NAME:MANVITH BALAJI

SECTION:A

MIS NO. :112315115

1.

Input code:

def memorize\_factorial():

    cache={}

    def closu(a):

        if a in cache:

            return cache[a]

        if(a==0 or a==1):

            return 1

        else:

            result=a\*closu(a-1)

            cache.update({a:result})

            return result

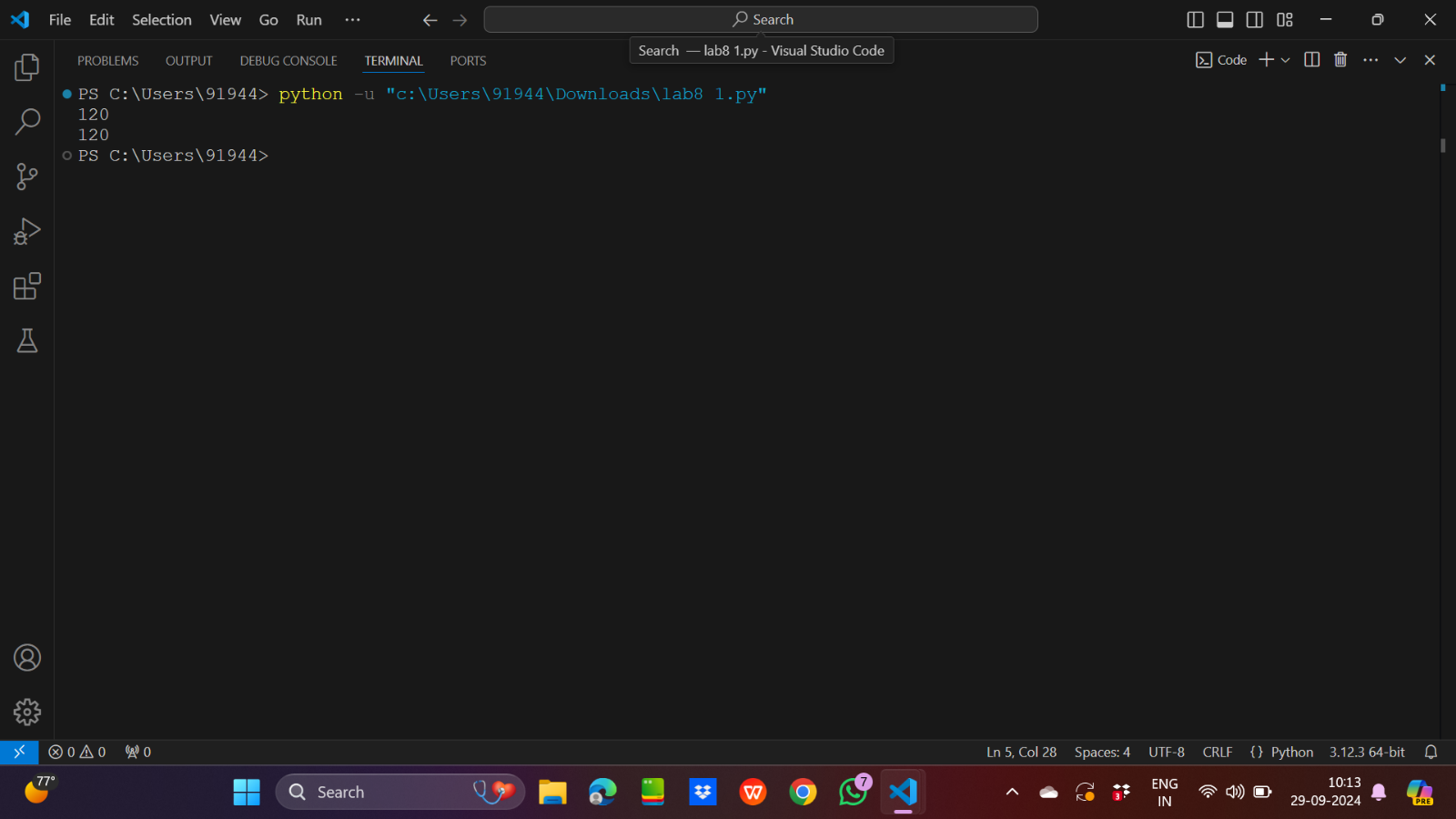
    return closu

fa=memorize\_factorial()

print(fa(5))

print(fa(5))

Output:



2.

Input code:

def multiply\_by\_two(x):return x\*2

def add\_three(x):return x+3

def create\_pipeline(l):

    def closu(x):

        for fu in l:

            x=fu(x)

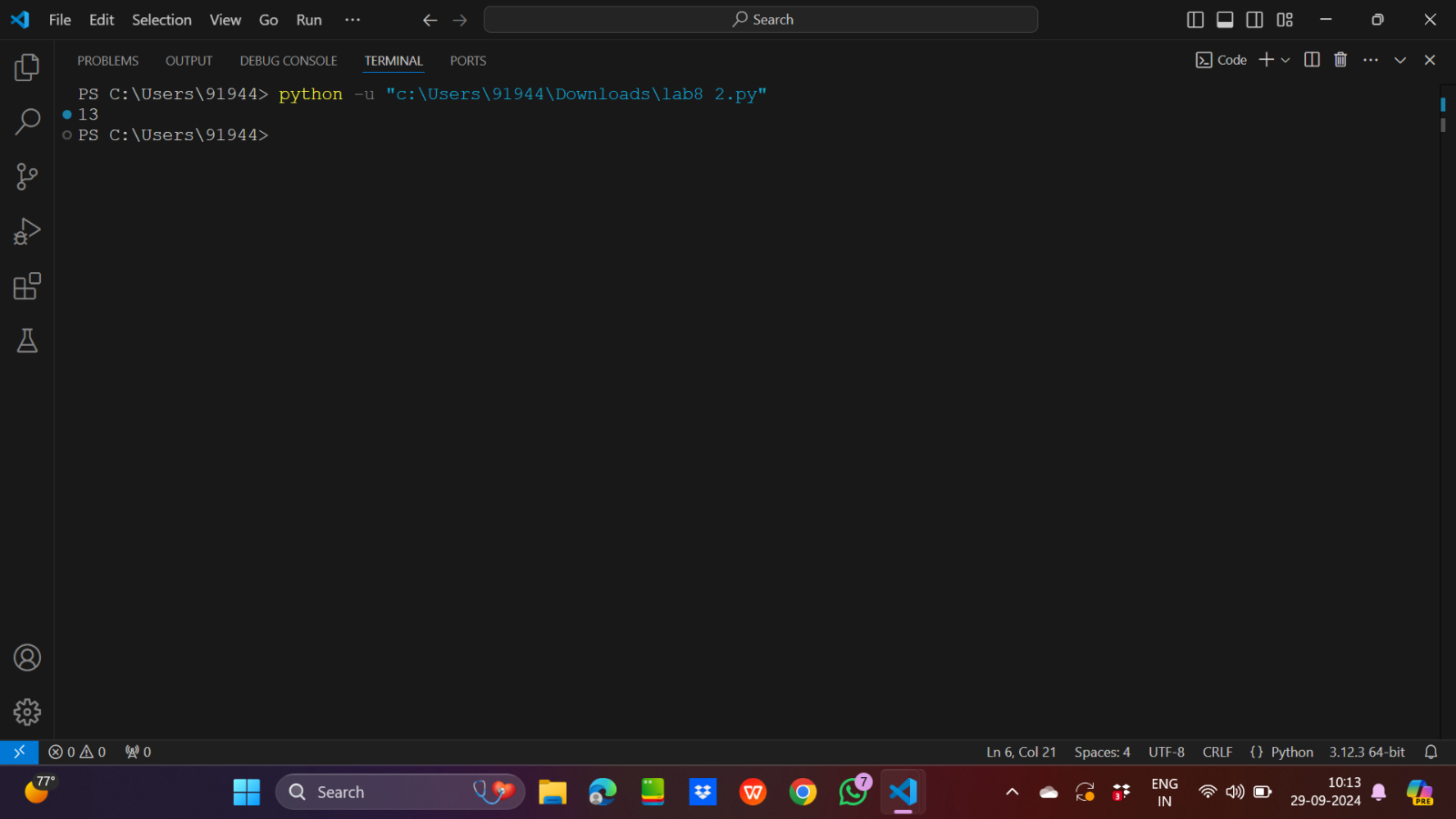
        return x

    return closu

pipeline=create\_pipeline([multiply\_by\_two,add\_three])

print(pipeline(5))

Output:



3.

Input code:

def multiply(a):

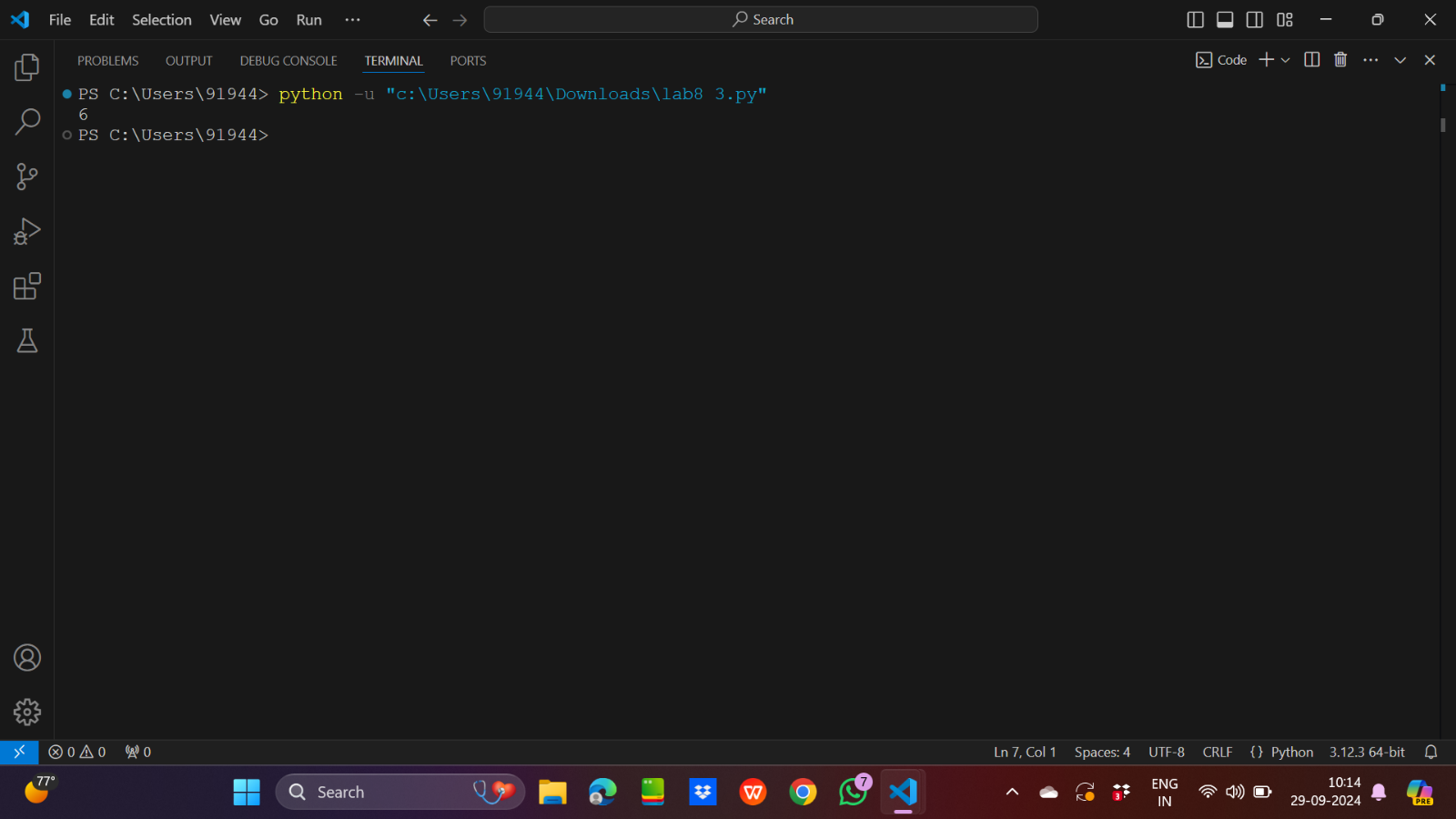
    def closu(b):

        return a\*b

    return closu

print(multiply(2)(3))

Output:



4.

Input code:

students=[

        {"name":"Alice","score":45},

        {"name":"Bob","score" :55},

        {"name":"Charlie","score":65},

        {"name":"David","score":75}

        ]

mylist=list(filter(lambda x:x["score"]>=50,students))

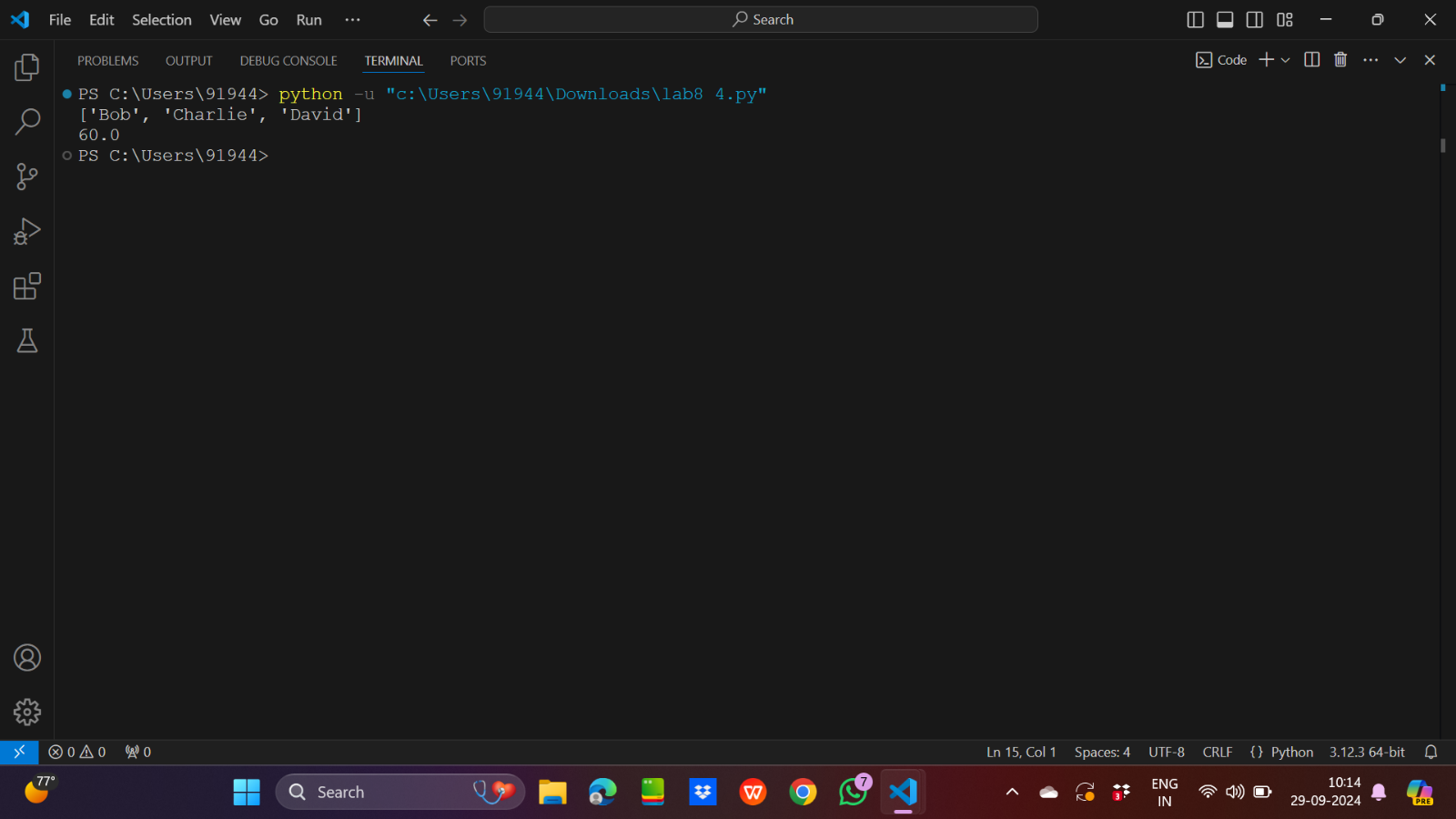
print(list(map(lambda x:x["name"],mylist)))

from functools import reduce

mylist2=list(map(lambda x:x["score"],students))

print(reduce(lambda x,y:x+y,mylist2)/len(students))

Output:



5.

Input code:

def bank\_account(balance):

    def deposit(amount):

        nonlocal balance

        balance=balance+amount

        print(f"Deposited:{amount},New Balance:{balance}")

    def withdraw(amount):

        nonlocal balance

        if(amount>balance):

            print("Insufficient funds!")

        else:

            balance=balance-amount

            print(f"Withdrew:{amount},New Balance:{balance}")

    return deposit,withdraw

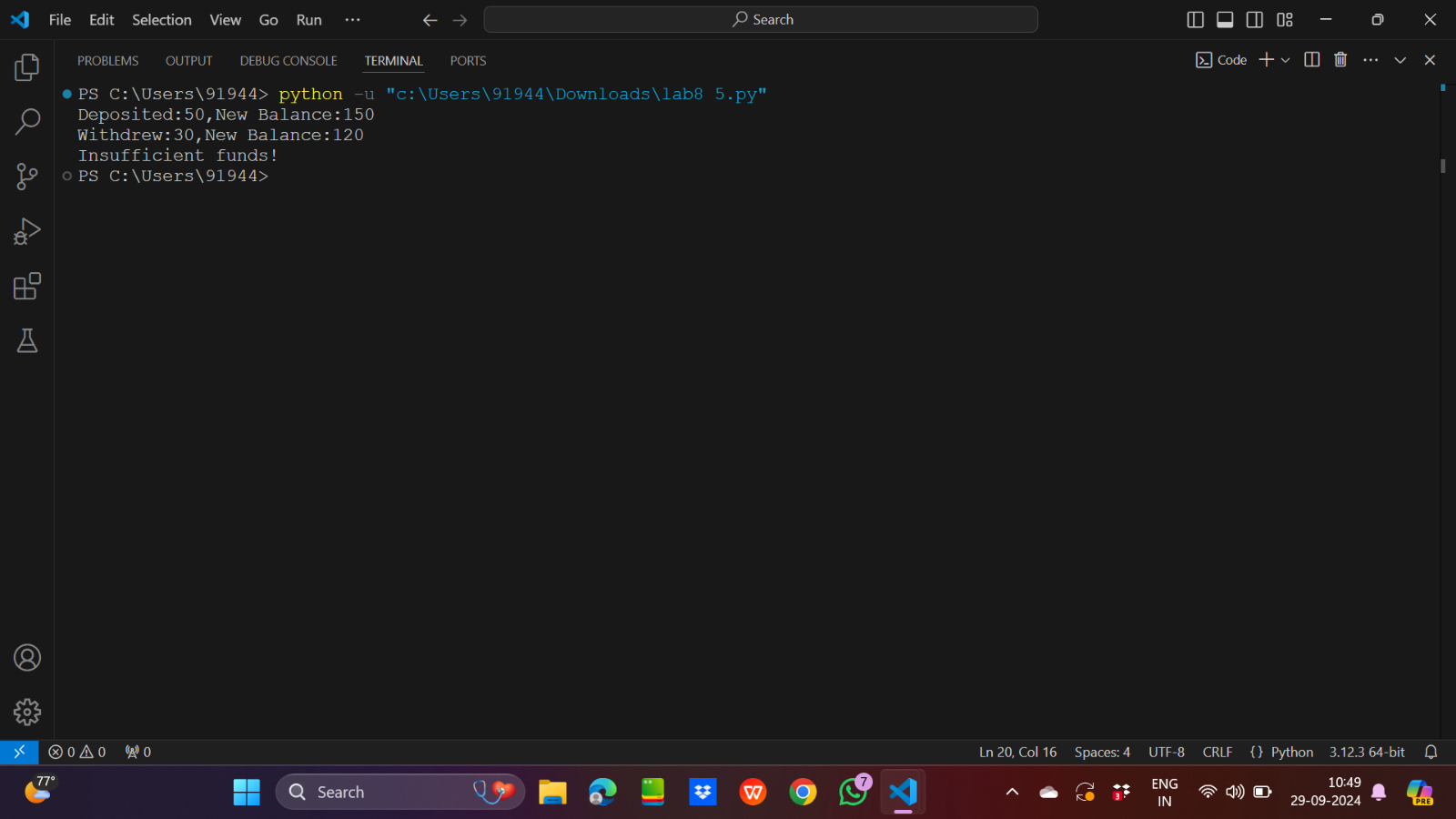
account=bank\_account(100)

account[0](50)

account[1](30)

account[1](200)

Output:



6.

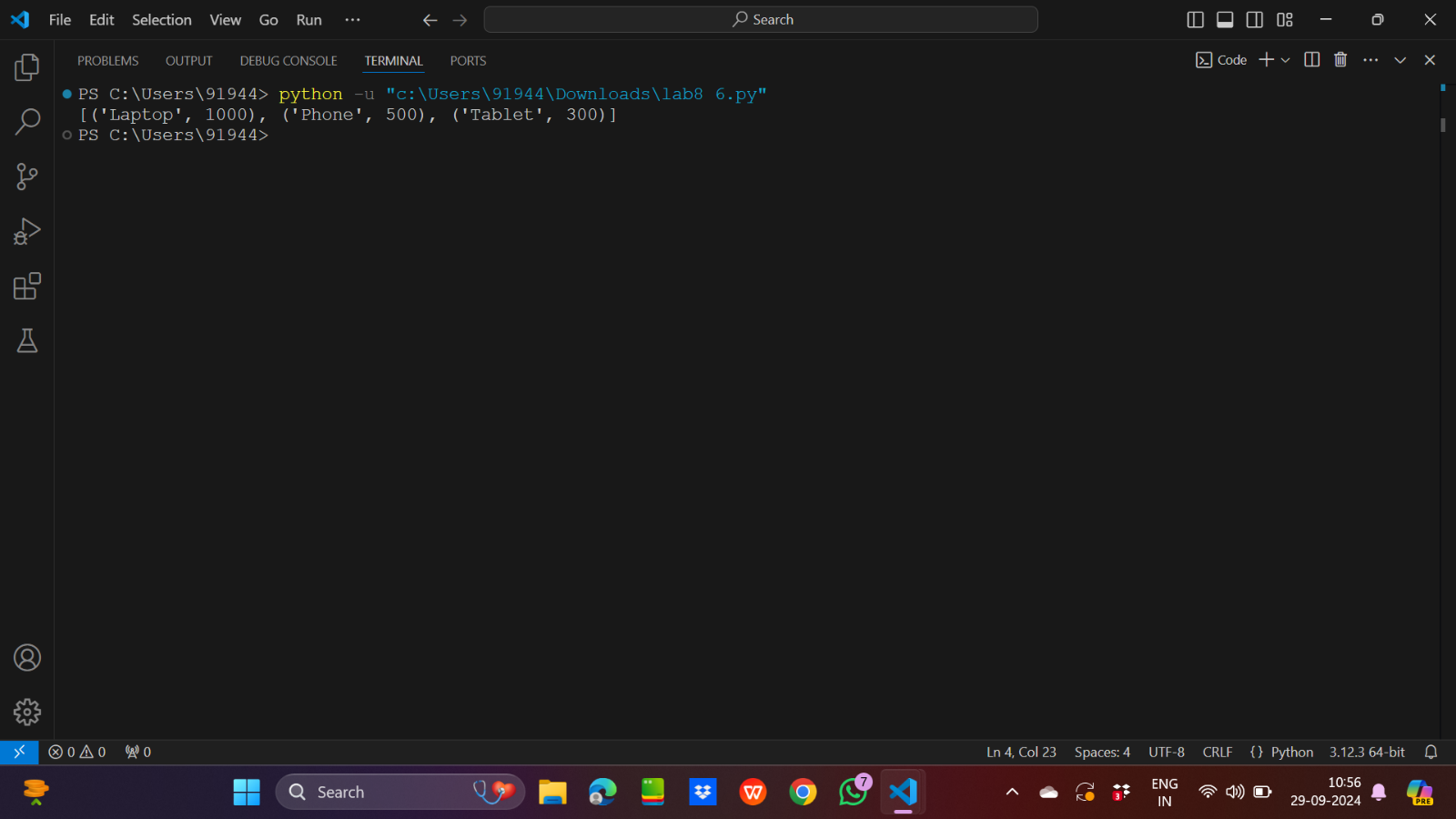
Input code:

products=[("Laptop",1000),("Phone",500),("Tablet",300)]

sorted\_products=sorted(products,key=lambda x:x[1],reverse=True)

print(sorted\_products)

Output:



7.

Input code:

import functools

def powe(a,b):

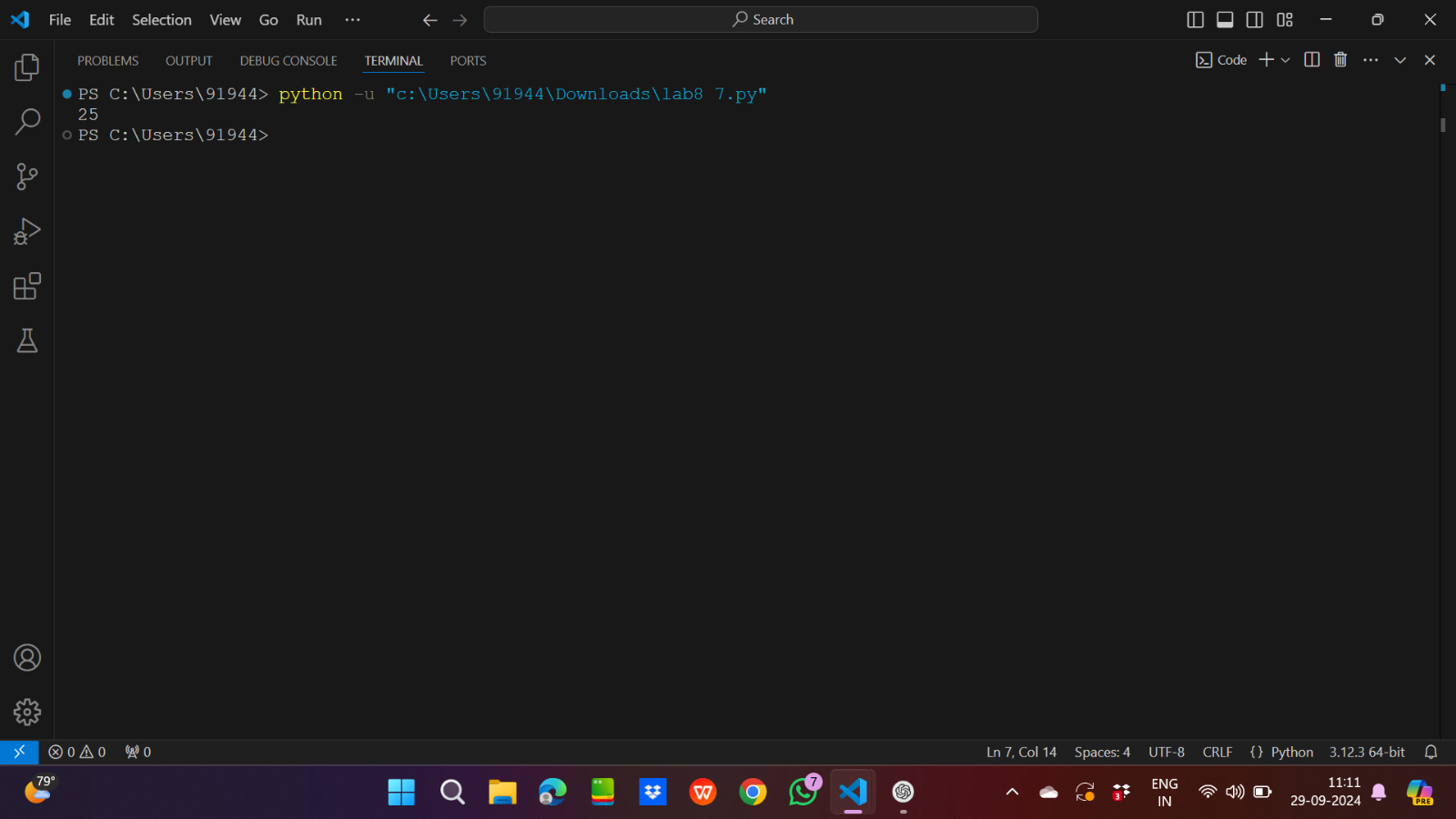
    return pow(a,b)

power\_of\_two=functools.partial(powe,b=2)

result=power\_of\_two(5)

print(result)

Output:



8.

Input code:

def polynomial\_creator(\*coeff):

    def polynomial(x):

        sum=0

        for i in range(len(coeff)-1,-1,-1):

            sum=sum+coeff[(len(coeff)-1)-i]\*pow(x,i)

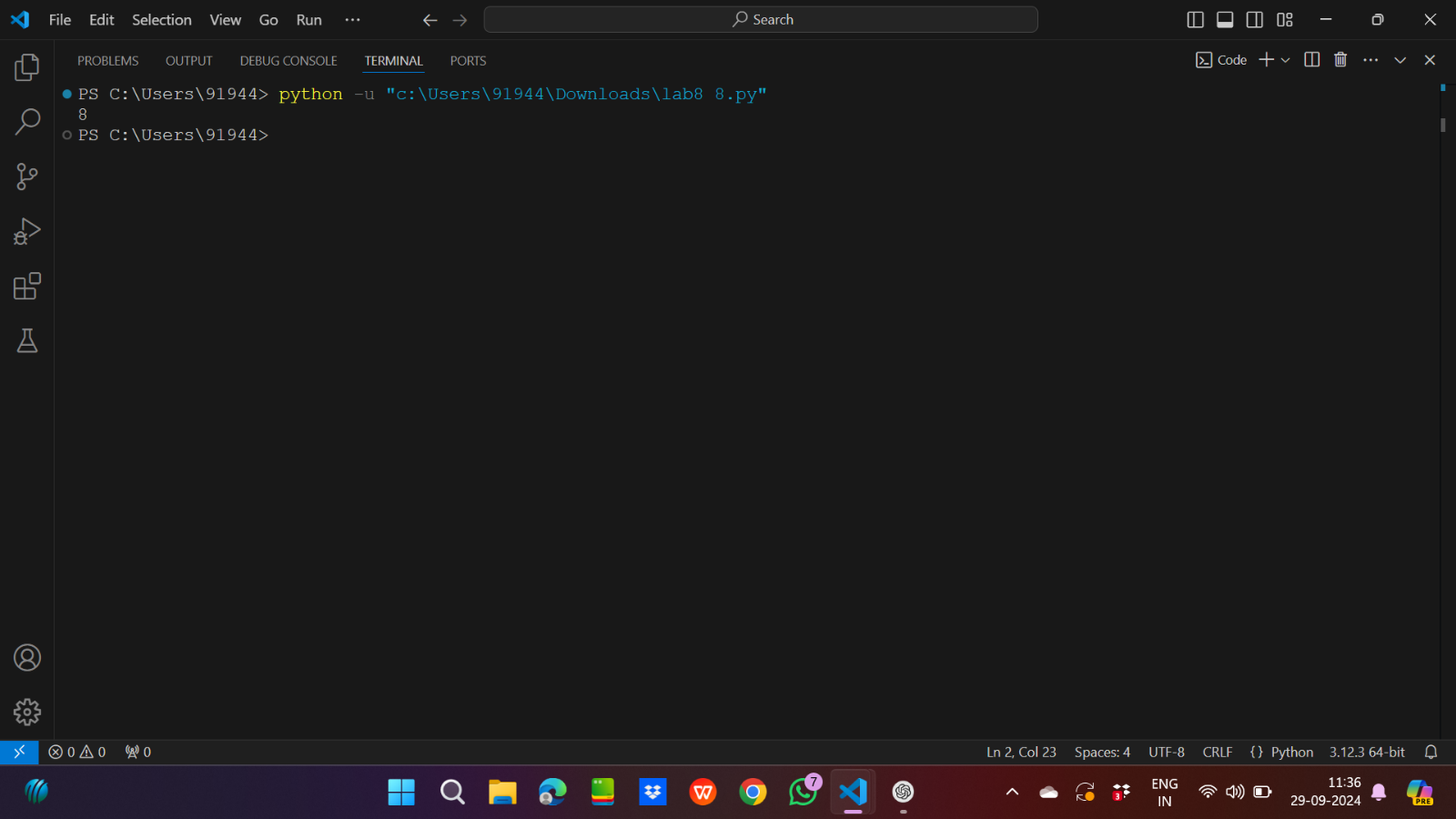
        return sum

    return polynomial

p=polynomial\_creator(3,0,-4)

print(p(2))

Output:



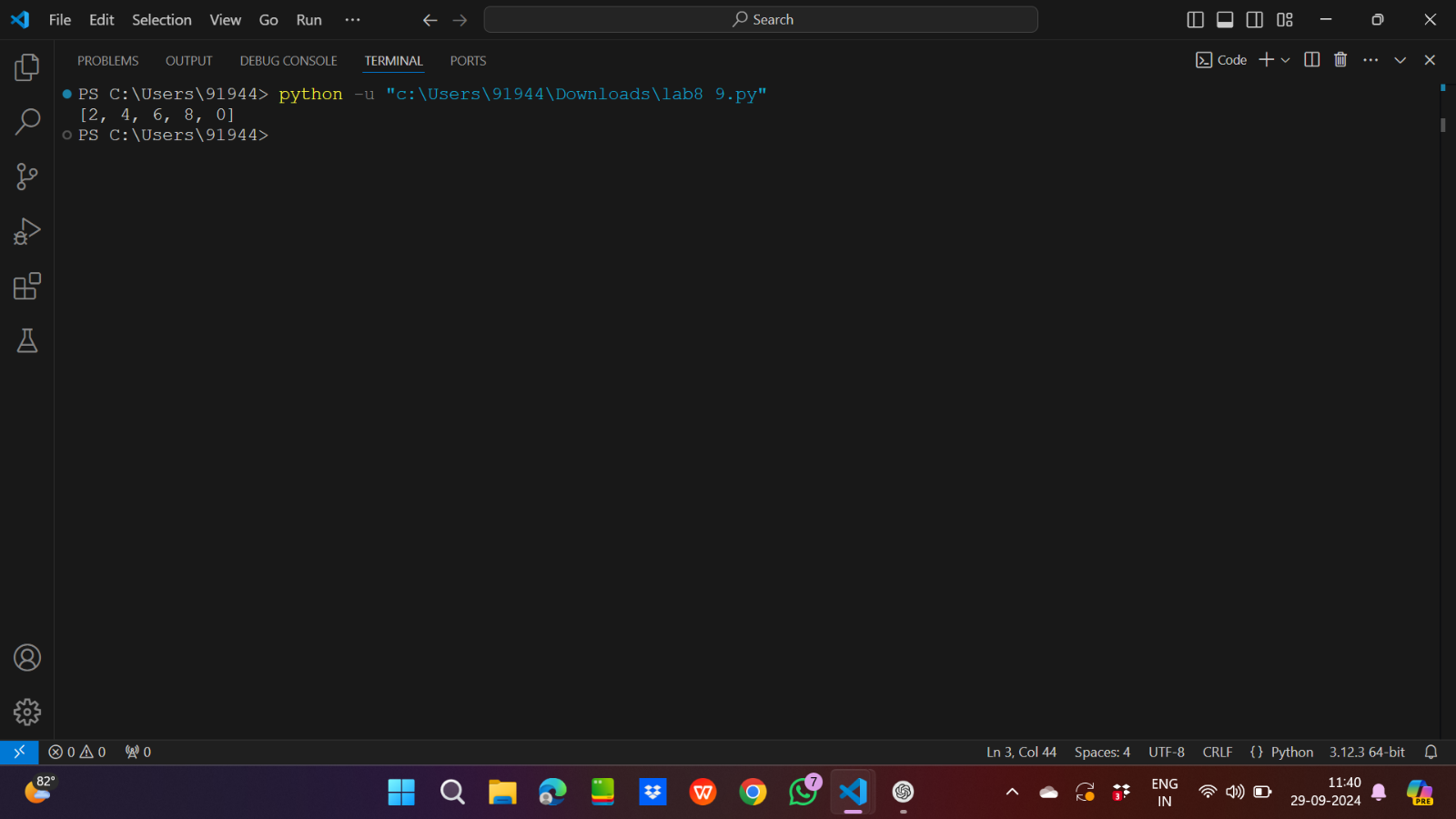
9.

Input code:

mylist=[1,2,3,4,5,6,7,8,9,0]

print(list(filter(lambda x:x%2==0,mylist)))

Output:



10.

Input code:

mylist=[1,2,3,4,5,6,7,8,9,10]

mylist2=list(filter(lambda x:x%2==0 ,mylist))

print(list(map(lambda x: x\*\*2,mylist2)))

Output:

