Assignment 7

PJ Grant

12/12/2024

In [1]:
```
from collections import defaultdict
```

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.

In [2]:
```
def palindrome(string):
    new_string = []
    string = string.strip().lower()

    for letter in string:
        new_string.append(letter)

    reversed_string = list(reversed(new_string))

    if new_string == reversed_string:
        print("This string is a palindrome")
    else:
        print("This string is not a palindrome")
```

In [3]:
```
string = 'civic'
palindrome(string)
```

```
This string is a palindrome
```

Question 2

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

In [4]:
```
def palindrome2(a_string):
    reversed_string = []
    a_string = a_string.strip().lower().replace(' ', '')
    characters = len(a_string) - 1
    count = 0
    while count <= characters:
        reversed_string.insert(0, a_string[count])
        count = count + 1
```

```
    reversed_string = ''.join(reversed_string)

    if a_string  == reversed_string:
        print("This string is a palindrome")
    else:
        print("This string is not a palindrome")
```

In [5]:
```
a_string = 'Taco cat'
palindrome2(a_string)
```

```
This string is a palindrome
```

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.

In [6]:
```
def two_sum(numbers, target):
    hash_map = defaultdict(int)

    for index, value in enumerate(numbers):
        complement = target - value

        if complement in hash_map:
            return [hash_map[complement], index]

        hash_map[value] = index
```

In [7]:
```
numbers = [1,2,3,4,5,6]
target = 11
two_sum(numbers,target)
```

Out[7]:
```
[4, 5]
```

Question 4

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

Negative indexing allows you to iterate through a variable from right to left (in reverse order). If I wanted to find the last element in a list I would type list[-1]

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.

In [8]:
```
def isomporphic_check(string_1, string_2):
    mapping = defaultdict()

    for index, letter in enumerate(string_1):
        if letter in mapping:
            if mapping[letter] != string_2[index]:
                return False
        elif string_2[index] not in mapping.values():
            mapping[letter] = string_2[index]
        else:
            return False
    return True

string_1 = "6677aa"
```

```
string_2 = "aa7766"
isomporphic_check(string_1, string_2)
```

Out[8]: True