1.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"].

class Solution(object):

def isNumber(self, s):

s = s.strip()

try:

s = float(s)

return True

except:

return False

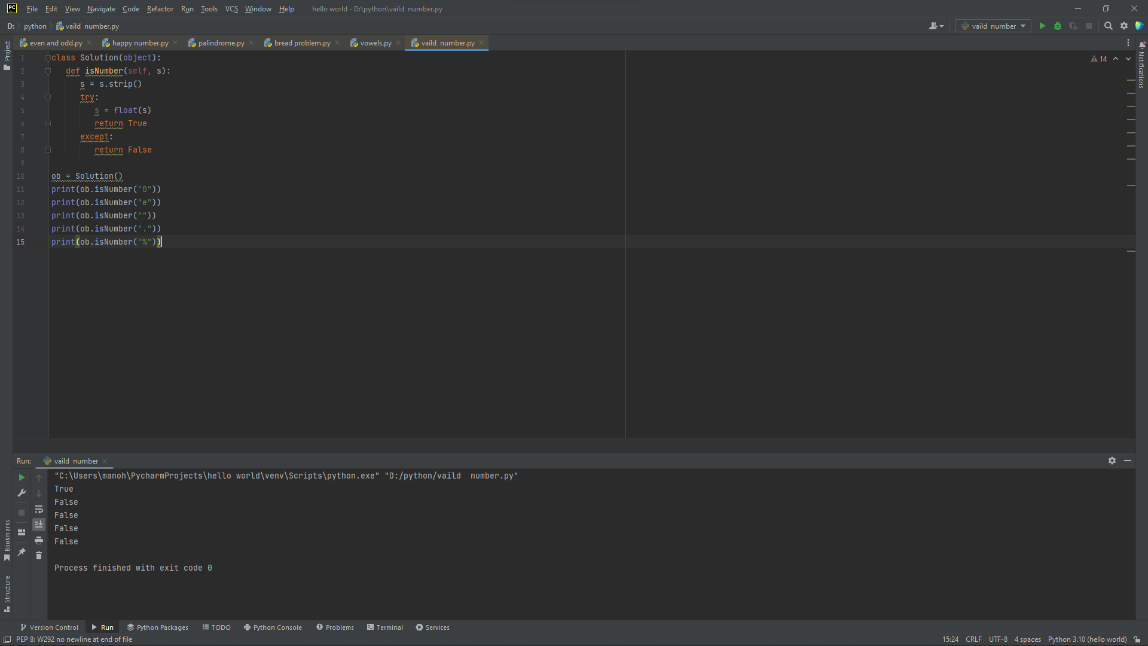
ob = Solution()

print(ob.isNumber("0"))

print(ob.isNumber("e"))

print(ob.isNumber(""))

print(ob.isNumber("."))

print(ob.isNumber("%"))

2.A party has been organised on a cruise. The party is organised for a limited time (T). The number of guests entering (E[i]) and leaving (L[i]) the party at every hour is represented as items of the list. The task is to find the maximum number of guests present on the cruise at any given instance within T hours. Sample Input: 5 ---> Value of T [7,0,5,1,3] ---> E[], element of E[0] to E[N-1], where input each element is separated by new line [1,2,1,3,4] -----> L[],element of L[0] to L[N-1], where input each element is separated by new line Sample Output: 8 -----> Maximum number of guests on cruise at an instance

E=[]

L=[]

T=int(input("range T:"))

for i in range(T):

e=int(input("E:"))

E.append(e)

for i in range(T):

l=int(input("L:"))

L.append(l)

