NAME:MANVITH BALAJI

SECTION:A

MIS NO. :112315115

1.

Input code:

from math import pi

class Circle:

    def \_\_init\_\_(self,radius):

        self.radius=radius

    def area(self):

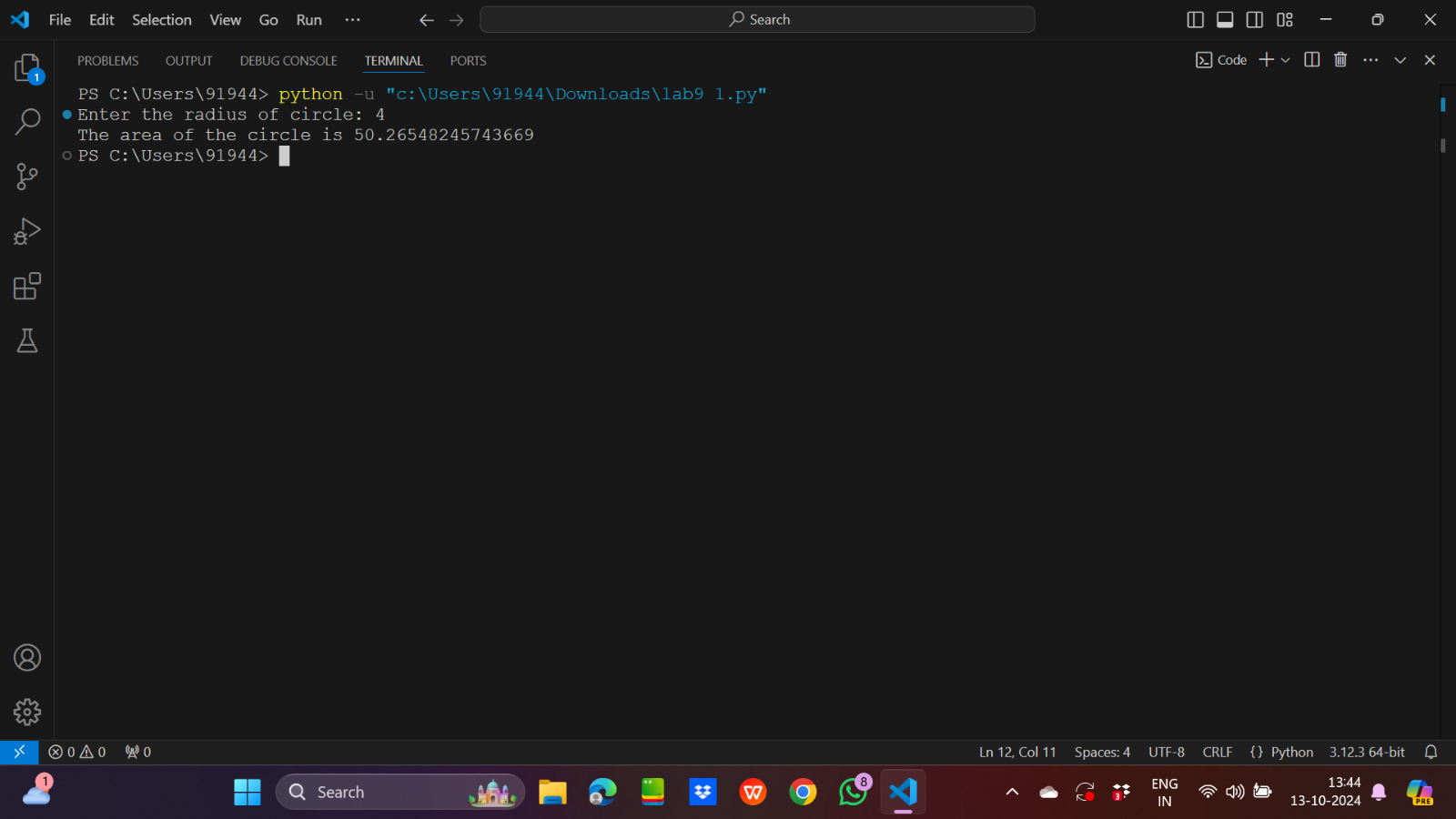
        print(f"The area of the circle is {pi\*self.radius\*\*2}")

r=int(input("Enter the radius of circle: "))

obj=Circle(r)

obj.area()

Output:



2.

Input code:

class Rectangle:

    def \_\_init\_\_(self,length,width):

        self.length=length

        self.width=width

    def area(self):

        print(f"The area of the rectangle is {self.length\*self.width}")

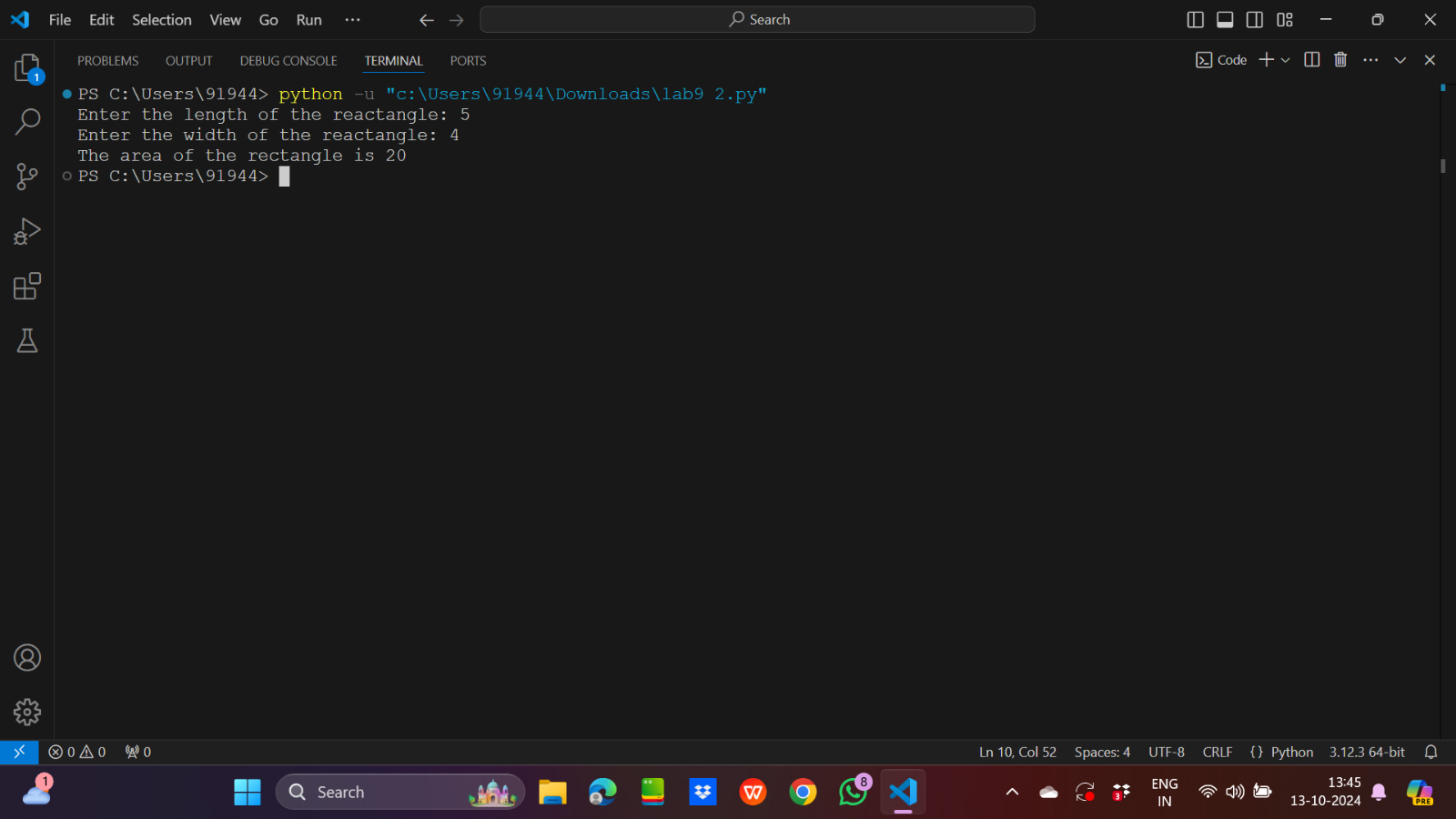
l=int(input("Enter the length of the reactangle: "))

w=int(input("Enter the width of the reactangle: "))

obj=Rectangle(l,w)

obj.area()

Output:



3.

Input code:

class gen\_ite7:

    def iterator(self,end):

        i=0

        while(i<end):

            if(i%7==0):

                yield i

            i=i+1

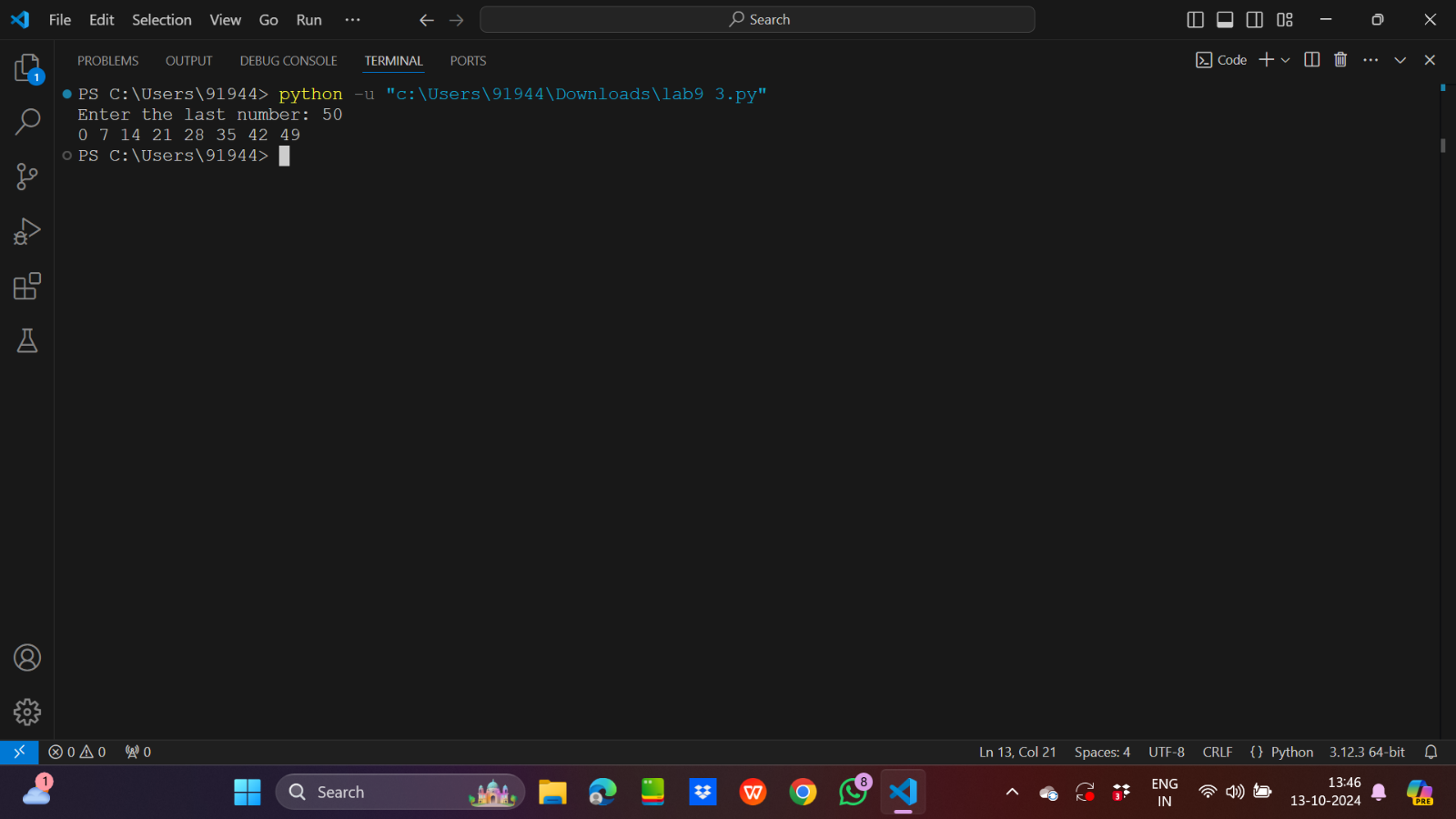
n=int(input("Enter the last number: "))

obj=gen\_ite7()

for i in obj.iterator(n):

    print(i,end=" ")

Output:



4.

