1,):
        reverse_list.append(str[i])

    reverse_str = "".join(reverse_list)

    return str == reverse_str

print(for_palindrome("radar"))
print(for_palindrome("tomorrow"))
```

```
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.

[27]:
```
def palindrome(str):
    str = str.lower()
    start = 0
    end = len(str)-1

    while start < end:
        if str[start] != str[end]:
```

```
            return False
        start += 1
        end -= 1
    return True

print(palindrome('radar'))
print(palindrome('tomorrow'))
```

```
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.

[30]:
```
# your code here
from collections import defaultdict

def two_sum(nums, target):
    hash_map = defaultdict(int)

    hash_map[119] = {}
    for i in range(len(nums)):
        if (target - nums[i]) in hash_map.keys():
            hash_map[119].update({(target - nums[i]): i})
        else:
            hash_map[target - nums[i]] = i

    for key, value in hash_map.items():
        comp = (target - key)
        if comp in hash_map.keys():
            if comp != key:
                return [value, hash_map[comp]]
            else:
                if comp in hash_map[119].keys():
                    return [value, hash_map[119][comp]]
```

```
print(two_sum([2,7,11,15], 9))
print(two_sum([3,2,4], 6))
print(two_sum([3,3], 6))
```

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

1.4 Question 4

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

[35]:
```
# your code here
num_list = list(range(1,15))

print(num_list)

print(num_list[-1])
print(num_list[-3])
print(num_list[-7])
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
14
12
8
```

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.

[38]:
```
# your code here
def check_isomorphic(str1, str2):
    mappings = []
```

```
    for i in range(len(str1)):
        if str1[i] in mappings and str2[i] in mappings:
            continue
        elif str1[i] not in mappings and str2[i] not in mappings:
            mappings.append(str1[i])
            mappings.append(str2[i])
        else:
            return False
    return True

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

```
True
False
```

[ ]: