Q1. Given two strings s and t, determine if they are isomorphic.

Two strings s and t are isomorphic if the characters in s can be replaced to get t.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.

**def** areIsomorphic(string1, string2):

    m **=** len(string1)

    n **=** len(string2)

**if** m !**=** n:

**return** False

    marked **=** [False] **\*** MAX\_CHARS

    map **=** [**-**1] **\*** MAX\_CHARS

**for** i **in** xrange(n):

**if** map[ord(string1[i])] **==** **-**1:

**if** marked[ord(string2[i])] **==** True:

**return** False

            marked[ord(string2[i])] **=** True

            map[ord(string1[i])] **=** string2[i]

**elif** map[ord(string1[i])] !**=** string2[i]:

**return** False

**return** True

Q2. isStrobogrammatic or not

class Solution(object):

def isStrobogrammatic(self, num):

maps = {("0", "0"), ("1", "1"), ("6", "9"), ("8", "8"), ("9", "6")}

i,j = 0, len(num) - 1

while i <= j:

if (num[i], num[j]) not in maps:

return False

i += 1

j -= 1

return True

Q3. Given two non-negative integers, num1 and num2 represented as string, return *the sum of* num1 *and* num2 *as a string*.

def addStrings(self, num1, num2):

result = ""

carry = 0

def equalizeNumberOfCharacters(num1, num2):

if len(num1) < len(num2):

while len(num1) != len(num2):

num1 = "0" + num1

else:

while len(num2) != len(num1):

num2 = "0" + num2

return [num1,num2]

num1, num2 = equalizeNumberOfCharacters(num1, num2)

num1Array = list(num1)

num2Array = list(num2)

while len(num1Array) != 0:

add = int(num1Array.pop()) + int(num2Array.pop()) + int(carry)

carry = add // 10

result = str(add % 10) + result

if carry != 0:

result = str(carry) + result

return result

Q4. Given a string s, reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

**def** reverseWords(string):

    st **=** list()

**for** i **in** range(len(string)):

**if** string[i] !**=** " ":

            st.append(string[i])

**else**:

**while** len(st) > 0:

                print(st[**-**1], end**=**"")

                st.pop()

            print(end**=**" ")

**while** len(st) > 0:

**print**(st[**-**1], end**=**"")

        st.pop()

Q5. Given a string s and an integer k, reverse the first k characters for every 2k characters counting from the start of the string.

If there are fewer than k characters left, reverse all of them. If there are less than 2k but greater than or equal to k characters, then reverse the first k characters and leave the other as original.

class Solution(object):

def reverseStr(self, s, k):

sLen = len(s)

builder = ""

left = 0

rev = False

for i in range(sLen):

if i % k == 0:

subStr = s[left:i]

if rev:

subStr = subStr[::-1]

builder += subStr

left = i

rev = not rev

subStr = s[left:sLen]

if rev:

subStr = subStr[::-1]

builder += subStr

return builder

q6. Given two strings s and goal, return true *if and only if* s *can become* goal *after some number of* ***shifts*** *on* s.

A **shift** on s consists of moving the leftmost character of s to the rightmost position

**def** check\_rotation(s, goal):

**if** (len(s) !**=** len(goal)):

        skip

    q1 **=** []

**for** i **in** range(len(s)):

        q1.insert(0, s[i])

    q2 **=** []

**for** i **in** range(len(goal)):

        q2.insert(0, goal[i])

    k **=** len(goal)

**while** (k > 0):

        ch **=** q2[0]

        q2.pop(0)

        q2.append(ch)

**if** (q2 **==** q1):

**return** True

        k **-=** 1

**return** False

Q7. Given two strings s and t, return true *if they are equal when both are typed into empty text editors*. '#' means a backspace character.

**def** removeBackspace(s) **-**> str:

    n **=** len(s)

    # To point at position after considering the backspaces

    idx **=** 0

**for** i **in** range(0, n):

**if**(s[i] !**=** '#'):

            s **=** s[:idx] **+** s[i] **+** s[idx**+**1:]

            idx **+=** 1

**elif**(s[i] **==** '#' **and** idx >**=** 0):

            idx **-=** 1

        # This idx can never point at negative index position

**if**(idx < 0):

            idx **=** 0

    ans **=** ""

**for** i **in** range(0, idx):

        ans **+=** s[i]

**return** ans

Q8. You are given an array coordinates, coordinates[i] = [x, y], where [x, y] represents the coordinate of a point. Check if these points make a straight line in the XY plane.

**def** isStraightLinePossible(arr, n):

    x0 **=** arr[0][0]

    y0 **=** arr[0][1]

    x1 **=** arr[1][0]

    y1 **=** arr[1][1]

    dx **=** x1 **-** x0

    dy **=** y1 **-** y0

**for** i **in** range(n):

        x **=** arr[i][0]

        y **=** arr[i][1]

**if** (dx **\*** (y **-** y1) !**=** dy **\*** (x **-** x1)):

            print("NO", end **=** "")

**return** False

**print**("YES", end **=** "")

**return** True