Input code:

class Shape:

    def area(self,a=0):

        self.a=a

        print(f"The area of the Shape is {self.a}")

class Square(Shape):

    def \_\_init\_\_(self,length):

        self.length=length

    def area(self):

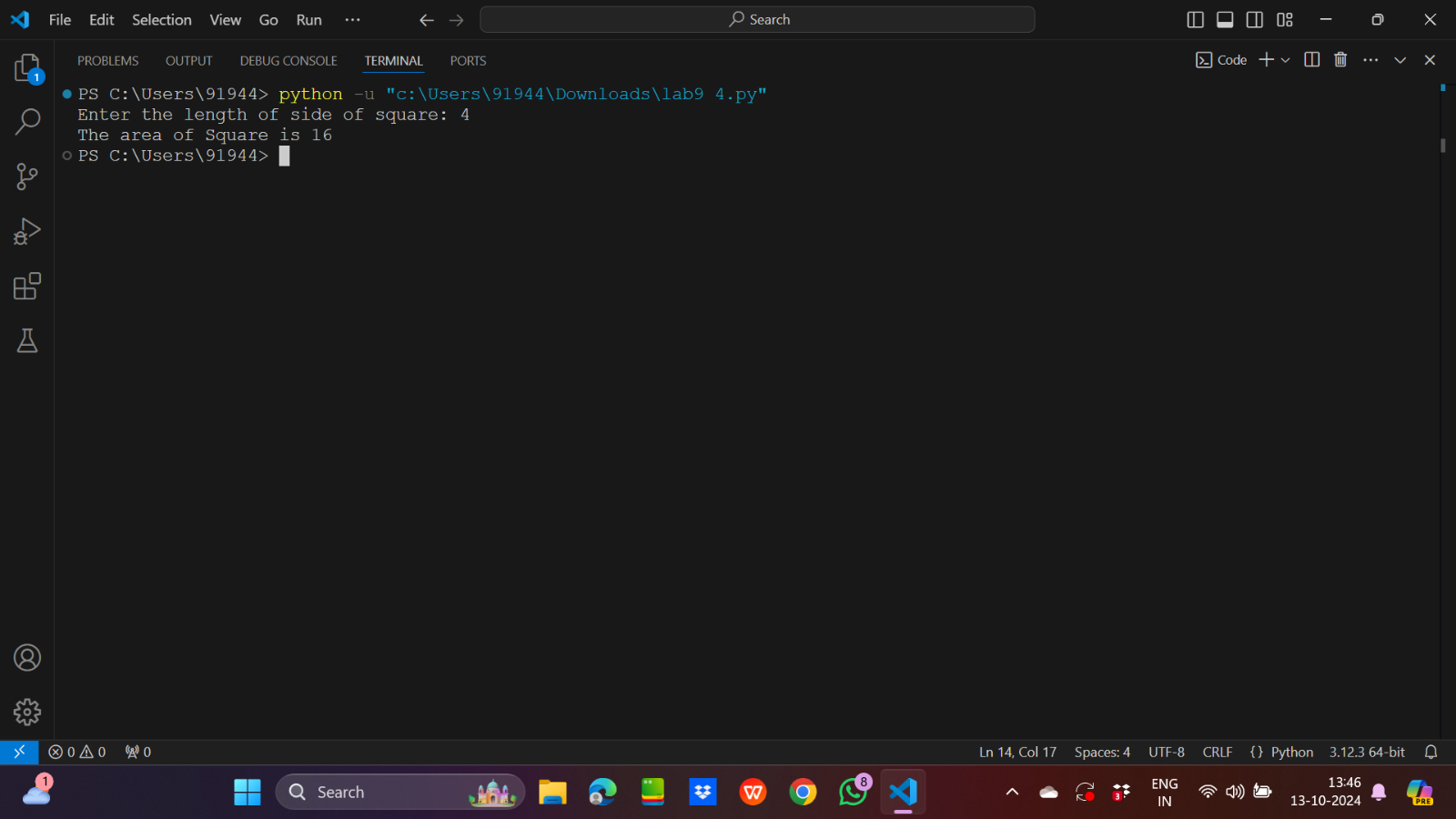
        print(f"The area of Square is {self.length\*\*2}")

side=int(input("Enter the length of side of square: "))

obj=Square(side)

obj.area()

Output:



5.

Input code:

class String:

    @classmethod

    def getString(cls):

        cls.x=input("Enter a string: ")

    @classmethod

    def printString(cls):

        cls.b=cls.x.upper()

        print(f"The string is {cls.b}")

    @classmethod

    def test(cls):

        if(cls.b.isupper()):

            print("The class methods are working good")

        else:

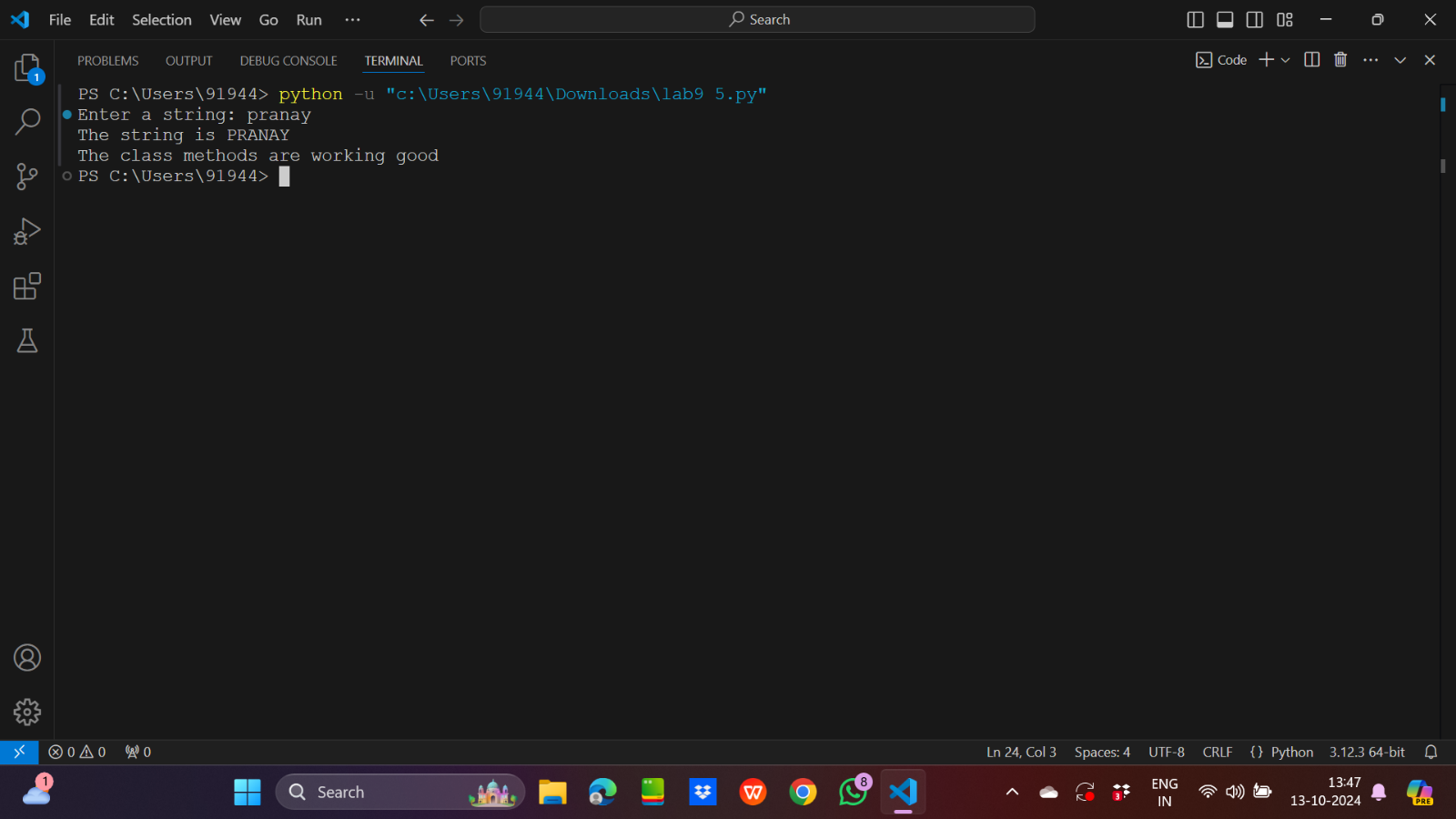
            print("The class methods are not working good")

String.getString()

String.printString()

String.test()

Output:



6.

Input code:

class Person:

    def getGender(self):

        print("The gender may be male or female")

class Male(Person):

     def getGender(self):

        print("Male")

class Female(Person):

     def getGender(self):

        print("Female")

obj=Person()

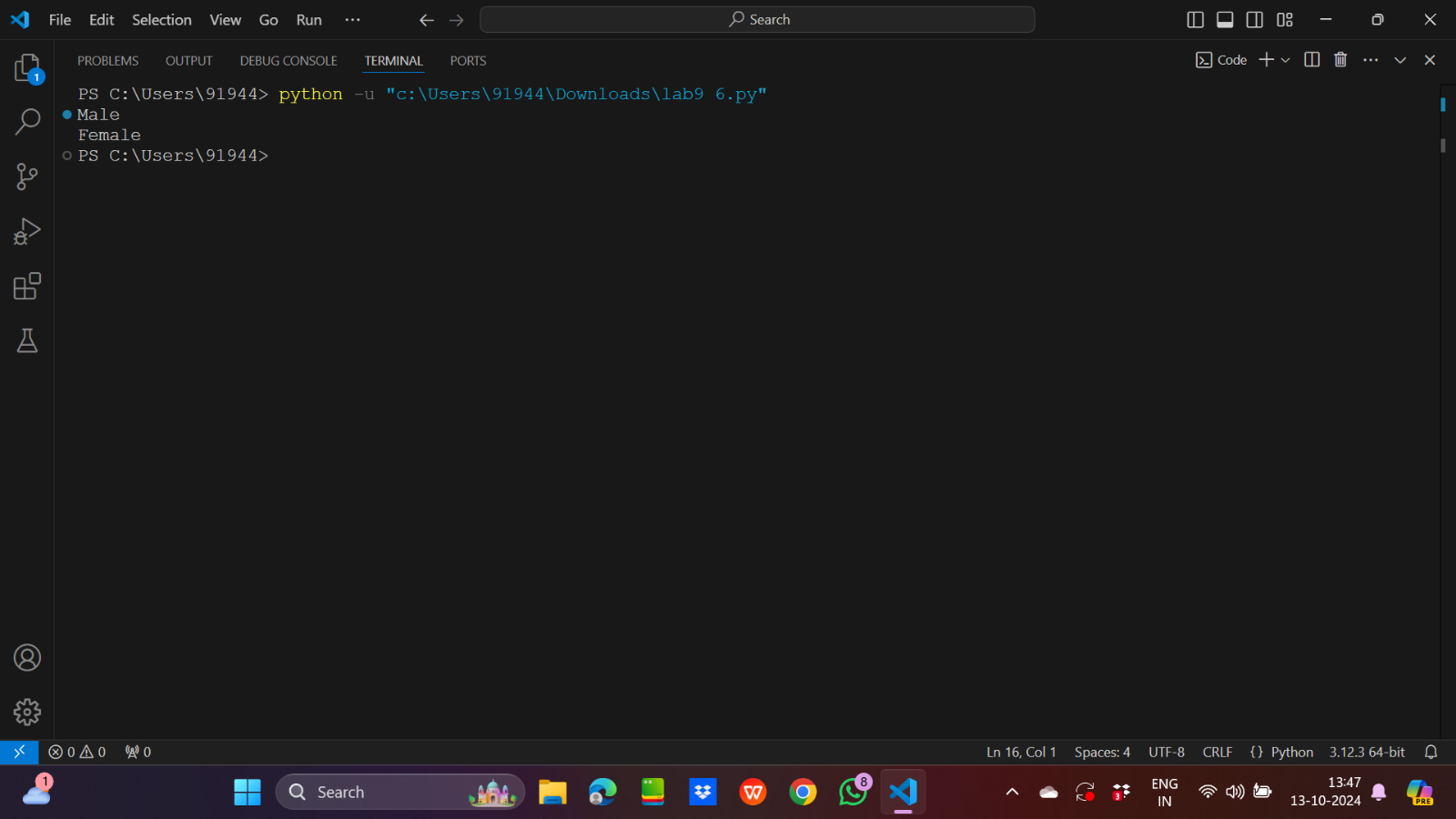
obj1=Male()

obj2=Female()

obj1.getGender()

obj2.getGender()

Output:



7.

Input code:

class Package:

    def \_\_init\_\_(self,package\_id,destination,weight,status):

        self.package\_id=package\_id

        self.destination=destination

        self.weight=weight

        self.status=status

    def deliver(self):

        self.status="delivered"

class Vehicle:

    def \_\_init\_\_(self,vehicle\_id,capacity,current\_packages):

        self.vehicle\_id=vehicle\_id

        self.capacity=capacity

        self.current\_packages=current\_packages

    def load\_package(self,pack\_obj):

        total\_weight=sum([i.weight for i in self.current\_packages])

        if total\_weight+pack\_obj.weight<=self.capacity:

            self.current\_packages.append(pack\_obj)

        else:

            print("Vehicle is full")

    def deliver\_packages(self):

        for i in self.current\_packages:

            i.deliver()

class Truck(Vehicle):

    def deliver\_packages(self):

        print("Vehicle Type:Truck")

        deli\_count=0

        for i in self.current\_packages:

            if i.status=="delivered":

                deli\_count=deli\_count+1

        print(f"No. of packages delivered:{deli\_count}")

class Drone(Vehicle):

    def load\_package(self):

        for i in self.current\_packages:

            if i.weight>5:

                raise ValueError("This package is more than 5kg")

    def deliver\_packages(self):

        print("Vehicle Type:Drone")

        for i in self.current\_packages:

            if i.status=="delivered":

                deli\_count=deli\_count+1

        print(f"No. of packages delivered:{deli\_count}")

class DeliverySystem:

    def assign\_vehicle(self,veh\_obj,list\_pack):

        self.veh\_obj=veh\_obj

        self.veh\_obj.current\_packages=list\_pack

    def dispatch(self,veh\_obj):

        self.veh\_obj=veh\_obj

        for i in self.veh\_obj.current\_packages:

            yield i.deliver()

def main():

    pack\_list1=[Package("P1","Mumbai",2,"pending"),Package("P2","Chennai",3,"intransit"),Package("P3","Pune",4,"intransit")]

    pack\_list2=[Package("P4","Mumbai",2,"pending"),Package("P5","Chennai",3,"pending"),Package("P6","Pune",4,"pending")]

    tru\_obj=Truck("V1-Truck",1000,pack\_list1)

    dron\_obj=Drone("V2-Drone",10,pack\_list2)

    deli\_obj=DeliverySystem()

    for i in deli\_obj.dispatch(tru\_obj):

        pass

    for i in deli\_obj.dispatch(dron\_obj):

