NAME : KOVI SAI GANESH

REGISTER NUMBER : 192124028

PYTHON PROGRAMS

DAY 1

1 . 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. Constraints • s and t consist of any valid ascii character.

Test Cases: 1.Input: s = "egg", t = "add" Output: true 2.Input: s = "foo", t = "bar" Output: false 3.Input: s = "paper", t = "title" Output: true 4.Input: s = "fry", t = "sky" Output: true 5. Input: s = "apples", t = "apple" Output: false

def isomorphic(str1,str2):

if len(str1) != len(str2):

return False

else:

map1,map2={},{}

for i in range(len(str1)):

ch1,ch2=str1[i],str2[i]

if ch1 not in map1:

map1[ch1]=ch2

if ch2 not in map2:

map2[ch2]=ch1

if ((map1[ch1] != ch2) or (map2[ch2]!=ch1)):

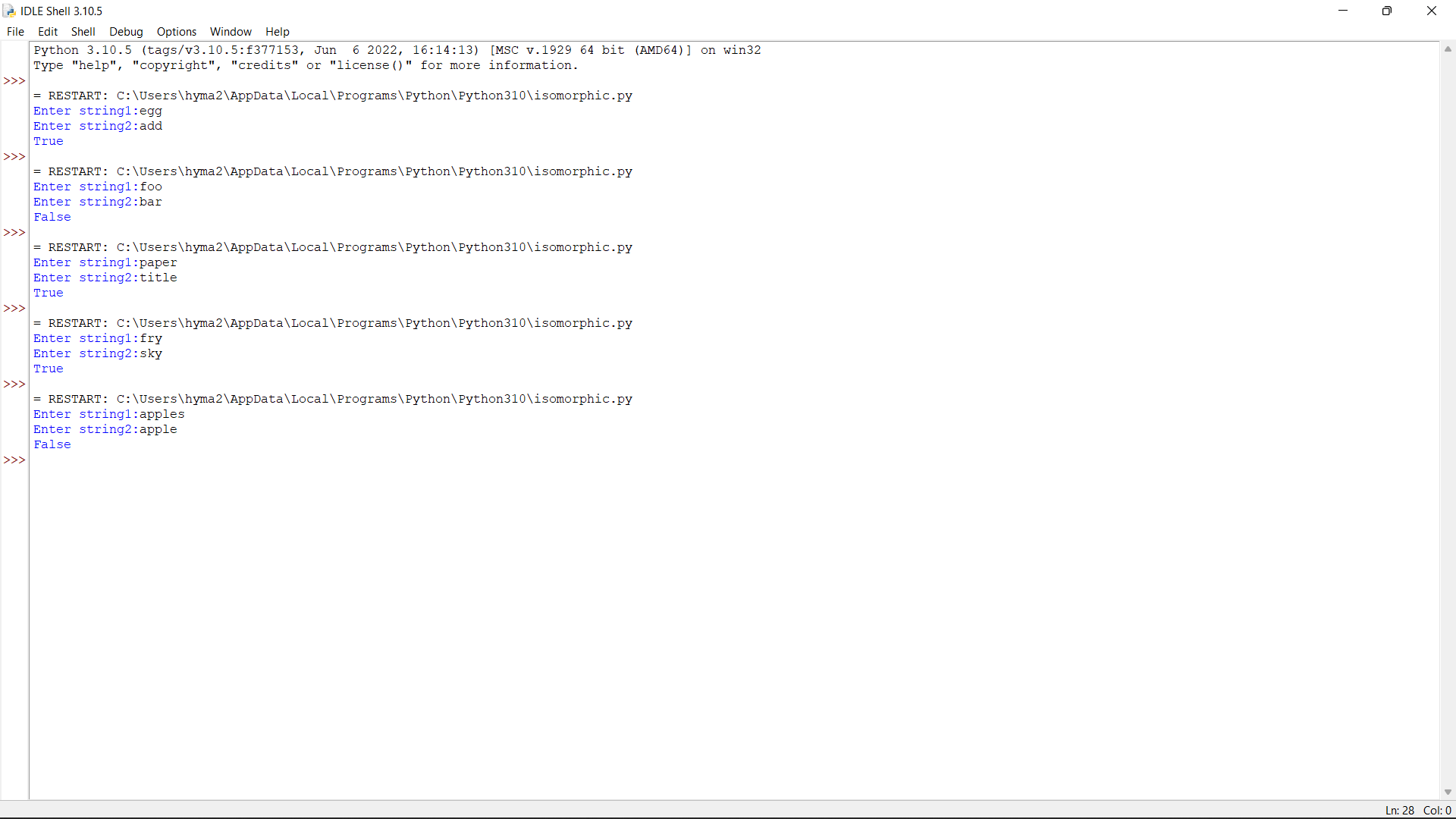
return False

return True

str1=input("Enter string1:")

str2=input("Enter string2:")

print(isomorphic(str1,str2))



2. Write a Python function sumsquare(l) that takes a nonempty list of integers and returns a list [odd,even], where odd is the sum of squares of all the odd numbers in l and even is the sum of squares of all the even numbers in l. Sample Input: Enter the number of elements:7 Enter the element: 18 Enter the element:9 Enter the element:1 Enter the element:12 Enter the element:13 Enter the element:4 Enter the element:30 Output: [251,1384] Test Cases: 1. 5, [1,2,3,4,5] 2. 8, [2,4,5,6,7,11,12,13] 3. -1, [12,10,11,1,2] 4. 0, [1,2,3,4,5] 5. -8, [2,4,5,6,7,11,12,13]

l=int(input("Enter number of elements:"))

lst=[]

sum1=0

sum2=0

for i in range(0,l):

ele=int(input())

lst.append(ele)

for i in range(0,l):

if(lst[i]%2==0):

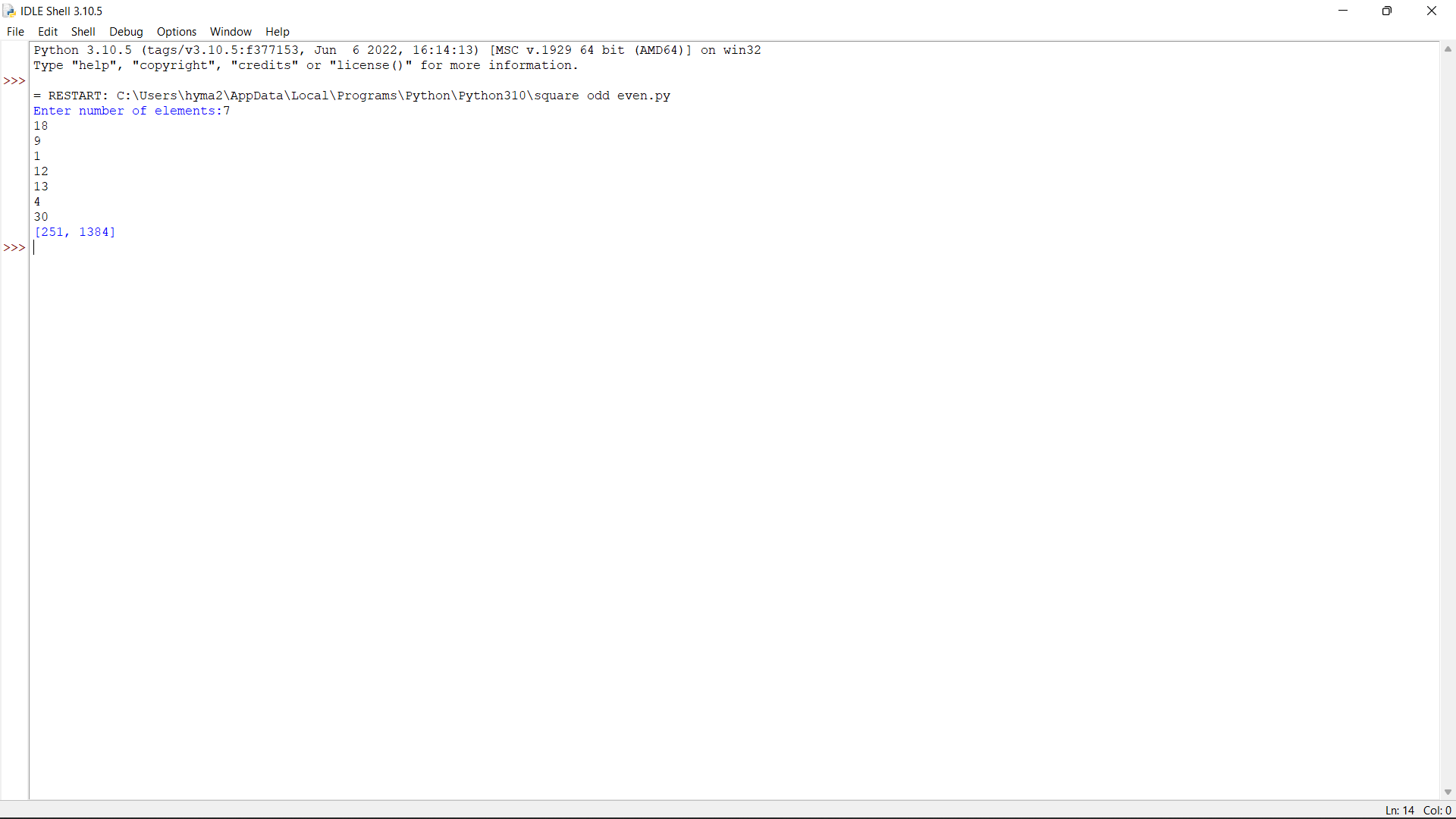
sum2=sum2+lst[i]\*\*2

else:

sum1=sum1+lst[i]\*\*2

l1=[sum1,sum2]

print(l1)



3. Write an algorithm to determine if a number n is happy. A happy number is a number defined by the following process: • Starting with any positive integer, replace the number by the sum of the squares of its digits. • Repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. • Those numbers for which this process ends in 1 are happy. Return true if n is a happy number, and false if not. Test Case 1. Input: n = 19 Output: true Explanation: 12 + 92 = 82 82 + 22 = 68 62 + 82 = 100 12 + 02 + 02 = 1 2. Input: n = 2 Output: false 3. Input: n=-1 4. Input: n=0 5. 5

def numSquareSum(n):

squareSum = 0

while(n):

squareSum += (n%10)\*(n%10)

n=int(n/10)

return squareSum

def isHappynumber(n):

slow=n

fast=n

while(True):

slow=numSquareSum(slow)

fast=numSquareSum(numSquareSum(fast))

if(slow!=fast):

continue

else:

break

return (slow==1)

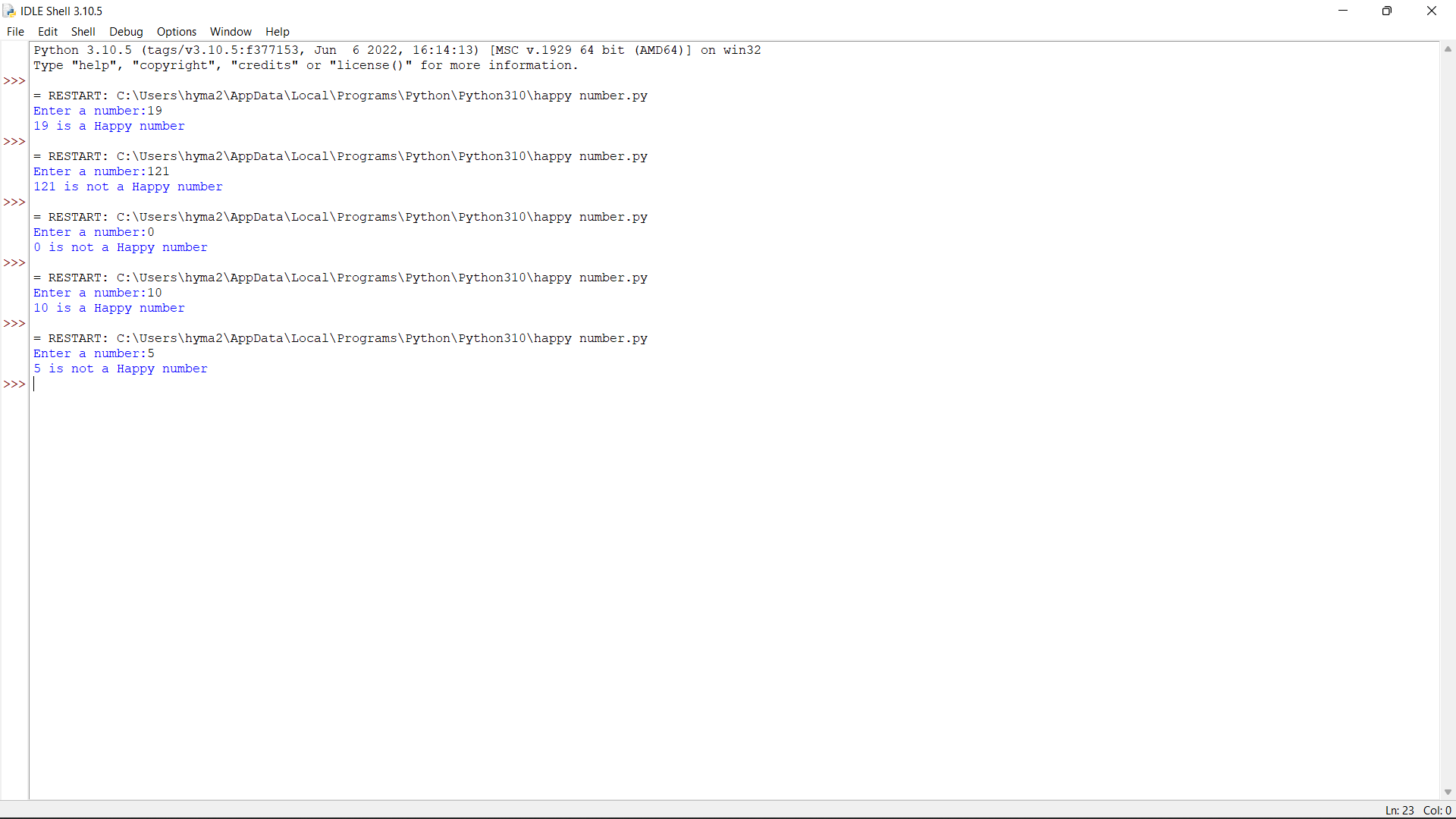
n =int(input("Enter a number:"))

if (isHappynumber(n)):

print(n , "is a Happy number")

else:

print(n , "is not a Happy number")



4. Given an integer x, return true if x is palindrome integer. An integer is a palindrome when it reads the same backward as forward. For example, 121 is a palindrome while 123 is not. Test cases: 1.Input: x = 121 Output: true Explanation: 121 reads as 121 from left to right and from right to left. 2.Input: x = -121 Output: false Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome. 3.Input: x = 10 Output: false Explanation: Reads 01 from right to left. Therefore it is not a palindrome. 4.x=abc 5.x=0

num=int(input("Enter a number:"))

temp=num

rev=0

while(num>0):

dig=num%10

rev=rev\*10+dig

num=num//10

if(temp==rev):

print("The number is palindrome!")

else:

print("Not a palindrome!")



5. A bakery sells loaves of bread for 185 rupees each. Day old bread is discounted by 60 percent. Write a program that begins by reading the number of loaves of day old bread being purchased from the user. Then your program should display the regular price for the bread, the discount because it is a day old, and the total price. All of the values should be displayed using two decimal places, and the decimal points in all of the numbers should be aligned when reasonable values are entered by the user. Sample Input: Enter the number of fresh loves purchased: 5 Enter the number of day old loaves purchased: 3 Sample Output: Regular price: Rs.185.00 Amount of new loaves: 925.00 Amount of day old loaves: 222.00 Total amount: Rs. 1147.00 Test cases: 1. 4,6 2. -1,5 3. 0,6 4. 7,8 5. 3,4

a=int(input("Enter number of fresh loaves:"))

b=int(input("Enter number of old loaves:"))

c=185\*a

d=185\*b\*0.6

e=185

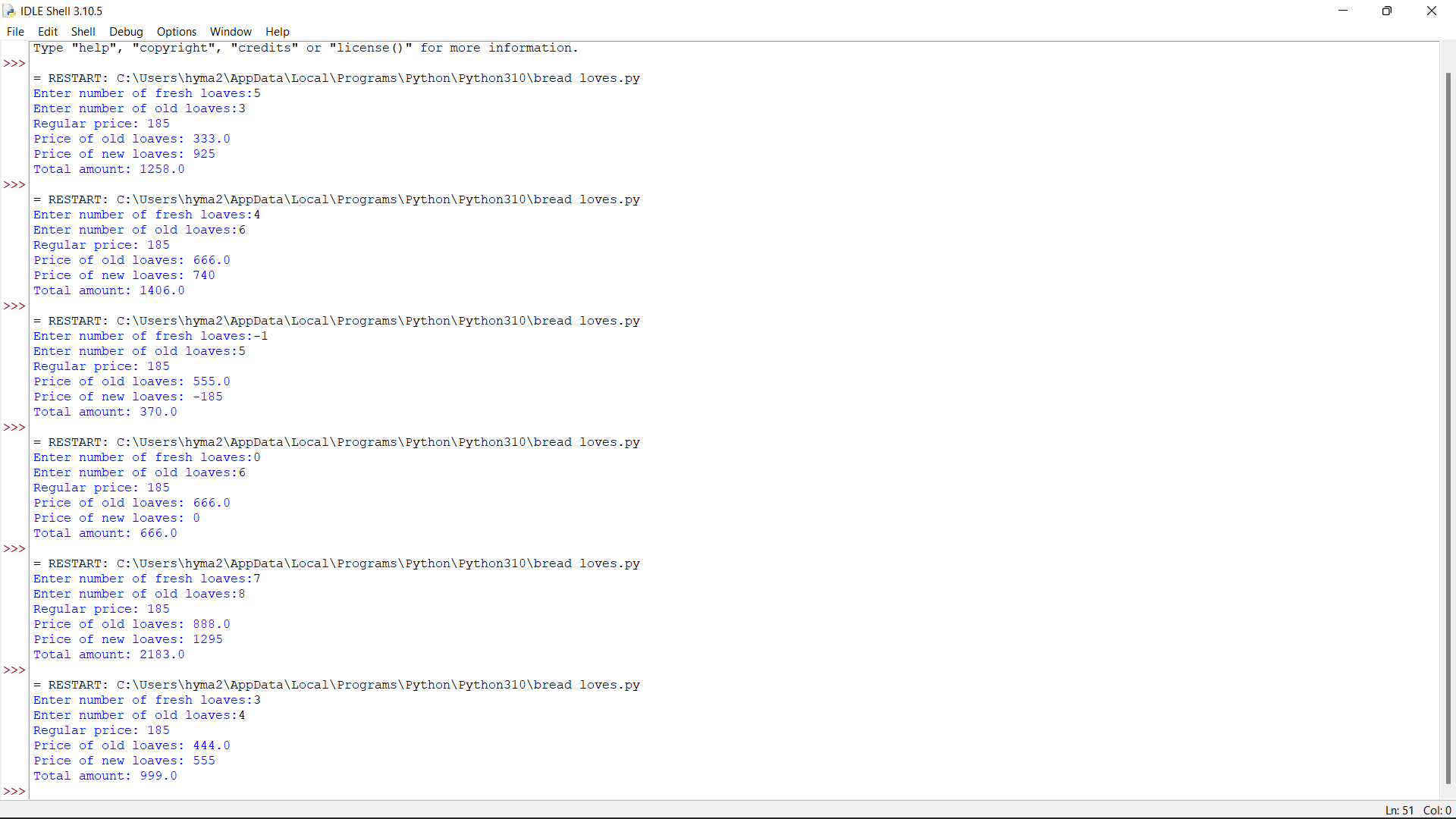
total=c+d

print("Regular price:",e)

print("Price of old loaves:",d)

print("Price of new loaves:",c)

print("Total amount:",total)



DAY 2

1q You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

def fib(n):

if n <= 1:

return n

return fib(n-1) + fib(n-2)

def countWays(s):

return fib(s + 1)

s = int(input("Enter Input : "))

print("Number of ways = ",countWays(s))

2q Most years have 365 days. However, the time required for the Earth to orbit the Sun is actually slightly more than that. As a result, an extra day, February 29, is included in some years to correct for this difference. Such years are referred to as leap years. The rules for determining whether or not a year is a leap year follow: Any year that is divisible by 400 is a leap year. Of the remaining years, any year that is divisible by 100 is not a leap year. Of the remaining years, any year that is divisible by 4 is a leap year. All other years are not leap years. Write a program that reads a year from the user and displays a message indicating whether or not it is a leap year. Sample Input: Enter Date : 1947 Sample Output: Given year is Non Leap Year Leap Year: 1944 Test cases: 1. 19.47 2. 1936 3. 0 4. 2000 5. -1428

n=int(input("Enter Your Year : "))

if(n>0):

if(n%400==0):

print("Your Year is a Leap Year")

elif((n%4==0)and (n%100!=0)):

print("Your Year is a Leap Year")

else:

print("Your Year is not a Leap Year")

else:

print("Invalid Year")

10q Python program to remove words that are common in two Strings Given two strings S1 and S2, representing sentences, the task is to print both sentences after removing all words which are present in both sentences. Test cases: 1.Input: S1 = “sky is blue in color”, S2 =”Raj likes sky blue color “ Output: is in Raj likes Explanation: The common words are [ sky, blue, color ]. Removing these words from the two sentences modifies the sentences to the specified output. 2.Input: S1 = “learn python“, S2 = “python is easy to learn“ 3.S1= “raju likes apple”, S2=”apple is red in color” 4.S1= “ sita likes orange” S2=”orange is rich in anti-oxidents” 5. S1=”raj is travelling to Chennai in train” S2=”the rain will reach Chennai at 8 pm

from collections import Counter

def removeCommonWords(sent1, sent2):

sentence1 = list(sent1.split())

sentence2 = list(sent2.split())

frequency1 = Counter(sentence1)

frequency2 = Counter(sentence2)

word = 0

for i in range(len(sentence1)):

if sentence1[word] in frequency2.keys():

sentence1.pop(word)

word = word-1

word += 1

word = 0

for i in range(len(sentence2)):

if sentence2[word] in frequency1.keys()

sentence2.pop(word)

word = word-1

word += 1

print(\*sentence1)

print(\*sentence2)

sentence1 = input("Enter Your Sentence 1 : ")

sentence2 = input("Enter Your Sentence 2 : ")

removeCommonWords(sentence1, sentence2)

4q . Merge Two Sorted Lists You are given the heads of two sorted linked lists list1 and list2. Merge the two lists in a one sorted list. The list should be made by splicing together the nodes of the first two lists. Return the head of the merged linked list. Constraints: • The number of nodes in both lists is in the range [0, 50]. • Both list1 and list2 are sorted in non-decreasing order. Test cases: 1.Input: list1 = [1,2,4], list2 = [1,3,4] Output: [1,1,2,3,4,4] 2.Input: list1 = [], list2 = [] Output: [] 3.Input: list1 = [], list2 = [0] Output: [0] 4.list1=[],list2=[1,2,3,4,5] 5.list1=[0,1,9], list2=[3,4,5]

lst1=[]

lst2=[]

n1=int(input("Enter number of elements in list1:"))

for i in range(0,n1):

a=int(input())

lst1.append(a)

print(lst1)

n2=int(input("Enter number of elements in list2:"))

for i in range(0,n2):

b=int(input())

lst2.append(b)

print(lst2)

lst3=lst1+lst2

lst3.sort()

print("The merged list is:",lst3)

7q Generate Parentheses Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses.

def printParenthesis(str, n):

if(n > 0):

\_printParenthesis(str, 0,

n, 0, 0)

return

def \_printParenthesis(str, pos, n,

open, close):

if(close == n):

for i in str:

print(i, end="")

print()

return

else:

if(open > close):

str[pos] = ')'

\_printParenthesis(str, pos + 1, n,

open, close + 1)

if(open < n):

str[pos] = '('

\_printParenthesis(str, pos + 1, n,

open + 1, close)

n = int(input("n="))

str = [" "] \* 2 \* n

printParenthesis(str , n)

9q The year is divided into four seasons: spring, summer, fall and winter. While the exact dates that the seasons change vary a little bit from year to year because of the way that the calendar is constructed, we will use the following dates for this exercise: Season First day Summer March 20 Spring June 21 Fall September 22 Winter December 21 Create a program that reads a month and day from the user. The user will enter the name of the month as a string, followed by the day within the month as an integer. Then your program should display the season associated with the date that was entered. Note: Enter First three letter for month example: Jan for January, Feb for February and so on....and first letter of the month should be capital Input: Enter the month: march Enter the date: 21 Output: The season is currently summer

month = input("Input the month (e.g. January, February etc.): ")

day = int(input("Input the day: "))

if month in ('January', 'February', 'March'):

season = 'winter'

elif month in ('April', 'May', 'June'):

season = 'spring'

elif month in ('July', 'August', 'September'):

season = 'summer'

else:

season = 'autumn'

if (month == 'March') and (day > 19):

season = 'spring'

elif (month == 'June') and (day > 20):

season = 'summer'

elif (month == 'September') and (day > 21):

season = 'autumn'

elif (month == 'December') and (day > 20):

season = 'winter'

print("Season is",season)

DAY 3

1Q In daily share trading, a buyer buys shares in the morning and sells them on the same day. If the trader is allowed to make at most 2 transactions in a day, whereas the second transaction can only start after the first one is complete (Buy->sell->Buy->sell). Given stock prices throughout the day, find out the maximum profit that a share trader could have made. Test Case: 1.Input: prices = [7,1,5,3,6,4] Output: 7 2.Input: prices = [7,6,4,3,1] Output: 0 3.Input: [10, 22, 5, 75, 65, 80] Output:87 4.Input: [2, 30, 15, 10, 8, 25, 80] Output:100 5.Input: [10, 22, 5, 75, 65, 80] Output:0

def maxProfit(price, n):

profit = [0]\*n

max\_price = price[n-1]

for i in range(n-2, 0, -1):

if price[i] > max\_price:

max\_price = price[i]

profit[i] = max(profit[i+1], max\_price - price[i])

min\_price = price[0]

for i in range(1, n):

if price[i] < min\_price:

min\_price = price[i]

profit[i] = max(profit[i-1], profit[i]+(price[i]-min\_price))

result = profit[n-1]

return result

a=[]

n=int(input("Enter number of items:"))

for i in range(n):

a.append(int(input()))

print ("Maximum profit is", maxProfit(a, len(a)))

2Q The Project manager has to submit the project within a week period. He didn’t find the proper combinations of team members to work on the project, Help him in finding the possible combinations available. Accept 3 digits and find all the combinations Sample Input: 123 Sample Output: 123 132 213 231 312 321 Test Cases: 1. 789 2. 1456 3.-856 4. 1001 5. 555

def comb(L):

for i in range(3):

for j in range(3):

for k in range(3):

# check if the indexes are not

# same

if (i!=j and j!=k and i!=k):

print(L[i], L[j], L[k])

a=[]

print("enter number:")

for i in range(3):

b=int(input())

a.append(b)

comb(a)

3Q Given an array of integers nums, return the number of good pairs. A pair (i, j) is called good if nums[i] == nums[j] and i < j.

def solve(nums):

count=0

n=len(nums)

for i in range(n):

for j in range(i+1,n):

if nums[i] == nums[j]:

count+=1

return count

a=[]

n=int(input("Enter number of elements:"))

print("Enter elements:")

for i in range(n):

b=int(input())

a.append(b)

print("Number of good pairs:",solve(a))

6Q Raju, has again started troubling people in your city. The people have turned on to you for getting rid of Raju. Raju presents to you a number consisting of numbers from 0 to 9 characters. He wants you to reverse it from the final answer such that the number becomes Mirror number. A Mirror is a number which equals its reverse. The hope of people are on you so you have to solve the riddle. You have to tell if some number exists which you would reverse to convert the number into Mirror Sample input: Enter the number: 123456 Sample output: Mirror image: 654321

def reverse(s):

str = " "

for i in s:

str= i+str

return str

s=input("Enter Your String : ")

print("Your String is : ", s)

print("Your Reverse String is : ", reverse(s))

10Q Given two strings word1 and word2, return the minimum number of operations required to convert word1 to word2. You have the following three operations permitted on a word: • Insert a character • Delete a character • Replace a character Test case: 1.Input: word1 = "horse", word2 = "ros" Output: 3 2.Input: word1 = "intention", word2 = "execution" Output: 5 3.Input: str1 = “sunday”, str2 = “saturday” Output: 3 4.Input: str1 = “cat”, str2 = “cut” Output: 1 5.Input: str1 = “girl”, str2 = “grill” Output: 2

def editDistance(str1, str2, m, n):

if m == 0:

return n

if n == 0:

return m

if str1[m-1] == str2[n-1]:

return editDistance(str1, str2, m-1, n-1)

return 1 + min(editDistance(str1, str2, m, n-1),

editDistance(str1, str2, m-1, n),

editDistance(str1, str2, m-1, n-1)

)

str1 =input("Enter Your String1 : ")

str2 = input("Enter Your String : ")

print (editDistance(str1, str2, len(str1), len(str2)))

5Q Given an array of integers where each element represents the max number of steps that can be made forward from that element. Write a function to return the minimum number of jumps to reach the end of the array (starting from the first element). If an element is 0, they cannot move through that element. If the end isn’t reachable, return -1. Test Case: 1. Input: arr[] = [1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9] Output: 3 (1-> 3 -> 9 -> 9) 2. Input: arr[] = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] Output: 10 3. Input: arr[] = [2,3,1,1,4] Output: 2 4. Input: arr[] = [1, 3, 6, 1, 0, 9] Output: 3 5. Input: arr[] = [2,3,0,1,4]

def minJumps(arr, l, h):

if (h == l):

return 0

if (arr[l] == 0):

return float('inf')

min = float('inf')

for i in range(l + 1, h + 1):

if (i < l + arr[l] + 1):

jumps = minJumps(arr, i, h)

if (jumps != float('inf') and

jumps + 1 < min):

min = jumps + 1

return min

arr=eval(input("Enter list:"))

n=len(arr)

print('Minimum number of jumps to reach',

'end is', minJumps(arr, 0, n-1))

DAY 4

2Q Write a program to find the number of student users in the college, get the total users, staff users details from the client. Note for every 3 staff user there is one Non-teaching staff user assigned by default. Sample Input: Total Users: 856 Staff Users: 126 Sample Output: Student Users: 688 Test Cases: 1. Total User: 0 2. Total User: -143 3. Total User: 1026, Staff User: 1026 4. Total User: 450, Staff User: 540 5. Total User: 600, Staff User: 450

a=float(input("Enter Total Number of Users : "))

b=float(input("Enter Number of Staff Users : "))

c=a-b-(b//3)

print("Student Users Are : " , c)

4Q Valid Palindrome A phrase is a palindrome if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers. Given a string s, return true if it is a palindrome, or false otherwise. Test Cases: 1.Input: s = "A man, a plan, a canal: Panama" Output: true 2.Input: s = "race a car" Output: false 3.Input: s = " " Output: true 4. s= “madam” 5.s= “honest”

def first\_letter\_index(str, left, right):

index = -1

for i in range(left, right + 1):

if str[i] >= 'a' and str[i] <= 'z' :

index = i

break

return index

def last\_letter\_index(str, left, right):

index = -1

for i in range(left, right - 1, -1) :

if str[i] >= 'a' and str[i] <= 'z':

index = i

break

return index

def solve(str):

left = 0

right = len(str) - 1

flag = True

for i in range(len(str)) :

left = first\_letter\_index(str, left, right)

right = last\_letter\_index(str, right, left)

if right < 0 or left < 0:

break

if str[left] == str[right]:

left += 1

right -= 1

continue

flag = False

break

return flag

s = input("enter string:")

print(solve(s))

5Q Given an array of integers where each element represents the max number of steps that can be made forward from that element. Write a function to return the minimum number of jumps to reach the end of the array (starting from the first element). If an element is 0, they cannot move through that element. If the end isn’t reachable, return -1. Test Case: 1.Input: arr[] = [1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9] Output: 3 (1-> 3 -> 9 -> 9) 2.Input: arr[] = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] Output: 10 3.Input: arr[] = [2,3,1,1,4] Output: 2 4.Input: arr[] = [1, 3, 6, 1, 0, 9] Output: 3 5.Input: arr[] = [2,3,0,1,4] Output: 2

def minJumps(arr, l, h):

if (h == l):

return 0

if (arr[l] == 0):

return float('inf')

min = float('inf')

for i in range(l + 1, h + 1):

if (i < l + arr[l] + 1):

jumps = minJumps(arr, i, h)

if (jumps != float('inf') and

jumps + 1 < min):

min = jumps + 1

return min

arr=eval(input("Enter list:"))

n=len(arr)

print('Minimum number of jumps to reach',

'end is', minJumps(arr, 0, n-1))

7Q Count Sorted Vowel Strings Given an integer n, return the number of strings of length n that consist only of vowels (a, e, i, o, u) and are lexicographically sorted. A string s is lexicographically sorted if for all valid i, s[i] is the same as or comes before s[i+1] in the alphabet. Test Cases: 1.Input: n = 1 Output: 5 Explanation: The 5 sorted strings that consist of vowels only are ["a","e","i","o","u"]. 2.Input: n = 2 Output: 15 Explanation: The 15 sorted strings that consist of vowels only are ["aa","ae","ai","ao","au","ee","ei","eo","eu","ii","io","iu","oo","ou","uu"]. Note that "ea" is not a valid string since 'e' comes after 'a' in the alphabet. 3.Input: n = 33 Output: 66045 4.n=55 5=32

def countstrings(n, start):

if n == 0:

return 1

cnt = 0

for i in range(start, 5):

cnt += countstrings(n - 1, i)

return cnt

def countVowelStrings(n):

return countstrings(n, 0)

n = int(input("Enter the N Value : "))

print(countVowelStrings(n))

9Q The year is divided into four seasons: spring, summer, fall and winter. While the exact dates that the seasons change vary a little bit from year to year because of the way that the calendar is constructed, we will use the following dates for this exercise: Season First day Summer March 20 Spring June 21 Fall September 22 Winter December 21 Create a program that reads a month and day from the user. The user will enter the name of the month as a string, followed by the day within the month as an integer. Then your program should display the season associated with the date that was entered. Note: Enter First three letter for month example: Jan for January, Feb for February and so on....and first letter of the month should be capital Input: Enter the month: march Enter the date: 21 Output: The season is currently summer Test Cases: 1. July, 29 2. September, 5 3. December, 30 4. March, 12 5. June, 27

month = input("Input the month (e.g. January, February etc.): ")

day = int(input("Input the day: "))

if month in ('January', 'February', 'March'):

season = 'winter'

elif month in ('April', 'May', 'June'):

season = 'summer'

elif month in ('July', 'August', 'September'):

season = 'spring'

else:

season = 'autumn'

if (month == 'March') and (day > 19):

season = 'summer'

elif (month == 'June') and (day > 20):

season = 'summer'

elif (month == 'September') and (day > 21):

season = 'autumn'

elif (month == 'December') and (day > 20):

season = 'winter'

print("Season is",season)

10Q 10. Scramble String We can scramble a string s to get a string t using the following algorithm: If the length of the string is 1, stop. If the length of the string is > 1, do the following: Split the string into two non-empty substrings at a random index, i.e., if the string is s, divide it to x and y where s = x + y. Randomly decide to swap the two substrings or to keep them in the same order. i.e., after this step, s may become s = x + y or s = y + x. Apply step 1 recursively on each of the two substrings x and y. Given two strings s1 and s2 of the same length, return true if s2 is a scrambled string of s1, otherwise, return false. Test cases: 1.Input: s1 = "great", s2 = "rgeat" Output: true 2.Input: s1 = "abcde", s2 = "caebd" Output: false 3.Input: s1 = "a", s2 = "a" Output: true 4.s1=”ab” s2= “ad” 5s1=10 s2=-5

def isScramble(S1: str, S2: str):

if len(S1) != len(S2):

return False

n = len(S1)

if not n:

return True

if S1 == S2:

return True

if sorted(S1) != sorted(S2):

return False

for i in range(1, n):

if (isScramble(S1[:i], S2[:i]) and

isScramble(S1[i:], S2[i:])):

return True

if (isScramble(S1[-i:], S2[:i]) and

isScramble(S1[:-i], S2[i:])):

return True

return False

S1 = input("enter string1:")

S2 = input("enter string2:")

if (isScramble(S1, S2)):

print("Yes")

else:

print("No")

DAY 5

Reverse Words in a String Given an input string s, reverse the order of the words. A word is defined as a sequence of non-space characters. The words in s will be separated by at least one space. Return a string of the words in reverse order concatenated by a single space. Note that s may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces. Test Cases: 1.Input: s = "the sky is blue" Output: "blue is sky the" 2.Input: s = " hello world " Output: "world hello" 3.Input: s = "a good example" Output: "example good a" 4. s= “apple is red” 5.s= “Red rose”

def reverse\_word(s, start, end):

while start < end:

s[start], s[end] = s[end], s[start]

start = start + 1

end -= 1

s = input("Enter Your String : " )

s = list(s)

start = 0

while True:

try:

end = s.index(' ', start)

reverse\_word(s, start, end - 1)

start = end + 1

except ValueError:

reverse\_word(s, start, len(s) - 1)

break

s.reverse()

s = "".join(s)

print(s)