**Data science lab**

**Lab cycle 1**

Submitted by,

Rony Kurian

S3 MCA

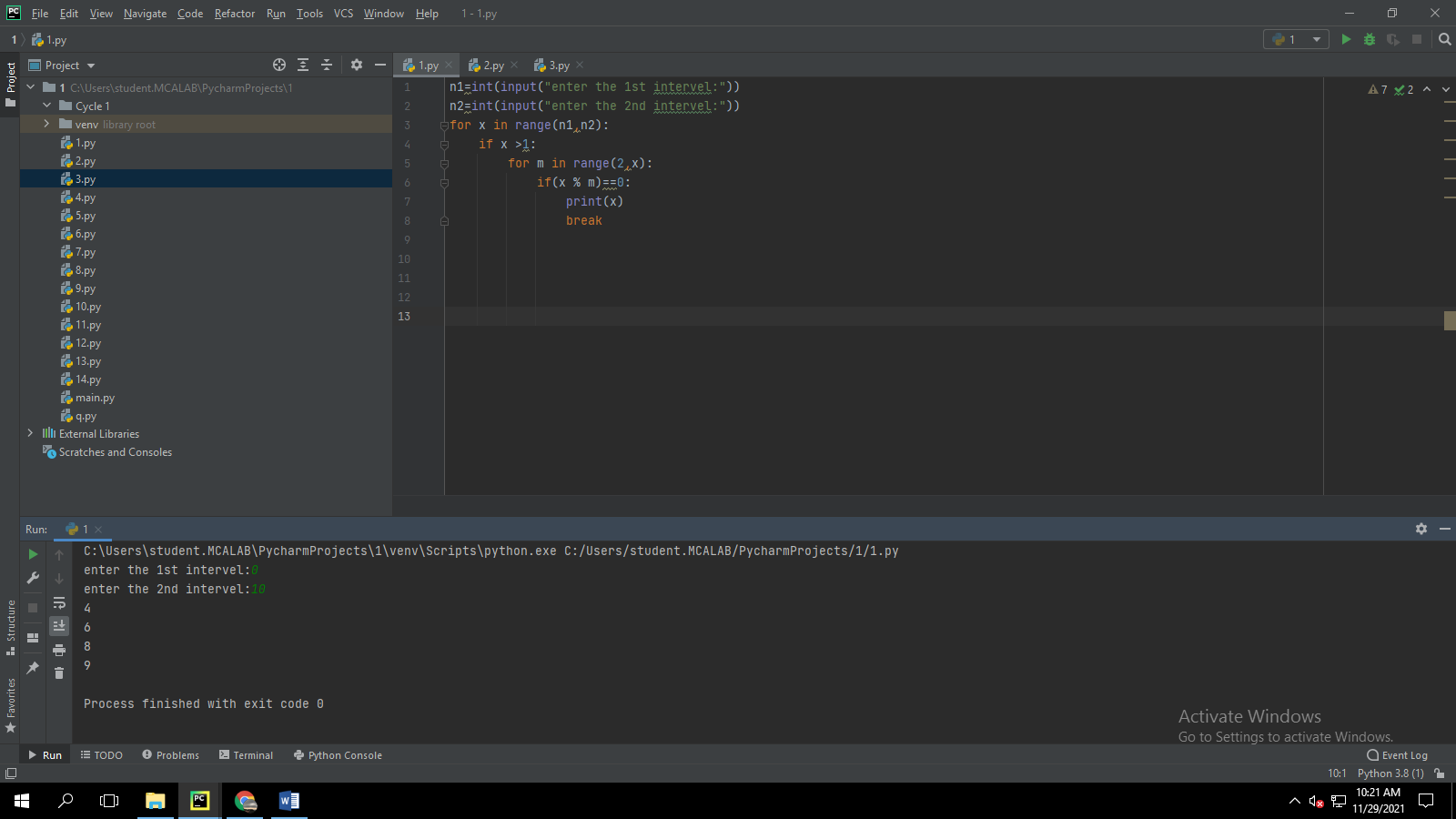
Roll no : 44

1. Program to Print all non-Prime Numbers in an Interval

**Code:**

n1=int(input("enter the 1st intervel:"))  
n2=int(input("enter the 2nd intervel:"))  
for x in range(n1,n2):  
 if x >1:  
 for m in range(2,x):  
 if(x % m)==0:  
 print(x)  
 break

**output:**

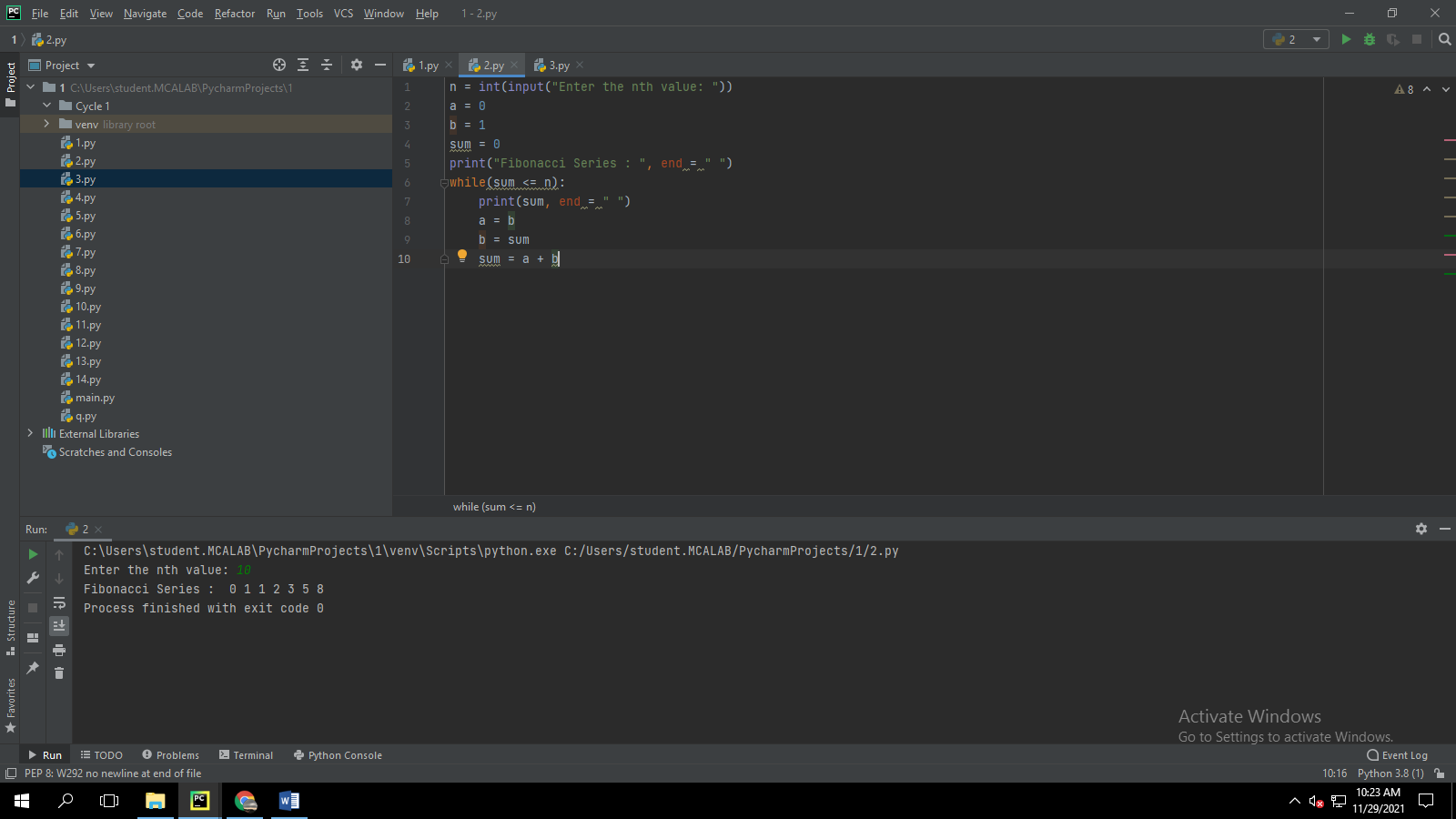
****

1. **Program to print the first N Fibonacci numbers.**

**Code:**

n = int(input("Enter the nth value: "))  
a = 0  
b = 1  
sum = 0  
print("Fibonacci Series : ", end = " ")  
while(sum <= n):  
 print(sum, end = " ")  
 a = b  
 b = sum  
 sum = a + b

**output;**

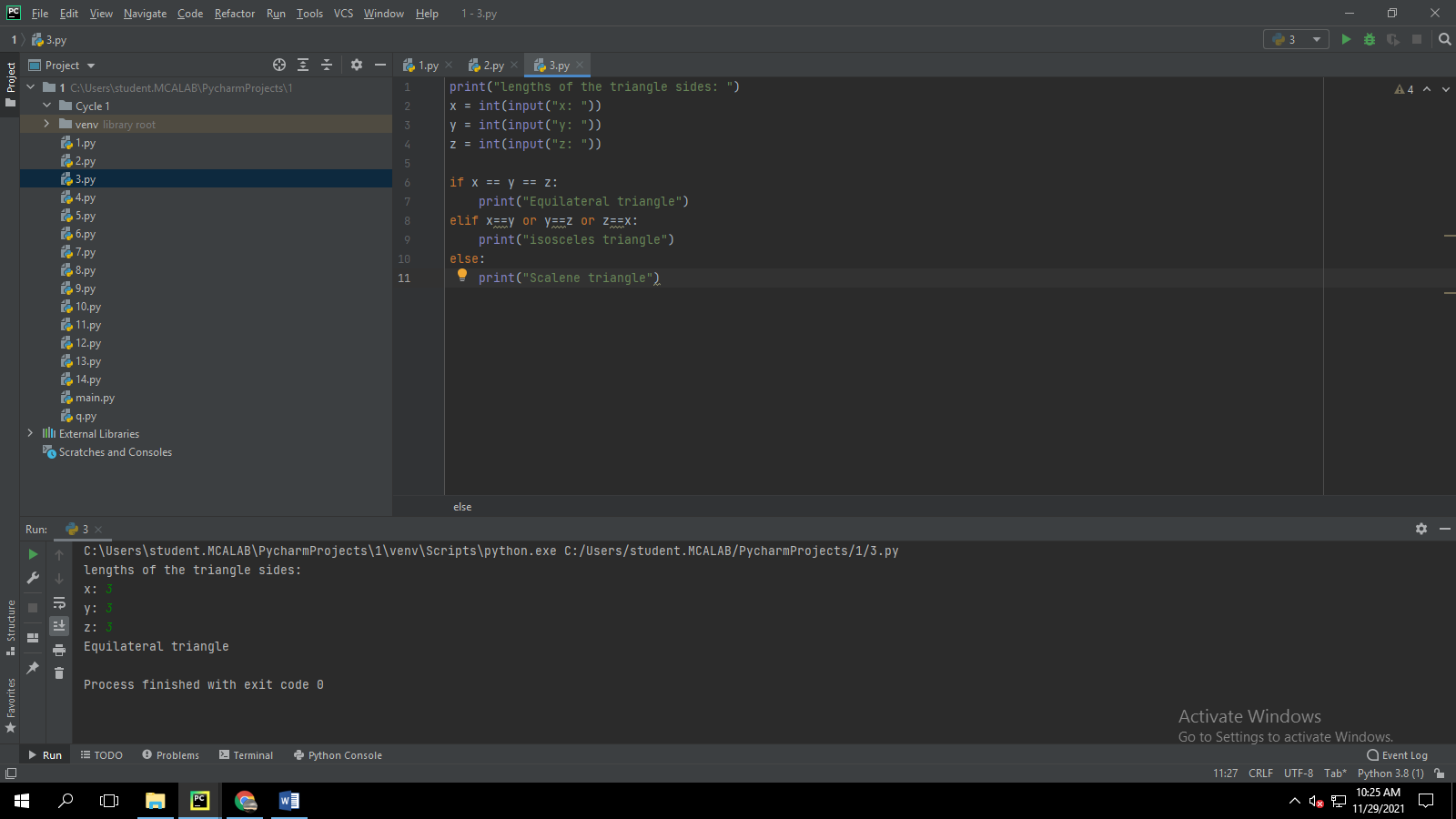
****

1. **Given sides of a triangle, write a program to check whether given triangle is an isosceles, equilateral or scalene.**

**Code:**

print("lengths of the triangle sides: ")  
x = int(input("x: "))  
y = int(input("y: "))  
z = int(input("z: "))  
  
if x == y == z:  
 print("Equilateral triangle")  
elif x==y or y==z or z==x:  
 print("isosceles triangle")  
else:  
 print("Scalene triangle")

**output:**

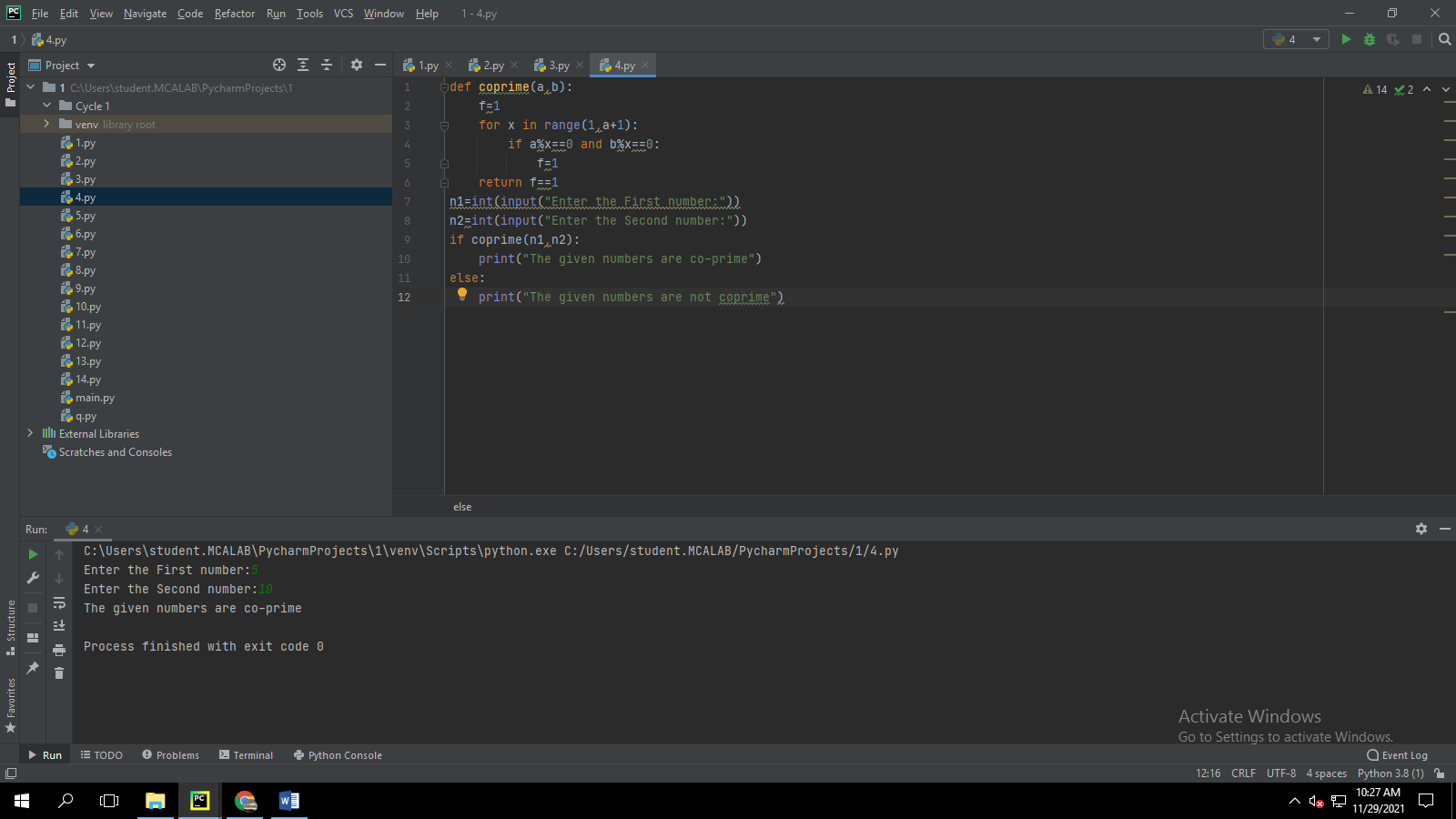
****

1. **Program to check whether given pair of number is coprime**

**Code:**

def coprime(a,b):  
 f=1  
 for x in range(1,a+1):  
 if a%x==0 and b%x==0:  
 f=1  
 return f==1  
n1=int(input("Enter the First number:"))  
n2=int(input("Enter the Second number:"))  
if coprime(n1,n2):  
 print("The given numbers are co-prime")  
else:  
 print("The given numbers are not coprime")

**output:**

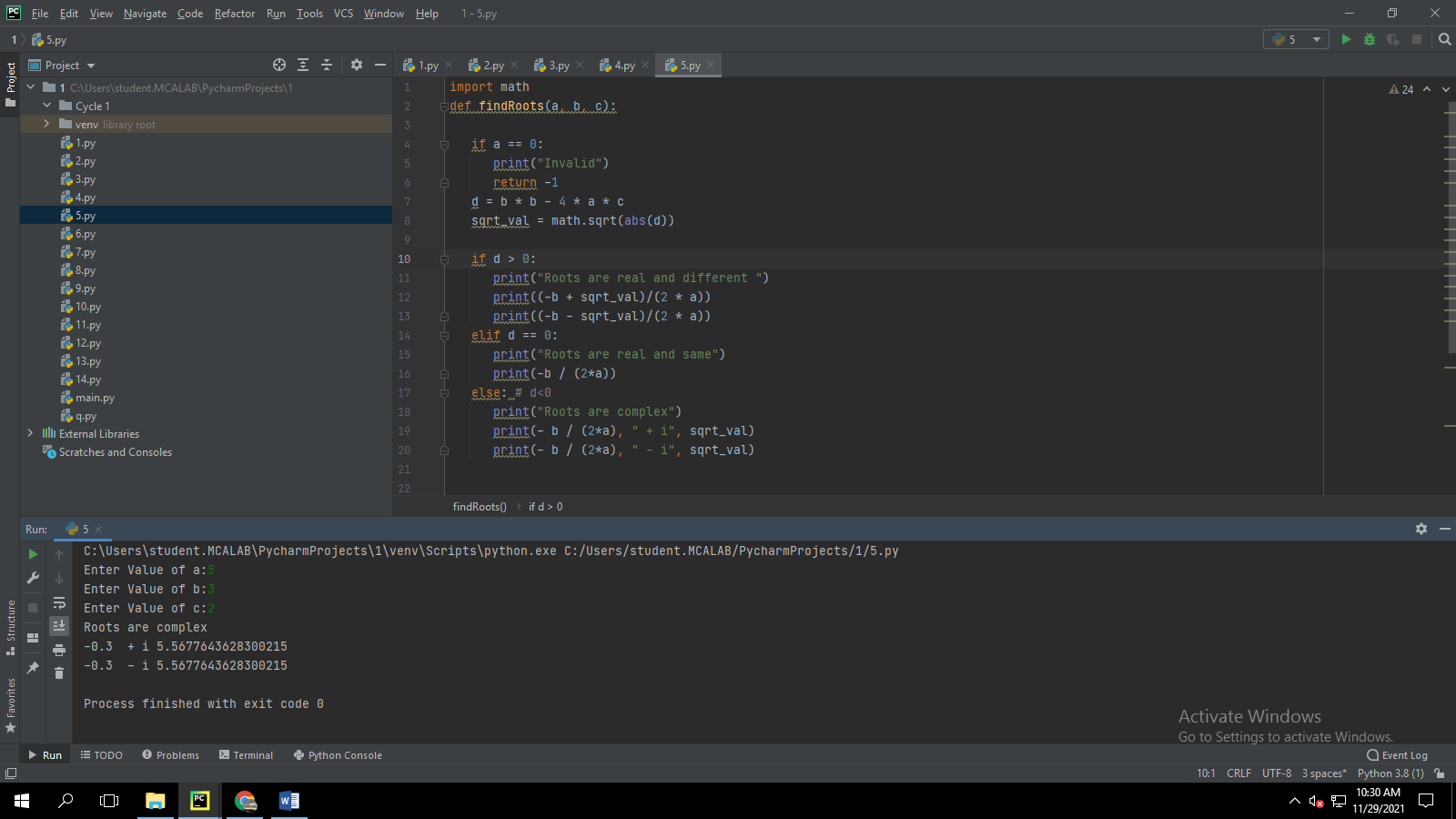
****

1. **Program to find the roots of a quadratic equation(rounded to 2 decimal places)\**

**Code:**

import math  
def findRoots(a, b, c):  
  
 if a == 0:  
 print("Invalid")  
 return -1  
 d = b \* b - 4 \* a \* c  
 sqrt\_val = math.sqrt(abs(d))  
  
 if d > 0:  
 print("Roots are real and different ")  
 print((-b + sqrt\_val)/(2 \* a))  
 print((-b - sqrt\_val)/(2 \* a))  
 elif d == 0:  
 print("Roots are real and same")  
 print(-b / (2\*a))  
 else: # d<0  
 print("Roots are complex")  
 print(- b / (2\*a), " + i", sqrt\_val)  
 print(- b / (2\*a), " - i", sqrt\_val)  
  
  
  
a = int(input("Enter Value of a:"))  
b = int(input("Enter Value of b:"))  
c = int(input("Enter Value of c:"))  
  
  
findRoots(a, b, c)

**output:**

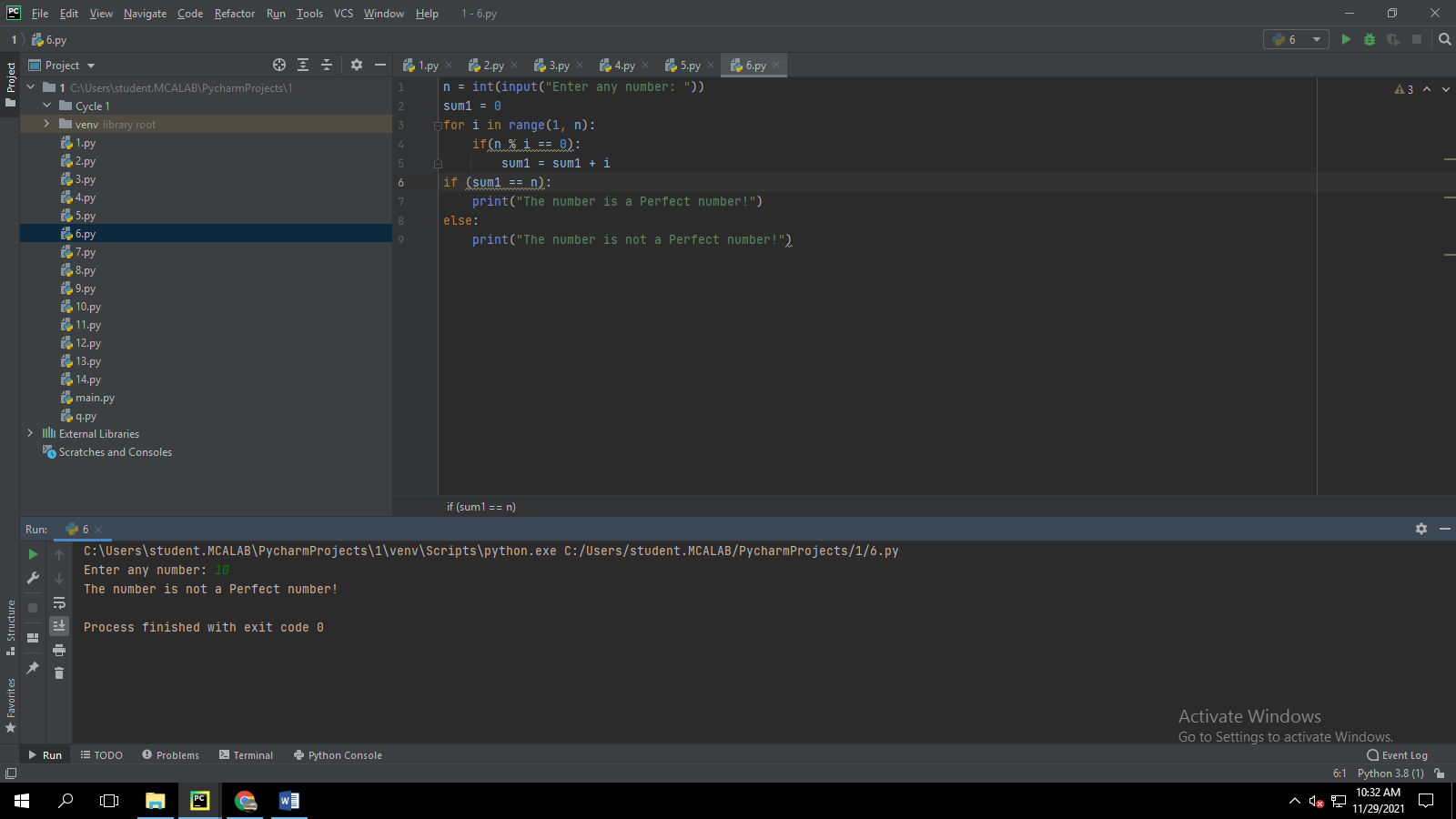
****

1. **Program to check whether a given number is perfect number or not(sum of factors =number)**

**Code:**

n = int(input("Enter any number: "))  
sum1 = 0  
for i in range(1, n):  
 if(n % i == 0):  
 sum1 = sum1 + i  
if (sum1 == n):  
 print("The number is a Perfect number!")  
else:  
 print("The number is not a Perfect number!")

**output:**

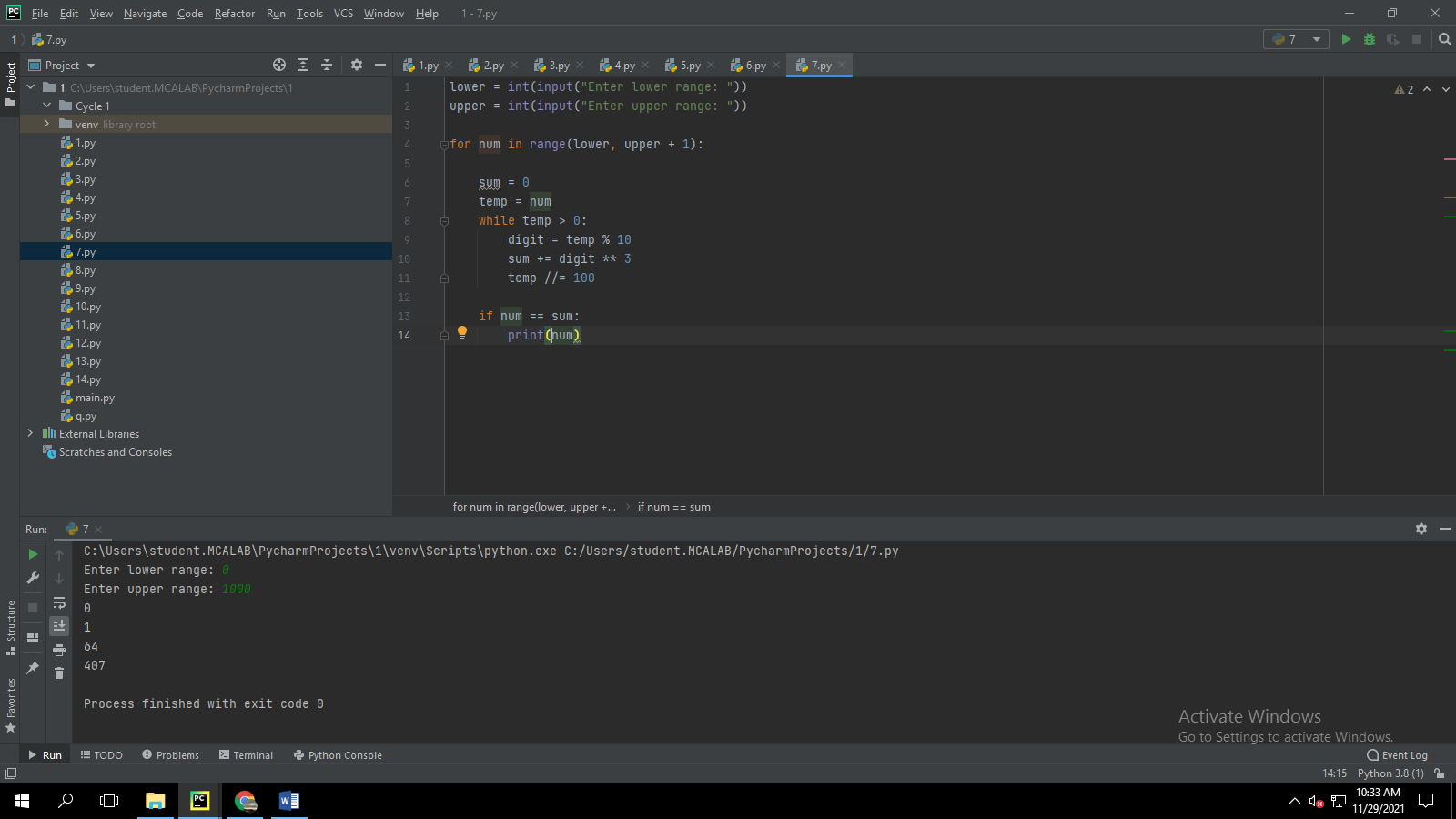
****

1. **Program to display amstrong numbers upto 1000**

**Code:**

lower = int(input("Enter lower range: "))  
upper = int(input("Enter upper range: "))  
  
for num in range(lower, upper + 1):  
  
 sum = 0  
 temp = num  
 while temp > 0:  
 digit = temp % 10  
 sum += digit \*\* 3  
 temp //= 100  
  
 if num == sum:  
 print(num)

**output:**

****

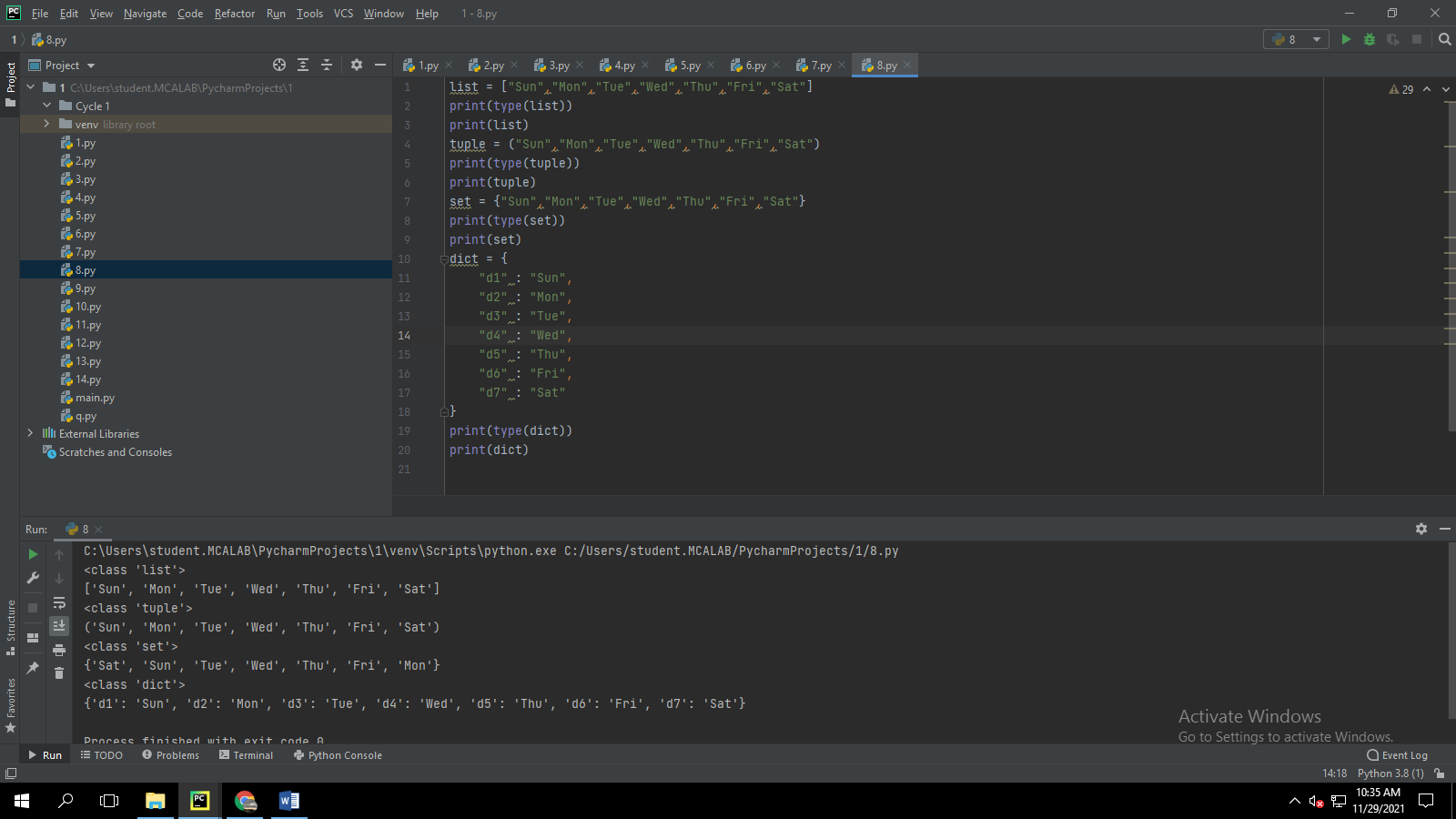
**8.** **Store and display the days of a week as a List, Tuple, Dictionary, Set. Also**

**demonstrate different ways to store values in each of them. Display its type also.**

**Code:**

list = ["Sun","Mon","Tue","Wed","Thu","Fri","Sat"]  
print(type(list))  
print(list)  
tuple = ("Sun","Mon","Tue","Wed","Thu","Fri","Sat")  
print(type(tuple))  
print(tuple)  
set = {"Sun","Mon","Tue","Wed","Thu","Fri","Sat"}  
print(type(set))  
print(set)  
dict = {  
 "d1" : "Sun",  
 "d2" : "Mon",  
 "d3" : "Tue",  
 "d4" : "Wed",  
 "d5" : "Thu",  
 "d6" : "Fri",  
 "d7" : "Sat"  
}  
print(type(dict))  
print(dict)

**output:**

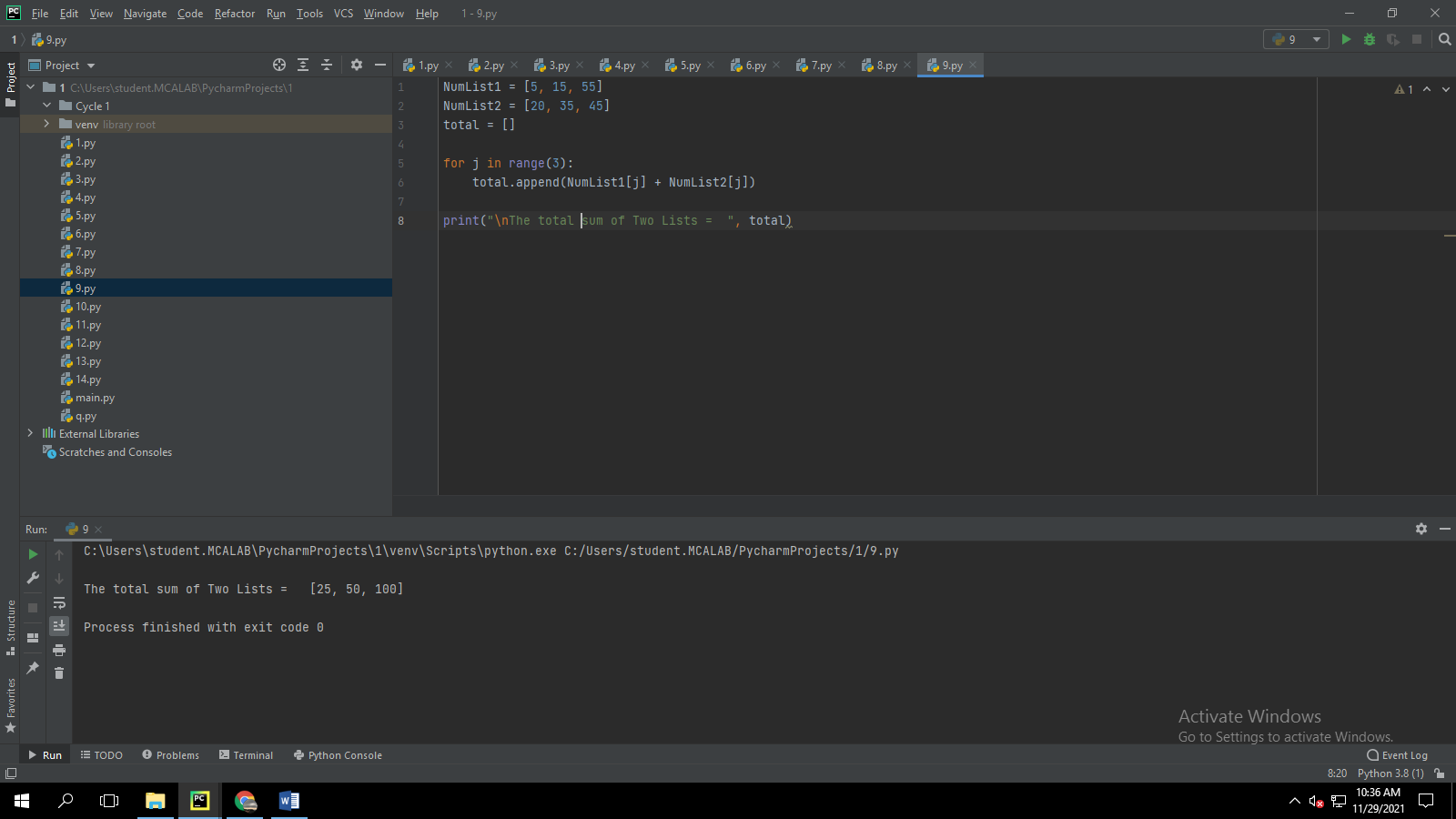
****

**9.Write a program to add elements of given 2 lists**

**Code:**

NumList1 = [5, 15, 55]  
NumList2 = [20, 35, 45]  
total = []  
  
for j in range(3):  
 total.append(NumList1[j] + NumList2[j])  
  
print("\nThe total sum of Two Lists = ", total)

**Output:**

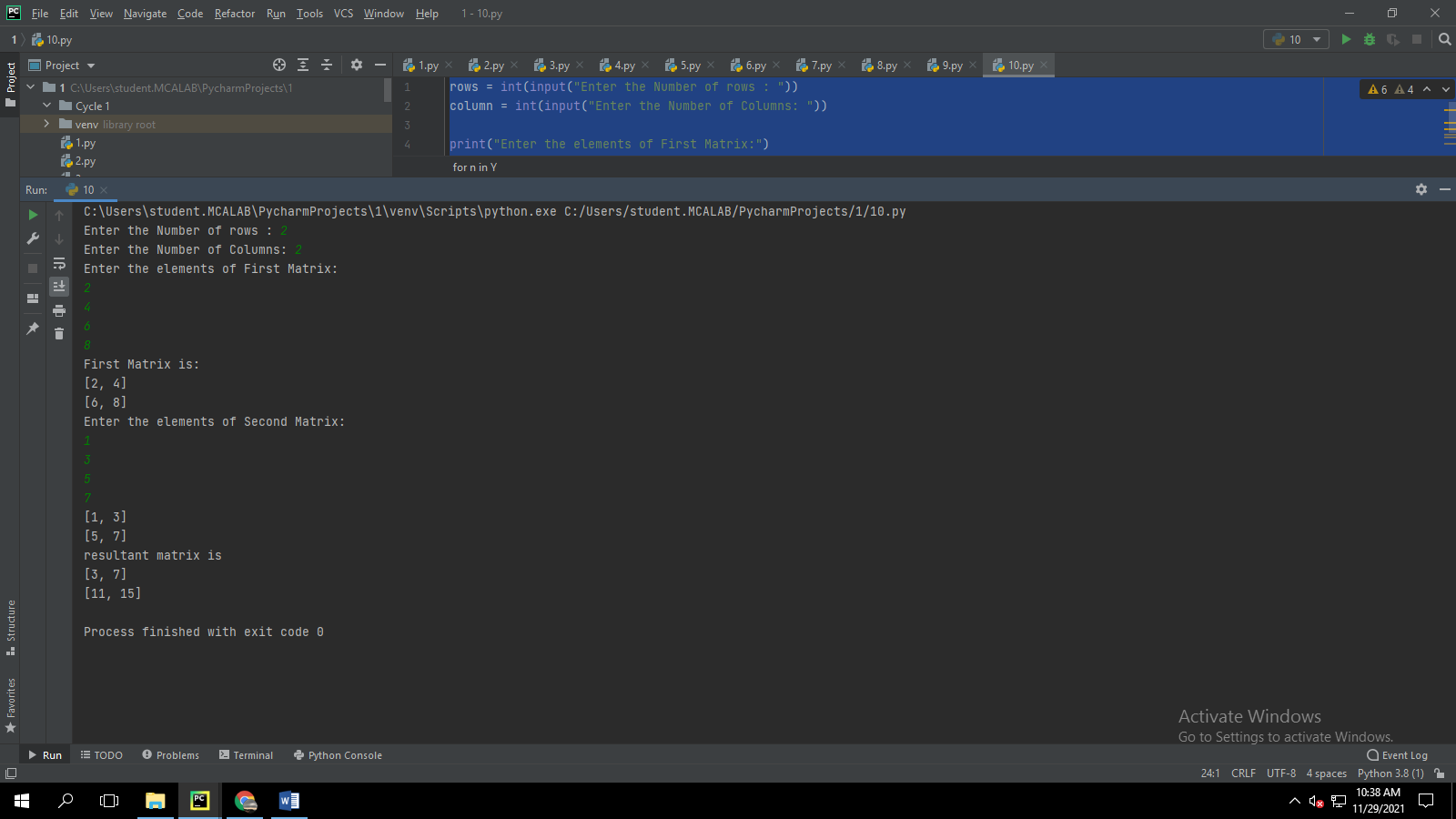
****

**10.** **Write a program to find the sum of 2 matrices using nested List.**

**Code:**

rows = int(input("Enter the Number of rows : "))  
column = int(input("Enter the Number of Columns: "))  
  
print("Enter the elements of First Matrix:")  
X = [[int(input()) for i in range(column)] for i in range(rows)]  
print("First Matrix is: ")  
for n in X:  
 print(n)  
  
print("Enter the elements of Second Matrix:")  
Y = [[int(input()) for i in range(column)] for i in range(rows)]  
for n in Y:  
 print(n)  
result = [[0 for i in range(column)] for i in range(rows)]  
  
for i in range(len(X)):  
 for j in range(len(X[0])):  
 result[i][j] = X[i][j] + Y[i][j]  
print("resultant matrix is")  
for r in result:  
 print(r)

**output:**

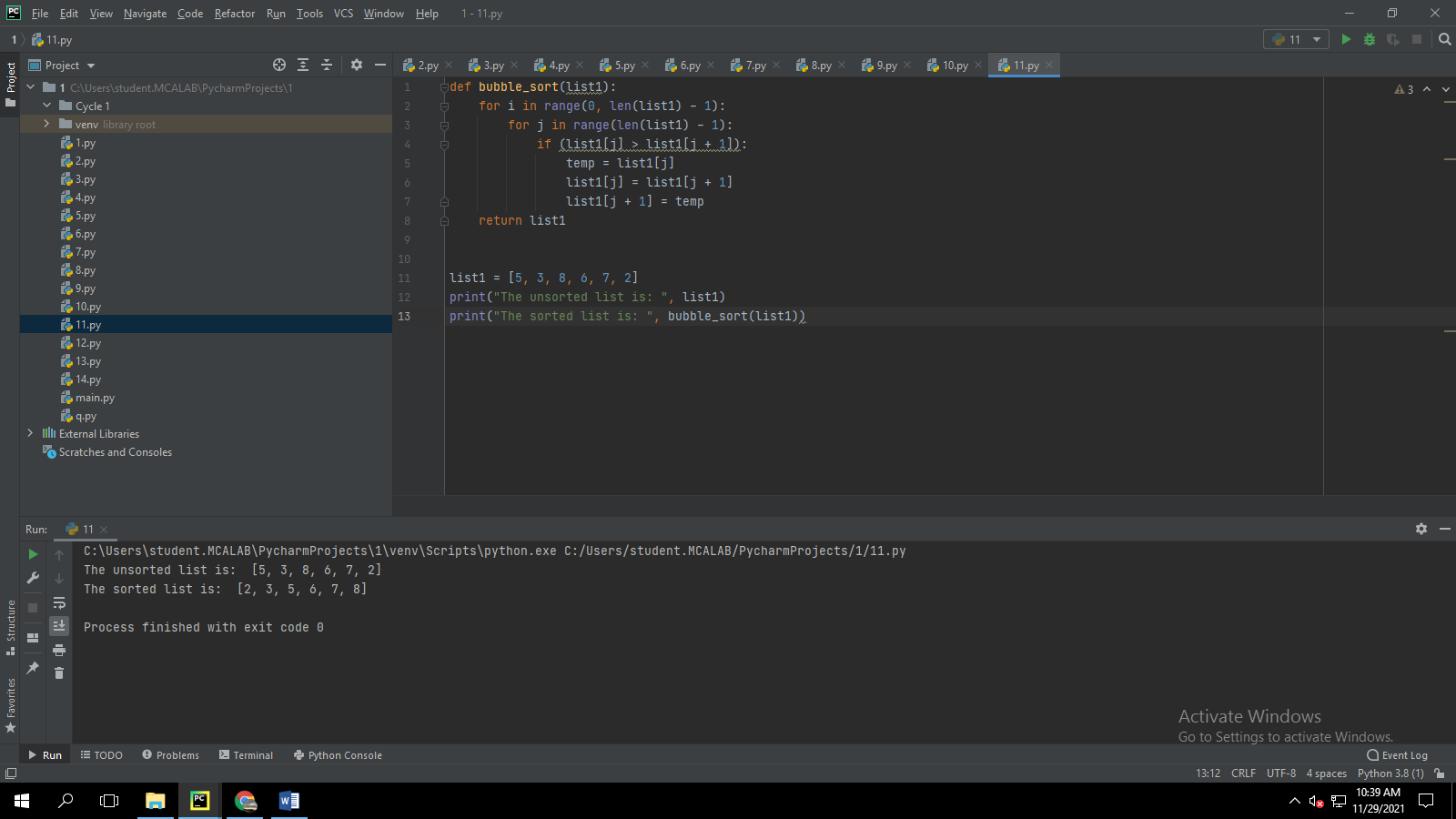
****

**11.** **Write a program to perform bubble sort on a given set of elements.**

**Code:**

def bubble\_sort(list1):  
 for i in range(0, len(list1) - 1):  
 for j in range(len(list1) - 1):  
 if (list1[j] > list1[j + 1]):  
 temp = list1[j]  
 list1[j] = list1[j + 1]  
 list1[j + 1] = temp  
 return list1  
  
  
list1 = [5, 3, 8, 6, 7, 2]  
print("The unsorted list is: ", list1)  
print("The sorted list is: ", bubble\_sort(list1))

**output:**

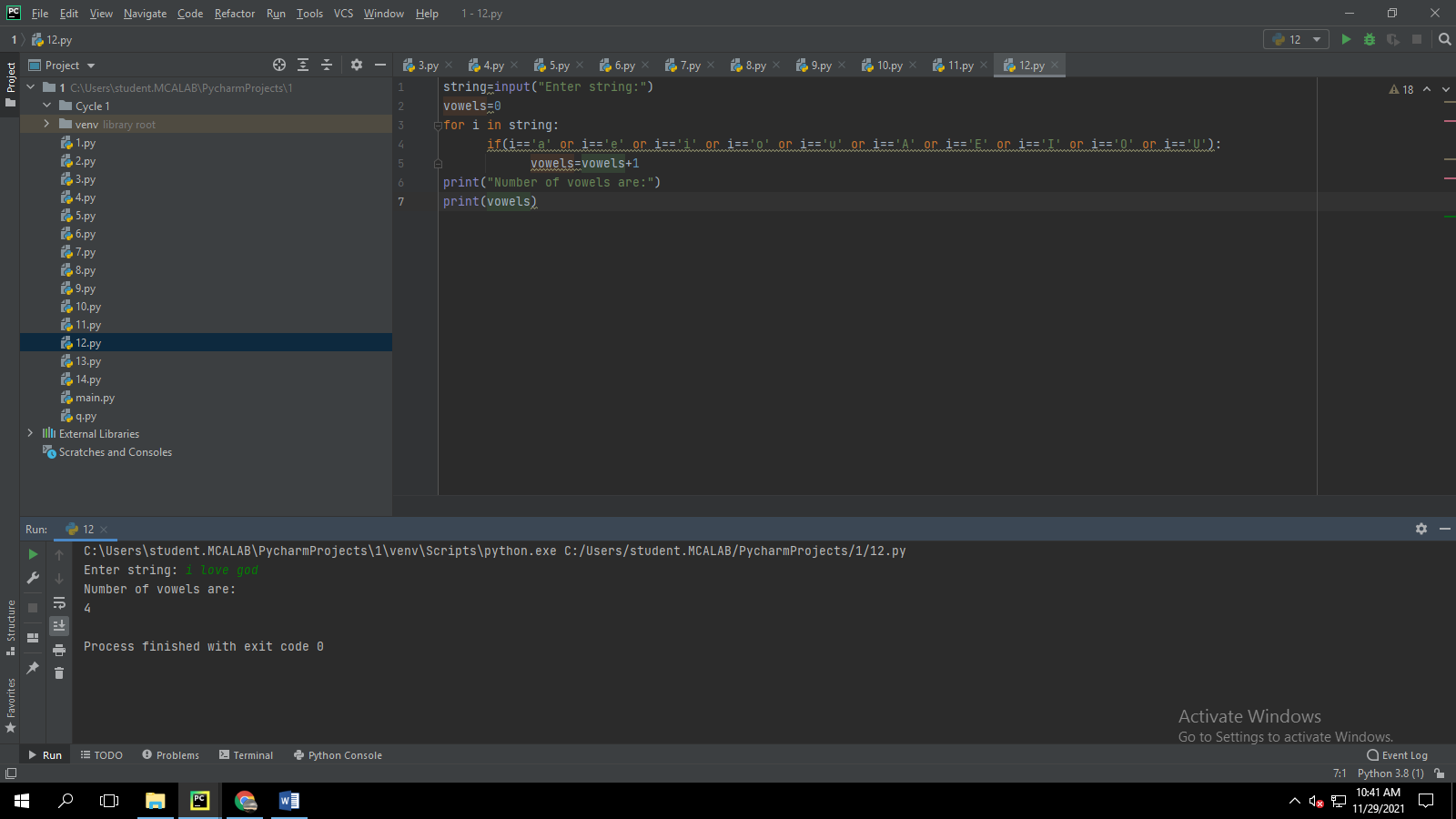
****

**12.** **Program to find the count of each vowel in a string(use dictionary)**

**Code:**

string=input("Enter string:")  
vowels=0  
for i in string:  
 if(i=='a' or i=='e' or i=='i' or i=='o' or i=='u' or i=='A' or i=='E' or i=='I' or i=='O' or i=='U'):  
 vowels=vowels+1  
print("Number of vowels are:")  
print(vowels)

