Name : Kovi Sai Ganesh

Register Number : 192124028

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:

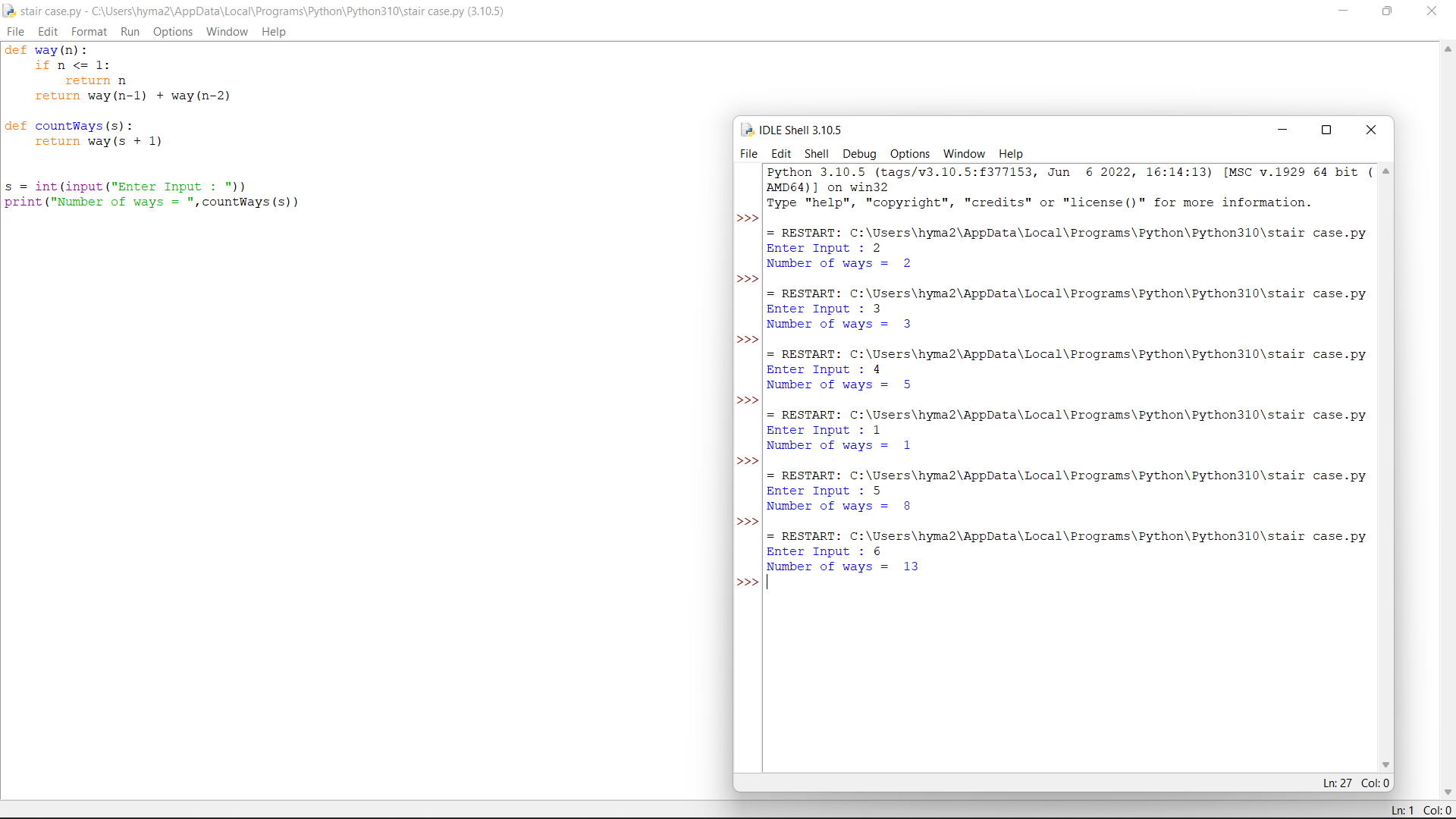
return n

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)):

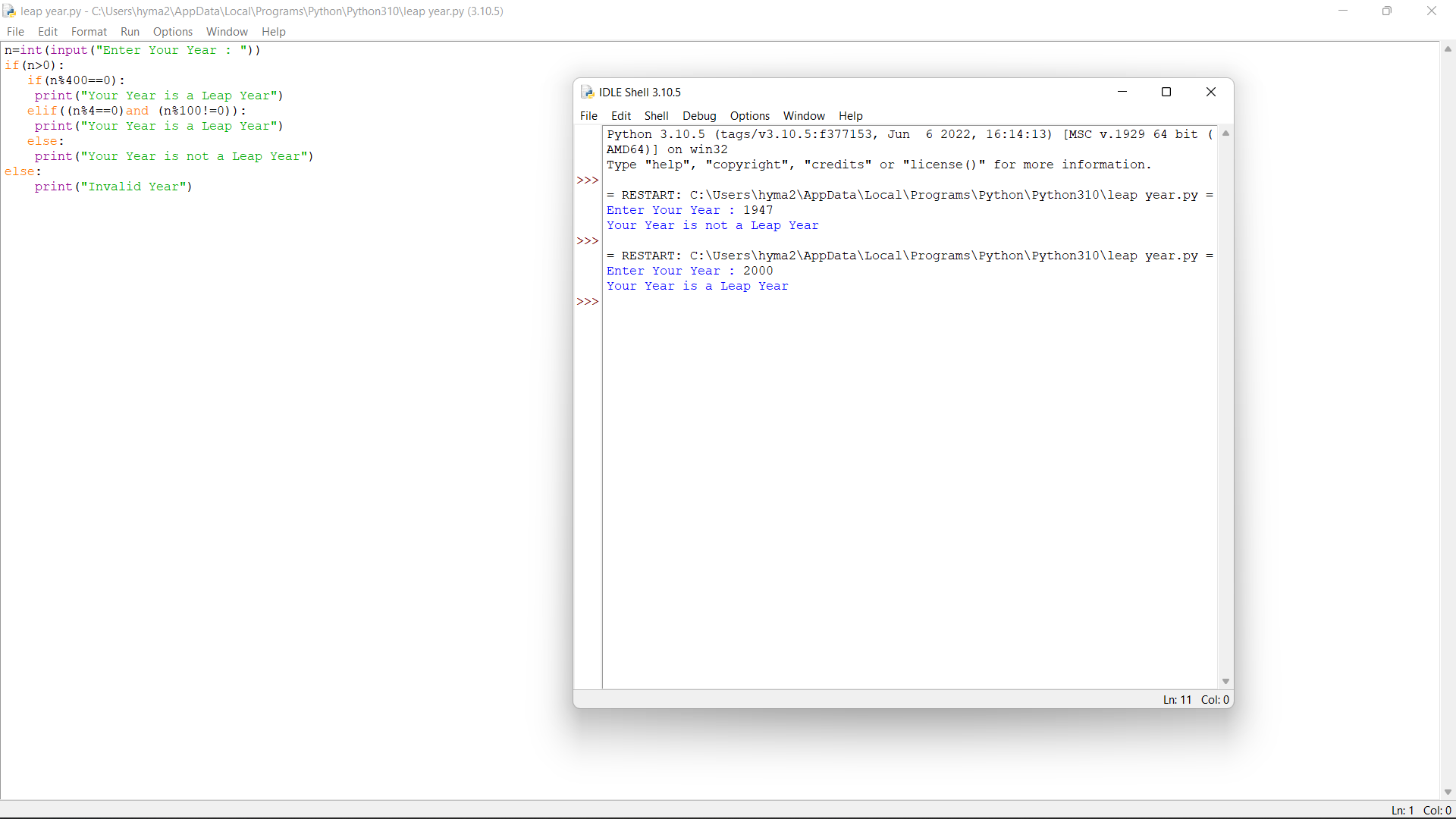
print("Your Year is a Leap Year")

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())

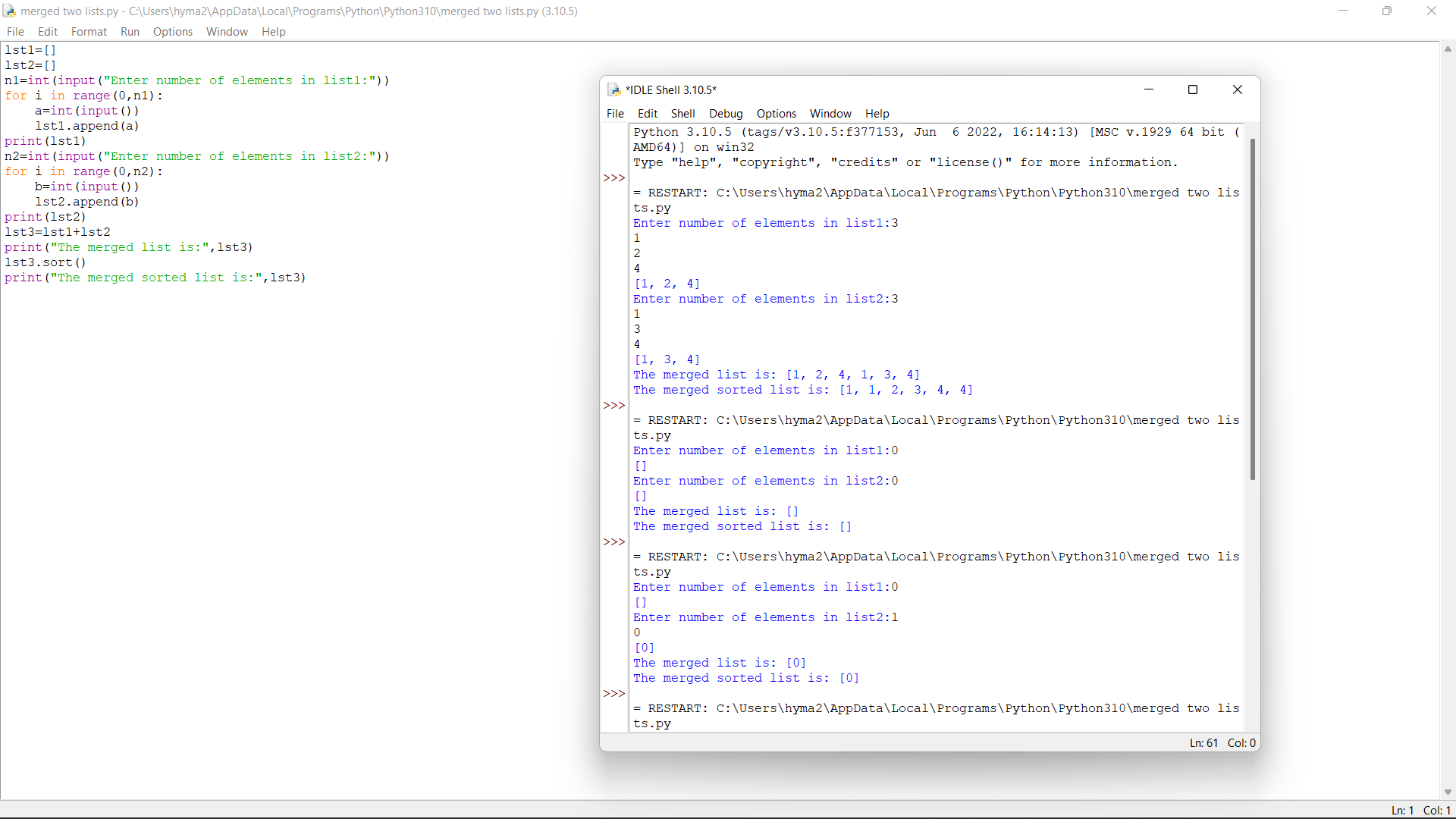
lst2.append(b)

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] = '('

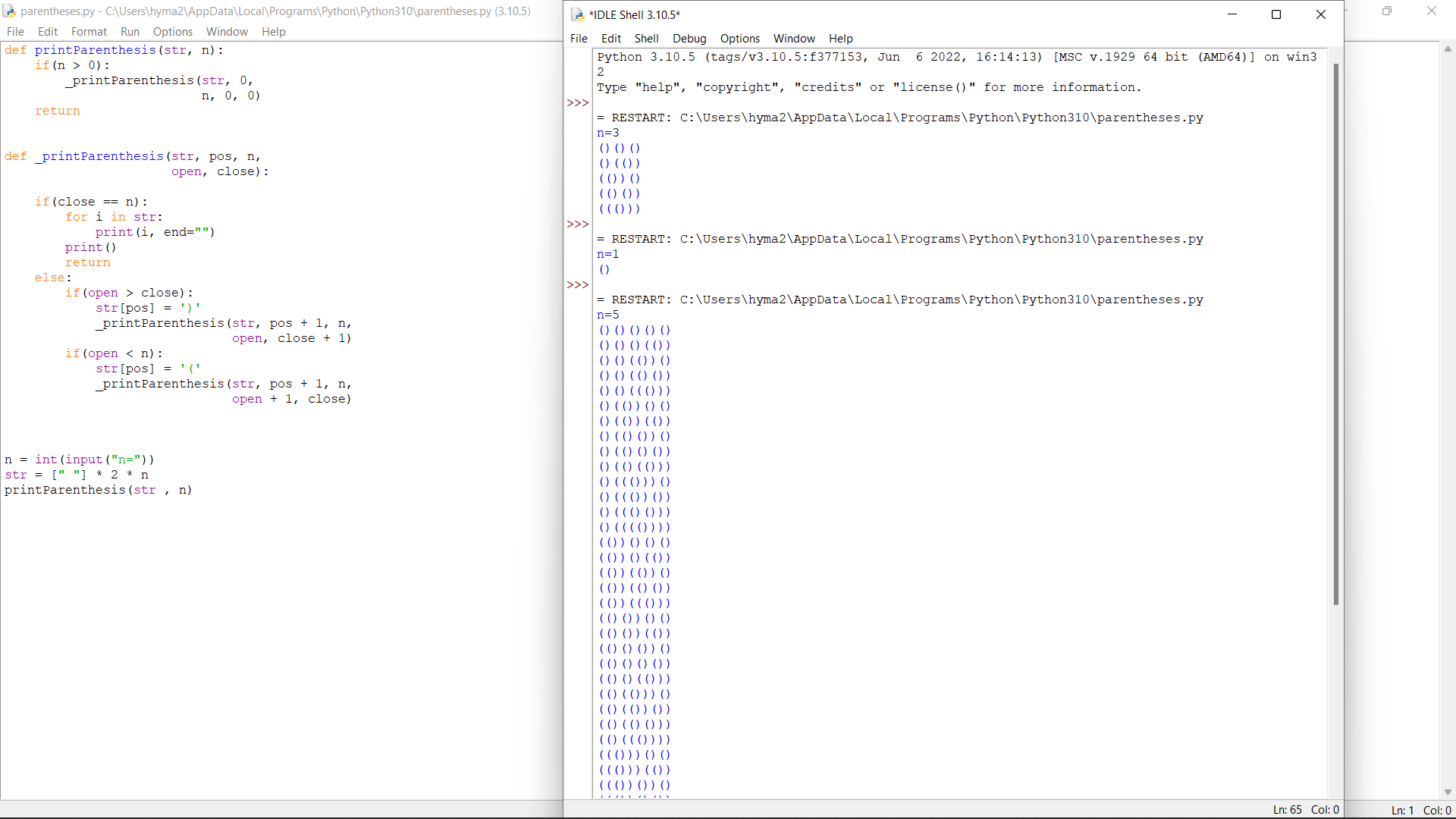
\_printParenthesis(str, pos + 1, n,

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)

8. Given an input string s and a pattern p, implement regular expression matching with support for '.' and '\*' where: • '.' Matches any single character. • '\*' Matches zero or more of the preceding element. The matching should cover the entire input string (not partial). Test case: 1.Input: s = "aa", p = "a" Output: false 2.Input: s = "aa", p = "a\*" Output: true 3.Input: s = "ab", p = ".\*" Output: true 4.Input: s = " aaa", p = "aa" Output: true 5.Input: s = "aab", p = "c\*a\*b" Output: true

class Solution(object):

def isMatch(self, text, pattern):

memo = {}

def dp(i, j):

if (i, j) not in memo:

if j == len(pattern):

ans = i == len(text)

else:

first\_match = i < len(text) and pattern[j] in {text[i], '.'}

if j+1 < len(pattern) and pattern[j+1] == '\*':

ans = dp(i, j+2) or first\_match and dp(i+1, j)

else:

ans = first\_match and dp(i+1, j+1)

memo[i, j] = ans

return memo[i, j]

return dp(0, 0)