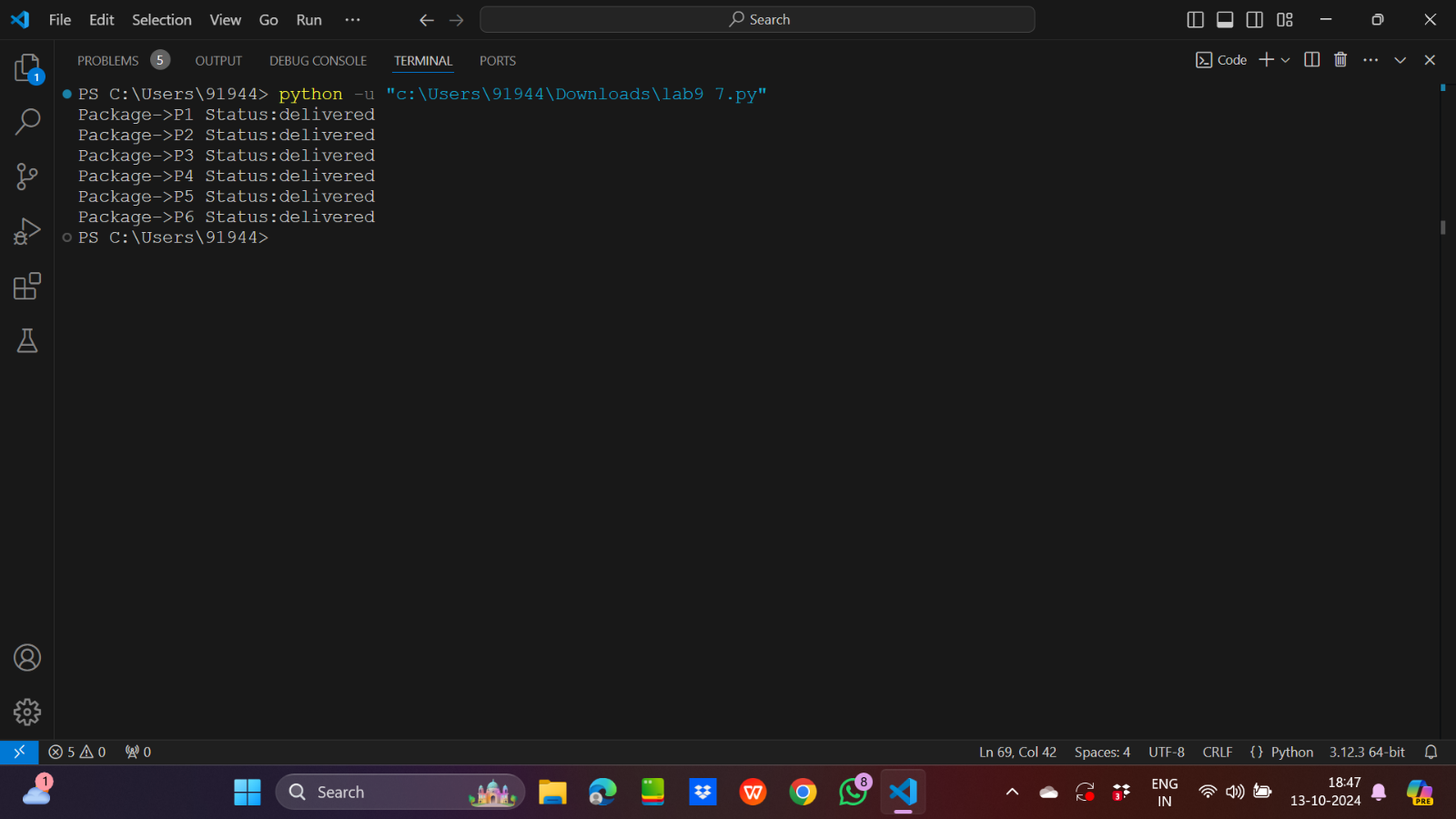
        pass

    for i in pack\_list1+pack\_list2:

        print(f"Package->{i.package\_id} Status:{i.status}")

main()

Output:



8.

Input code:

from datetime import \*

from random import \*

count=0

def sensor\_data():

    while True:

        yield (f"{datetime.now()}",randint(-10,50))

def filter\_by\_threshold():

    for i in sensor\_data():

        if i[1]>=0 and i[1]<=40:

            yield i

mylist=[]

def smooth\_temperature():

    for i in filter\_by\_threshold():

        mylist.append(i[1])

        if(len(mylist)<3):

           smoothed\_temperature=sum(mylist)/len(mylist)

        else:

           smoothed\_temperature=(mylist[-1]+mylist[-2]+mylist[-3])/3

        yield (f"{datetime.now()}",smoothed\_temperature)

def convert\_to\_farenheit():

    for i in smooth\_temperature():

        temperature\_farenheit=(1.4\*i[1])+32

        yield (f"{datetime.now()}",temperature\_farenheit)

def main():

    for i in convert\_to\_farenheit():

        print(i)

        global count

        count=count+1

        if(count==20):

            break

main()

Output:

