**Name-Ujjawal Mandhani**

**Batch-F6**

**Enroll no-9918103237**

**OSS LAB-3**

**Question 1: Write a program to count frequency of characters in a paragraph.**

def count\_letters(para):

result = {}

for letter in para:

if letter not in result:

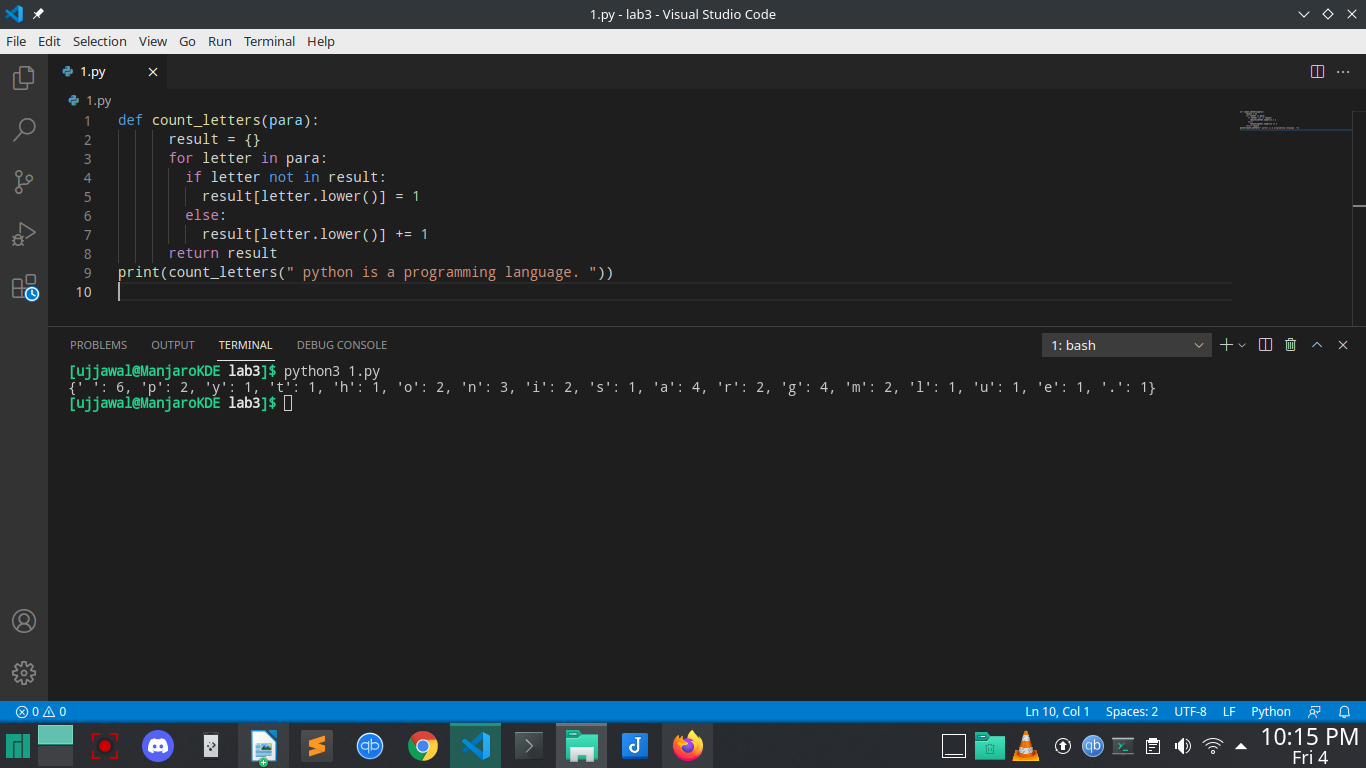
result[letter.lower()] = 1

else:

result[letter.lower()] += 1

return result

print(count\_letters(" python is a programming language. "))



**Question 2:****Write a program wrap.py that takes filename and number of characters per line (width)**

**as arguments. The program must wrap the lines of the file longer than entered width.**

def wrap(f,n):

fi=open(f,"w+")

rl=fi.readlines()

for i in range(0,len(rl)):

if len(rl[i]) > n:

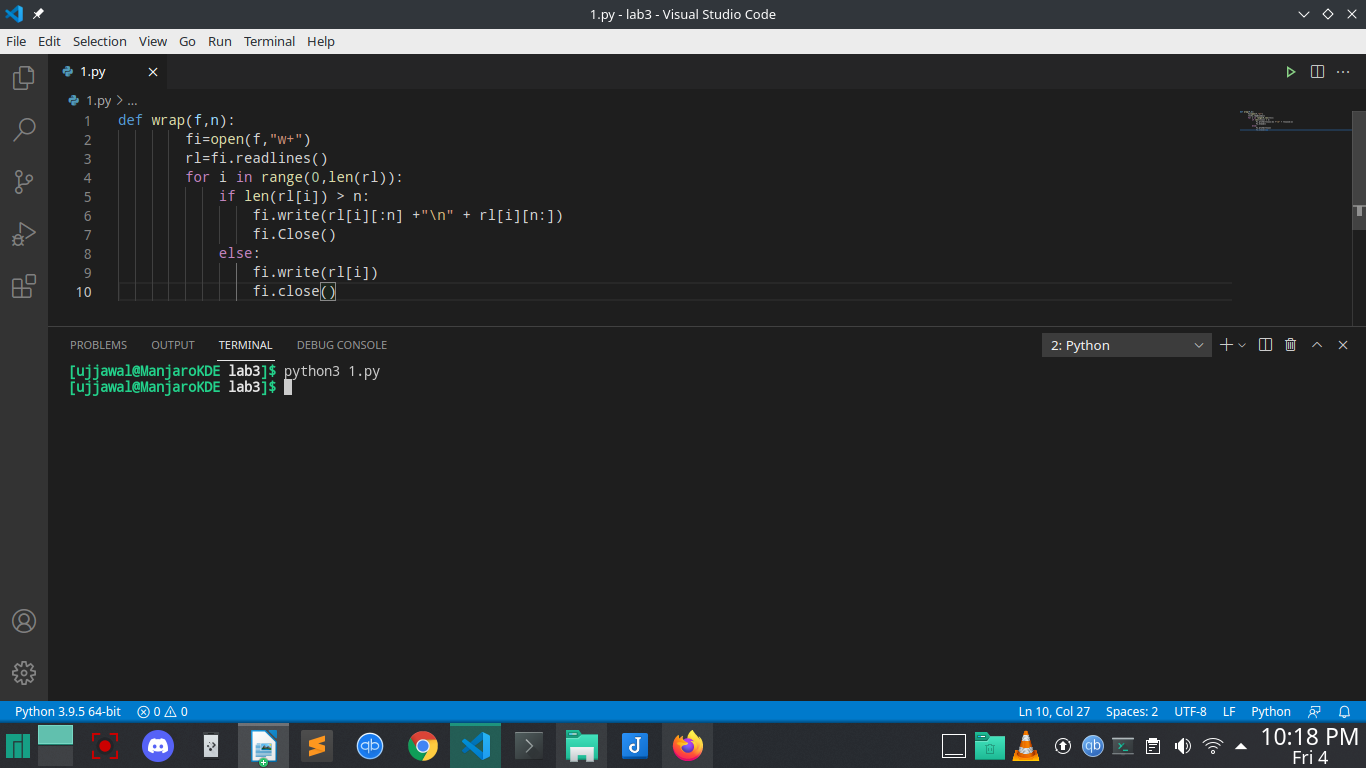
fi.write(rl[i][:n] +"\n" + rl[i][n:])

fi.Close()

else:

fi.write(rl[i])

**fi.close()**



**Question 3:****Python has a built-in function ‘map’ that applies a function to each element of a list.**

**Provide an implementation for map using list comprehensions.**

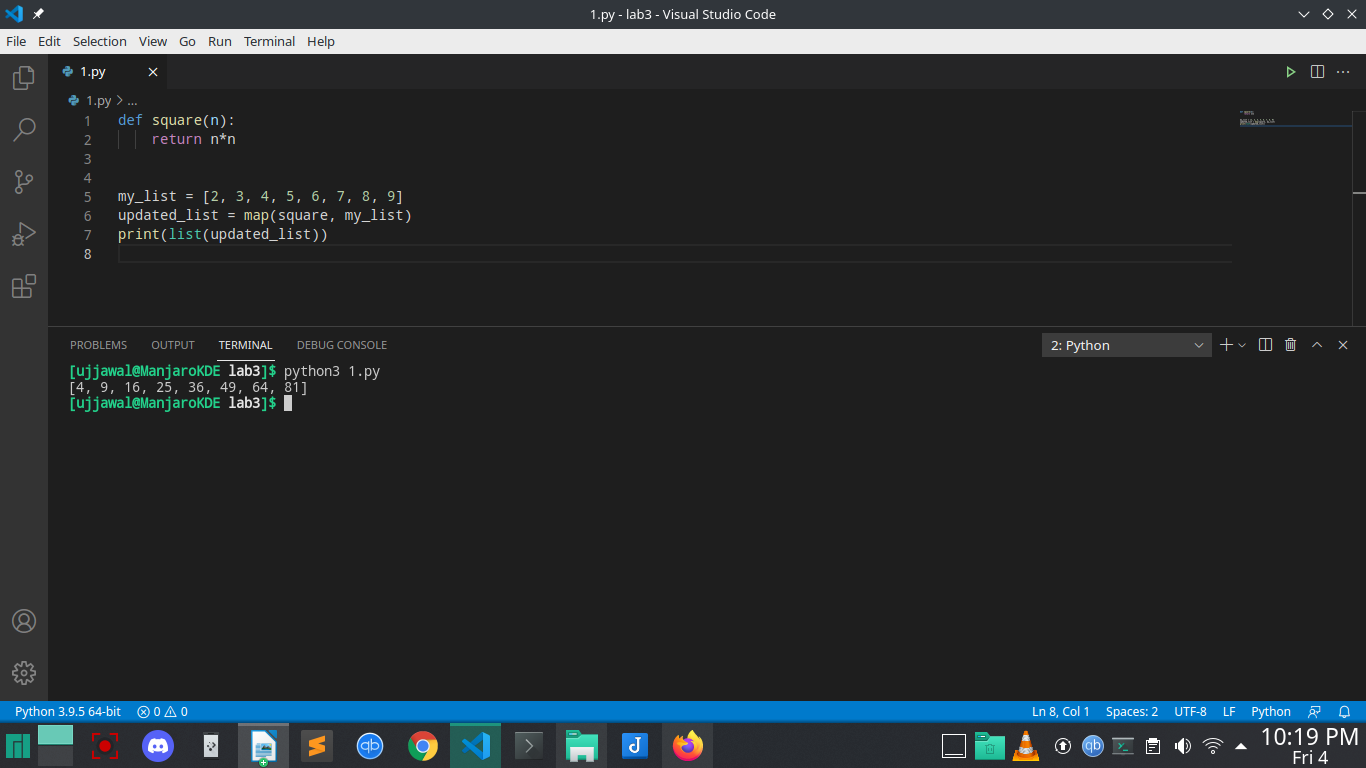
def square(n):

return n\*n

my\_list = [2,3,4,5,6,7,8,9]

updated\_list = map(square, my\_list)

print(list(updated\_list))



**Question 4:****Python has a built-in function ‘filter(f, a)’ that returns items of the list ‘a’ for which**

**f(item) returns true. Provide an implementation for filter using list comprehensions**

def is\_even(x):

if x % 2 == 0:

return True

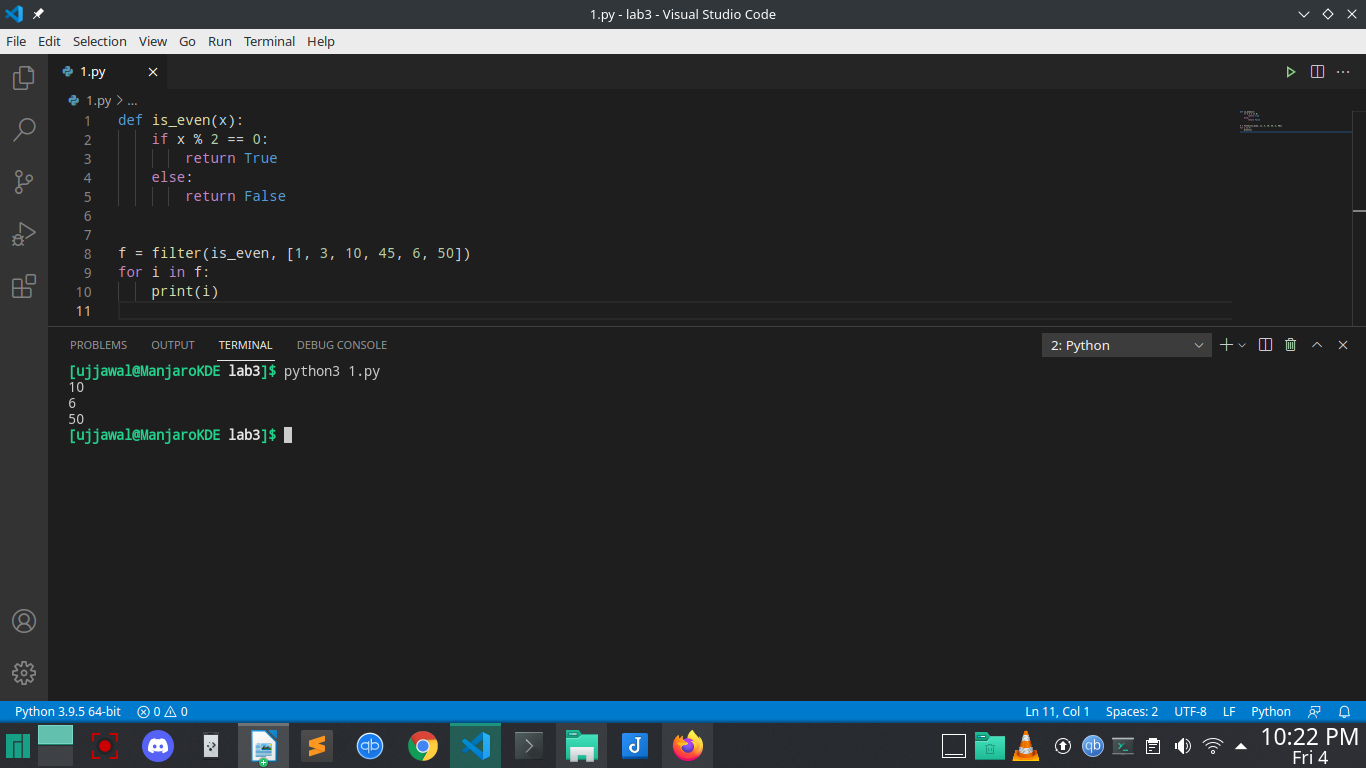
else:

return False

f = filter(is\_even, [1, 3, 10, 45, 6, 50])

for i in f:

print(i)



**Question 5:****Write a function triplets that takes a number ‘n’ as argument and returns a list of triplets**

**such that sum of first two elements of the triplet equals the third element using**

**numbers below n. Please note that (a,b, c) and (b, a, c) represent same triplet.**

def write\_triplets(n):

to\_return = []

if n >= 2:

for i in range(2, n):

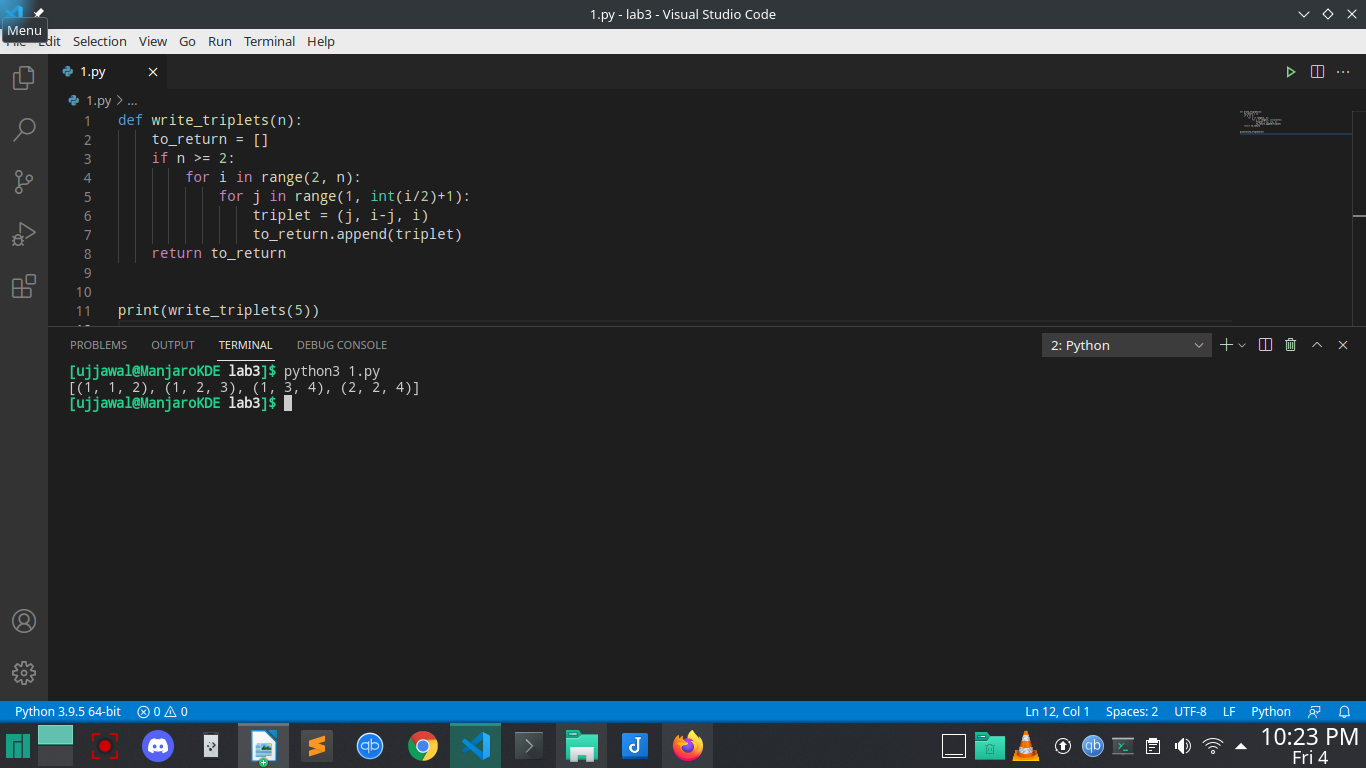
for j in range(1, int(i/2)+1):

triplet = (j, i-j, i)

to\_return.append(triplet)

return to\_return

print(write\_triplets(5))



**Question 6:Write a python function ‘parse\_csv’ to parse csv (comma separated values) files.**

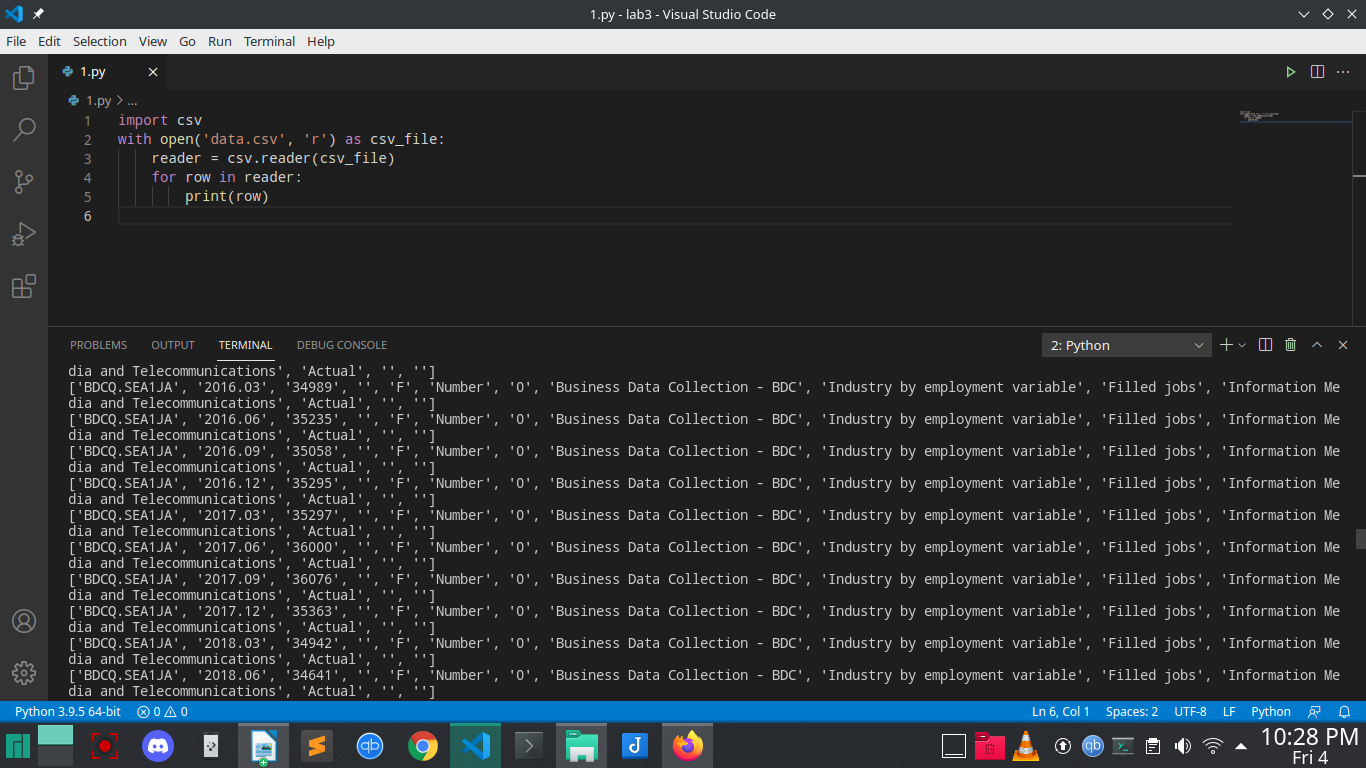
import csv

with open('data.csv', 'r') as csv\_file:

reader = csv.reader(csv\_file)

for row in reader:

print(row)



**Question 7:****Generalize the above implementation of csv parser to support any delimiter and**

**comments.**

def parse\_csv(filename, d, c):

lines=open(filename).readlines()

new=[]

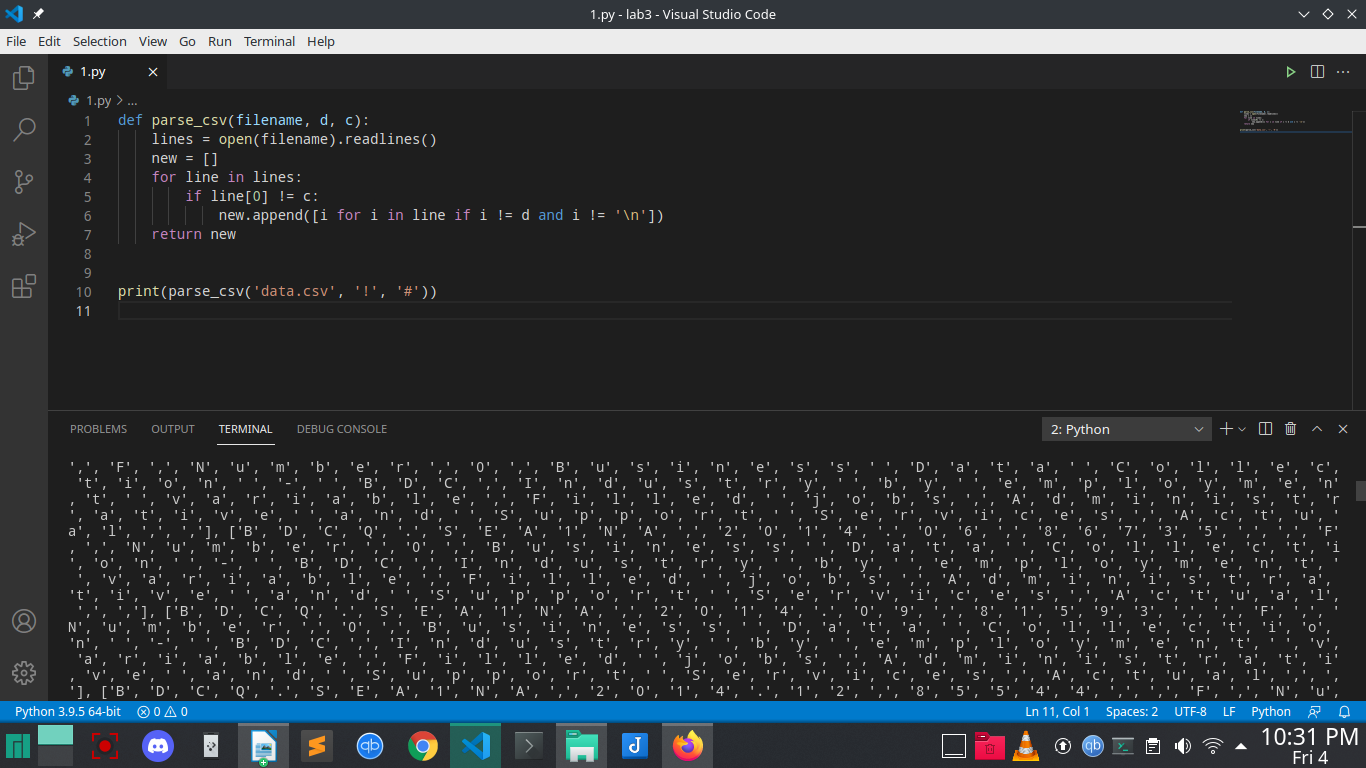
for line in lines:

if line[0] != c:

new.append([i for i in line if i!=d and i!='\n'])

return new

print(parse\_csv('data.txt', '!', '#'))



**Question 8:****Write a function ‘mutate’ to compute all words generated by a single mutation on a**

**given word. A mutation is defined as inserting a character, deleting a character,**

**replacing a character, or swapping 2 consecutive characters in a string. For simplicity**

**consider only letters from a to z.**

def mutate(d):

ret=[d]

i=0

l=len(d)

alp=map(chr,range(97,123))

while i<l:

cop=d

ret.append(cop[:i]+cop[i+1:])

if i<l-2:

ret.append(cop[:i]+cop[i+1]+cop[i]+cop[i+2:])

elif i<l-1:

ret.append(cop[:i]+cop[i+1]+cop[i])

for x in alp:

ret.append(cop[:i]+x+cop[i+1:])

for x in alp:

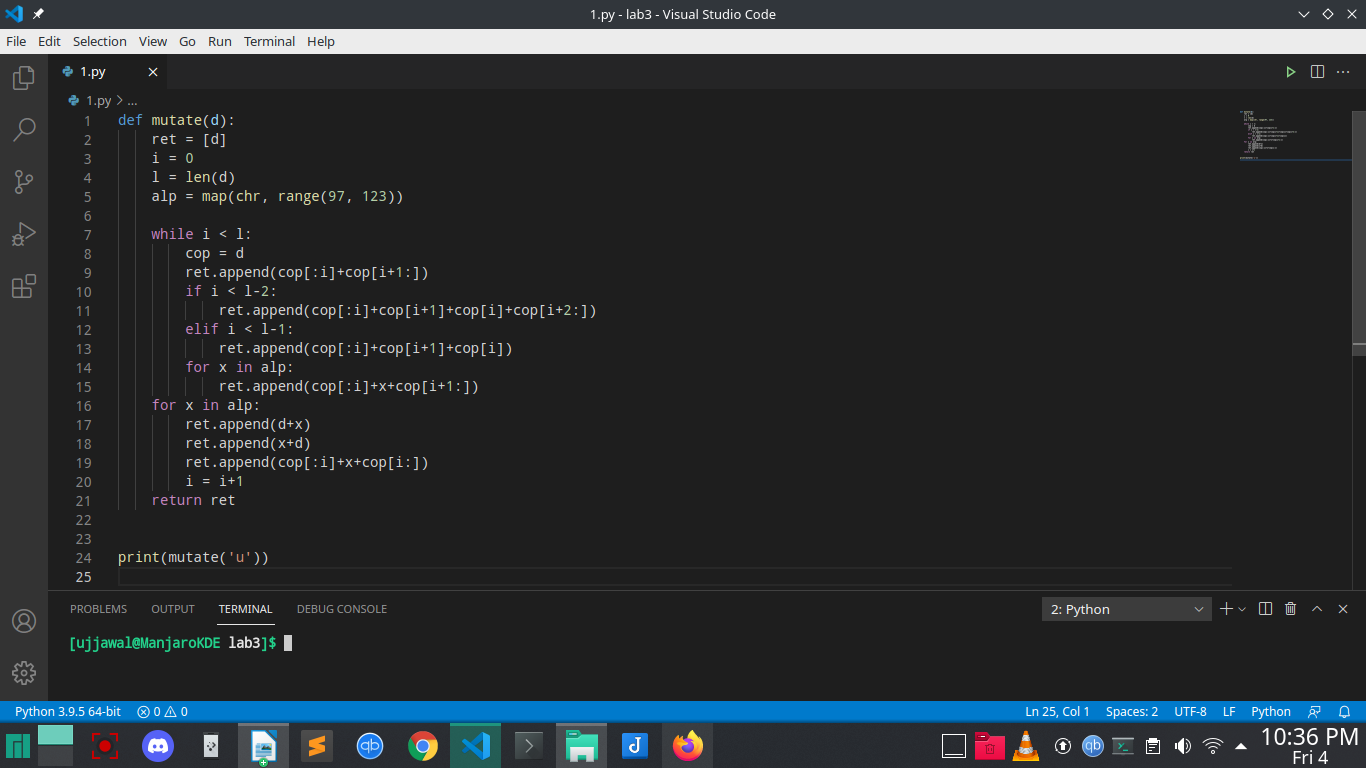
ret.append(d+x)

ret.append(x+d)

ret.append(cop[:i]+x+cop[i:])

i=i+1

return ret



**Question 9:****Write a function ‘nearly\_equal’ to test whether two strings are nearly equal. Two strings**

‘**a’ and ‘b’ are nearly equal when ‘a’ can be generated by a single mutation on ‘b’.**

def nearly\_equal(str1,str2):

count=0

i=j=0

while(i<len(str1) and j<len(str2)):

if(str1[i]!=str2[j]):

count=count+1

if(len(str1)>len(str2)):

i=i+1

elif(len(str1)==len(str2)):

pass

else:

i=i-1

if(count>1):

return False

i=i+1

j=j+1

if(count<2):

return True

str1=input("Enter first string::\n")

str2=input("Enter second string::\n")

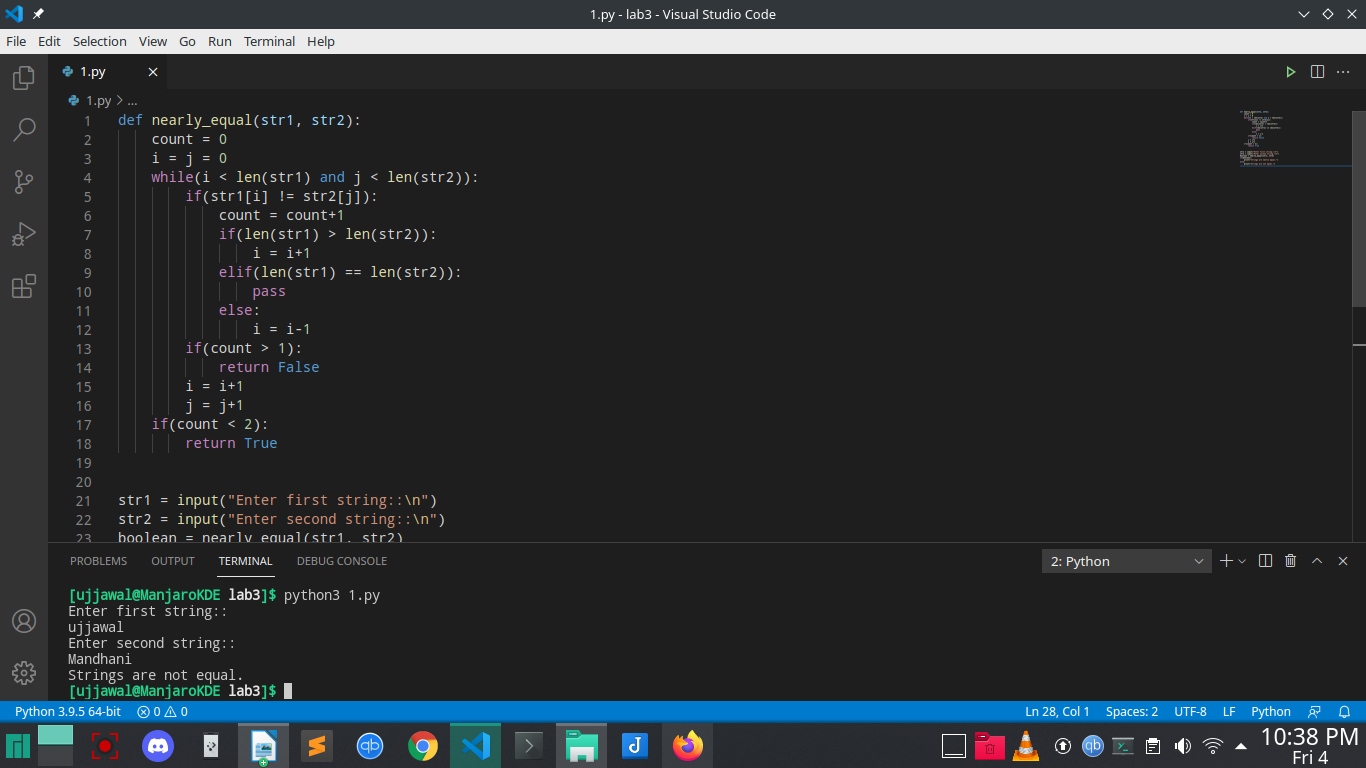
boolean=nearly\_equal(str1,str2)

if(boolean):

print("Strings are nearly equal.")

else:

print("Strings are not equal.")



**Question 10:****Write a program to find anagrams in a given list of words. Two words are called**

**anagrams if one word can be formed by rearranging letters of another. For example**

‘**eat’, ‘ate’ and ‘tea’ are anagrams.**

from collections import Counter

texts = ["bcda", "abce", "cbda", "cbea", "adcb"]

str = "abcd"

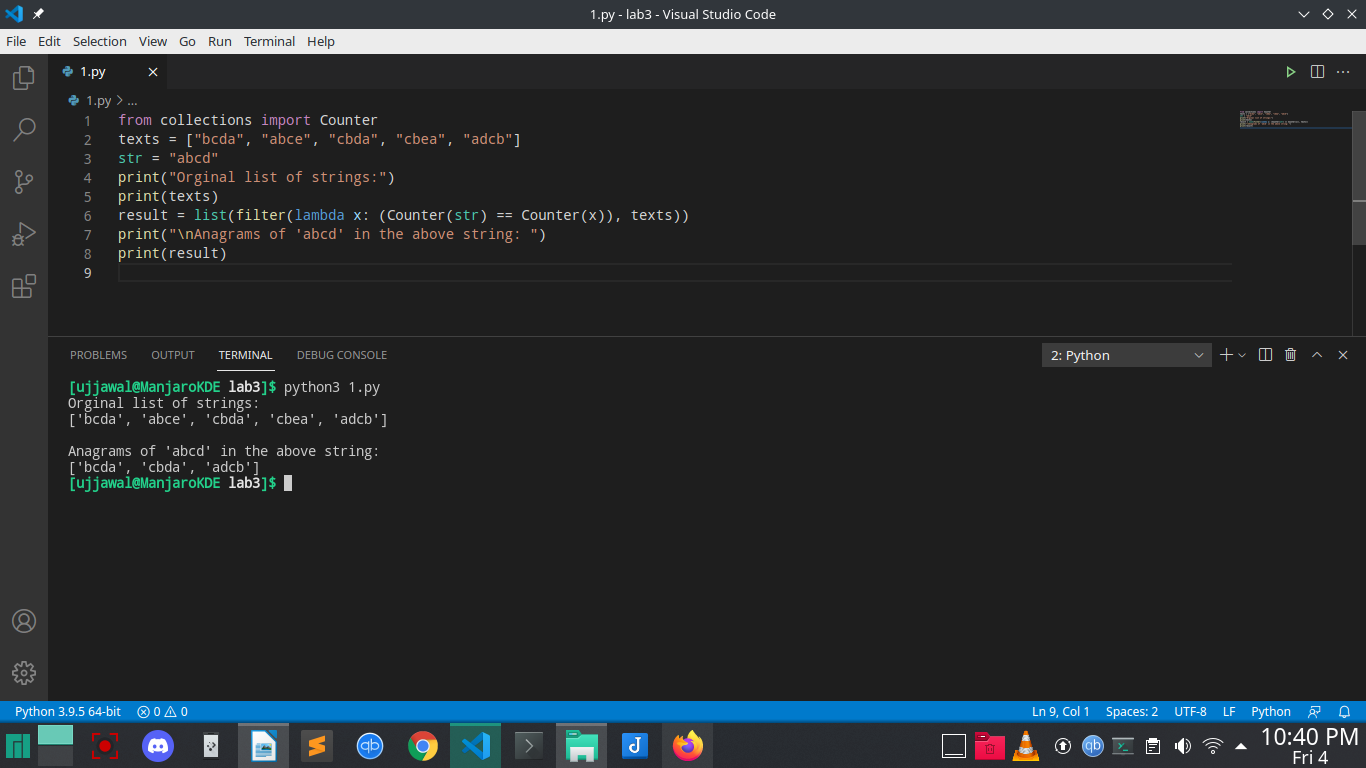
print("Orginal list of strings:")

print(texts)

result = list(filter(lambda x: (Counter(str) == Counter(x)), texts))

print("\nAnagrams of 'abcd' in the above string: ")

print(result)

****