**Python HARD Challenges**

**1: Maximize It:**

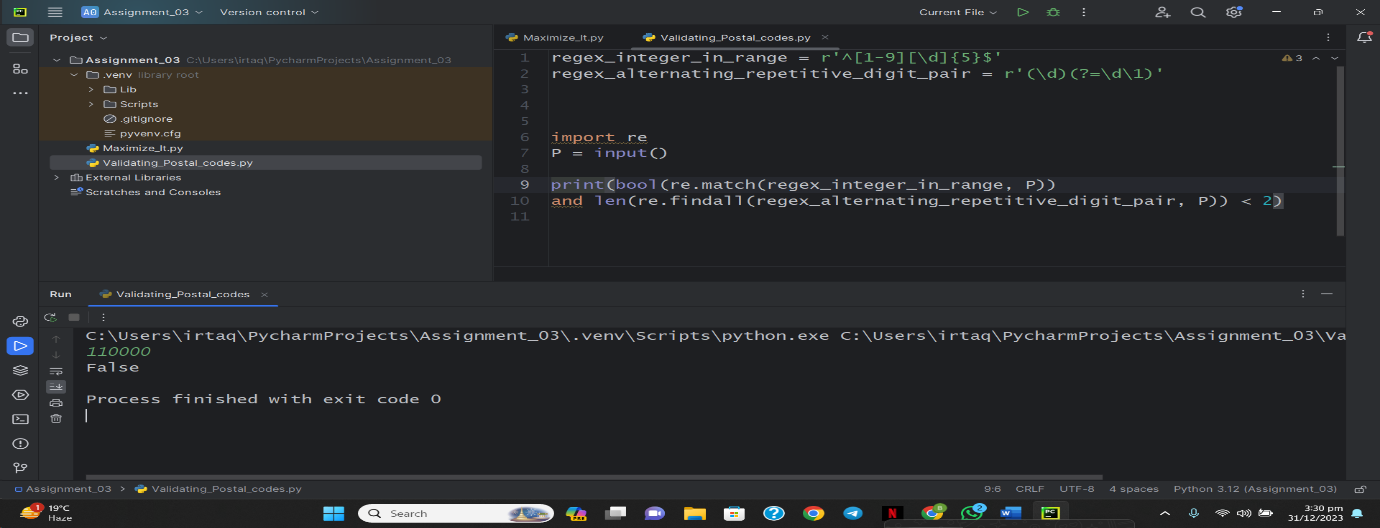
from itertools import product  
K, M = map(int, input().split())  
N = (list(map(int, input().split()))[1:] for \_ in range(K))  
max\_lst = []  
for item in product(\*N):  
 S = 0  
 for val in item:  
 S += val\*\*2  
 S\_max = S % M  
 max\_lst.append(S\_max)  
print(max(max\_lst))

A computer screen shot of a program

Description automatically generated

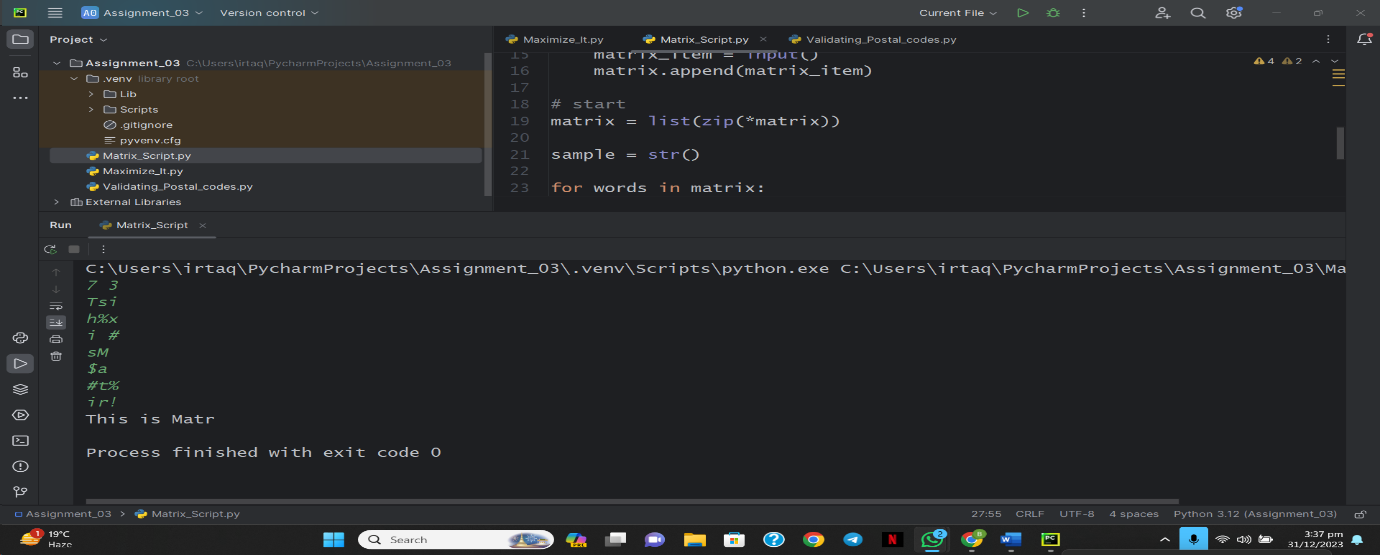
**2: Validating Postal Codes:**

regex\_integer\_in\_range = r'^[1-9][\d]{5}$'  
regex\_alternating\_repetitive\_digit\_pair = r'(\d)(?=\d\1)'  
  
import re  
P = input()  
  
print(bool(re.match(regex\_integer\_in\_range, P))  
and len(re.findall(regex\_alternating\_repetitive\_digit\_pair, P)) < 2)



**3: Matrix Script:**

import math  
import os  
import random  
import re  
import sys  
  
first\_multiple\_input = input().rstrip().split()  
  
n = int(first\_multiple\_input[0])  
m = int(first\_multiple\_input[1])  
  
matrix = []  
  
for \_ in range(n):  
 matrix\_item = input()  
 matrix.append(matrix\_item)  
  
# start  
matrix = list(zip(\*matrix))  
  
sample = str()  
  
for words in matrix:  
 for char in words:  
 sample += char  
  
print(re.sub(r'(?<=\w)([^\w\d]+)(?=\w)', ' ', sample))



**Python Medium Challenges**

**1: Write a Function**

def is\_leap(year):

    leap = False

    if (year % 400 == 0) and (year % 100 == 0):

        leap = True

    elif (year % 4 ==0) and (year % 100 != 0):

        leap = True

    else:

        pass

    return leap

year = int(input())

A computer screen shot of a program

Description automatically generatedprint(is\_leap(year))

**2: The Minion Game**

def minion\_game(string):

    s=len(string)

    vowel = 0

    consonant = 0

    for i in range(s):

        if string[i] in 'AEIOU':

            vowel+=(s-i)

        else:

            consonant+=(s-i)

    if vowel < consonant:

        print('Stuart ' + str(consonant))

    elif vowel > consonant:

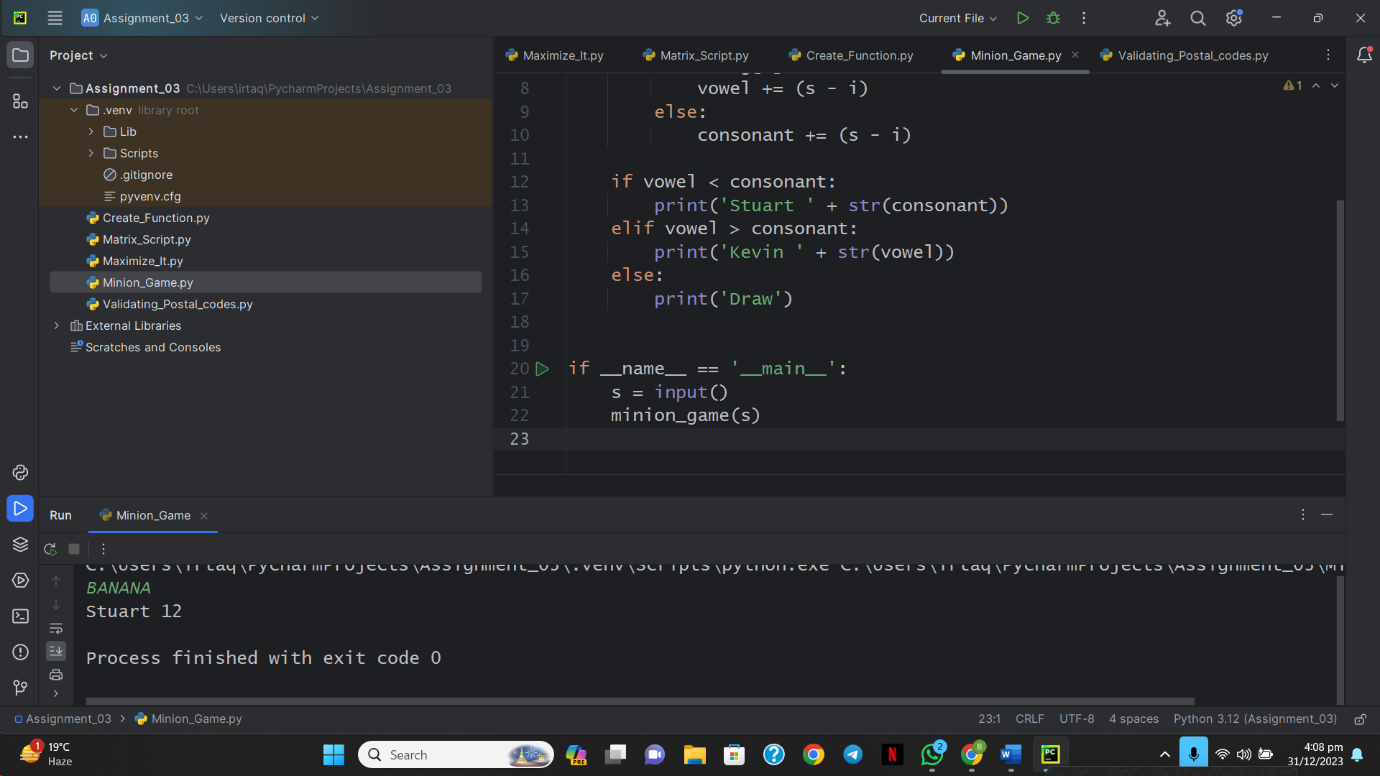
        print('Kevin ' + str(vowel))

    else:

        print('Draw')

if \_\_name\_\_ == '\_\_main\_\_':

    s = input()

    minion\_game(s)

**3: Merge the Tools**

def merge\_the\_tools(string, k):

    temp = []

    len\_temp = 0

    for item in string:

        len\_temp += 1

        if item not in temp:

            temp.append(item)

        if len\_temp == k:

            print(''.join(temp))

            temp = []

            len\_temp = 0

if \_\_name\_\_ == '\_\_main\_\_':

    string, k = input(), int(input())

    merge\_the\_tools(string, k)

**4: Time Delta**

import math

import os

import random

import re

import sys

from datetime import datetime

def time\_delta(t1, t2):

    t1 = datetime.strptime(t1, '%a %d %b %Y %H:%M:%S %z')

    t2 = datetime.strptime(t2, '%a %d %b %Y %H:%M:%S %z')

    return str(int(abs((t1-t2).total\_seconds())))

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    t = int(input())

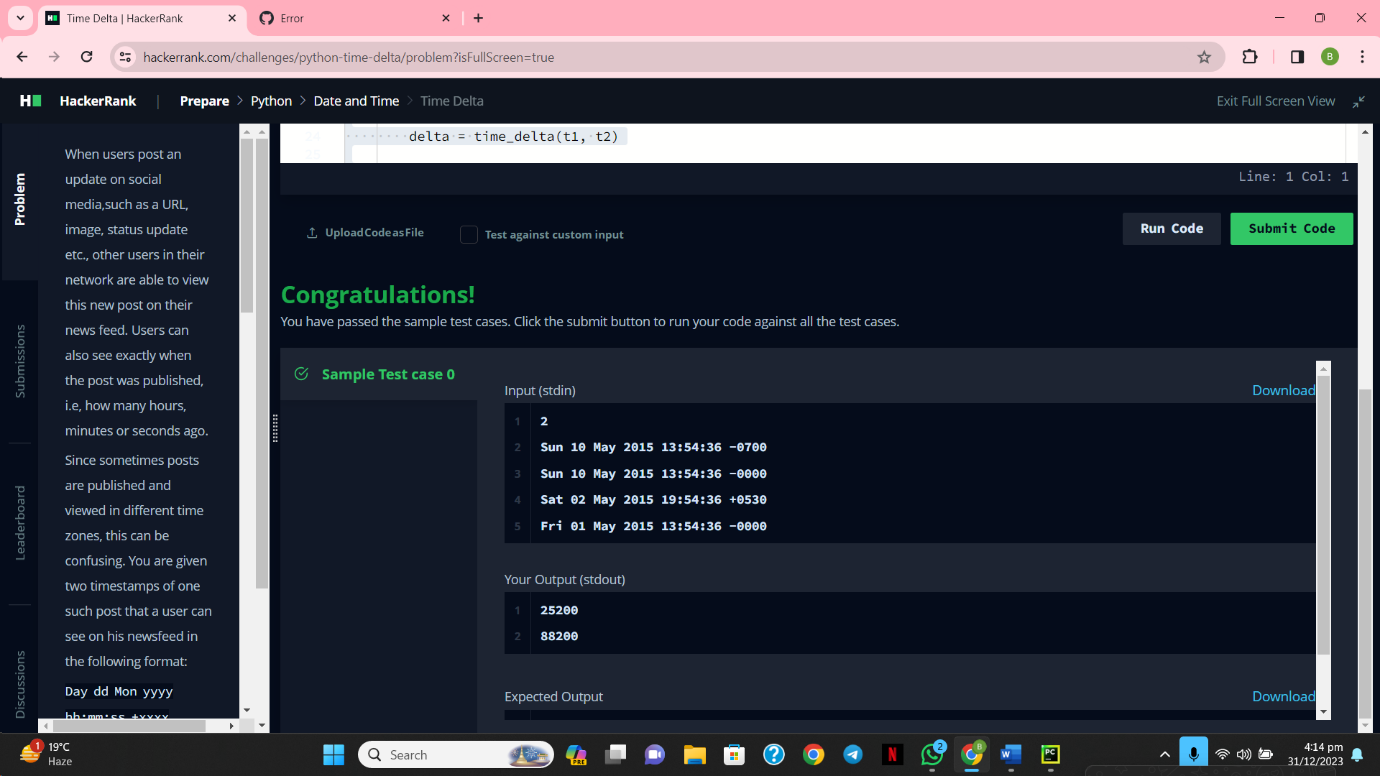
    for t\_itr in range(t):

        t1 = input()

        t2 = input()

        delta = time\_delta(t1, t2)

        fptr.write(delta + '\n')

    fptr.close()

**5: Find Angle MBC**

import math

ab=int(input())

bc=int(input())

ca=math.hypot(ab, bc)

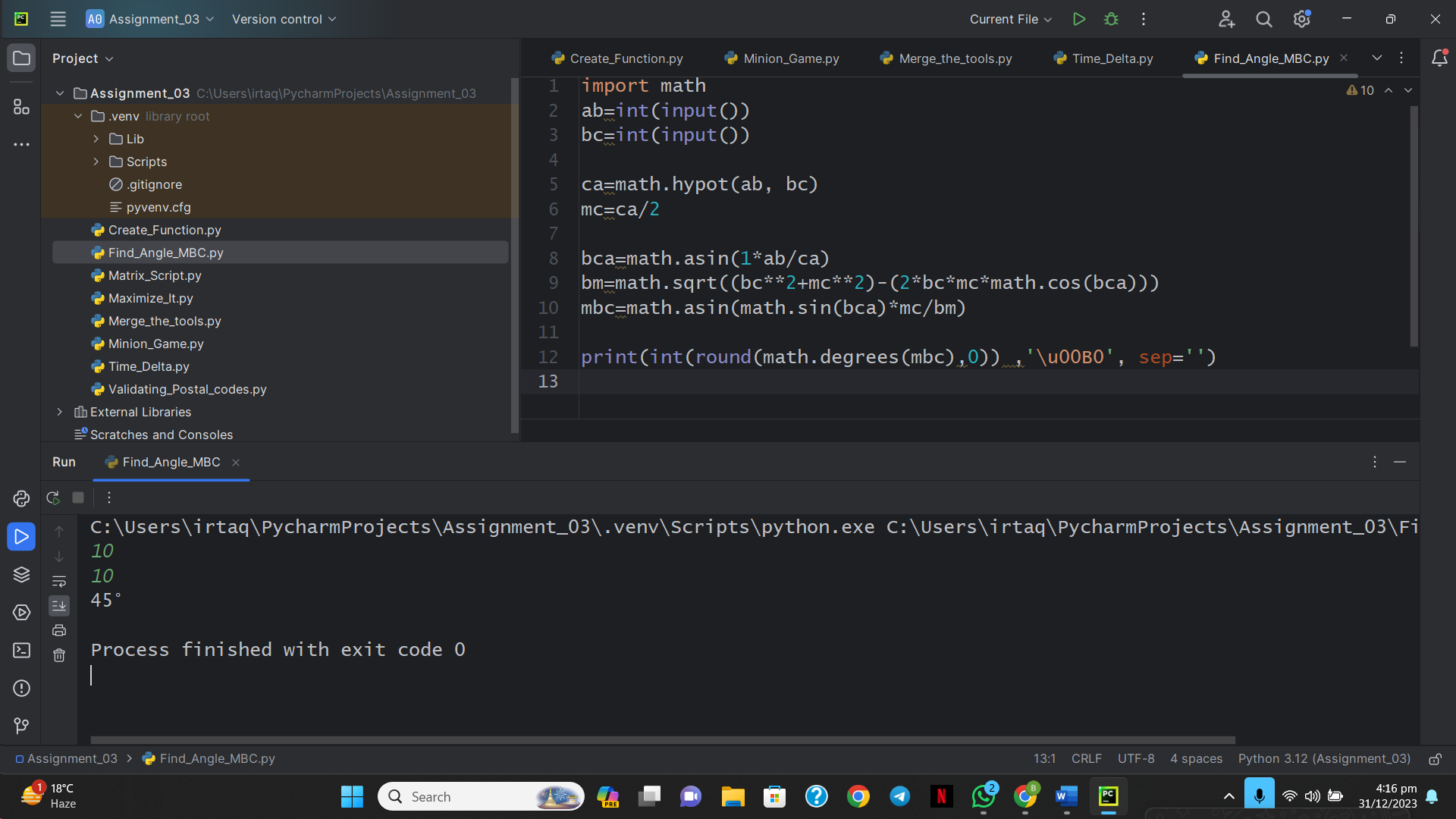
mc=ca/2

bca=math.asin(1\*ab/ca)

bm=math.sqrt((bc\*\*2+mc\*\*2)-(2\*bc\*mc\*math.cos(bca)))

mbc=math.asin(math.sin(bca)\*mc/bm)

print(int(round(math.degrees(mbc),0)) ,'\u00B0', sep='')



**6: No Idea**

if \_\_name\_\_ == "\_\_main\_\_":

    happiness = 0

    n, m = map(int, input().strip().split(' '))

    elements\_arr = list(map(int, input().strip().split(' ')))

    A = set(map(int, input().strip().split(' ')))

    B = set(map(int, input().strip().split(' ')))

    for i in elements\_arr:

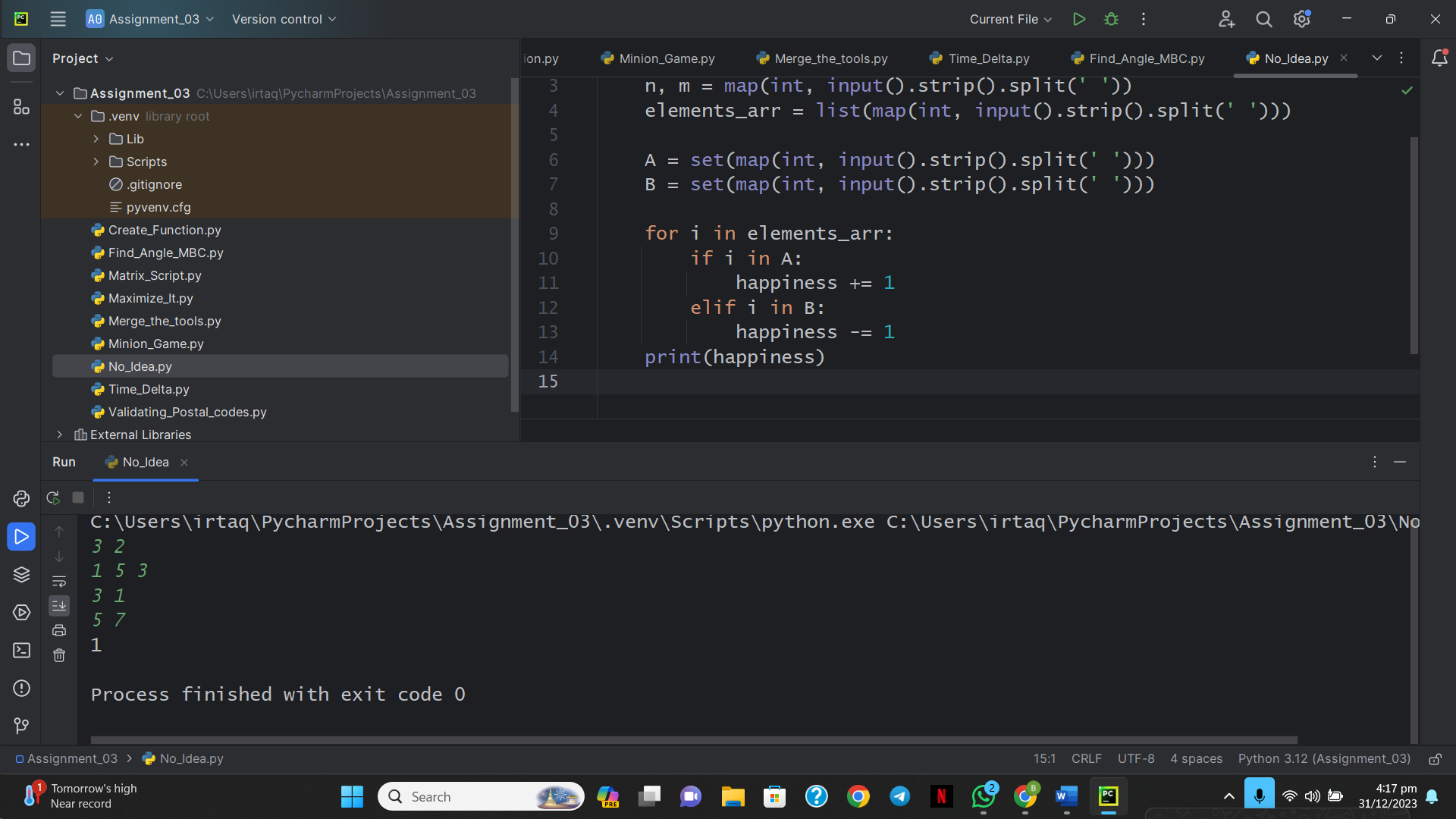
        if i in A:

            happiness += 1

        elif i in B:

            happiness -= 1

    print(happiness)



**7: Word Order**

from collections import Counter

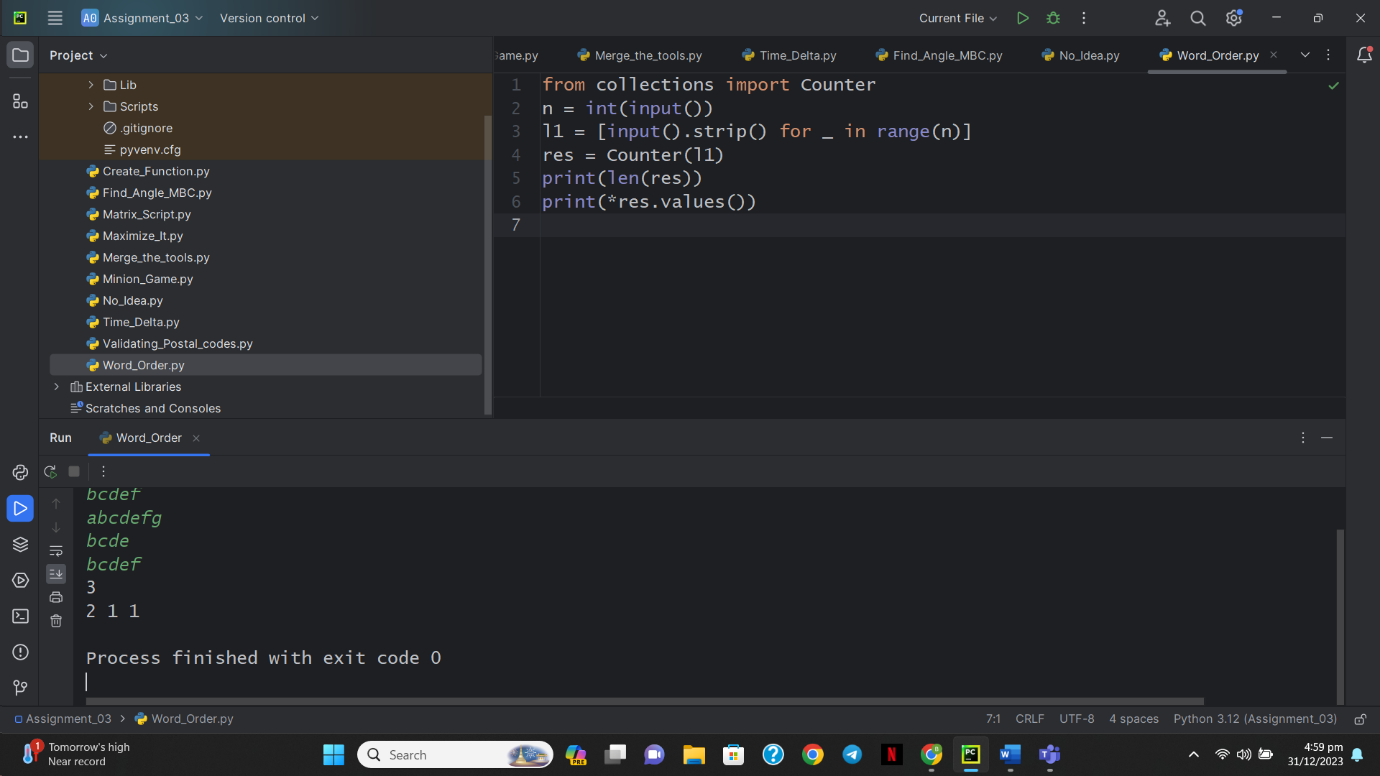
n = int(input())

l1 = [input().strip() for \_ in range(n)]

res = Counter(l1)

print(len(res))

print(\*res.values())



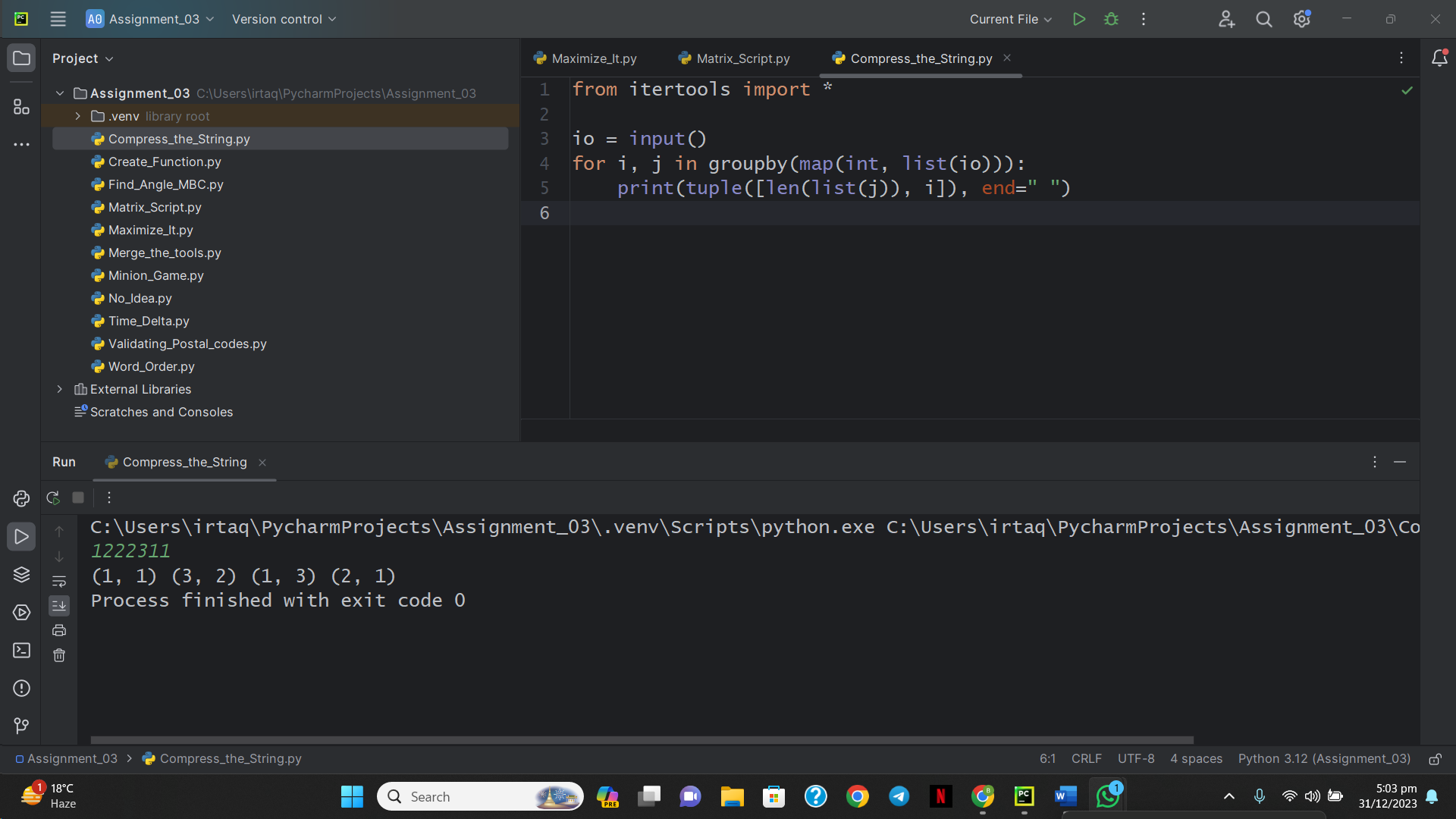
**8: Compress the String**

from itertools import \*

io = input()

for i, j in groupby(map(int,list(io))):

    print(tuple([len(list(j)), i]), end=" ")



**9: Company Logo**

import math

import os

import random

import re

import sys

from collections import Counter

if \_\_name\_\_ == '\_\_main\_\_':

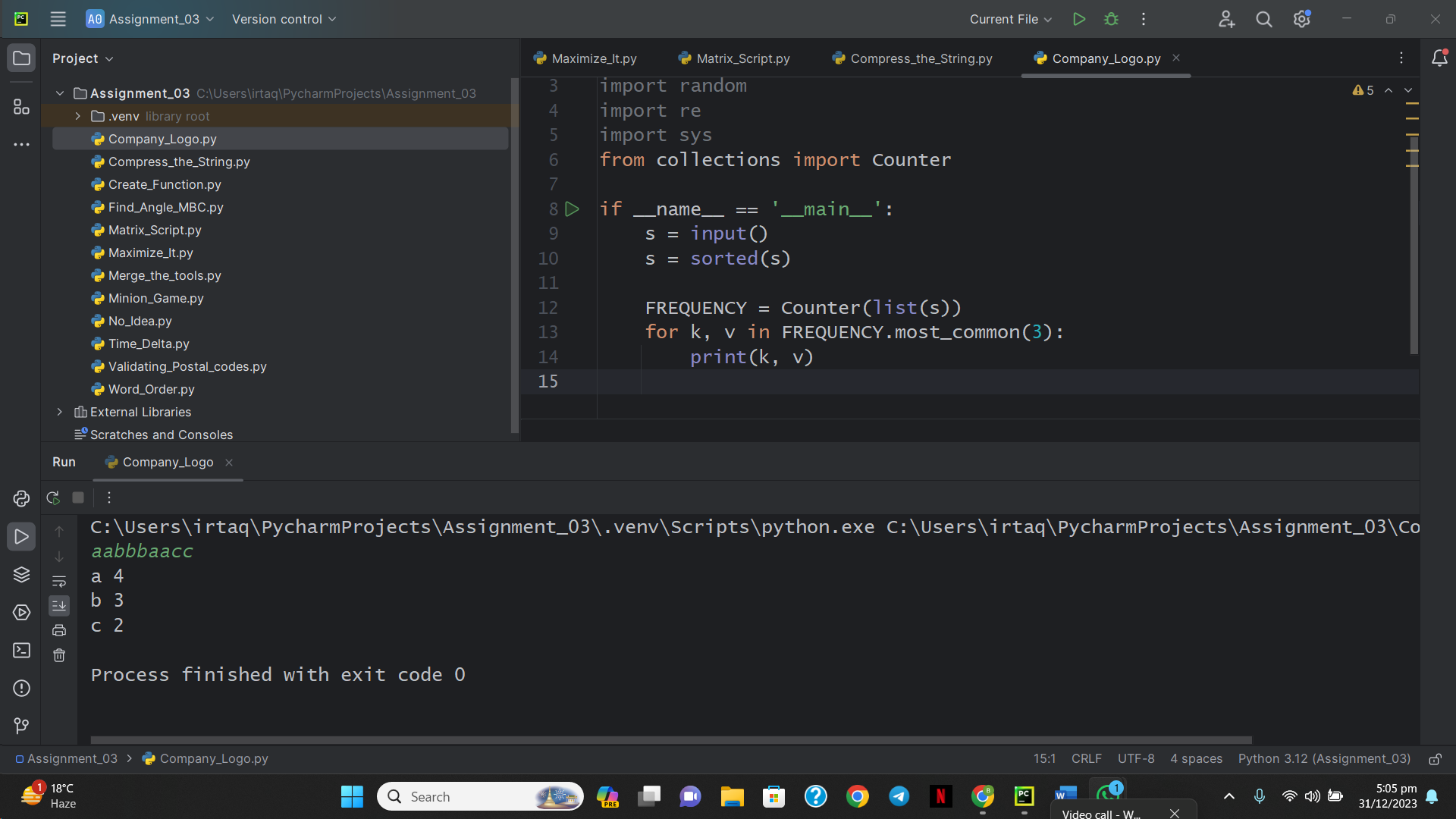
    s = input()

    s = sorted(s)

    FREQUENCY = Counter(list(s))

    for k, v in FREQUENCY.most\_common(3):

        print(k, v)



**10: Piling Up**

ANS = []

T = int(input())

for \_ in range(T):

    n = int(input())

    sl = list(map(int, input().split()))

    for \_ in range(n-1):

        if sl[0] >= sl[len(sl)-1]:

            a = sl[0]

            sl.pop(0)

        elif sl[0] < sl[len(sl)-1]:

            a = sl[len(sl)-1]

            sl.pop(len(sl)-1)

        else:

            pass

        if len(sl) == 1:

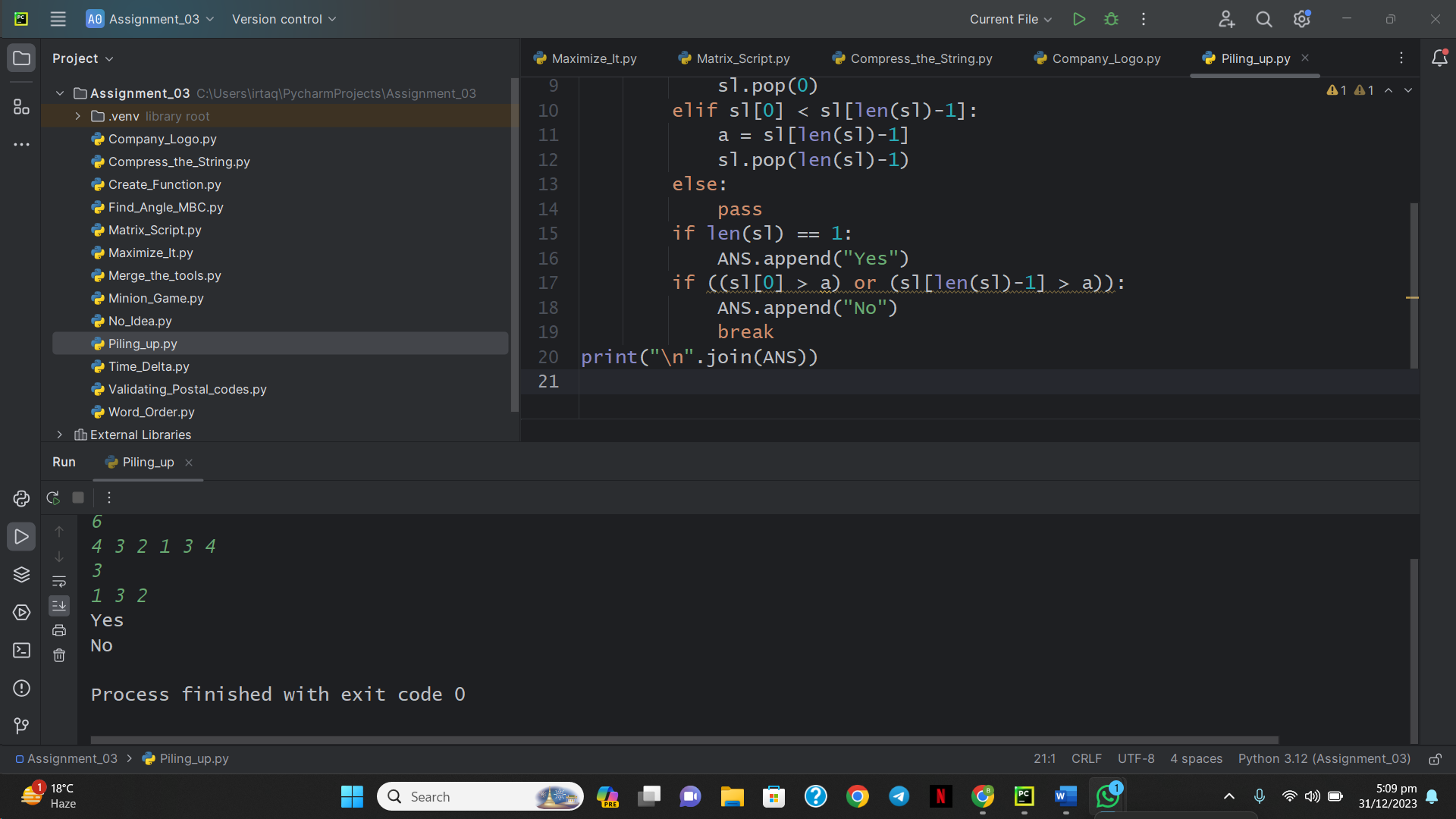
            ANS.append("Yes")

        if ((sl[0] > a) or (sl[len(sl)-1] > a)):

            ANS.append("No")

            break

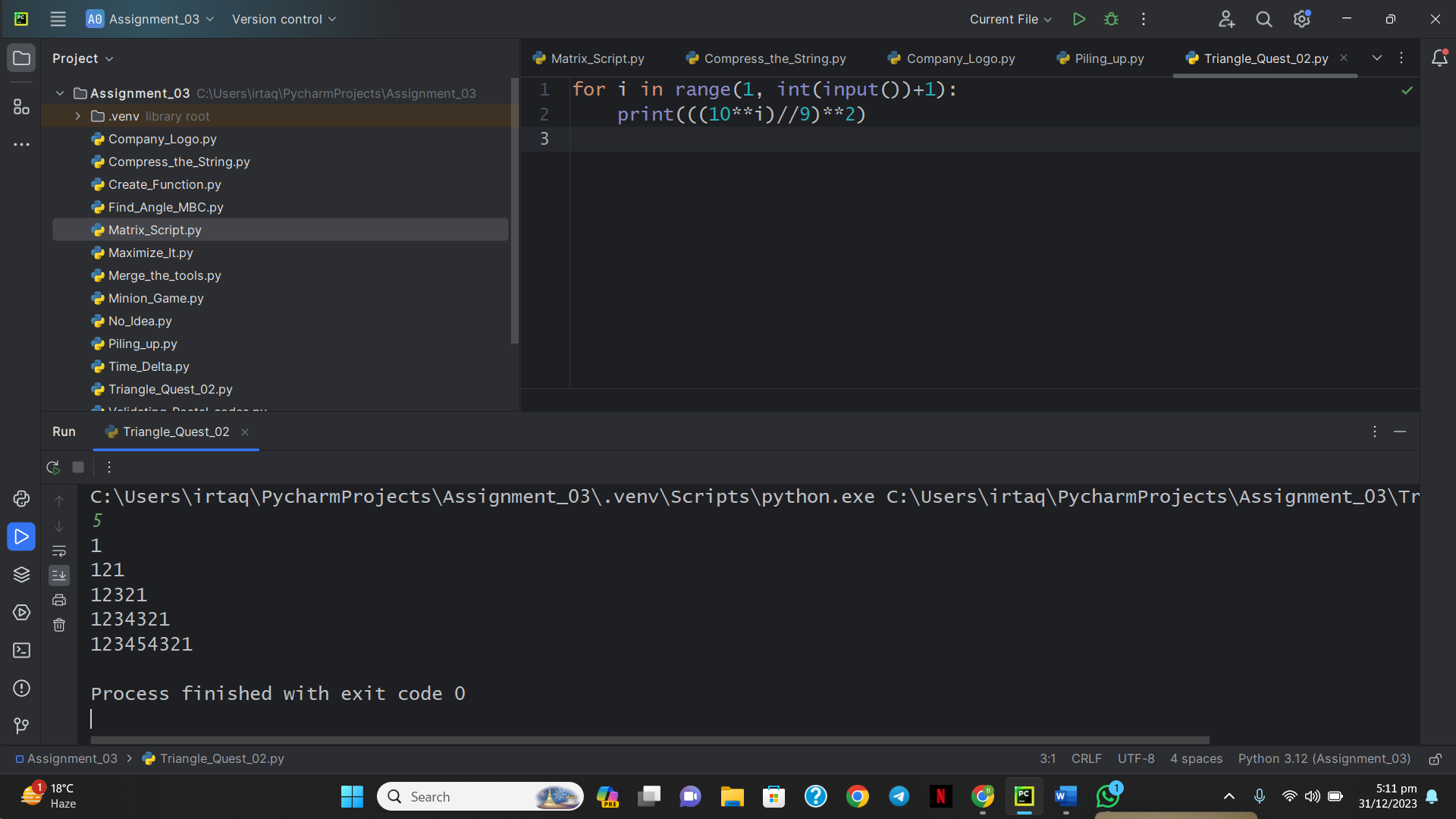
print("\n".join(ANS))



**11: Triangle Quest 2**

for i in range(1, int(input())+1):

    print(((10\*\*i)//9)\*\*2)



**12: Iterables and Iterators**

from itertools import combinations

N = int(input())

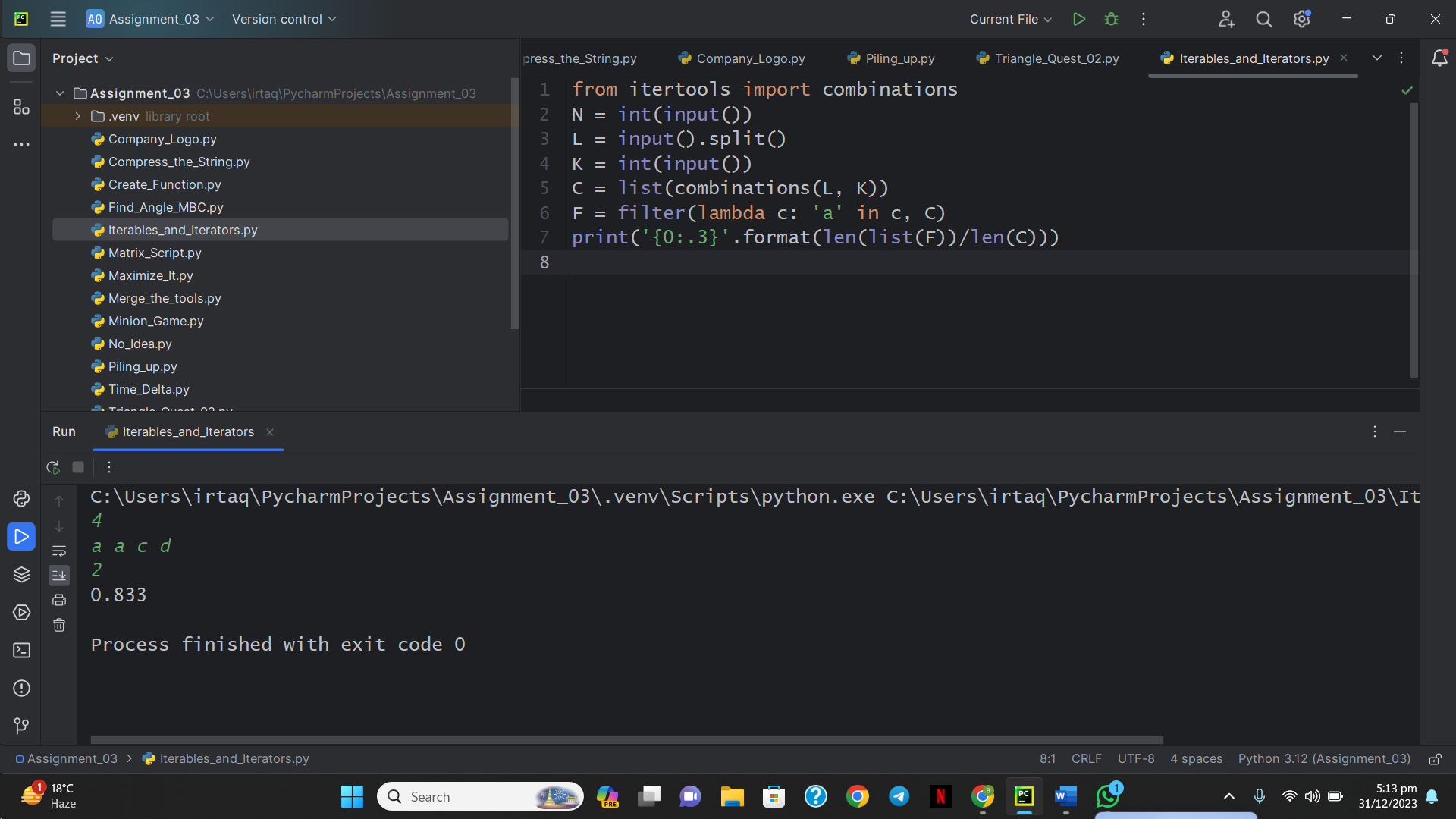
L = input().split()

K = int(input())

C = list(combinations(L, K))

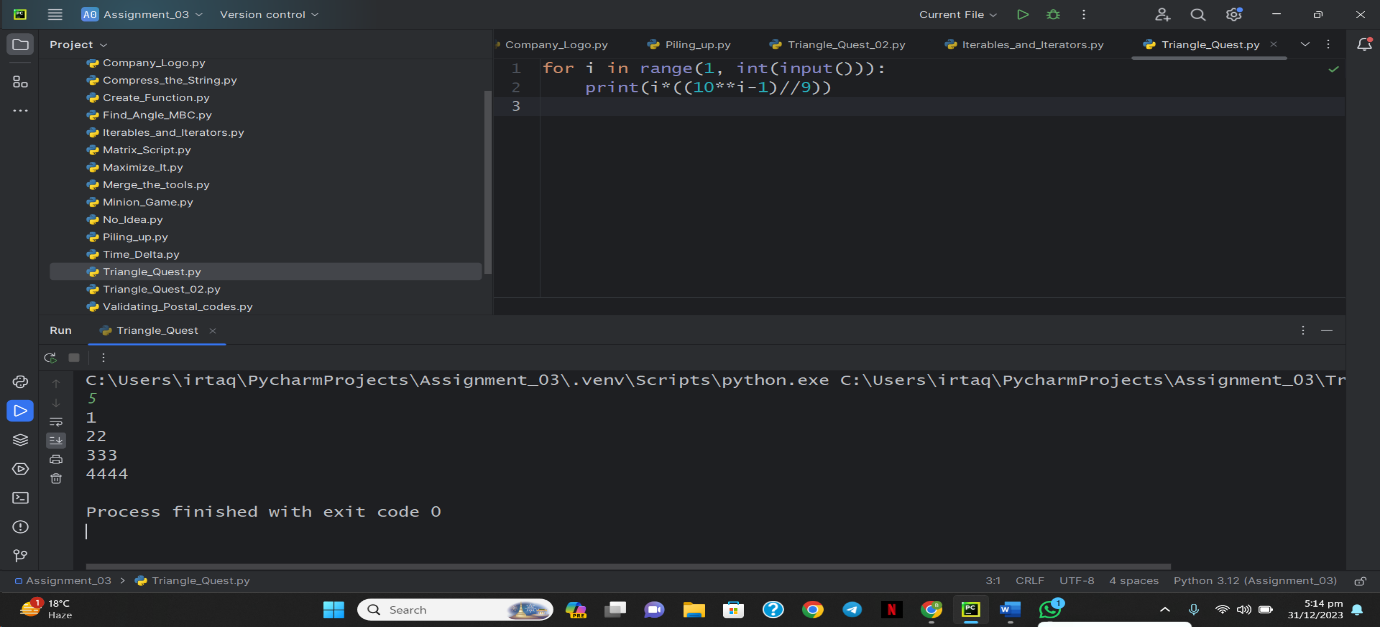
F = filter(lambda c: 'a' in c, C)

print('{0:.3}'.format(len(list(F))/len(C)))



**13: Triangle Quest**

for i in range(1, int(input())):

    print(i\*((10\*\*i-1)//9))

**14: Classes: Dealing with Complex Numbers**

import math

class Complex(object):

    def \_\_init\_\_(self, real, imaginary):

        self.real = real

        self.imaginary = imaginary

    def \_\_add\_\_(self, no):

        return Complex(self.real + no.real,self.imaginary + no.imaginary)

    def \_\_sub\_\_(self, no):

        return Complex(self.real - no.real,self.imaginary - no.imaginary)

    def \_\_mul\_\_(self, no):

        prod = complex(self.real , self.imaginary)\*complex(no.real , no.imaginary)

        return Complex(prod.real , prod.imag)

    def \_\_truediv\_\_(self, no):

        div = complex(self.real , self.imaginary)/complex(no.real , no.imaginary)

        return Complex(div.real , div.imag)

    def mod(self):

        m = math.sqrt(self.real\*\*2 + self.imaginary\*\*2)

        return Complex(m,0)

    def \_\_str\_\_(self):

        if self.imaginary == 0:

            result = "%.2f+0.00i" % (self.real)

        elif self.real == 0:

            if self.imaginary >= 0:

                result = "0.00+%.2fi" % (self.imaginary)

            else:

                result = "0.00-%.2fi" % (abs(self.imaginary))

        elif self.imaginary > 0:

            result = "%.2f+%.2fi" % (self.real, self.imaginary)

        else:

            result = "%.2f-%.2fi" % (self.real, abs(self.imaginary))

        return result

if \_\_name\_\_ == '\_\_main\_\_':

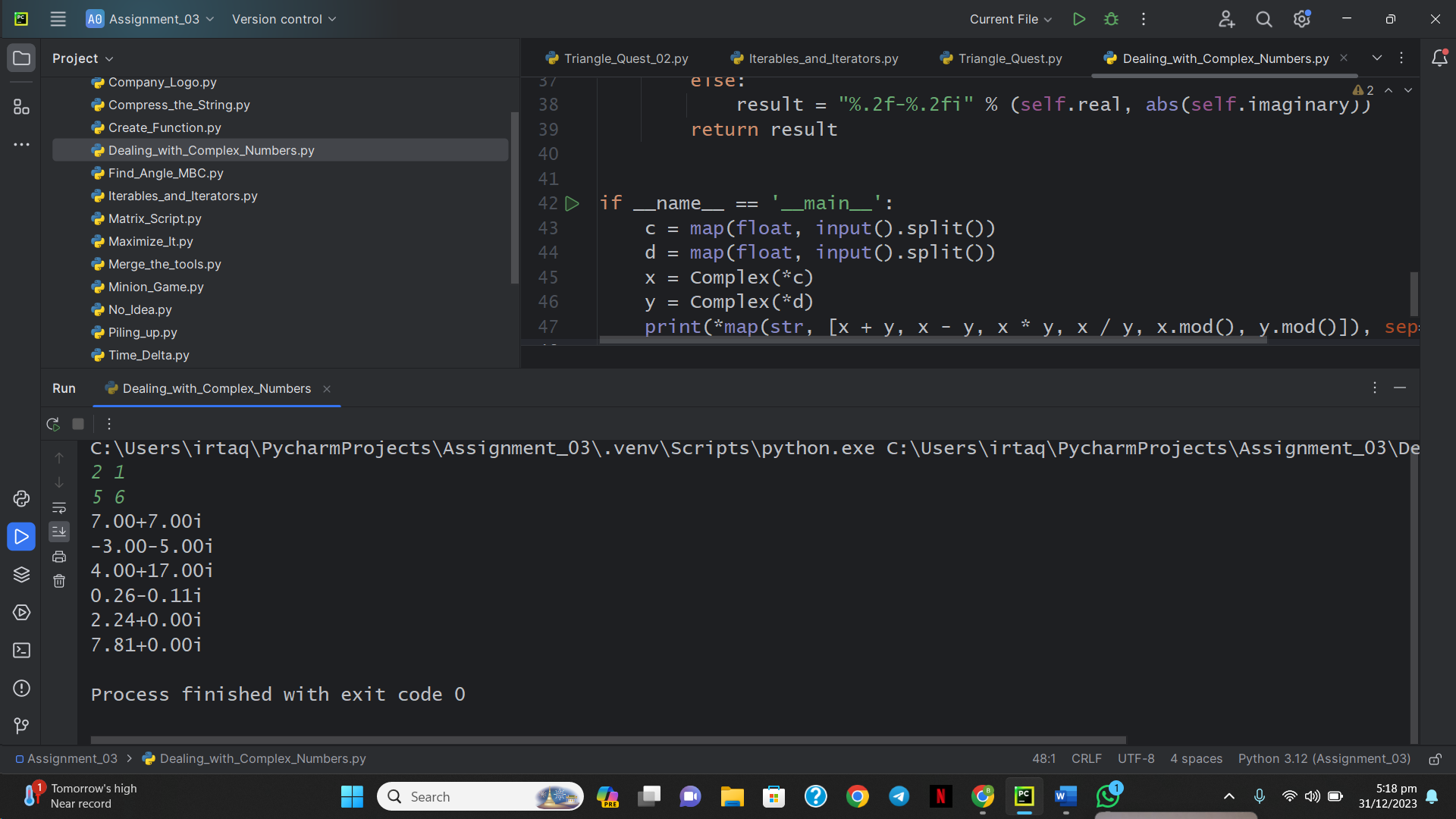
    c = map(float, input().split())

    d = map(float, input().split())

    x = Complex(\*c)

    y = Complex(\*d)

    print(\*map(str, [x+y, x-y, x\*y, x/y, x.mod(), y.mod()]), sep='\n')



**15: Athlete Sort**

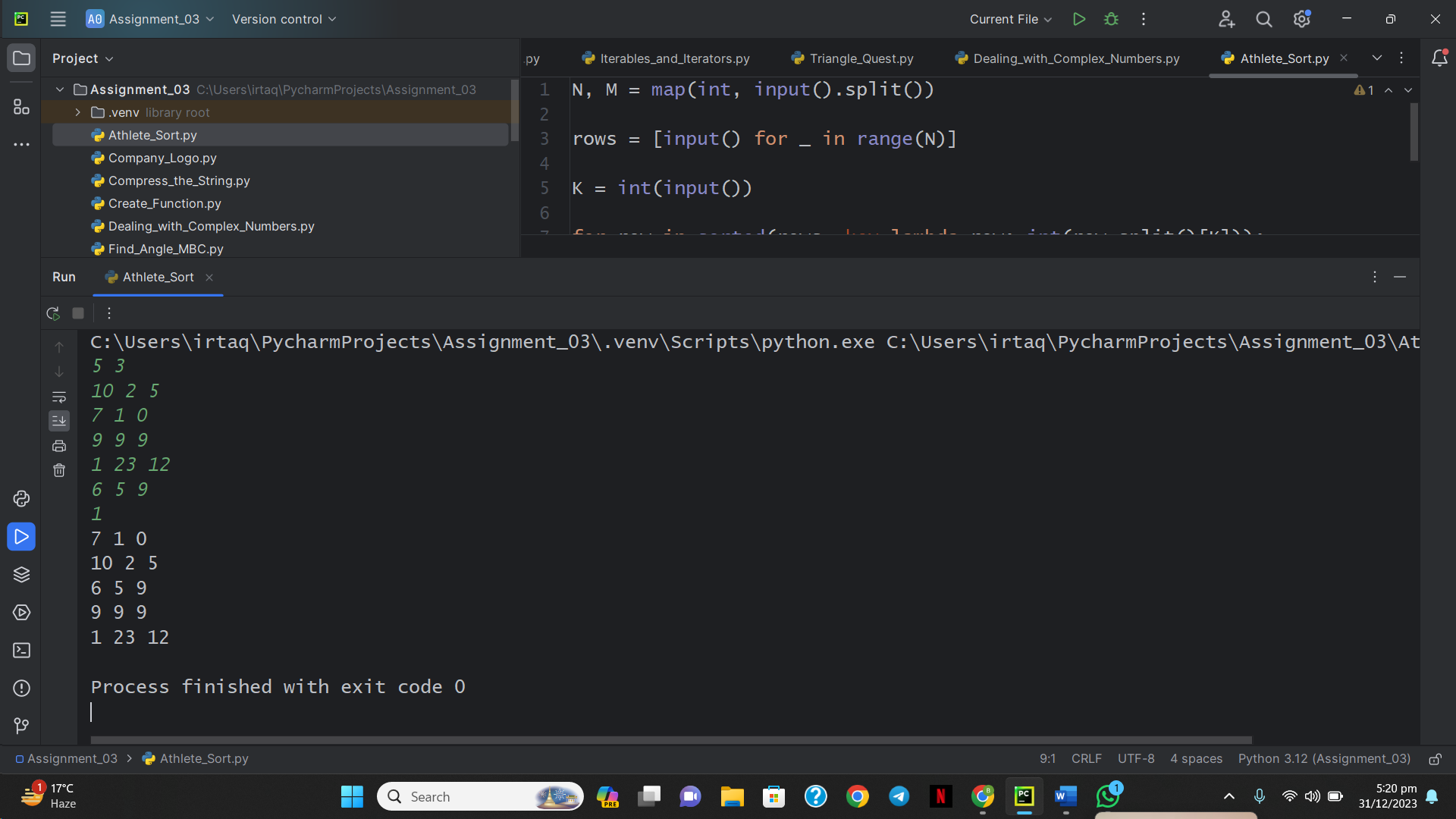
N, M = map(int, input().split())

rows = [input() for \_ in range(N)]

K = int(input())

for row in sorted(rows, key=lambda row: int(row.split()[K])):

    print(row)



**16: ginortS**

s = input()

upper, lower, odd, even =[], [], [], []

for char in s:

    if char.isnumeric():

        if int(char) % 2==0:

            even.append(char)

        else:

            odd.append(char)

    else:

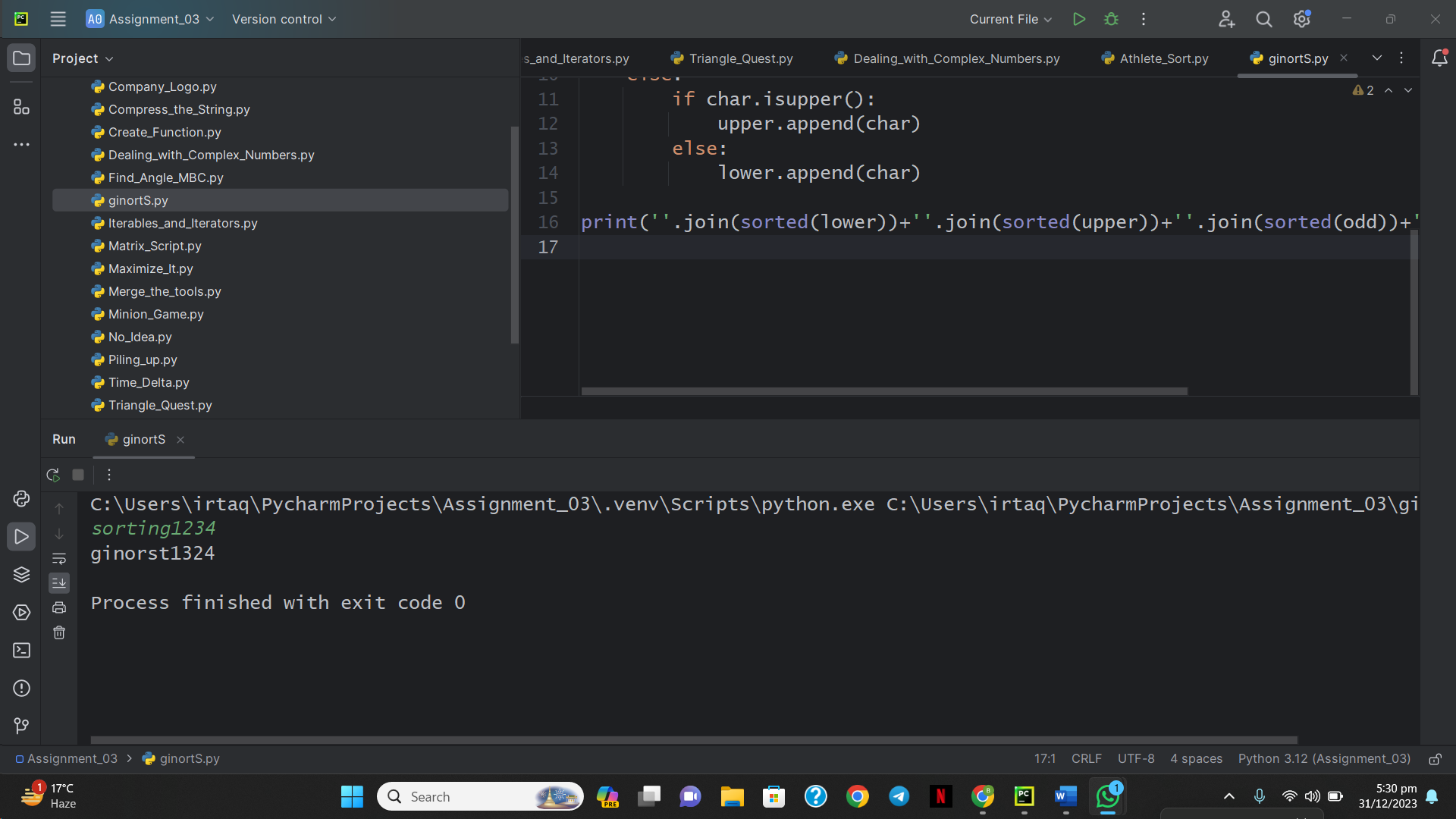
        if char.isupper():

            upper.append(char)

        else:

            lower.append(char)

print(''.join(sorted(lower))+''.join(sorted(upper))+''.join(sorted(odd))+''.join(sorted(even)))



**17: Validating Email Address with a Filter**

def fun(s):

    try:

        username, url = s.split('@')

        website, extension = url.split('.')

    except ValueError:

        return False

    if username.replace('-', '').replace('\_', '').isalnum() is False:

        return False

    elif website.isalnum() is False:

        return False

    elif len(extension) > 3:

        return False

    else:

        return True

def filter\_mail(emails):

    return list(filter(fun, emails))

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    emails = []

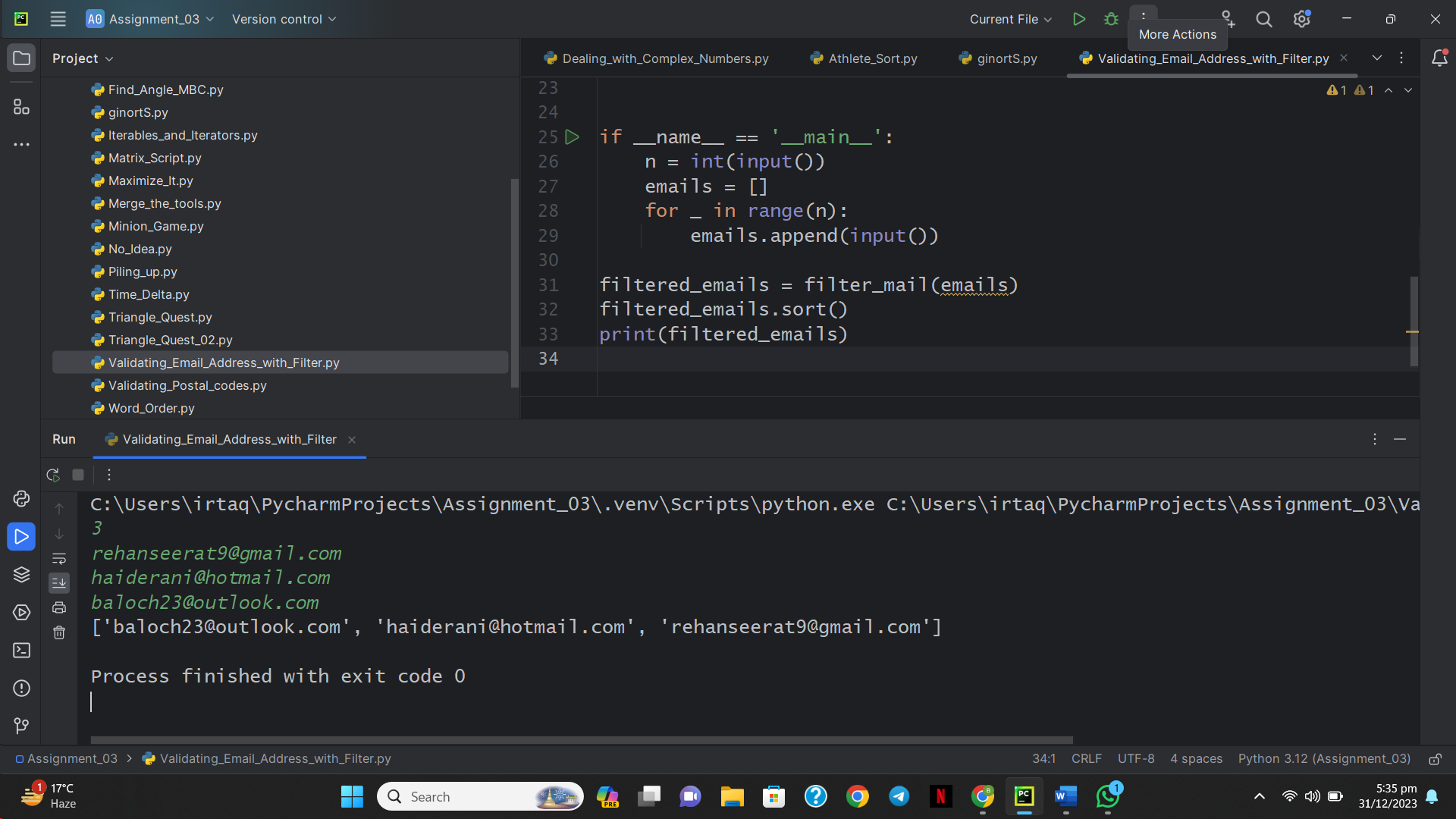
    for \_ in range(n):

        emails.append(input())

filtered\_emails = filter\_mail(emails)

filtered\_emails.sort()

print(filtered\_emails)



**18: Reduce Function**

rom fractions import Fraction

from functools import reduce

def product(fracs):

    t = Fraction(reduce(lambda x, y: x \* y, fracs))

    return t.numerator, t.denominator

if \_\_name\_\_ == '\_\_main\_\_':

    fracs = []

    for \_ in range(int(input())):

        fracs.append(Fraction(\*map(int, input().split())))

    result = product(fracs)

    print(\*result)



**19: Regex Substitution**

import re

def change(match):

    if match.group(1) == '&&':

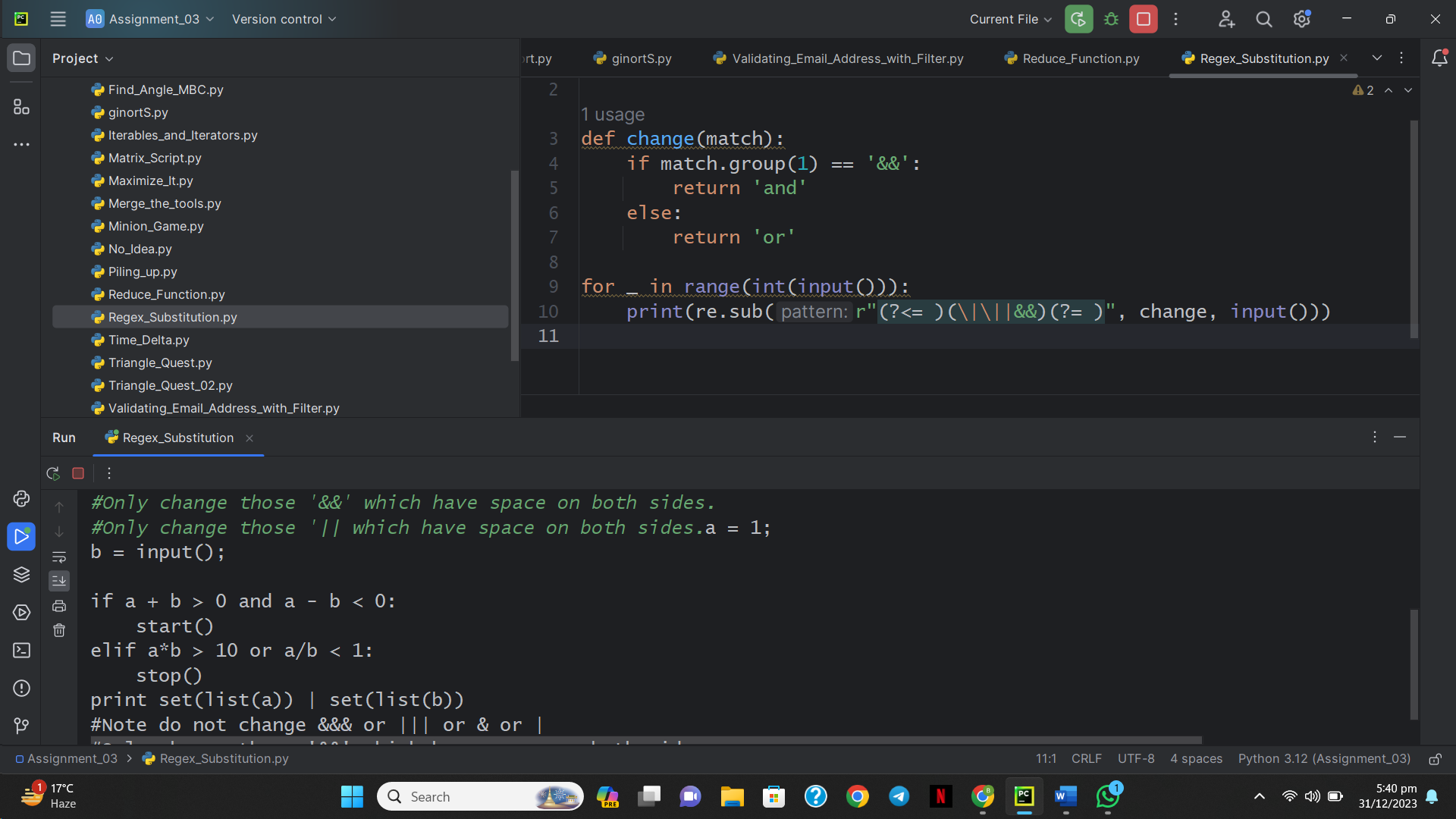
        return 'and'

    else:

        return 'or'

for \_ in range(int(input())):

    print(re.sub(r"(?<= )(\|\||&&)(?= )", change, input()))



**20: Validating Credit Card Numbers**

import re

pattern = re.compile(

    r'^'

    r'(?!.\*(\d)(-?\1){3})'

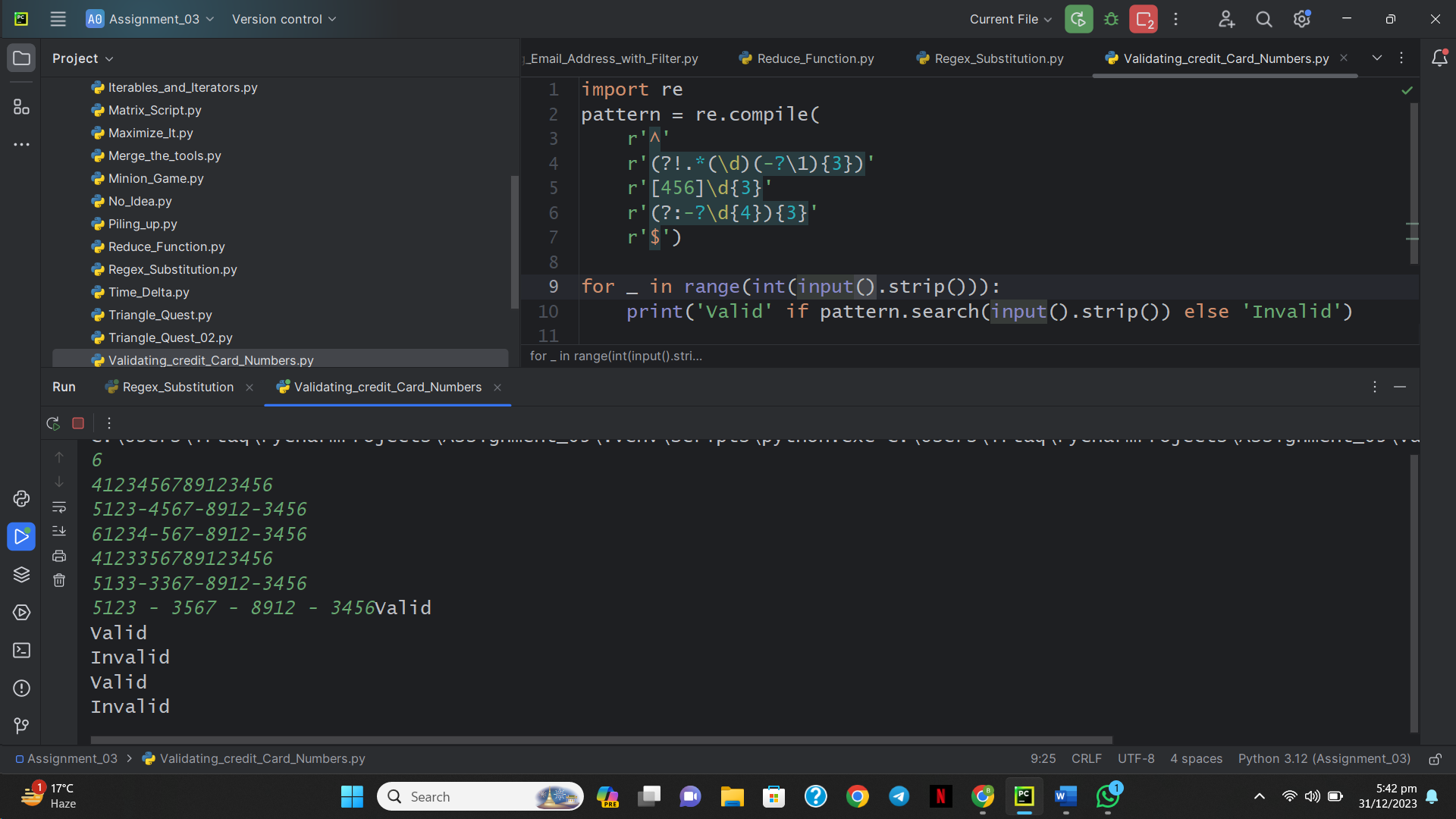
    r'[456]\d{3}'

    r'(?:-?\d{4}){3}'

    r'$')

for \_ in range(int(input().strip())):

    print('Valid' if pattern.search(input().strip()) else 'Invalid')



**21: Words Score**

def is\_vowel(letter):

    return letter in ['a', 'e', 'i', 'o', 'u', 'y']

def score\_words(words):

    score = 0

    for word in words:

        num\_vowels = 0

        for letter in word:

            if is\_vowel(letter):

                num\_vowels += 1

        if num\_vowels % 2 == 0:

            score += 2

        else:

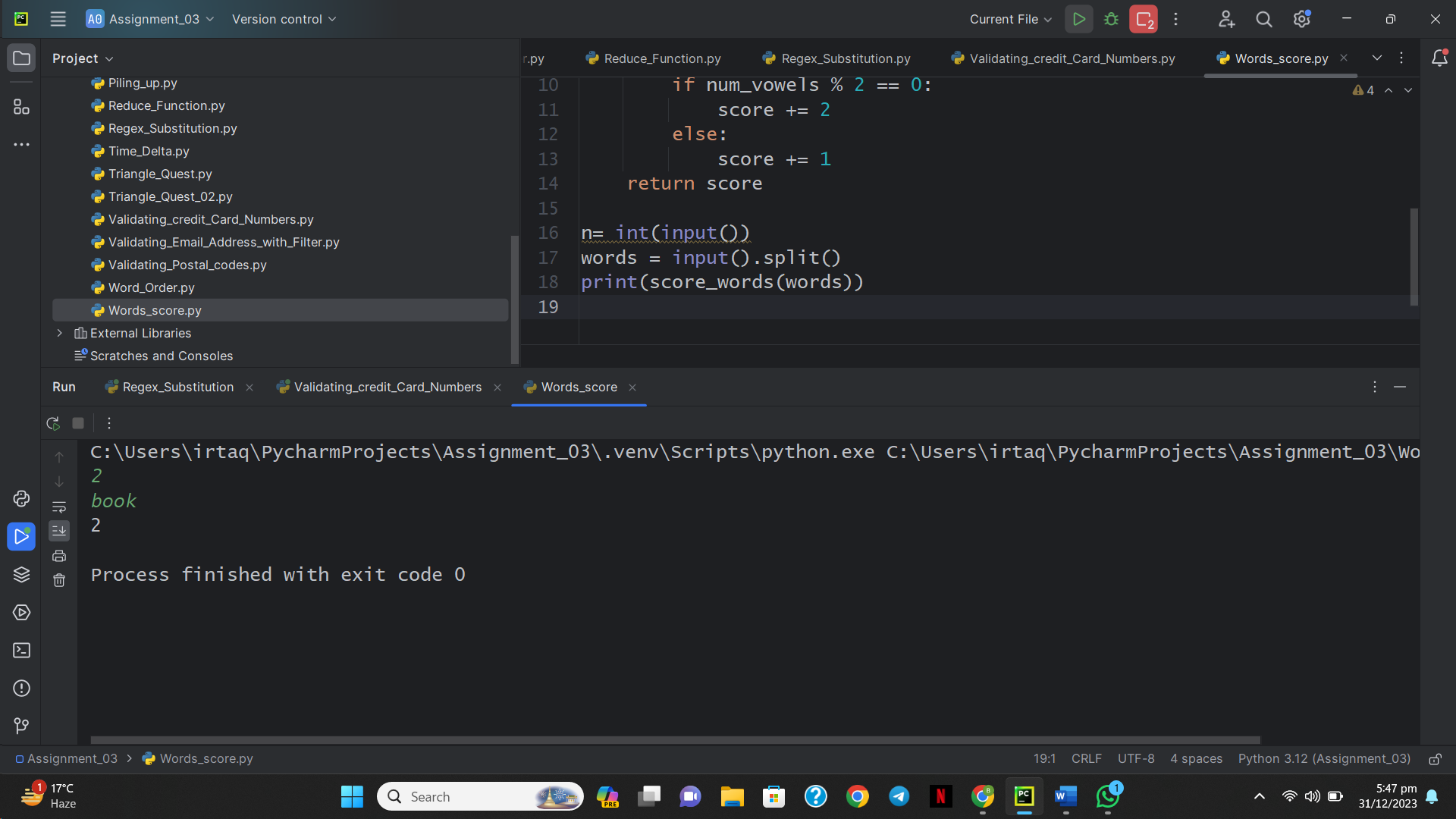
            score += 1

    return score

n= int(input())

words = input().split()

print(score\_words(words))



**22: Default Arguments**

class EvenStream(object):

    def \_\_init\_\_(self):

        self.current = 0

    def get\_next(self):

        to\_return = self.current

        self.current += 2

        return to\_return

class OddStream(object):

    def \_\_init\_\_(self):

        self.current = 1

    def get\_next(self):

        to\_return = self.current

        self.current += 2

        return to\_return

def print\_from\_stream(n, stream=None):

    if stream is None:

        stream = EvenStream()

    for \_ in range(n):

        print(stream.get\_next())

queries = int(input())

for \_ in range(queries):

    stream\_name, n = input().split()

    n = int(n)

    if stream\_name == "even":

        print\_from\_stream(n)

    else:

        print\_from\_stream(n, OddStream())

