PROGRAMMING IN PYTHON LAB FILE

NAME- Anish Bansal

ROLL NO- R134218020

SAP ID- 500068395

CSE-CSF

B1

INTRODUCTION

1) **Say "Hello, World!" With Python**

   Here is a sample line of code that can be executed in Python:

   print("Hello, World!")

   You can just as easily store a string as a variable and then print it to stdout:

   my_string = "Hello, World!"
   print(my_string)

   The above code will print Hello, World! on your screen. Try it yourself in the editor below!

   Input Format

   You do not need to read any input in this challenge.

   Output Format

   Print Hello, World! to stdout.

   Sample Output 0

   Hello, World!

   CODE:

   print("Hello, World!")

2) Python If-Else

Task

Given an integer, , perform the following conditional actions:

If is odd, print Weird
If is even and in the inclusive range of to , print Not Weird
If is even and in the inclusive range of to , print Weird
If is even and greater than , print Not Weird
Input Format

A single line containing a positive integer, .

Constraints

0<=n<=100

Output Format

Print Weird if the number is weird. Otherwise, print Not Weird.

CODE:

```
import math
import os
import random
import re
import sys



if __name__ == '__main__':
    n = int(input().strip())
    if(n%2!=0):
        print("Weird")

    elif(n%2==0 and n>=2 and n<=5):
        print("Not Weird")
```

```
elif(n%2==0 and n>=6 and n<=20):
    print("Weird")

elif(n%2==0 and n>=20):
    print("Not Weird")
```

3) Arithmetic Operators

Task

The provided code stub reads two integers from STDIN, and . Add code to print three lines where:

The first line contains the sum of the two numbers.
The second line contains the difference of the two numbers (first - second).
The third line contains the product of the two numbers.
Example

Print the following:

```
8
-2
15
```

CODE:

```
if __name__ == '__main__':
    a = int(input())
    b = int(input())
    print(a+b)
    print(a-b)
    print(a*b)
```

4) Python: Division

Task

The provided code stub reads two integers, a and b , from STDIN.

Add logic to print two lines. The first line should contain the result of integer division, a//b . The second line should contain the result of float division, a / b.

No rounding or formatting is necessary.

Example
a=3
B=5

The result of the integer division . =0
The result of the float division is .=0.6
Print:

0
0.6

CODE:

```
if __name__ == '__main__':
    a = int(input())
    b = int(input())
    print(int(a/b))
    print(float(a/b))
```

Basic Data Types

1) **List Comprehensions**

Let's learn about list comprehensions! You are given three integers x, y and z representing the dimensions of a cuboid along with an integer n. Print a list of all possible coordinates given by (i, j, k) on a 3D grid where the sum of $i + j + k$ is not equal to n. Here, $0 \leq i \leq x; 0 \leq j \leq y; 0 \leq k \leq z$. Please use list comprehensions rather than multiple loops, as a learning exercise.

Example

$x = 1$

$y = 1$

$z = 2$

$n = 3$

All permutations of $[i, j, k]$ are:

$[[0, 0, 0], [0, 0, 1], [0, 0, 2], [0, 1, 0], [0, 1, 1], [0, 1, 2], [1, 0, 0], [1, 0, 1], [1, 0, 2], [1, 1, 0], [1, 1, 1], [1, 1, 2]]$.

Print an array of the elements that do not sum to $n = 3$.

$[[0, 0, 0], [0, 0, 1], [0, 0, 2], [0, 1, 0], [0, 1, 1], [1, 0, 0], [1, 0, 1], [1, 1, 0], [1, 1, 2]]$

CODE:

```
if __name__ == '__main__':
    x = int(input())
    y = int(input())
    z = int(input())
    n = int(input())
    lst1=[x for x in range(0,x+1)]
    lst2=[y for y in range(0,y+1)]
    lst3=[z for z in range(0,z+1)]
    lst4=[[a,b,c] for a in lst1 for b in lst2 for c in lst3 if a+b+c!=n]
    print(lst4)
```

2) Find the Runner-Up Score!

Given the participants' score sheet for your University Sports Day, you are required to find the runner-up score. You are given n scores. Store them in a list and find the score of the runner-up.

Input Format

The first line contains n. The second line contains an array $A[\,]$ of n integers each separated by a space.

Constraints

- $2 \leq n \leq 10$
- $-100 \leq A[i] \leq 100$

Output Format

Print the runner-up score.

Sample Input 0

```
5
2 3 6 6 5
```

CODE:

```
if __name__ == '__main__':
    ma2=-99999
    n = int(input())
    arr = list(map(int, input().split()))
    ma=max(arr)
    for i in range(0,len(arr)):
        if(arr[i]>ma2 and arr[i]!=ma):
            ma2=arr[i]
    print(ma2)
```

3) Nested Lists

Given the names and grades for each student in a class of N students, store them in a nested list and print the name(s) of any student(s) having the second lowest grade.

Note: If there are multiple students with the second lowest grade, order their names alphabetically and print each name on a new line.

Example

$records = [[\text{"chi"}, 20.0], [\text{"beta"}, 50.0], [\text{"alpha"}, 50.0]]$

The ordered list of scores is $[20.0, 50.0]$, so the second lowest score is 50.0. There are two students with that score: $[\text{"beta"}, \text{"alpha"}]$. Ordered alphabetically, the names are printed as:

```
alpha
beta
```

Input Format

The first line contains an integer, N, the number of students.

The $2N$ subsequent lines describe each student over 2 lines.

- The first line contains a student's name.

- The second line contains their grade.

CODE:

```
if __name__ == '__main__':
    lst=[]
    score1=[]
    least2=9999.9
    for _ in range(int(input())):
        name = input()
        score = float(input())
        lst2=[name,score]
        lst.append(lst2)
    for i in range(0,len(lst)):
        score1.append(lst[i][1])
    least=min(score1)
    for i in range(0,len(lst)):
        if(lst[i][1]<least2 and lst[i][1]!=least ):
            least2=lst[i][1]
```

```
lst.sort()
for i in range(0,len(lst)):
    if(lst[i][1]==least2):
        print(lst[i][0])
```

4) Finding the percentage

The provided code stub will read in a dictionary containing key/value pairs of name:[marks] for a list of students. Print the average of the marks array for the student name provided, showing 2 places after the decimal.

Example

marks key:value pairs are

'alpha': [20, 30, 40]

'beta': [30, 50, 70]

query_name = 'beta'

The **query_name** is 'beta'. beta's average score is $(30 + 50 + 70)/3 = 50.0$.

Input Format

The first line contains the integer n, the number of students' records. The next n lines contain the names and marks obtained by a student, each value separated by a space. The final line contains **query_name**, the name of a student to query.

Constraints

- $2 \leq n \leq 10$
- $0 \leq marks[i] \leq 100$
- length of marks arrays $= 3$

Output Format

Print one line: The average of the marks obtained by the particular student correct to 2 decimal places.

CODE:

```
if __name__ == '__main__':
    n = int(input())
    student_marks = {}
    for _ in range(n):
        name, *line = input().split()
        scores = list(map(float, line))
        student_marks[name] = scores
    query_name = input()
    a=student_marks.get(query_name,0)
```

```
sum1=0
for i in a:
    sum1=sum1+i
N=len(a)
avg=sum1/N
print("{0:1.2f}".format(avg))
```

Strings

1) sWAP cASE

You are given a string and your task is to swap cases. In other words, convert all lowercase letters to uppercase letters and vice versa.

For Example:

```
Www.HackerRank.com → wWW.hACKERrANK.COM
Pythonist 2 → pYTHONIST 2
```

Input Format

A single line containing a string S.

Constraints

$0 < len(S) \leq 1000$

Output Format

Print the modified string S.

CODE:

```
def swap_case(s):
    s2=s.swapcase()
    return s2

if __name__ == '__main__':
    s = input()
    result = swap_case(s)
    print(result)
```

2) String Split and Join

In Python, a string can be split on a delimiter.

Example:

```
>>> a = "this is a string"
>>> a = a.split(" ") # a is converted to a list of strings.
>>> print a
['this', 'is', 'a', 'string']
```

Joining a string is simple:

```
>>> a = "-".join(a)
>>> print a
this-is-a-string
```

Task

You are given a string. Split the string on a " " (space) delimiter and join using a - hyphen.

Input Format

The first line contains a string consisting of space separated words.

Output Format

Print the formatted string as explained above.

CODE:

```
def split_and_join(line):
    a=line.split(" ")
    a="-".join(a)
    return a
if __name__ == '__main__':
    line = input()
    result = split_and_join(line)
    print(result)
```

3) What's Your Name?

You are given the firstname and lastname of a person on two different lines. Your task is to read them and print the following:

> Hello `firstname` `lastname`! You just delved into python.

Input Format

The first line contains the first name, and the second line contains the last name.

Constraints

The length of the first and last name ≤ 10.

Output Format

Print the output as mentioned above.

CODE:

```
def print_full_name(a, b):
    print("Hello {} {}! You just delved into python.".format(a,b))

if __name__ == '__main__':
    first_name = input()
    last_name = input()
    print_full_name(first_name, last_name)
```

4) Find a string

In this challenge, the user enters a string and a substring. You have to print the number of times that the substring occurs in the given string. String traversal will take place from left to right, not from right to left.

NOTE: String letters are case-sensitive.

Input Format

The first line of input contains the original string. The next line contains the substring.

Constraints

$1 \leq len(string) \leq 200$

Each character in the string is an ascii character.

Output Format

Output the integer number indicating the total number of occurrences of the substring in the original string.

CODE:

```
def count_substring(string, sub_string):
    n=0
    for i in range(0,len(string)):
        if(string[i]==sub_string[0]):
         if(string[i:i+len(sub_string)]==sub_string):
            n=n+1

    return n

if __name__ == '__main__':
    string = input().strip()
    sub_string = input().strip()

    count = count_substring(string, sub_string)
    print(count)
```

Collections

1) **collections.Counter()**

Task

$Raghu$ is a shoe shop owner. His shop has X number of shoes.

He has a list containing the size of each shoe he has in his shop.

There are N number of customers who are willing to pay x_i amount of money only if they get the shoe of their desired size.

Your task is to compute how much money $Raghu$ earned.

Input Format

The first line contains X, the number of shoes.

The second line contains the space separated list of all the shoe sizes in the shop.

The third line contains N, the number of customers.

The next N lines contain the space separated values of the $shoe\ size$ desired by the customer and x_i, the price of the shoe.

Constraints

$0 < X < 10^3$

$0 < N \leq 10^3$

$20 < x_i < 100$

$2 < shoe\ size < 20$

Output Format

Print the amount of money earned by $Raghu$.

CODE:

```
from collections import Counter
X=int(input())
amount=0
lst=list(map(int,input().split(" ")))


N=int(input())
c1=Counter(lst)


for i in range(N):
   size,cost=map(int,input().split())
```

```
    if(c1[size]>0):
        c1[size]-=1
        amount+=cost


print(amount)
```

2) Collections.deque()

Task

Perform append, pop, popleft and appendleft methods on an empty deque d.

Input Format

The first line contains an integer N, the number of operations.

The next N lines contains the space separated names of methods and their values.

Constraints

$0 < N \leq 100$

Output Format

Print the space separated elements of deque d.

Sample Input

```
6
append 1
append 2
append 3
appendleft 4
pop
popleft
```

Sample Output

```
1 2
```

CODE:

```
from collections import deque
d=deque()
N=int(input())
for i in range(N):
    t=input()
    t2=t.split()

    if(t2[0]=="append"):
        t2[1]=int(t2[1])
        d.append(t2[1])
    elif(t2[0]=="appendleft"):
        t2[1]=int(t2[1])
        d.appendleft(t2[1])
    elif(t2[0]=="pop"):
        d.pop()
    elif(t2[0]=="popleft"):
        d.popleft()

for i in d:
    print(i,end=" ")
```

Errors and Exceptions

1) **Exceptions**

Task

You are given two values a and b.

Perform integer division and print a/b.

Input Format

The first line contains T, the number of test cases.

The next T lines each contain the space separated values of a and b.

Constraints

- $0 < T < 10$

Output Format

Print the value of a/b.

In the case of ZeroDivisionError or ValueError, print the error code.

CODE:

```
n=int(input())
for i in range(n):
    num1,num2=input().split()
    try:
        z=int(num1)
        x=int(num2)
        print(int(z/x))
    except ZeroDivisionError:
        print("Error Code: integer division or modulo by zero")
    except ValueError as e:
        print("Error Code:",e)
```

Built-Ins

1) **Zipped!**

Task

The National University conducts an examination of N students in X subjects.

Your task is to compute the average scores of each student.

$$Average\ score = \frac{Sum\ of\ scores\ obtained\ in\ all\ subjects\ by\ a\ student}{Total\ number\ of\ subjects}$$

The format for the general mark sheet is:

```
Student ID → ___1_____2_____3_____4_____5__
Subject 1   |  89    90    78    93    80
Subject 2   |  90    91    85    88    86
Subject 3   |  91    92    83    89    90.5
            |_____________________________
Average        90    91    82    90    85.5
```

Input Format

The first line contains N and X separated by a space.

The next X lines contains the space separated marks obtained by students in a particular subject.

Constraints

$0 < N \leq 100$

$0 < X \leq 100$

Output Format

Print the averages of all students on separate lines.

The averages must be correct up to 1 decimal place.

CODE:

```
N,X=map(int,input().split())
sub_marks=[]

for i in range(X):
    temp=list(map(float,input().split()))
    sub_marks.append(temp)

t=[]

for j in sub_marks:
    t=t+[j]

z=zip(*t)
```

```
for i in z:
    sum=0
    for j in i:
        sum=sum+j
    print("{:1.1f}".format(sum/X))
```

2) Python Evaluation

Task

You are given an expression in a line. Read that line as a string variable, such as var, and print the result using eval(var).

NOTE: Python2 users, please import `from __future__ import print_function`.

Constraint

Input string is less than 100 characters.

Sample Input

```
print(2 + 3)
```

Sample Output

```
5
```

CODE:

```
s=input()
eval(s)
```

3) Any or All

Task

You are given a space separated list of integers. If all the integers are positive, then you need to check if any integer is a palindromic integer.

Input Format

The first line contains an integer N. N is the total number of integers in the list.

The second line contains the space separated list of N integers.

Constraints

$0 < N < 100$

Output Format

Print `True` if all the conditions of the problem statement are satisfied. Otherwise, print `False`.

Sample Input

```
5
12 9 61 5 14
```

Sample Output

True

CODE:

```
n=int(input())
a=list(input().split())
b=[]
i2=True
for i in a:
   temp=i[-1::-1]
   b.append(temp)
   if(int(i)<0):
      i2=False
c=dict(zip(a,b))

X=[]
for i in c:
   t=i==c[i]
   X.append(t)
i1=any(X)
```

print(all([i2,i1]))

Python Functionals

1) Map and Lambda Function

Input Format

One line of input: an integer N.

Constraints

$0 \leq N \leq 15$

Output Format

A list on a single line containing the cubes of the first N fibonacci numbers.

Sample Input

```
5
```

Sample Output

```
[0, 1, 1, 8, 27]
```

CODE:

```
def fibonacci(n):
    if(n==0):
        return []
    if(n==1):
        return [0]
    if(n==2):
        return [0,1]


    fib=[0,1]
    for i in range(2,n):
```

```
        sum=fib[i-1]+fib[i-2]
        fib.append(sum)
    return fib

    # return a list of fibonacci numbers


if __name__ == '__main__':
    n = int(input())
    print(list(map(cube, fibonacci(n))))
```

2) Validating Email Addresses With a Filter

Input Format

The first line of input is the integer N, the number of email addresses.

N lines follow, each containing a string.

Constraints

Each line is a non-empty string.

Output Format

Output a list containing the valid email addresses in lexicographical order. If the list is empty, just output an empty list, [].

Sample Input

```
3
lara@hackerrank.com
brian-23@hackerrank.com
britts_54@hackerrank.com
```

Sample Output

```
['brian-23@hackerrank.com', 'britts_54@hackerrank.com', 'lara@hackerrank.com']
```

CODE:

```
def fun(s):
    try:
```

```
        s1,s2=s.split("@")
        s2,s3=s2.split(".")


    except Exception:
        return False

    if(len(s1)==0 or len(s2)==0 or len(s3)==0):
        return False
    for i in range(0,len(s1)):
        t=s1[i].isalnum()
        if(t or s1[i]=="-"or s1[i]=="_"):
            continue
        else:
            return False
    if(s2.isalnum()):
       b=2
    else:
        return False

    n=len(s3)
    if(n>3 or n==0):
        return False
    return True
    # return True if s is a valid email, else return False


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)

3) Reduce Function

Input Format

First line contains n, the number of rational numbers.

The i^{th} of next n lines contain two integers each, the numerator(N_i) and denominator(D_i) of the i^{th} rational number in the list.

Constraints

- $1 \leq n \leq 100$
- $1 \leq N_i, D_i \leq 10^9$

Output Format

Print only one line containing the numerator and denominator of the product of the numbers in the list in its simplest form, i.e. numerator and denominator have no common divisor other than 1.

Sample Input 0

```
3
1 2
3 4
10 6
```

Sample Output 0

```
5 8
```

Explanation 0

Required product is $\frac{1}{2}\frac{3}{4}\frac{10}{6} = \frac{5}{8}$

CODE:

```
from fractions import Fraction
from functools import reduce

def product(fracs):
```

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

FILE HANDLING

1. **Assume a file city.txt with details of 5 cities in given format (cityname population(in lakhs) area(in sq KM)):**

Example:

Dehradun 5.78 308.20

Delhi 190 1484

……

Open file city.txt and read to:

a. **Display details of all cities [3]**
b. **Display city names with population more than 10Lakhs [4]**
c. **Display sum of areas of all cities [3]**

CODE:

```
import os


str1="Dehradun 5.78 308.20"
str2="Delhi 190 1484"
```

```
str3="Chandigarh 10 114"
str4="Ahemdabad 80 464"
str5="Mumbai 204 603.4"
str1=str1+(20-len(str1))*" "
str2=str2+(20-len(str2))*" "
str3=str3+(20-len(str3))*" "
str4=str4+(20-len(str4))*" "
str5=str5+(20-len(str5))*" "

with open("city.txt","w+b") as f:
    f.write(str1.encode())
    f.write(str2.encode())
    f.write(str3.encode())
    f.write(str4.encode())
    f.write(str5.encode())



import os

totlen=os.path.getsize("city.txt")
size=20

lst=[]
f=open("city.txt","rb")
```

```
print("Details of all cities: ")
for i in range(5):
    r1=f.read(size)
    r1=r1.decode()
    lst.append(r1)
f.close()



for i in range(len(lst)):
    lst[i]=(lst[i].split())

for i in lst:
    print("City= {}\tPopulation= {} lakhs\tArea= {} sq KM".format(i[0],i[1],i[2]))

print("\n")
print("Cities with population more than 10 lakh:")
for i in lst:
    pop=float(i[1])
    if(pop>10):
        print(i[0])

sum=0.0

for i in lst:
    area=float(i[2])
```

```
    sum=area+sum


print("\n\nSum of area of all cities = {}".format(sum))
```

OUTPUT:

```
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: F:\python file exp\file1b.py ==========
Details of all cities:
City= Dehradun   Population= 5.78 lakhs   Area= 308.20 sq KM
City= Delhi      Population= 190 lakhs    Area= 1484 sq KM
City= Chandigarh         Population= 10 lakhs     Area= 114 sq KM
City= Ahemdabad Population= 80 lakhs     Area= 464 sq KM
City= Mumbai     Population= 204 lakhs    Area= 603.4 sq KM


Cities with population more than 10 lakh:
Delhi
Ahemdabad
Mumbai


Sum of area of all cities = 2973.6
>>>
```

2. **Assuming suitable data create a file "temp.txt" which stores the sales of 10 products quarterly in the given format where sales_first is no of sales in the first quarter. Productname sales_first sales_second sales_third sales_fourth**

e.g. TV 45 78 89 90

mobile 123 678 781 772

…….

Write python script to:

a) **Display all product details**

b) **Find average sale of all products**

c) **Find product with maximum sales**

CODE:

```
f=open("temp.txt","r")
```

```
avgs=[]
maxim=-1
maxstr=""
sum=0.0
for line in f:
   lst=line.split()
   print("Product: {}   sale in 1st quator: {}   sale in 2nd quator: {}   sale in 3rd quator: {}   sale in 4th quator: {}".format(lst[0],lst[1],lst[2],lst[3],lst[4]))
   sum=float(lst[1])+float(lst[2])+float(lst[3])+float(lst[4])
   if(sum>maxim):
      maxim=sum
      maxstr=line

   sum=sum/10
   avgs.append(sum)

for i in range(10):
   print("\n\nAverage of product {} = {}".format(i+1,avgs[i]))

lst=maxstr.split()
print("\n\n Product with maximum sale is {}".format(lst[0]))
```

OUTPUT:

```
========== RESTART: F:\python file exp\file2.py ==========
Product: TV   sale in 1st quator: 45   sale in 2nd quator: 78   sale in 3rd quator: 89   sale in 4th quator: 90
Product: mobile   sale in 1st quator: 123   sale in 2nd quator: 678   sale in 3rd quator: 781   sale in 4th quator: 772
Product: AC   sale in 1st quator: 200   sale in 2nd quator: 312   sale in 3rd quator: 323   sale in 4th quator: 234
Product: Fridge   sale in 1st quator: 389   sale in 2nd quator: 361   sale in 3rd quator: 260   sale in 4th quator: 310
Product: Laptop   sale in 1st quator: 223   sale in 2nd quator: 778   sale in 3rd quator: 881   sale in 4th quator: 882
Product: Heater   sale in 1st quator: 259   sale in 2nd quator: 341   sale in 3rd quator: 529   sale in 4th quator: 350
Product: Geyser   sale in 1st quator: 65   sale in 2nd quator: 42   sale in 3rd quator: 26   sale in 4th quator: 100
Product: Gaming_Console   sale in 1st quator: 165   sale in 2nd quator: 142   sale in 3rd quator: 26   sale in 4th quator: 100
Product: Cooler   sale in 1st quator: 472   sale in 2nd quator: 319   sale in 3rd quator: 854   sale in 4th quator: 190
Product: Lamp   sale in 1st quator: 37   sale in 2nd quator: 72   sale in 3rd quator: 28   sale in 4th quator: 46


Average of product 1 = 30.2


Average of product 2 = 235.4


Average of product 3 = 106.9


Average of product 4 = 132.0


Average of product 5 = 276.4


Average of product 6 = 147.9


Average of product 7 = 23.3


Average of product 8 = 43.3
```

```
Average of product 9 = 183.5


Average of product 10 = 18.3


 Product with maximum sale is Laptop
>>>
```

3. **Assume a file movie.txt with movie details, separated by spaces, in given format (movie_name director_first_name production_cost(in crores) Year_of_release):**

Example:

Lagaan Ashutosh 98 2001

Dangal Nitesh 110 2016

……

Open file movie.txt and write python script to:

d. Count number of movies in the file.

e. Add a new movie detail (War Amit 180 2019) at the end of file.

f. Display details of all movies where production cost is more than 80 Crores

g. Display first five movie details.

h. Display director name who has worked in more than two movies.

CODE:

```
count=0
f=open("movie.txt","r+")
pos=0
for line in f:
    count+=1

f.write("\nWar Amit 180 2019")
print("Number of movies= {}".format(count))
print("Movie added\n\n")

f.seek(0,0)

print("Movies with production cost more than 80 crores: ")

for line in f:
    temp=line.split()
    cost=float(temp[2])
    if(cost>80.0):
        print(temp[0])

f.seek(0,0)
```

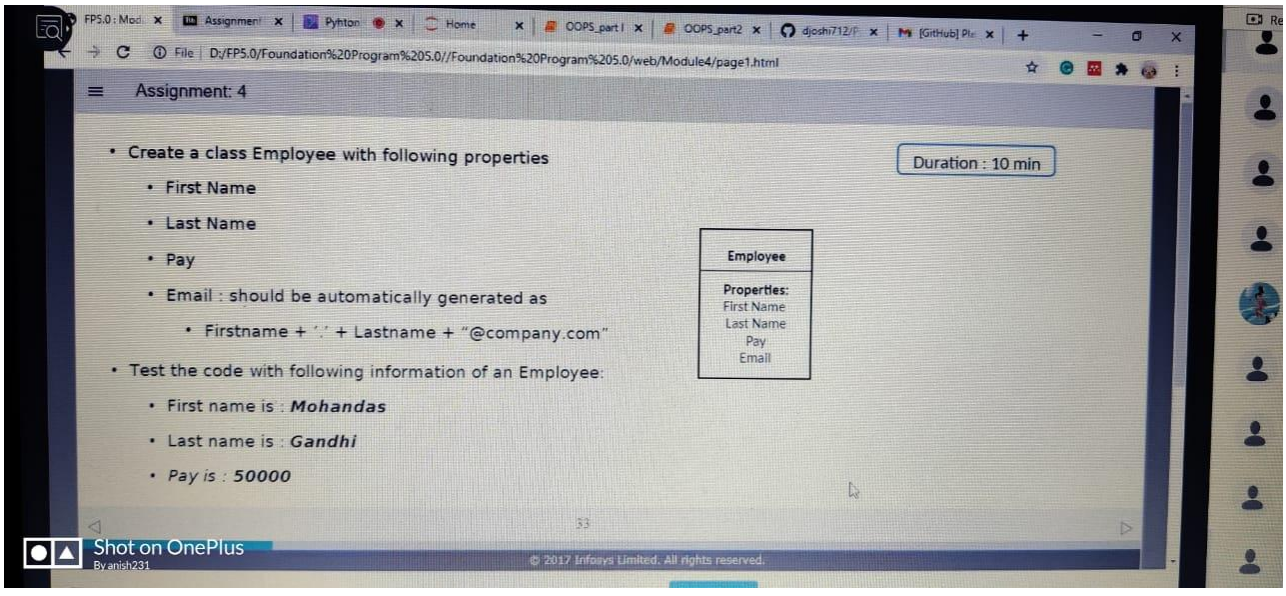
```
count=0
print("\n\n 1st 5 movies:")
for line in f:
    print(line)
    count+=1
    if(count==5):
        break

f.seek(0,0)

d={}
for line in f:
    temp=line.split()
    d[temp[1]]=d.get(temp[1],0)+1

print(" names of director worked in 2 or more movies: ")
for i,j in d.items():
    if(j>=2):
        print(i,end=" ")
```

OUTPUT:

```
>>>
====================== RESTART: F:\python file exp\file3.py ======================
Number of movies= 5
Movie added


Movies with production cost more than 80 crores:
Lagaan
Dangal
Fan
War


 1st 5 movies:
Lagaan Ashutosh 98 2001

Dangal Nitesh 110 2016

Swades Ashutosh 21 2004

Fan Maneesh 120 2016

Race Abbas 46 2008

 names of director worked in 2 or more movies:
Ashutosh
>>>
```

OBJECT ORIENTED PROGRAMMING

1)

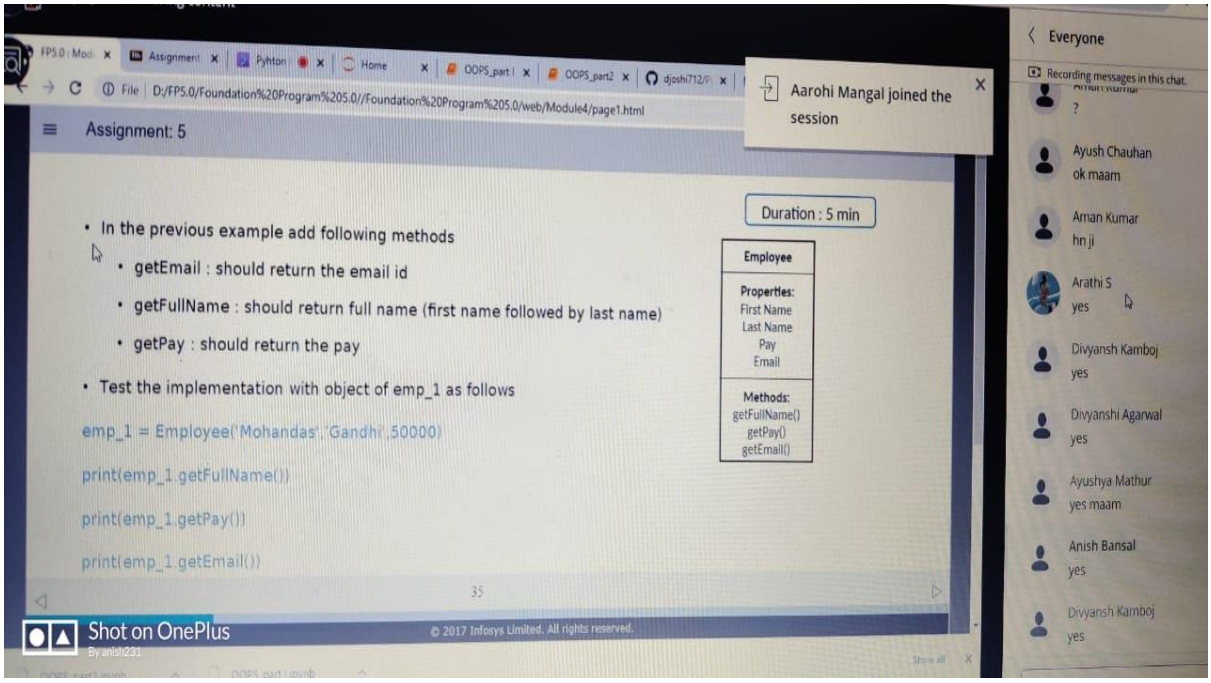


CODE:

```
class Employee:

    def __init__(self,fname,lname,pay):
```

```
        self.Firstname=fname
        self.Lastname=lname
        self.Pay=pay
        self.Email=fname+"."+lname+"@company.com"

e=Employee("Mohandas","Gandhi",50000)
```

2)



CODE:

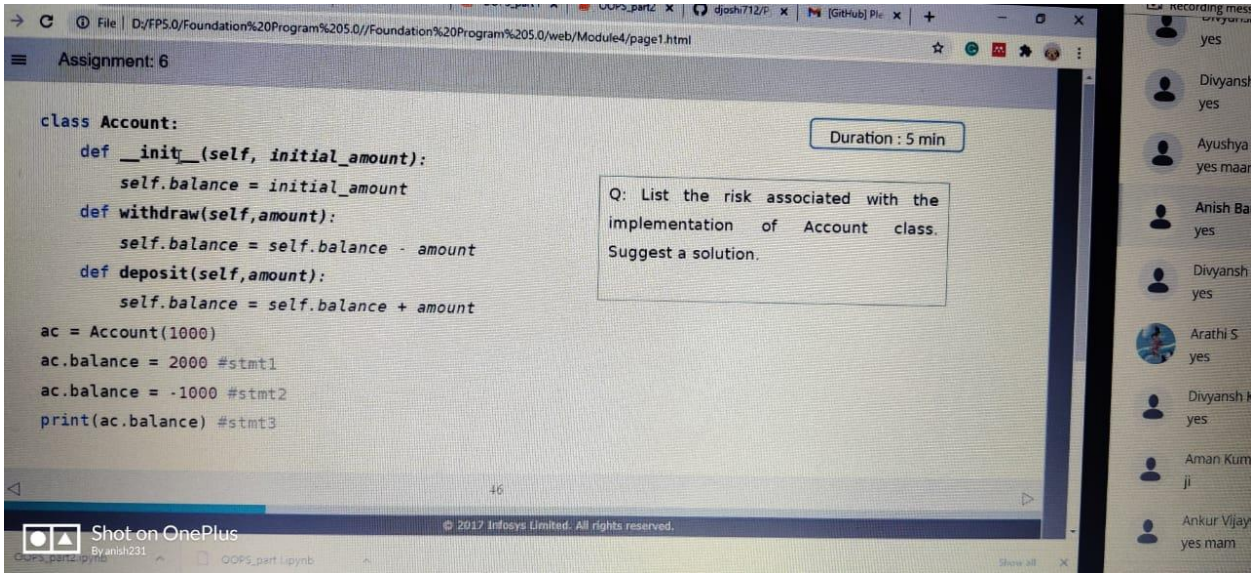
```
class Employee:
    def __init__(self,fname,lname,pay):
        self.Firstname=fname
        self.Lastname=lname
        self.Pay=pay
```

```
        self.Email=fname+"."+lname+"@company.com"

    def getEmail(self):
        return self.Email

    def getFullName(self):
        return self.Firstname+" "+ self.Lastname

    def getPay(self):
        return self.Pay



emp_1=Employee("Mohandas","Gandhi",50000)
print(emp_1.getFullName())
print(emp_1.getPay())
print(emp_1.getEmail())
```

OUTPUT:

```
>>>
====================== RESTART: F:\python oop exp\A2.py ======================
Mohandas Gandhi
50000
Mohandas.Gandhi@company.com
>>>
```

3).



The risk associated with the implementation of Account class is that value of variable “Balance” can be changed outside the Account class and amount of it can be changed without even depositing or withdrawing amount from the account. Value of variable Balance should only be initialized with some value at the time of the creation of the object and after that its values should only be varied by calling withdraw and deposit methods.

The solution for this is there should be abstraction and “Balance” variable should be declared private.

CODE:

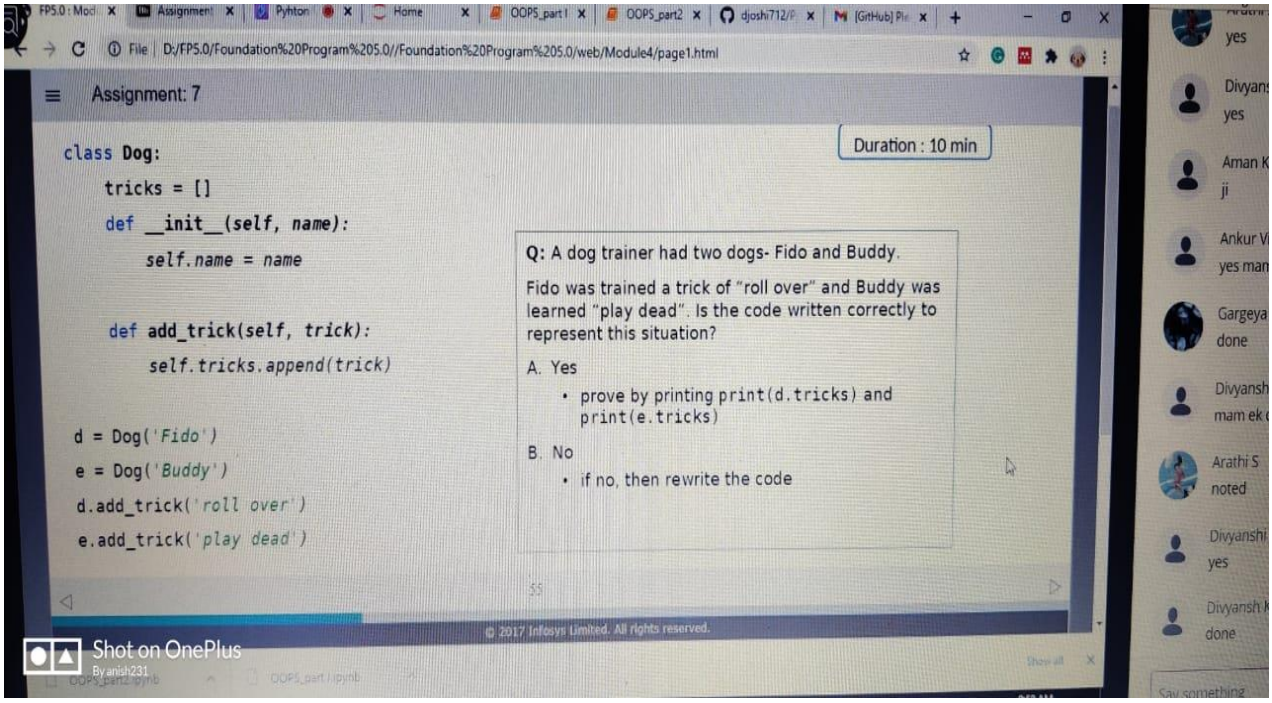
```
class Account:
    def __init__(self,initial_amount):
        self.__balance=initial_amount


    def withdraw(self,amount):
        self._Account__balance=self._Account__balance-amount


    def deposit(self,amount):
        self._Account__balance=self._Account__balance+amount
```

```
    def getbalanc(self):
        return self._Account__balance+0


ac=Account(1000)
ac.deposit(1000)
ac.withdraw(50)
amt=ac.getbalanc()
print("Balance= {}".format(amt))
```

OUTPUT:

```
>>>
========== RESTART: F:\python oop exp\A3.py ==========
Balance= 1950
>>>
```

4)



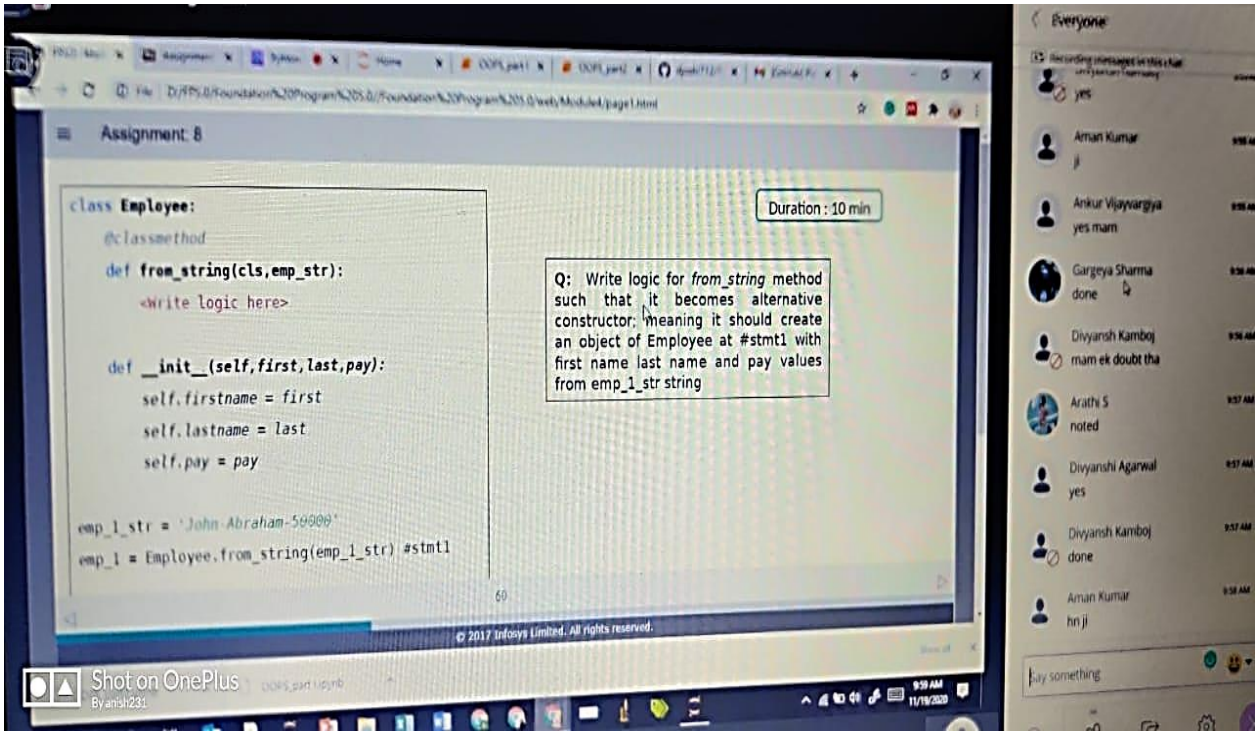
CODE:

```
class Dog:


    def __init__(self,name):
        self.name=name
        self.tricks=[]


    def add_trick(self,trick):
        self.tricks.append(trick)



d=Dog("Fido")
e=Dog("Buddy")
d.add_trick("roll over")
e.add_trick("play dead")
print(d.tricks)
print(e.tricks)
```

OUTPUT:

```
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: F:\python oop exp\A4.py ==========
['roll over']
['play dead']
>>>
```

5)



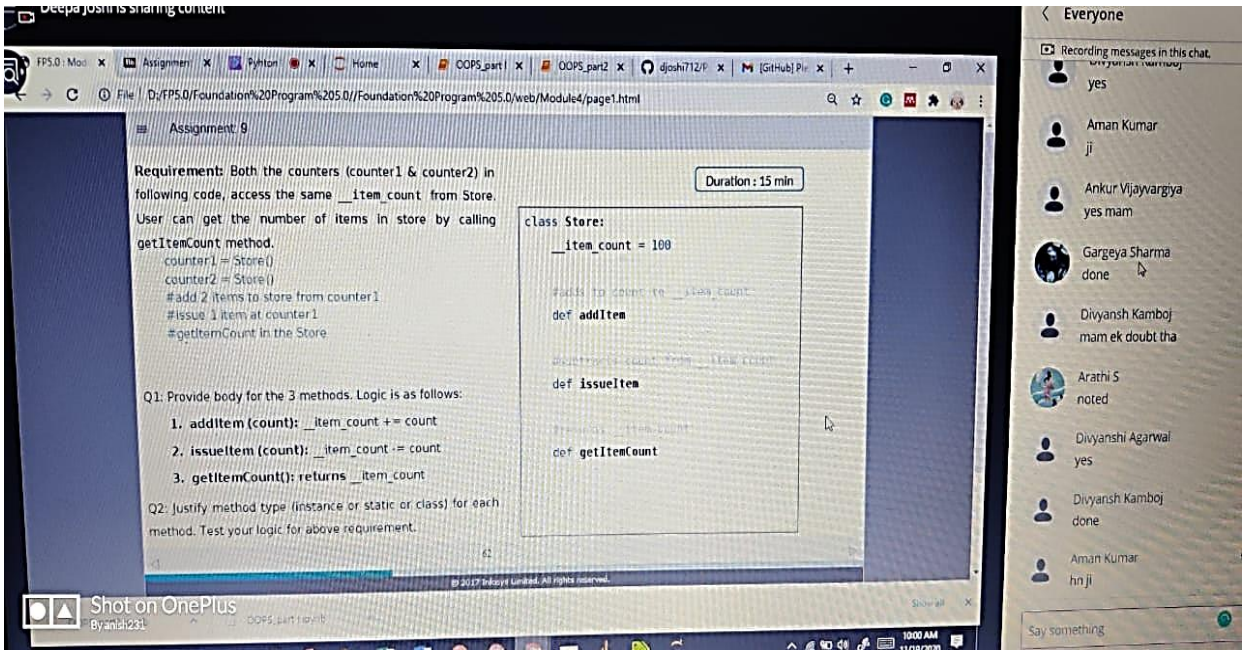
CODE:

```
class Employee:
    @classmethod
    def from_string(cls,emp_str):
        fname,lname,pay=emp_str.split("-")
        return cls(fname,lname,pay)

    def __init__(self,first,last,pay):
        self.firstname=first
        self.lastname=last
        self.pay=pay

emp_1_str="John-Abraham-50000"
emp_1 = Employee.from_string(emp_1_str) #stmt1
print(emp_1.firstname+emp_1.lastname+emp_1.pay)
```

OUTPUT:

```
>>>
========== RESTART: F:\python oop exp\A5.py ==========
JohnAbraham50000
>>>
```

6)



All methods are class methods

```
class Store:
    __item_count=100


    @classmethod
    def addItem(cls,count):
        cls._Store__item_count+=count


    @classmethod
    def issueItem(cls,count):
```

```
        cls._Store__item_count-=count


    @classmethod
    def getItemCount(cls):
        return cls._Store__item_count


counter1=Store()
counter2=Store()
counter1.addItem(2)
counter1.issueItem(1)
a=Store.getItemCount()
print(a)
```

OUTPUT:

```
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: F:\python oop exp\A6.py ==========
101
>>>
```