Sum=0

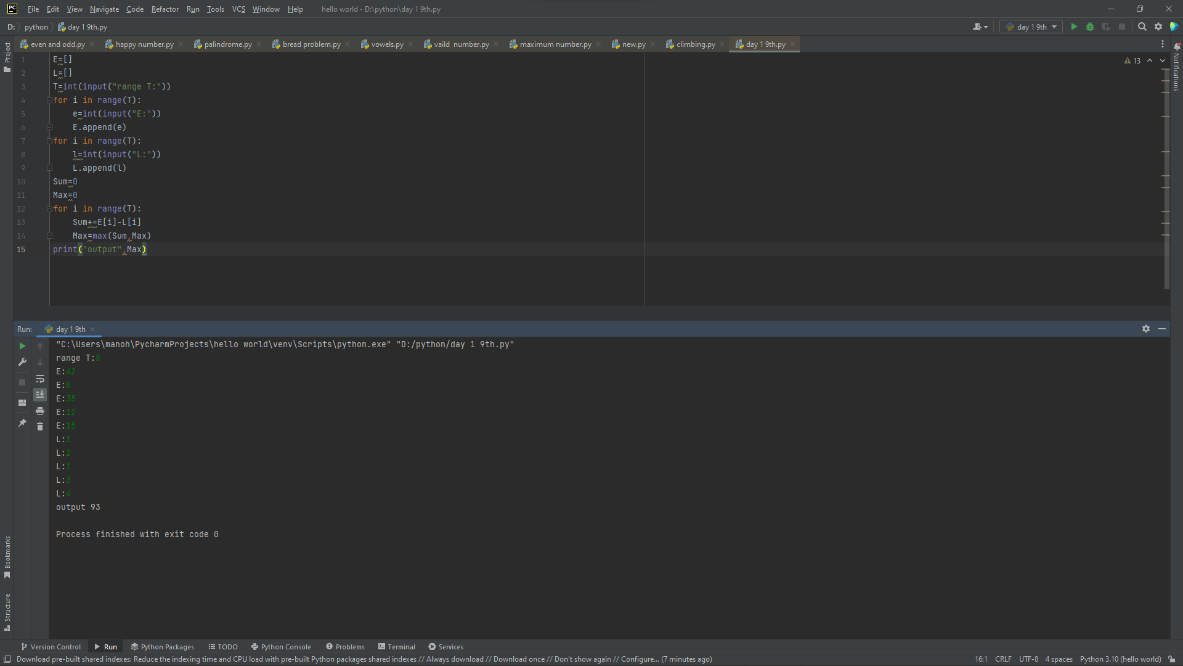
Max=0

for i in range(T):

Sum+=E[i]-L[i]

Max=max(Sum,Max)

print("output",Max)



3.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

def addFrequencyToCharacter(s):

frequency = [0] \* 26

n = len(s)

for i in range(n):

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

for i in range(n):

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

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

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

else:

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

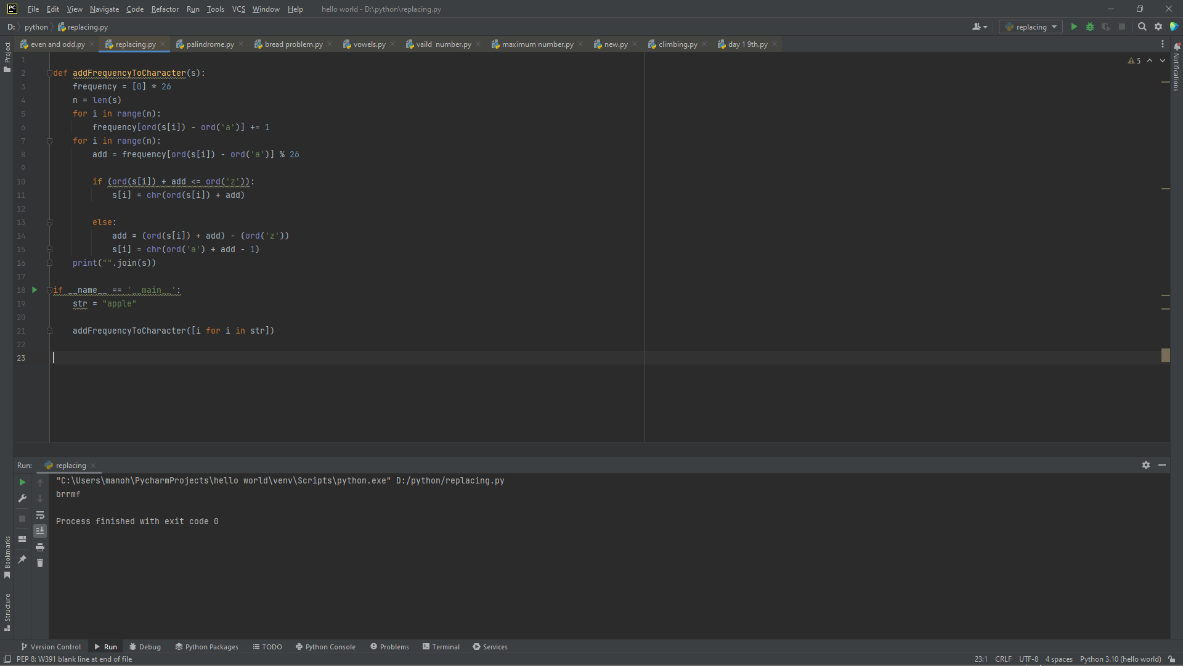
s[i] = chr(ord('a') + add - 1)

print("".join(s))

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

str = input("S=")

addFrequencyToCharacter([i for i in str])



4.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:

month = input ("Enter the name of the month:")

day = int (input ("Enter the day number:"))

if month == "January" or month == "February":

season = "Winter"

elif month == "March":

if day < 20:

season = "Winter"

else :

season = "summer"

elif month == "April" or month == "May":

season = "summer"

elif month == "June":

if day < 21:

season = "summer"

else :

season = "spring"

elif month == "July" or month == "August":

season = "Summer"

elif month == "September":

if day < 22:

season = "fall"

else :

season = "winter"

elif month == "October" or month == "November":

season = "fall"

elif month == "December":

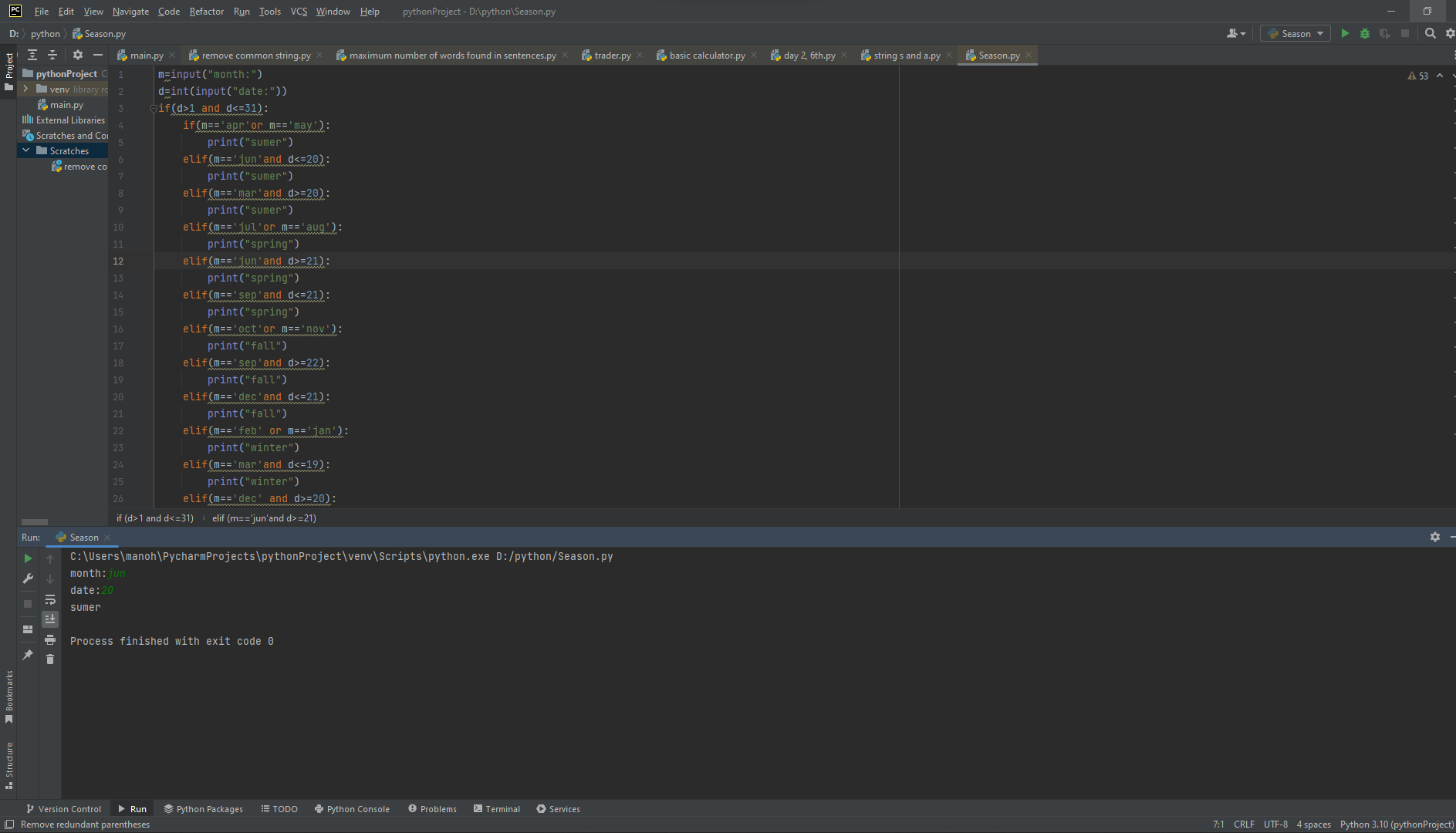
if day < 21:

season = "winter"

else :

season = "summer"

print (month, day, "is in", season)



5.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.

def commonWords(sent1,sent2):

sen1=set(sent1)

sen2=set(sent2)

common=list(sen1.intersection(sen2))

return common

def removeCommonWords(sent1,sent2):

sentence1=list(sent1.split())

sentence2=list(sent2.split())

commonlist=commonWords(sentence1,sentence2)

word=0

for i in range(len(sentence1)):

if sentence1[word] in commonlist:

sentence1.pop(word)

word=word-1

word+=1

word=0

for i in range(len(sentence2)):

if sentence2[word] in commonlist:

sentence2.pop(word)

word=word-1

word+=1

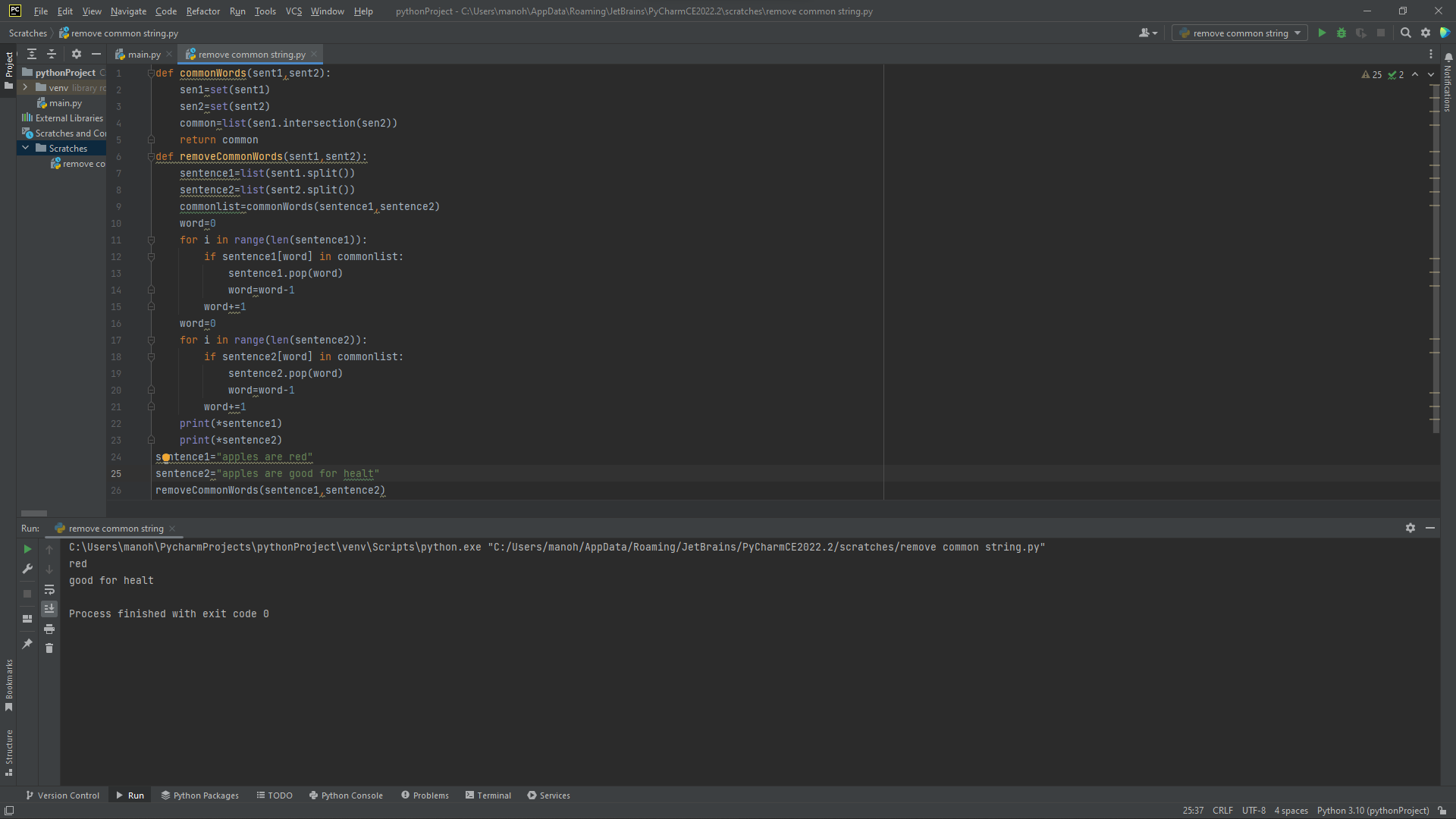
print(\*sentence1)

print(\*sentence2)

sentence1="apples are red"

sentence2="apples are good for healt"

removeCommonWords(sentence1,sentence2)



6.Given a list of strings strs, group the anagrams together. You can return the answer in any order. An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. strs[i] consists of lowercase English letters.

class Solve():

def Anagrams( self, li ):

dictionary = {}

for word in li:

sortedWord = ''.join(sorted(word))

if sortedWord not in dictionary:

dictionary[sortedWord] = [word]

else:

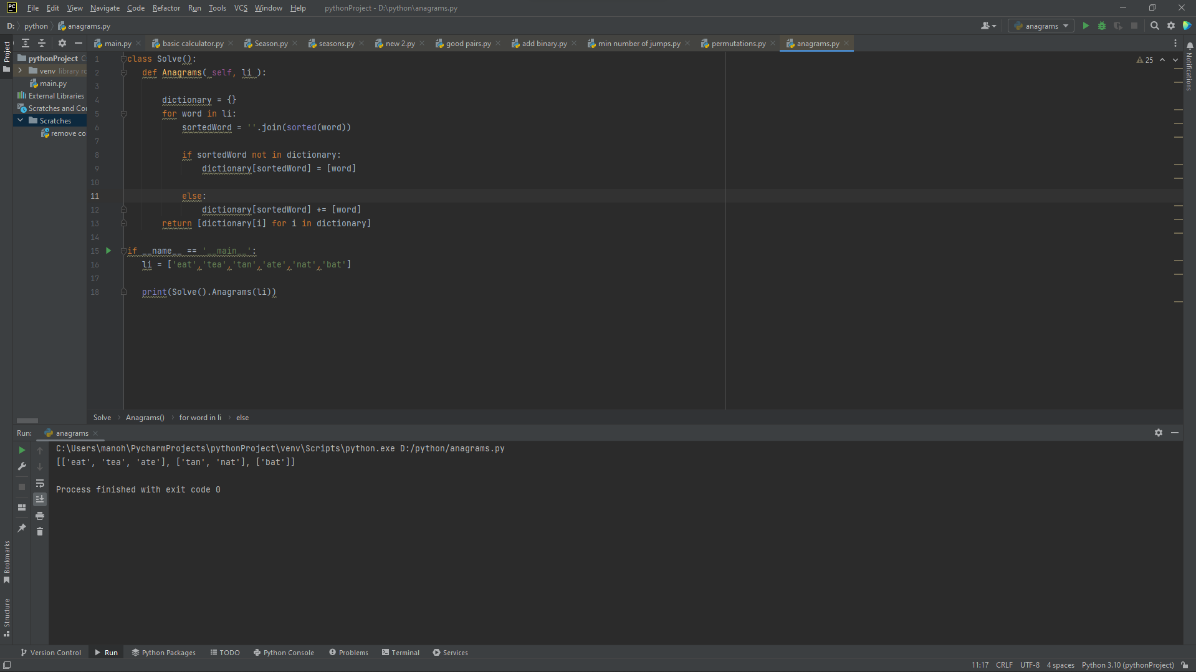
dictionary[sortedWord] += [word]

return [dictionary[i] for i in dictionary]

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

li = ['eat','tea','tan','ate','nat','bat']

print(Solve().Anagrams(li))



7.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

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),

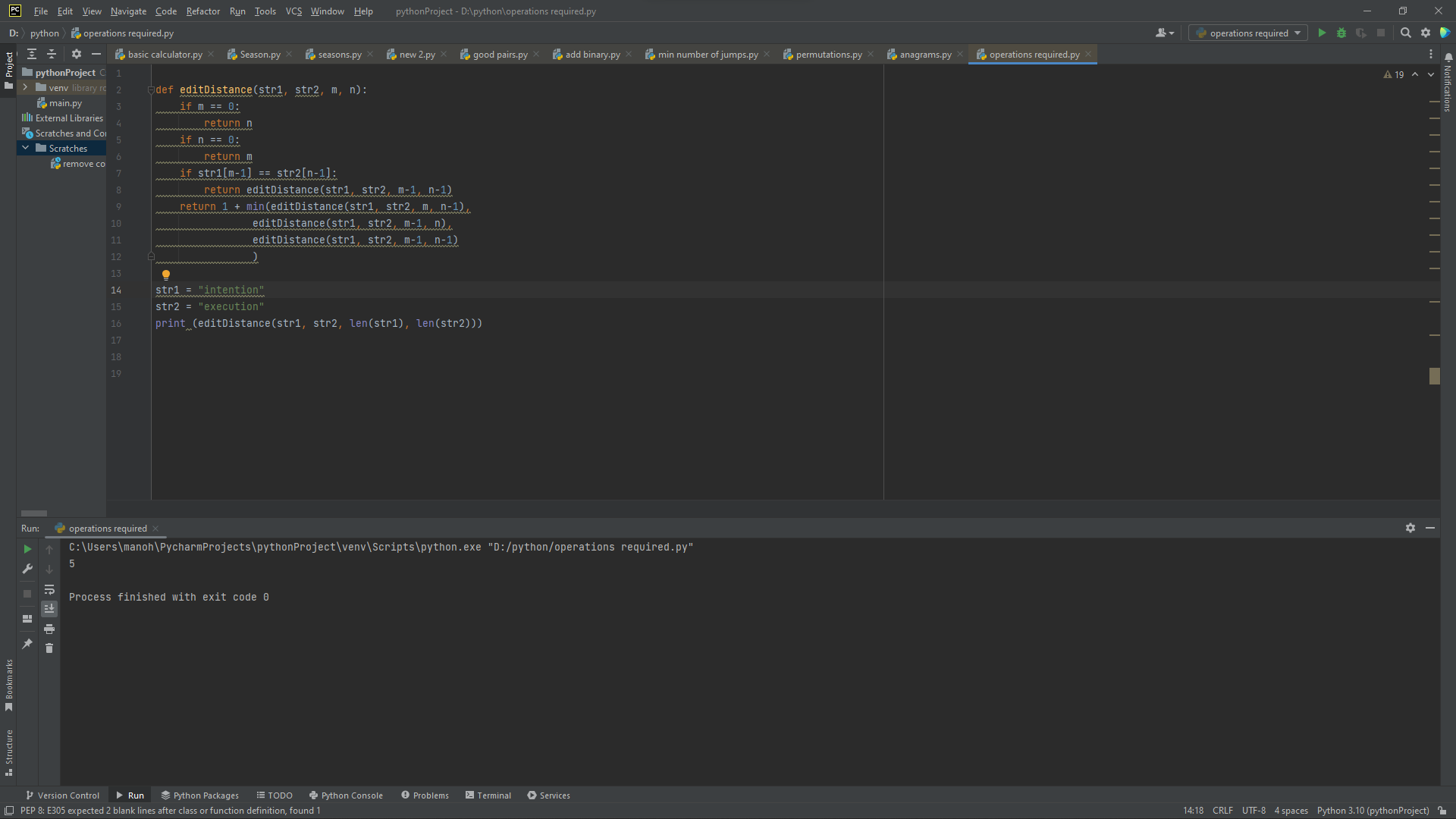
editDistance(str1, str2, m-1, n-1)

)

str1 = "intention"

str2 = "execution"

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



8.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.

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

if \_\_name\_\_ == "\_\_main\_\_":

S1 = "abced"

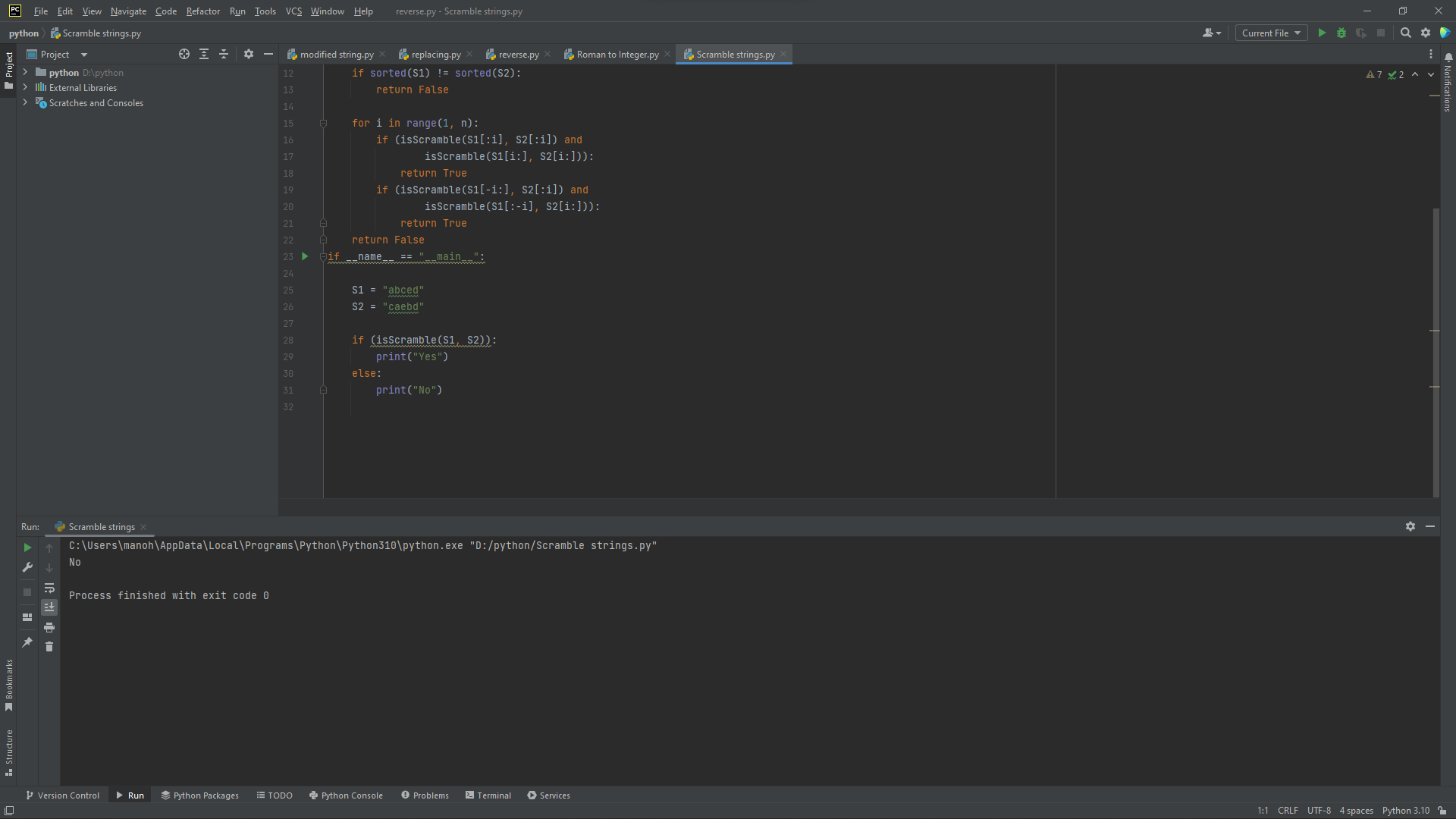
S2 = "caebd"

if (isScramble(S1, S2)):

print("Yes")

else:

print("No")



10.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.

string = input("Enter the string: ")

s = string.split()[::-1]

l = []

for i in s:

l.append(i)

print(" ".join(l))

