**NAME: YOHENBA KSHETRIMAYUM**

**REG NO: RA1911003010904**

**WEEK: 3**

**Experiment Number:3**

**DATE: 18/2/2021**

**Aim: To solve allotted week 3(SET 14) python exercises**

1. Write a Python class to find a pair of elements (indices of the two numbers) from a given

array whose sum equals a specific target number.

Input: numbers= [10,20,10,40,50,60,70], target=50

Output: 3, 4

class Find\_pair():

    def pair(self, nums, target):

        l = len(nums)

        for i in range(0,l):

            for j in range(i-1,i+1):

                if nums[i] + nums[j] == target:

*# print(nums[i],nums[j])*

                    print(j+1,i+1)

                    return

x = [10,20,10,40,50,60,70]

*# x= [15, 20 , 13, 24, 34, 10 , 40 , 45]*

target= 50

y = Find\_pair()

y.pair(x,target)



2. Write a Python class which has two methods get\_String and print\_String. get\_String

accept a string from the user and print\_String print the string in upper case.

class String\_print():

    def get\_String(self):

*self*.str = input()

    def print\_String(self):

        print(*self*.str.upper())

str1 = String\_print()

str1.get\_String()

str1.print\_String()



3. Write a Python class named Rectangle constructed by a length and width and a method

which will compute the area of a rectangle.

class Rectangle():

    def input(self):

*self*.length = int(input("length: "))

*self*.breath = int(input("breadth: "))

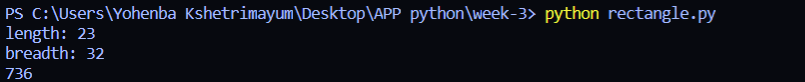
    def output(self):

        print(*self*.length\**self*.breath)

x = Rectangle()

x.input()

x.output()



4. Write a Python class to find validity of a string of parentheses, '(', ')', '{', '}', '[' and ']. These

brackets must be close in the correct order, for example "()" and "()[]{}" are valid but "[)",

"({[)]" and "{{{" are invalid.

class py\_solution:

   def is\_valid\_parenthese(self, str1):

        stack, pchar = [], {"(": ")", "{": "}", "[": "]"}

        for parenthese in str1:

            if parenthese in pchar:

                stack.append(parenthese)

            elif len(stack) == 0 or pchar[stack.pop()] != parenthese:

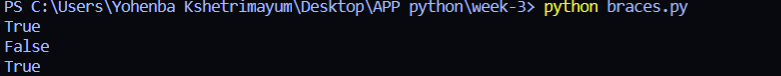
                return False

        return len(stack) == 0

print(py\_solution().is\_valid\_parenthese("()"))

print(py\_solution().is\_valid\_parenthese("()[{)}"))

print(py\_solution().is\_valid\_parenthese("{}[]"))



5. Write a Python class to get all possible unique subsets from a set of distinct integers.

Input : [4, 5, 6]

Output : [[], [6], [5], [5, 6], [4], [4, 6], [4, 5], [4, 5, 6]]

class py\_solution:

    def sub\_sets(self, sset):

        return *self*.subsetsRecur([], sorted(sset))

    def subsetsRecur(self, current, sset):

        if sset:

            return *self*.subsetsRecur(current, sset[1:]) + *self*.subsetsRecur(current + [sset[0]], sset[1:])

        return [current]

print(py\_solution().sub\_sets([4,5,6]))

