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)):

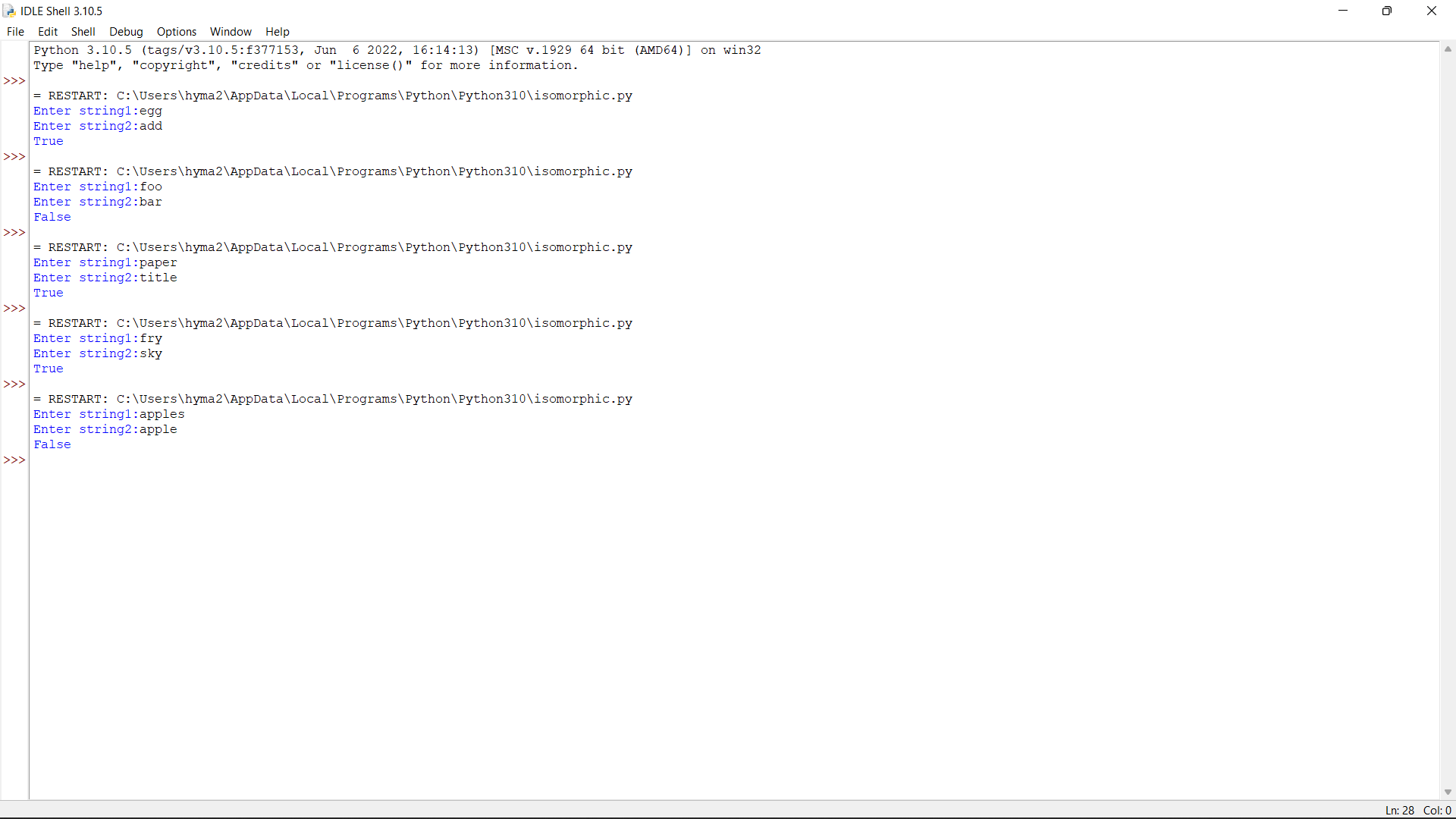
return False

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):

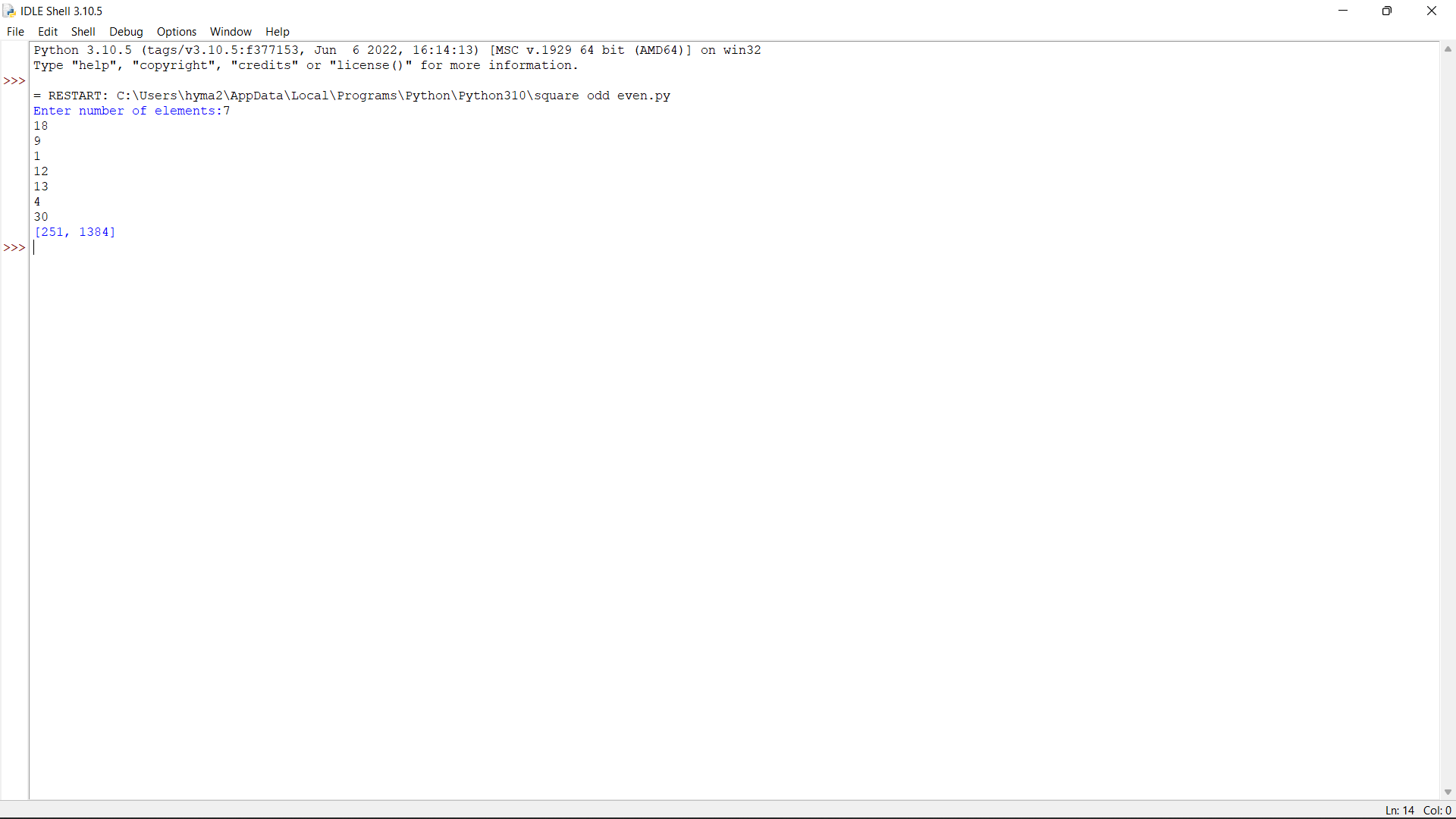
sum2=sum2+lst[i]\*\*2

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)

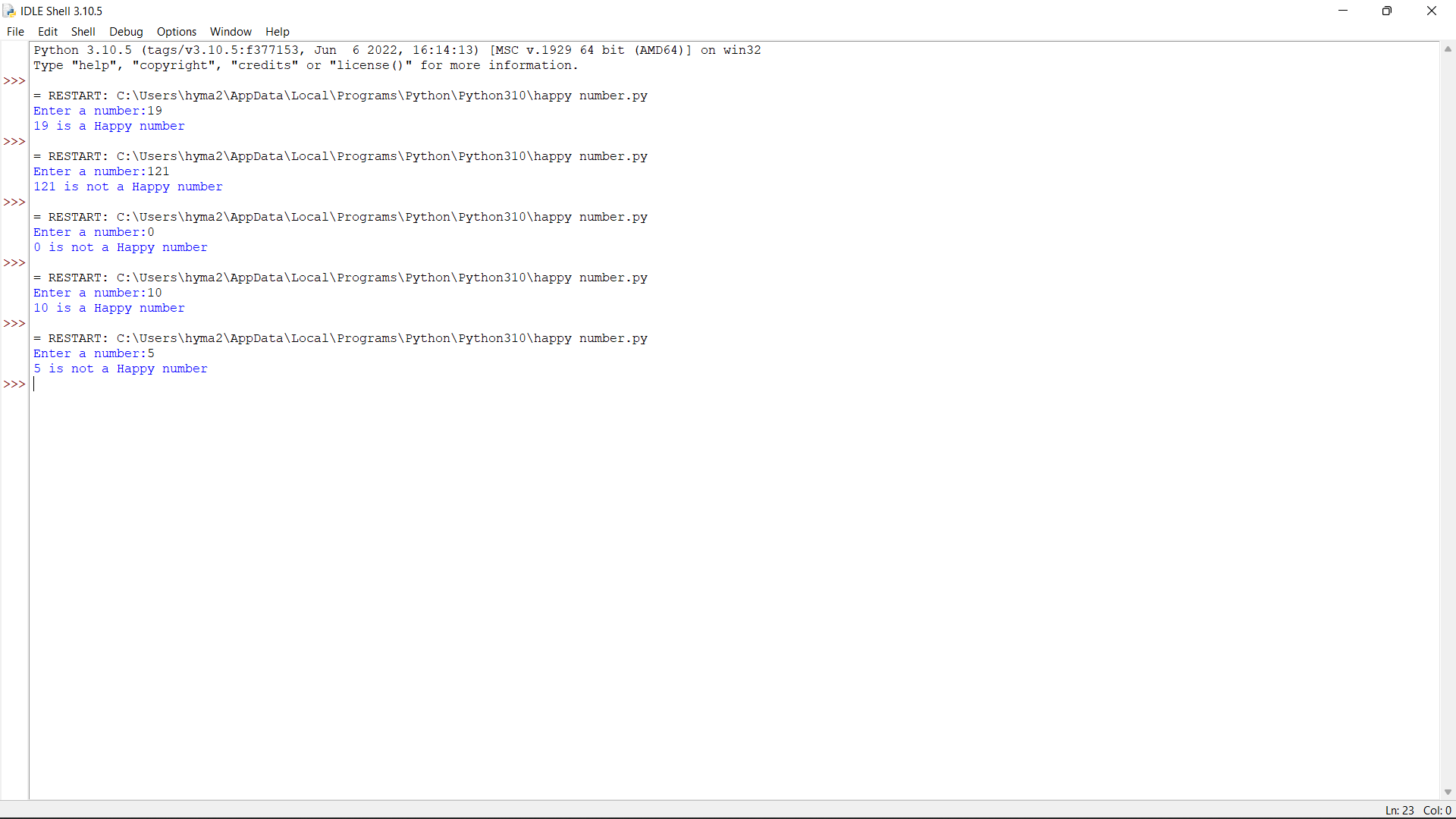
n =int(input("Enter a number:"))

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

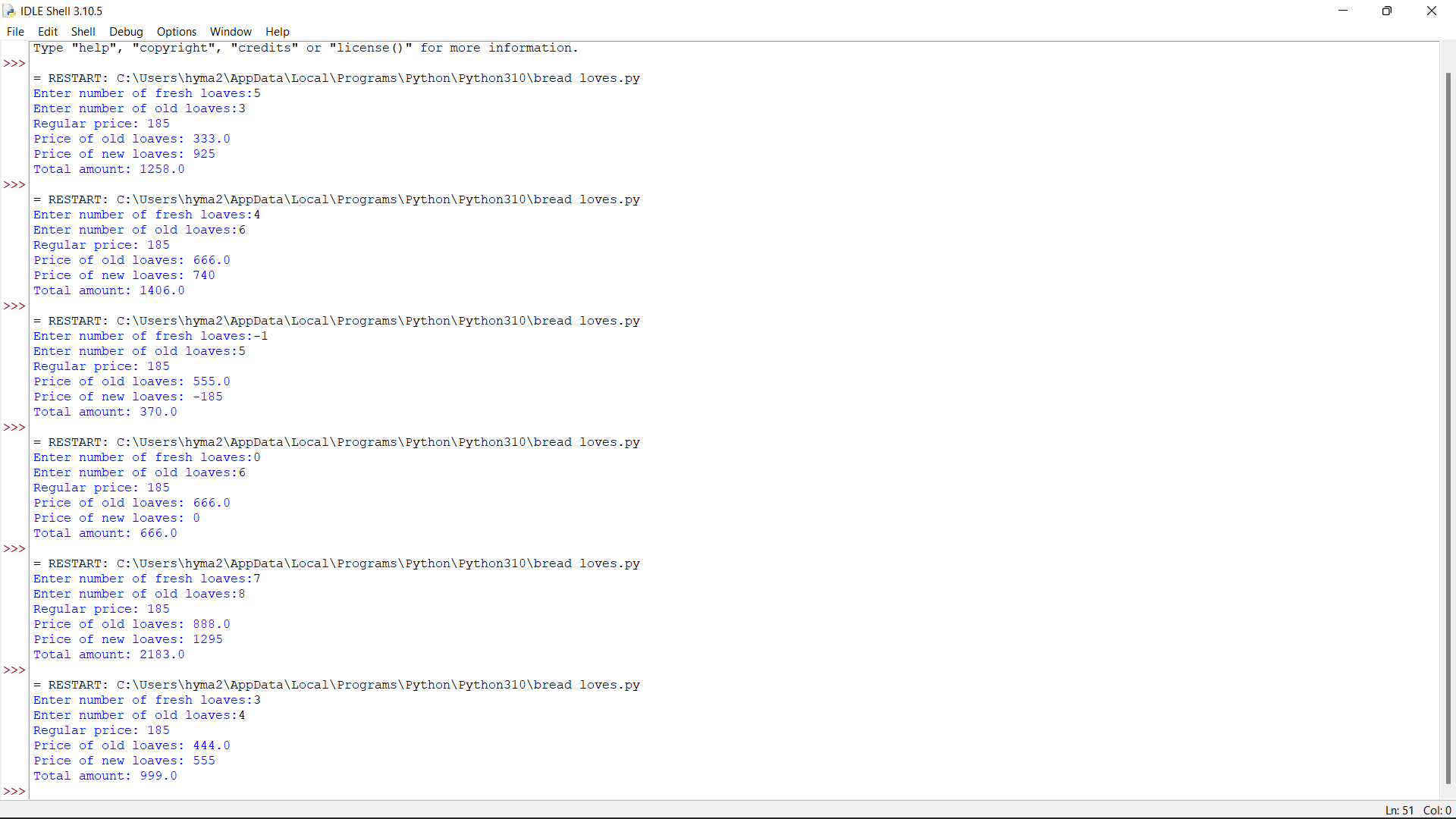
total=c+d

print("Regular price:",e)

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

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

print("Total amount:",total)



6. Given n non-negative integers a1,a2,a3,…an where each represents a point at coordinate (i, ai) . „ n „ vertical lines are drawn such that the two endpoints of line i is at (i, ai) and (i,0). Find two lines, which together with x-axis forms a container, such that the container contains the most water. The program should return an integer which corresponds to the maximum area of water that can be contained (maximum area instead of maximum volume sounds weird but this is the 2D plane we are working with for simplicity). Note: You may not slant the container.

Test case:

1.Input: array = [1, 5, 4, 3] Output: 6

2.Input: array = [3, 1, 2, 4, 5] Output: 12

3.Input: array = [1,8,6,2,5,4,8,3,7] Output: 49

4.Input: array = [1,1] Output: 1

5.Input: array = [7,3] Output: 3

def maxArea(A, Len) :

area = 0

for i in range(Len) :

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

# Calculating the max area

area = max(area, min(A[j], A[i]) \* (j - i))

return area

# Driver code

a = [ 1, 5, 4, 3 ]

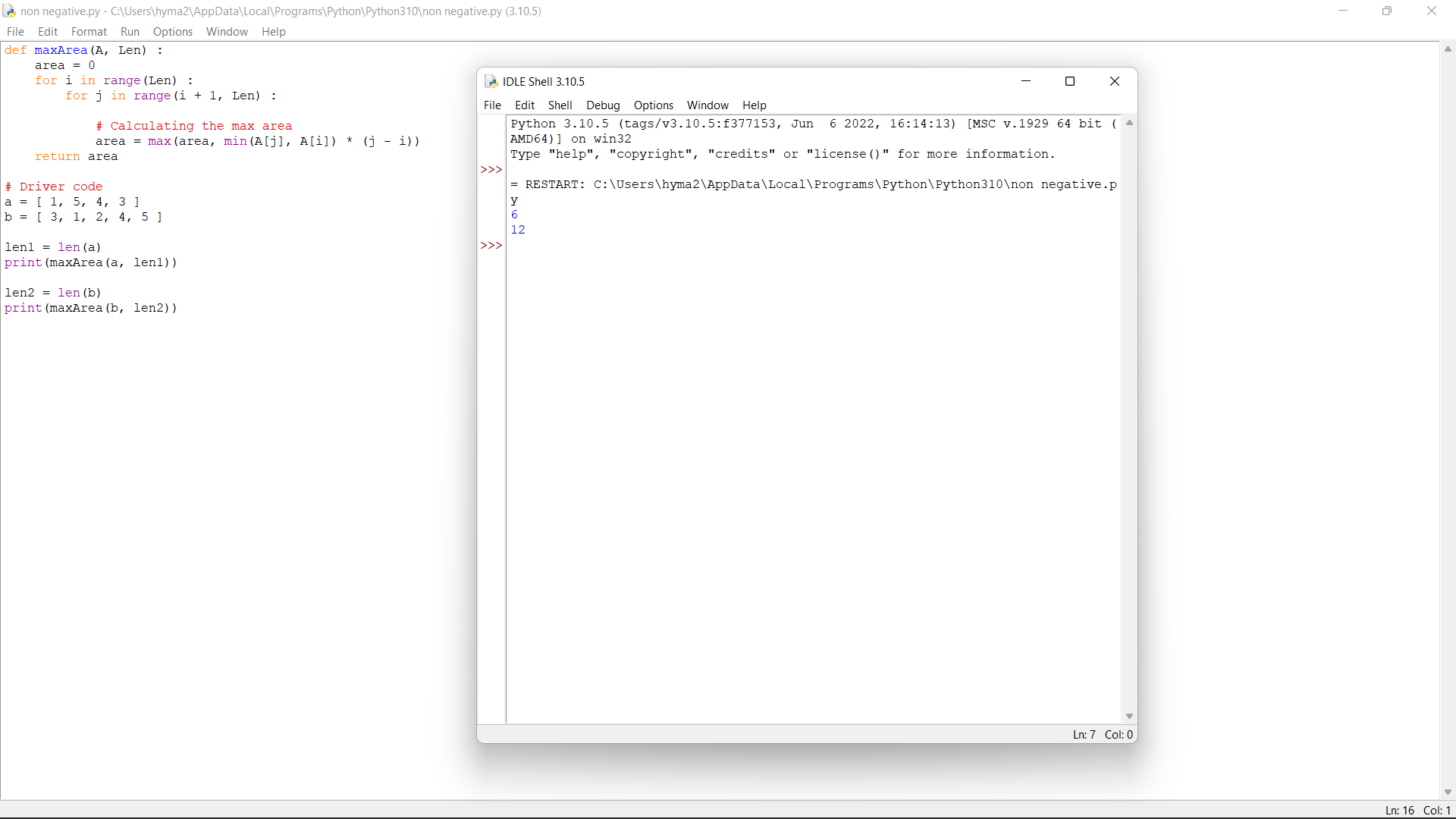
b = [ 3, 1, 2, 4, 5 ]

len1 = len(a)

print(maxArea(a, len1))

len2 = len(b)

print(maxArea(b, len2))



