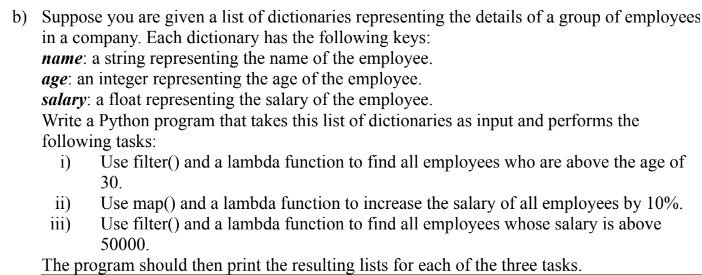
**13/05/2023:** P1 Set 2 (Q.2 (b))

****

employees = [

{"name": "John Doe", "age": 25, "salary": 40000.0},

{"name": "Jane Doe", "age": 35, "salary": 60000.0},

{"name": "Bob Smith", "age": 45, "salary": 80000.0},

{"name": "Alice Johnson", "age": 30, "salary": 55000.0},

{"name": "Mike Williams", "age": 28, "salary": 45000.0},

]

# filtering employees > 30

print("employees above the age of 30: ")

aboveAge = filter(lambda y: y['age'] > 30, employees)

for y in aboveAge:

    print(y)

# increase salary by 10%

print("\nemployees with increased salary")

increSalary = map(lambda y: {'name': y['name'], 'age': y['age'], 'salary': round(y['salary']\*1.1)}, employees)

for y in increSalary:

    print(y)

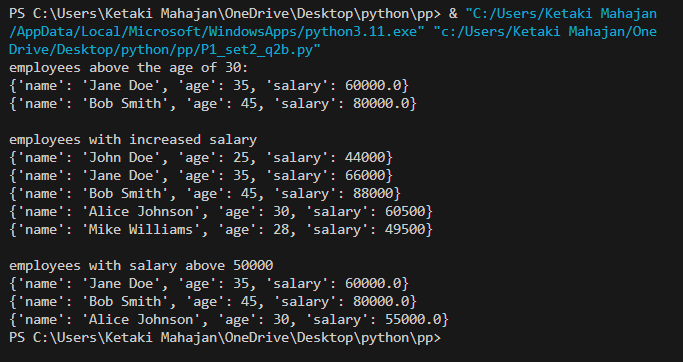
# filtering employees with salary > 50000

print("\nemployees with salary above 50000")

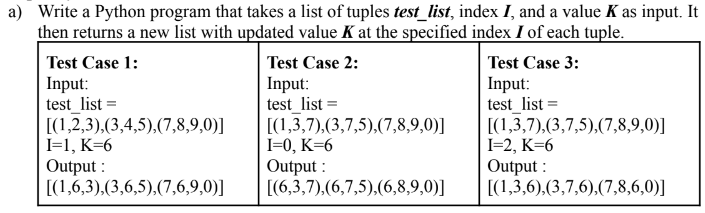
aboveSalary = filter(lambda y: y['salary'] > 50000, employees)

for y in aboveSalary:

    print(y)

****

**12/05/2023:** P1 Set 2 (Q.2 (a))

****

def updated\_tuple(tupList, i, k):

    result = [] # initializing list

    for tup in tupList: # iterating through each tuple

        tupTolist = list(tup) # converting to list

        tupTolist[i] = k # update values at index i with k

        result.append(tuple(tupTolist))

    return result

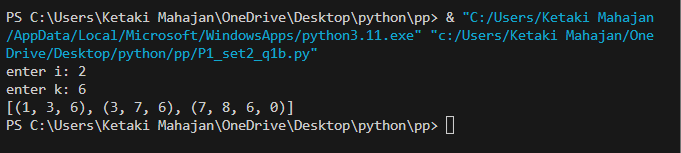
tupList = [(1,3,7),(3,7,5),(7,8,9,0)]

i = int(input("enter i: "))

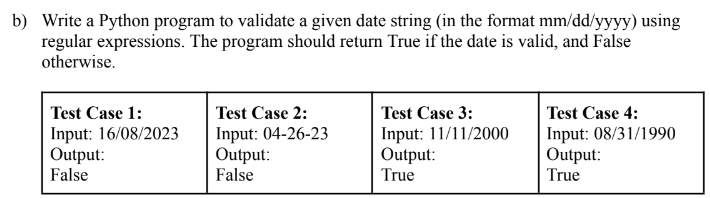
k = int(input("enter k: "))

result = updated\_tuple(tupList, i, k)

print(result)

****

**11/05/2023:** P1 Set 2 (Q.1 (b))

****

import re

def is\_valid(date):

    pattern = r'^([1-9]|[1-3][0-1])/([1-9]|[1-1][0-2])/([0-9]{4})$'

    # str day from 1-31, slash, month 1-12, slash, 4-digit year

    if re.match(pattern, date): # using re.match to match input with pattern

        return True

    else:

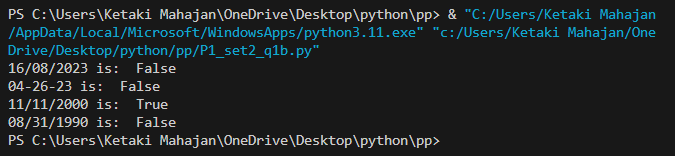
        return False

print("16/08/2023 is: ", is\_valid('16/08/2023')) # T

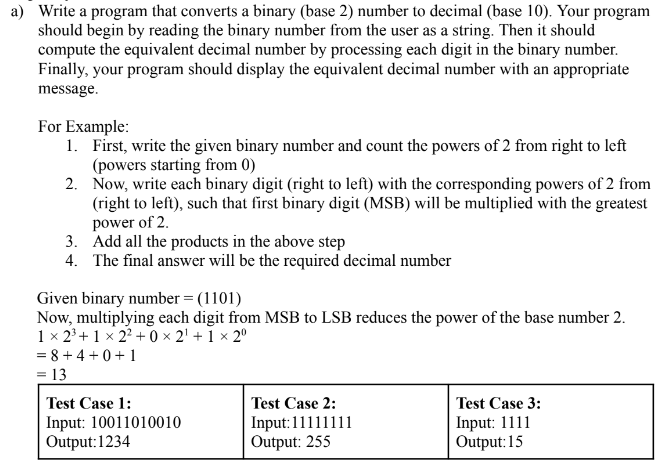
print("04-26-23 is: ", is\_valid('04-26-23')) # F

print("11/11/2000 is: ", is\_valid('11/11/2000')) # T

print("08/31/1990 is: ", is\_valid('08/31/1990')) # T

****

**10/05/2023:** P1 Set 2 (Q.1 (a))



bin\_num = input("enter binary number: ")

dec\_num = 0 # initializes deciaml num

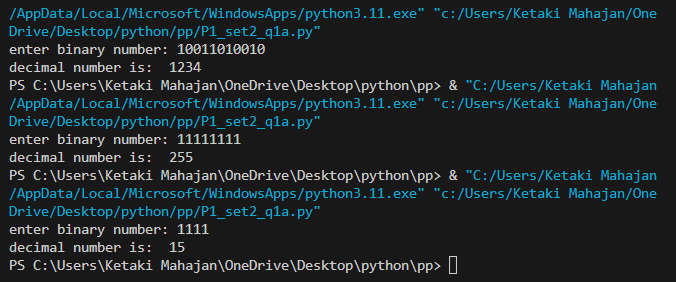
power = len(bin\_num) - 1 # highest power of binary

for num in bin\_num: # loops through each num in binary

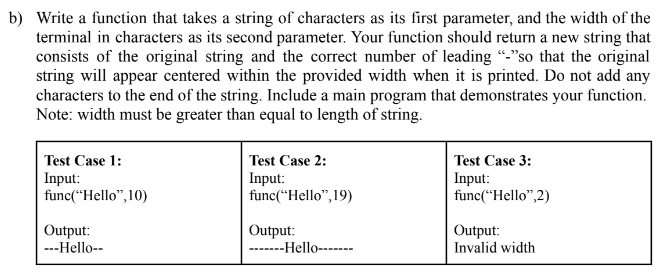
    dec\_num = dec\_num + (int(num) \* (2 \*\* power)) # adds powers to decimal

    power = power - 1 # decrements power

print("decimal number is: ", dec\_num)

****

**09/05/2023:** P1 Set 1 (Q.2 (b))

****

def center(input\_string, width):

    if len(input\_string) >= width:

        return input\_string

    dashes = width - len(input\_string) # 10-5=5

    leftDashes = dashes // 2 # 5//2=2

    rightDashes = dashes - leftDashes # 5-2=3

    result = ('-' \* rightDashes) + input\_string + ('-' \* leftDashes) # concatenating strings

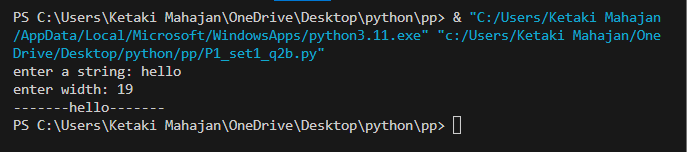
    return result

input\_string = input("enter a string: ") # hello

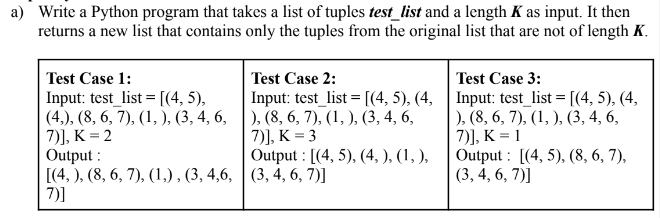
width = int(input("enter width: ")) # 10

result = center(input\_string, width)

print(result)

****

**08/05/2023:** P1 Set 1 (Q.2 (a))

****

tuplist = [(4, 5), (4,), (8, 6, 7), (1,), (3, 4, 6, 7)] #initializing list

k = int(input("enter value of k: ")) # k=2

new\_tuplist = []

# filtering tuples not of length k

for tup in tuplist: # iterating through each tuple e.g (4,5), (4,), (8,6,7)...etc

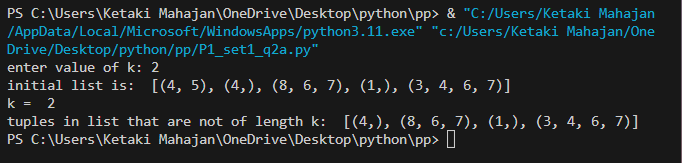
    if len(tup) != k: # checking if tuple is not equal to k and adding into empty list

        new\_tuplist.append(tup)

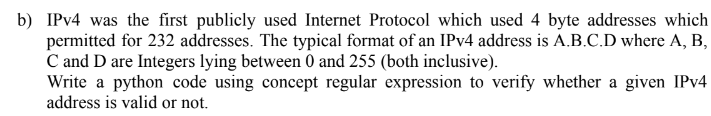
print("initial list is: ", str(tuplist))

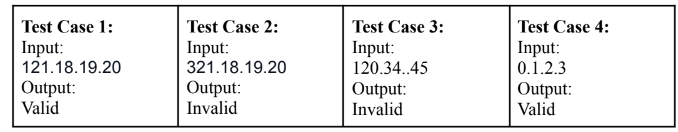
print("k = ", k)

print("tuples in list that are not of length k: " , str(new\_tuplist))

****

**07/05/2023:** P1 Set 1 (Q.1 (b))

****



import re

def is\_valid(ip\_address):

    pattern = r'^([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5]\.){3}[0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5]$'

    # num from 0-9, 10-99, 100-199, 200-249, 250-255) and . repeats 3 times, same pattern without . for last num

    if re.match(pattern, ip\_address): # using re.match to match input with pattern

        return True

    else:

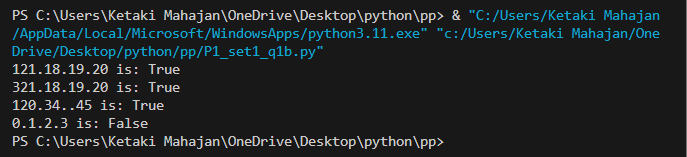
        return False

print("121.18.19.20 is:", is\_valid('121.18.19.20')) # true

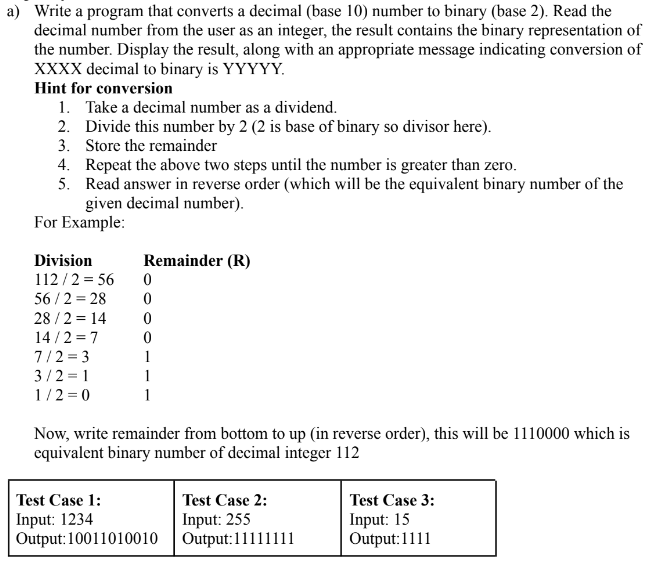
print("321.18.19.20 is:", is\_valid('321.18.19.20')) # false

print("120.34..45 is:", is\_valid('120.34..45')) # false

print("0.1.2.3 is:", is\_valid('0.1.2.3 is')) # true

****

**06/05/2023:** P1 Set 1 (Q.1 (a))

****

dec\_num = int(input("enter a decimal number: "))

# converting decimal to binary

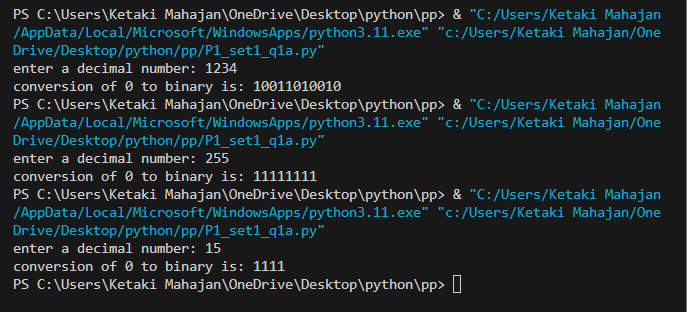
bin\_num = "" # initializing empty string to store the binary number

while dec\_num > 0:

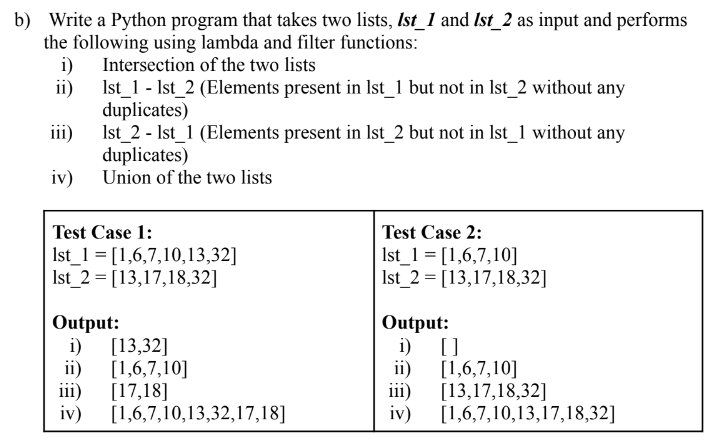
    bin\_num = str(dec\_num % 2) + bin\_num # get remainder 0 or 1 and add to empty string

    dec\_num = dec\_num // 2 # rounds to whole number

print(f"conversion of {dec\_num} to binary is: {bin\_num}")

****

**05/05/2023:** P2 Set 2 (Q.2 (b))

****

list1 = [1, 6, 7, 10, 13, 32]

list2 = [13, 17, 18, 32]

intersection = list(filter(lambda x: x in list1, list2)) # intersection of two lists

print("intersection of list 1 and list 2 is: ", intersection)

difference1\_2 = list(filter(lambda x: x not in list2, list1)) # elements in list1 but not in list2

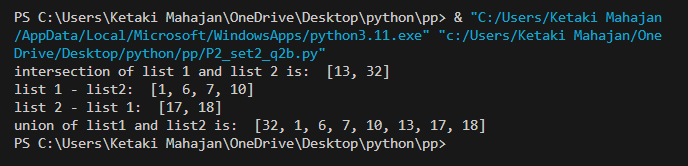
print("list 1 - list2: ", difference1\_2)

difference2\_1 = list(filter(lambda x: x not in list1, list2)) # elements in list2 but not in list1

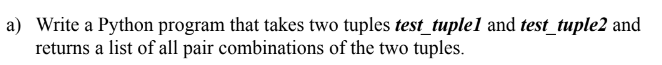
print("list 2 - list 1: ", difference2\_1)

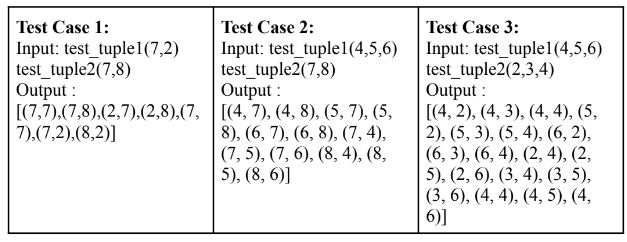
union = list(set(list1 + list2)) # union of the two lists

print("union of list1 and list2 is: ", union)

****

**04/05/2023:** P2 Set 2 (Q.2 (a))

****

****

print("ketaki mahajan / P1-2 / 16014022050")

def combinations(tuple1, tuple2):

    res = []

    for tup1 in tuple1:

        for tup2 in tuple2:

            res.append((tup1, tup2))

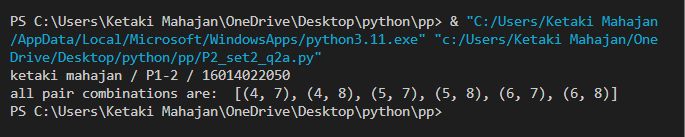
    return res

tuple1 = (4, 5, 6)

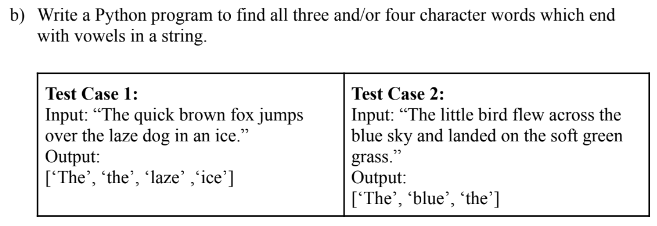
tuple2 = (7, 8)

res = combinations(tuple1, tuple2)

print("all pair combinations are: ", res)

****

**03/05/2023:** P2 Set 2 (Q.1 (b))

****

print("ketaki mahajan / P1-2 / 16014022050")

string = ("The quick brown fox jumps over the laze dog in an ice.")

new\_string = string.replace(".", "") # ignoring . as last characters of word becomes . and not considered as vowel

print(string)

words = new\_string.split() # split the string into a list of words

vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']

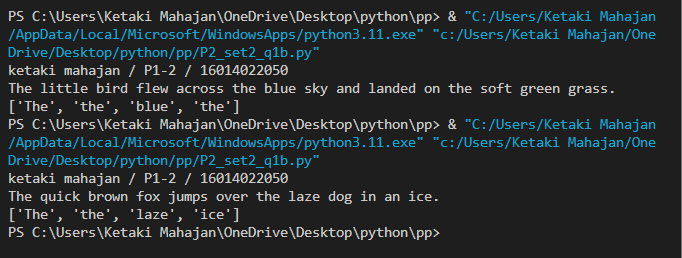
res = []

for word in words: # iterate each letter in the list

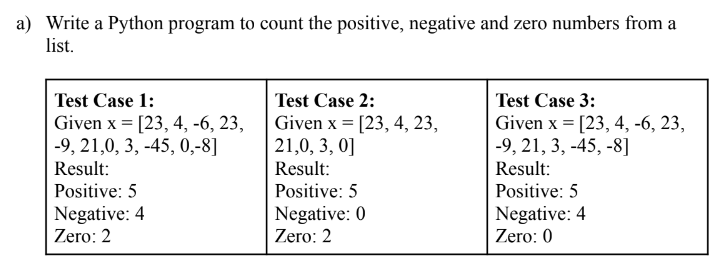
    if len(word) in [3, 4] and (word[-1] in vowels): # checks if word is 3/4 characters long and ends with a vowel

        res.append(word)

print(res)

****

**02/05/2023:** P2 Set 2 (Q.1 (a))

****

print("ketaki mahajan / P1-2 / 16014022050")

numbers = input("enter numbers: ").split() # input list (as string)

numbers = list(map(int, numbers)) # converts string list to int list

positive = 0

negative = 0

zero = 0

for num in numbers: # iterates through each number in list to check whether positive/negative/zero

    if num > 0:

        positive += 1

    elif num < 0:

        negative += 1

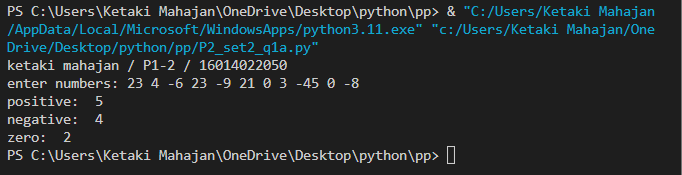
    else:

        zero += 1

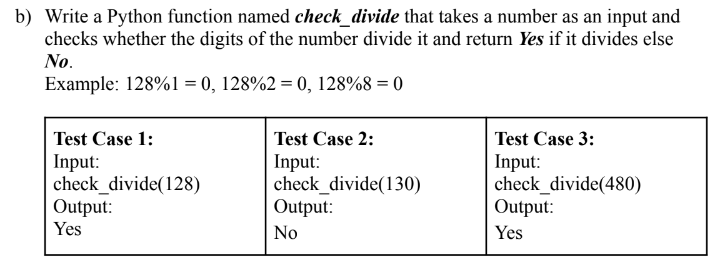
print("positive: ", positive)

print("negative: ", negative)

print("zero: ", zero)

****

**01/05/2023:** P2 Set 1 (Q.2 (b))



def check\_divide(number):

    # converts number to string to get individual digits

    number\_str = str(number)

    # loops through each digit of the number

    for digit in number\_str:

        if (int(digit) != 0) and (number % int(digit) == 0): # converts digit integer and checks if divisible by original number

            continue

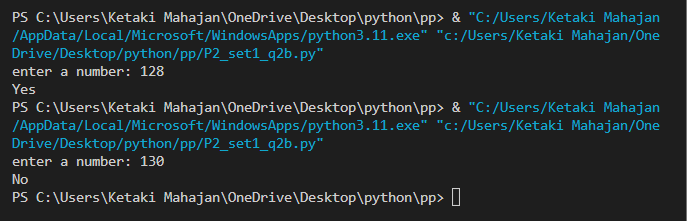
        else:

            return "No"

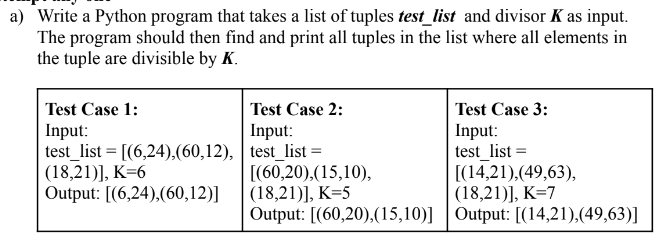
    return "Yes"

number = int(input("enter a number: "))

print(check\_divide(number))



**30/04/2023:** P2 Set 1 (Q.2 (a))



tuplist = [(6,24),(60,12), (18,21)] #initializing list

k = int(input("enter value of k: ")) # k=6

selected\_tuplist = []

# filtering tuples with values divisible by k

for tup in tuplist: # iterating through each tuple e.g (6,24), (60,12) etc..

    count = 0

    for element in tup: # iterating through each element in tuple e.g 6, 24 etc..

        if (element % k == 0):

            count += 1 # increment count if element is divisible by k

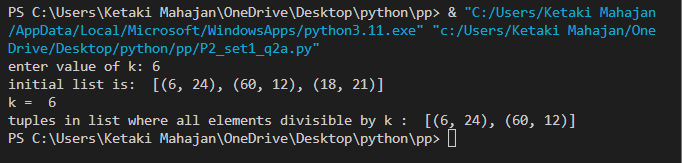
    if count == len(tup): # if count equal length of tuple, it means all elements divisible by k

        selected\_tuplist.append(tup) # appending divisible tuple to emply list

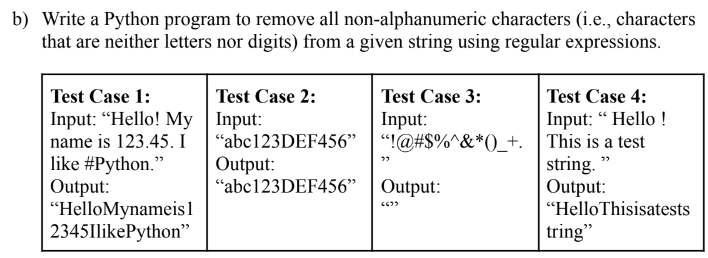
print("initial list is: ", str(tuplist))

print("k = ", k)

print("tuples in list where all elements divisible by k : " , str(selected\_tuplist)) # printing tuple list with all elements divisible by k



**29/04/2023:** P2 Set 1 (Q.1 (b))



import re

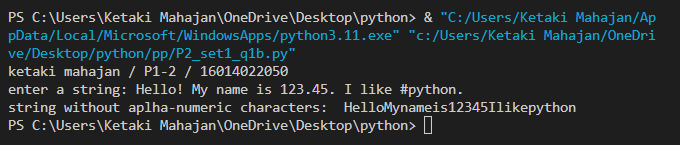
print("ketaki mahajan / P1-2 / 16014022050")

str = input("enter a string: ")

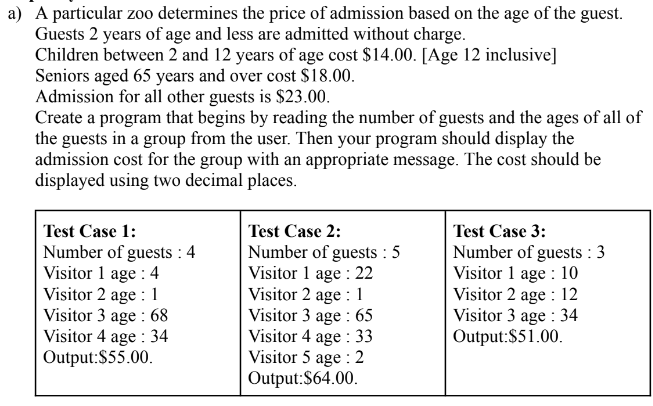
pattern = r'[^A-Za-z0-9]+' #matches characters that are not A-Z, a-z, 0-9

new\_str = re.sub(pattern, '', str) #searches for non-alphanumeric characters and replaces with empty string

print("string without aplha-numeric characters: ", new\_str) # prints new string



**28/04/2023:** P2 Set 1 (Q.1 (a))



guest = int(input("enter number of guests: "))

total\_cost = 0 #initializing total cost

for i in range(guest): #loops through each guest

    age = int(input("Visitor {} age: ".format(i+1)))

    if age <= 2: #finding cost based on age

        cost = 0

    elif age <= 12:

        cost = 14

    elif age >= 65:

        cost = 18

    else:

        cost = 23

    total\_cost += cost

print("The total admission cost is ${:.2f}.".format(total\_cost))