**output:**

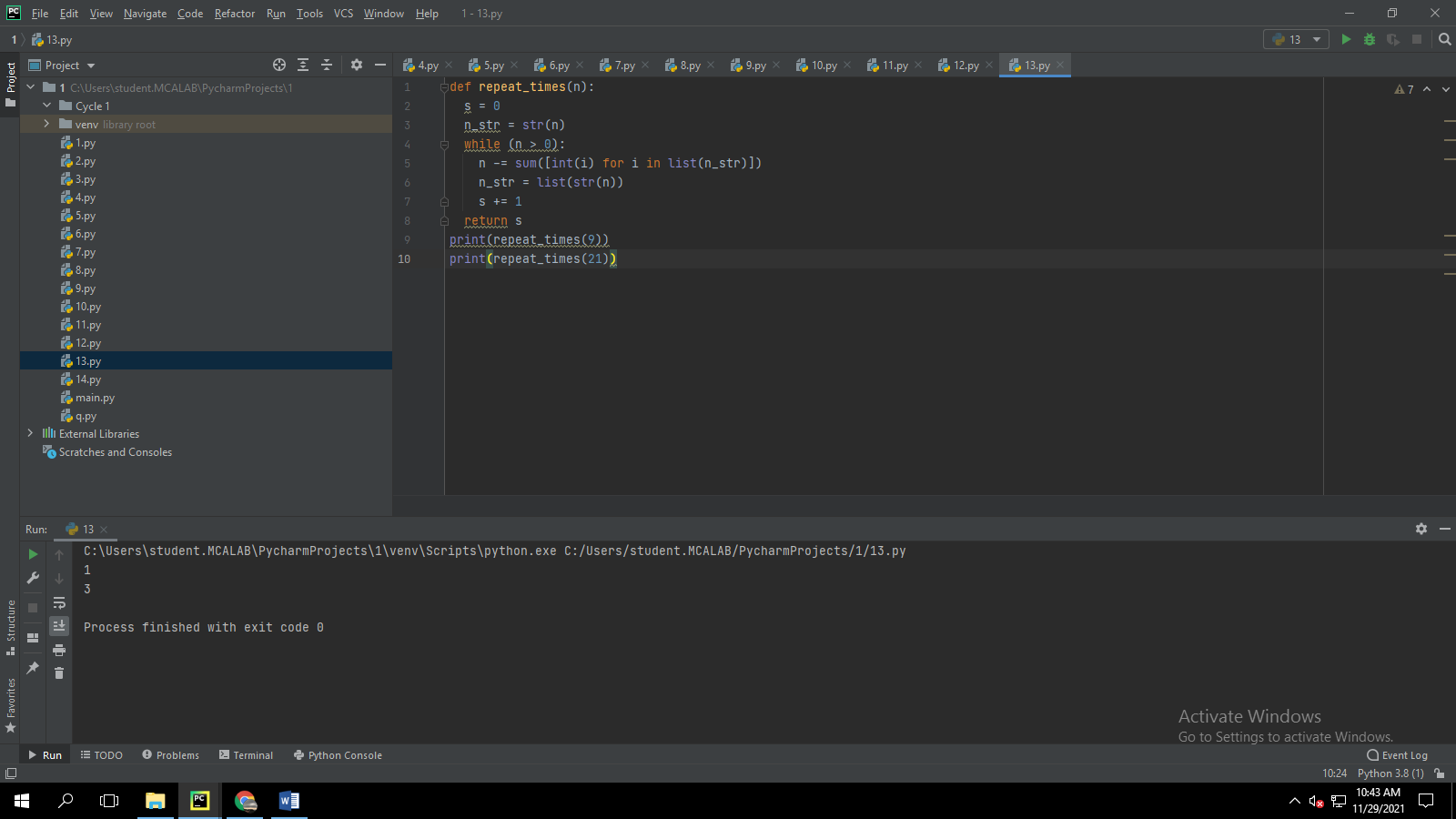
****

**13.** **Write a Python program that accept a positive number and subtract from this number the sum of its digits and so on. Continues this operation until the number is positive**

**Code:**

def repeat\_times(n):  
 s = 0  
 n\_str = str(n)  
 while (n > 0):  
 n -= sum([int(i) for i in list(n\_str)])  
 n\_str = list(str(n))  
 s += 1  
 return s  
print(repeat\_times(9))  
print(repeat\_times(21))

**output:**

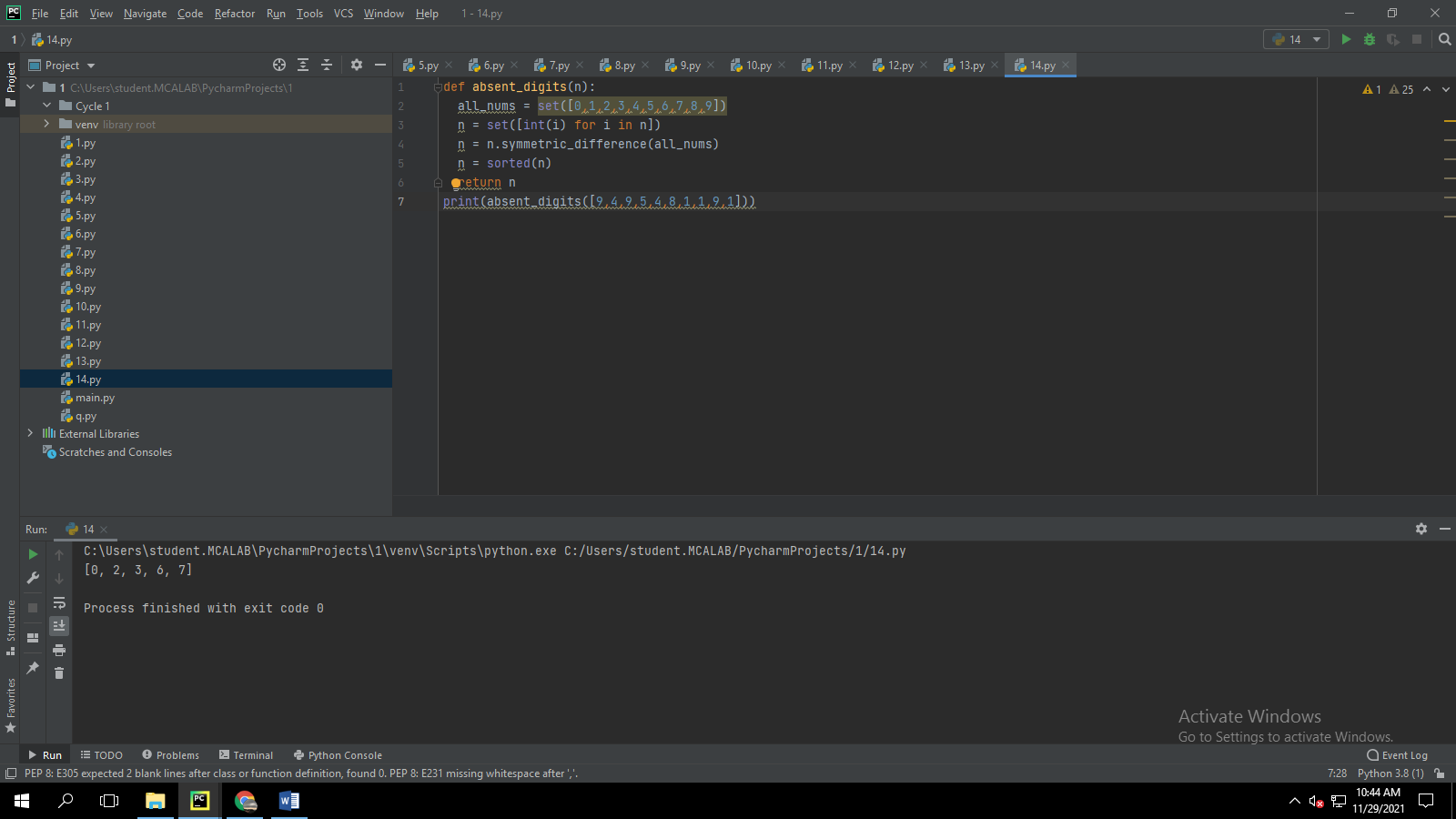
****

**14.** **Write a Python program that accepts a 10 digit mobile number, and find the digits which are absent in a given mobile number.**

**Code:**

def absent\_digits(n):  
 all\_nums = set([0,1,2,3,4,5,6,7,8,9])  
 n = set([int(i) for i in n])  
 n = n.symmetric\_difference(all\_nums)  
 n = sorted(n)  
 return n  
print(absent\_digits([9,4,9,5,4,8,1,1,9,1]))

**output:**

****