7. 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:

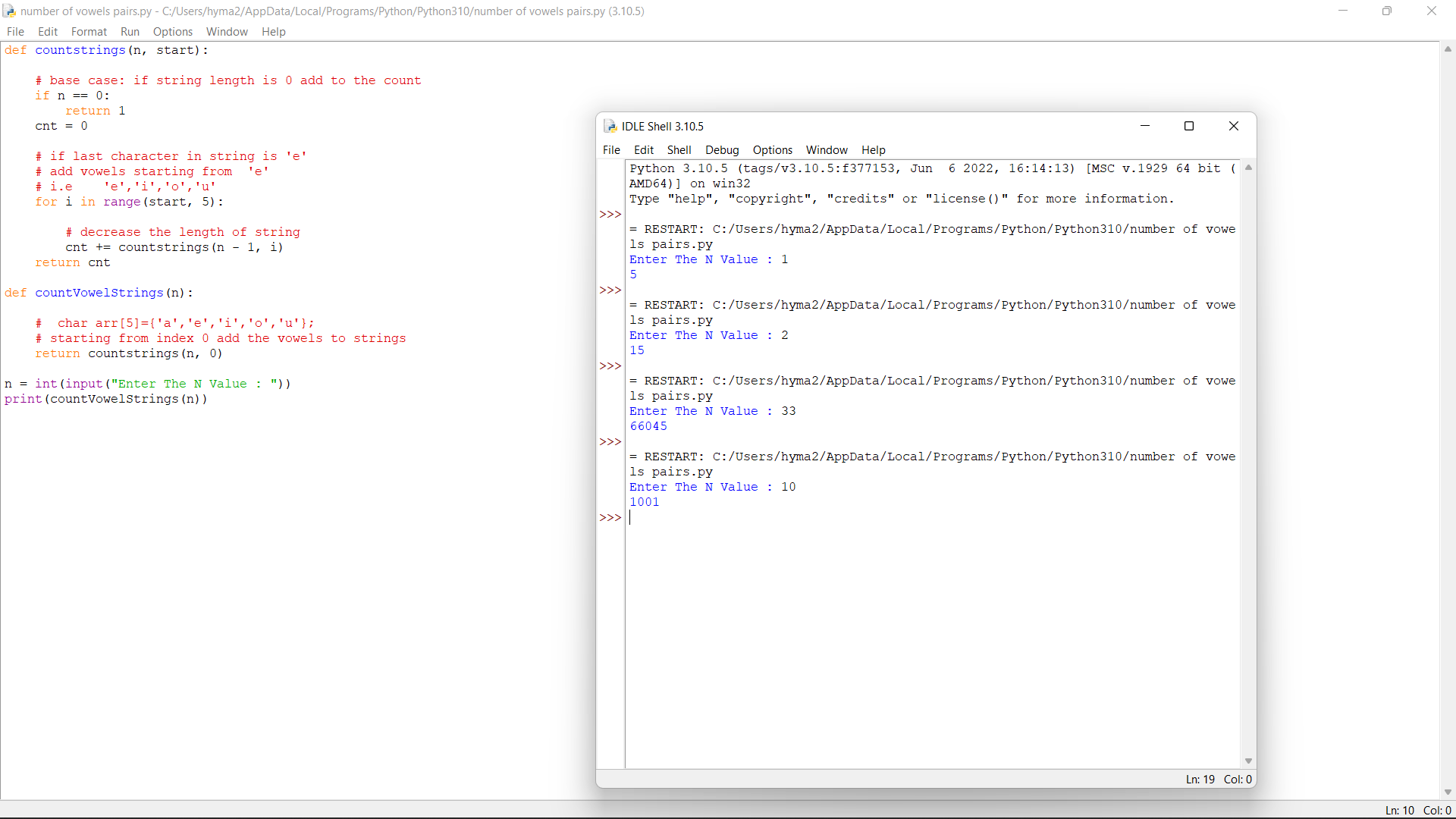
1.Input: n = 1 Output: 5 Explanation: The 5 sorted strings that consist of vowels only are ["a","e","i","o","u"].

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=-5

5.n=10



8. A valid number can be split up into these components (in order): 1. A decimal number or an integer. 2. (Optional) An 'e' or 'E', followed by an integer. A decimal number can be split up into these components (in order): 1. (Optional) A sign character (either '+' or '-'). 2. One of the following formats: 1. One or more digits, followed by a dot '.'. 2. One or more digits, followed by a dot '.', followed by one or more digits. 3. A dot '.', followed by one or more digits. An integer can be split up into these components (in order): 1. (Optional) A sign character (either '+' or '-'). 2. One or more digits. For example, all the following are valid numbers: ["2", "0089", "-0.1", "+3.14", "4.", "-.9", "2e10", "-90E3", "3e+7", "+6e-1", "53.5e93", "-123.456e789"], while the following are not valid numbers: ["abc", "1a", "1e", "e3", "99e2.5", "--6", "-+3", "95a54e53"]. Given a string s, return true if s is a valid number. Test cases: 1.Input: s = "0" Output: true 2.Input: s = "e" Output: false 3.Input: s = " " Output: false 4.Input: s = "." Output: false 5.Input: s = "%" Output: false

def isNumber(s):

valid\_patterns = ('xd','x.d','xd.','xd.d','xsd','xsd.',

'xsd.d','xded','x.ded','xd.ed','xd.ded',

'xsded','xsd.ed','xsd.ded', 'xdesd', 'x.desd',

'xd.esd', 'xd.desd', 'xsdesd', 'xsd.esd',

'xsd.desd', 'xs.d', 'xs.ded', 'xs.desd'

)

patttern = 'x'

for ch in s:

if ch == '+' or ch == '-': patttern += 's'

elif ch == 'e' or ch == 'E': patttern += 'e'

elif ch.isdigit():

if patttern[-1] != 'd': patttern += 'd'

elif ch == '.': patttern += '.'

else: return False

return patttern in valid\_patterns

10.Modify string by replacing characters by alphabets whose distance from that character is equal to its frequency Given a string S consisting of N lowercase alphabets, the task is to modify the string S by replacing each character with the alphabet whose circular distance from the character is equal to the frequency of the character in S. Testcases: 1.Input: S=“ghee” Output: hggi Explanation: The following modifications are done on the string S: • The frequency of „g‟ in the string is 1. Therefore, „g‟ is replaced by „h‟. • The frequency of „e‟ in the string is 2. Therefore, „e‟ is replaced by „g‟. • The frequency of „e‟ in the string is 2. Therefore, „e‟ is replaced by „g‟. • The frequency of „h‟ in the string is 1. Therefore, „h‟ is converted to „i‟. modified string S is “hggi”. 2.S= “elephant” 3.S= “apple” 4. S= “orange” 5. S= “lion”

def addFrequencyToCharacter(s):

# Stores frequency of characters

frequency = [0] \* 26

# Stores length of the string

n = len(s)

# Traverse the given string S

for i in range(n):

# Increment frequency of

# current character by 1

frequency[ord(s[i]) - ord('a')] += 1

# Traverse the string

for i in range(n):

# Store the value to be added

# to the current character

add = frequency[ord(s[i]) - ord('a')] % 26

# Check if after adding the

# frequency, the character is

# less than 'z' or not

if (ord(s[i]) + add <= ord('z')):

s[i] = chr(ord(s[i]) + add)

# Otherwise, update the value of

# add so that s[i] doesn't exceed 'z'

else:

add = (ord(s[i]) + add) - (ord('z'))

s[i] = chr(ord('a') + add - 1)

# Print the modified string

print("".join(s))

# Driver Code

if \_\_name\_\_ == '\_\_main\_\_':

str = "apple"

addFrequencyToCharacter([i for i in str])

