Python – First Problem Statement

Question

Write a python code to print the count of valid strings and invalid strings from a given list of strings.

A string is considered as valid if it contains combinations of alphabets (in upper case or lower case) with/without spaces.

Define a function to check if a given string is valid or not i.e if it contains combinations of alphabets (in upper case or lower case) with/without spaces.

This function will take a string as input and return True if the string is valid, otherwise will return False.

Example :

If the string given as argument is "Hello Good Morning", the function will return True.

If the string given as argument is "Purabi@gmail.com", the function will return False.

Using this function in main section, build the logic to print the count of valid strings and invalid strings from a given list of strings.

Refer to below instructions and sample input-output for more clarity on the requirement.

Instructions to write main section of the code:

a. You would require to write the main section completely, hence please follow the below instructions for the same.

b. You would require to write the main program which is inline to the "sample input description section" mentioned below and to read the data in the same sequence.

1. Create a list of strings. To create the list,

i. First read the number of elements/strings you want to store in the list.

ii. Read a string and add it to the list. This point repeats for the number of elements/strings to be stored in the list (considered in the first line of input i.e. in point #1.i).

2. Next print the message "Count Of Valid Strings=". Followed by this print the count of valid strings found in the list.

3. Next print the message "Count Of Invalid Strings=". Followed by this print the count of invalid strings found in the list.

You can use/refer the below given sample input and output to verify your solution.

Sample Input (below) description:

The first line of input taken in the main section contains an integer value representing the number of elements/strings to be added to the list.

The next lines of inputs are the strings to be added to the list one after another and is repeated for the number of elements/strings given in the first line of input. (each line represents one string to be added to the list).

Sample testcase :

Python – Second Problem Statement

Question

Write a code to automate the process of calculation of bonus amount for Employees in an Organization.

Define a class to create Employee objects with the below attributes:

employee_name of type String

designation of type String

salary of type Number

overTimeContribution of type dictionary having name of the month and overtime in hours (hours contributed as overtime for that month) as key:value pairs (month:overtime).

overTimeStatus of type String representing whether employee is eligible for overtime or not.

Define the required method in the Employee class which takes all parameters in the above sequence except the attribute overTimeStatus and sets the value of attributes to parameter values while creating an object of the class. For the attribute overTimeStatus a default value is set as False.

Define another class to create an Organization object with the below attributes:

employee_list of type List having Employee objects.

Define the required method in the Organization class which takes all parameters in the above sequence and sets the value of attributes to parameter values while creating an object of the class.

Define another method in the Organization class to check if the employee is eligible for overtime bonus or not and update the value for the attribute overTimeStatus.

This method takes the following as argument:

1. a number representing the overtime threshold.

This method will update the value for the attribute overTimeStatus to True if the following condition is fulfilled:

- the value for the total overtime hours for all months recorded for the Employee is not less than the overtime threshold (the value passed as argument).

- if above condition is not fulfilled, the value for the attribute overTimeStatus will remain as False.

Define another method in the Organization class to calculate the total bonus amount to be paid by the Organization for all the Employees in the Organization.

The bonus amount is calculated only for eligible Employees (having overTimeStatus as True) based on the total overtime hours for all months recorded for the Employee and the rate per hour provided.

This method will take as argument a number representing the rate per hour and will return the total amount the Organization will require to pay as bonus to all eligible Employees of the Organization.

Note:

All searches should be case insensitive.

Consider no two employees have the same name.

Instructions to write main section of the code:

a. You would require to write the main section completely, hence please follow the below instructions for the same.

b. You would require to write the main program which is inline to the "sample input description section" mentioned below and to read the data in the same sequence.

c. Create the respective objects (Employee and Organization) with the given sequence of arguments as per the requirement defined in the respective classes referring to the below instructions.

i. Create a list of Employee objects. To create the list,

1. Take a number as input representing the count of Employee objects to be created and added to the list.

2. Create an Employee object after reading the data related to it i.e. employeeName, designation, salary and overTimeContribution and add the object to the list of Employee objects which will be provided to the Organization object. This point repeats for the number of Employee objects (considered in the first line of input, point #c.i.1) to be created.

- To create the overTimeContribution dictionary for each Employee object, in the first step, take as input a number representing the count of months for which overtime hours to be recorded (i.e, the count of elements you want to add to the dictionary). Then create a (month:overtime) key:value pair after reading the data related to it and add the pair to the dictionary, this step is repeated for the number of element count read in the first step. Refer to sample input-outputs for more clarity.

ii. Create Organization object by passing the list of Employee objects (created in point# c.i) as the argument.

d. Take a number as input representing the overtime threshold to be passed as the argument to the method to check if the employee is eligible for overtime bonus or not.

e. Take a number as input representing the rate per hour to be passed as argument to the method to calculate the total bonus amount to be paid by the Organization for all the employees in the Organization .

f. Display the name and the updated overTimeStatus of all the Employees from the employeeList of the Organization object created in point #c.ii after checking the eligibility for overtime bonus. In the output, Name and the overTimeStatus values should be separated by a space. Refer to sample input-outputs for more clarity.

g. Display the calculated total bonus amount to be paid by the Organization for all the employees in the Organization.

You may refer to the sample outputs to have better clarity on the display formats.

You can use/refer the below given sample input and output to verify your solution.

Input Format for Custom Testing:

a. The first input taken in the main section is the number of Employee objects to be added to the list of employees.

b. The next set of inputs are the values related to attributes of Employee objects to be added to the employee List.

- the employee name, designation, salary and the count of elements to be added in the overTimeContribution dictionary of the Employee followed by the values for month and overtime hours for each element to be added to the overTimeContribution dictionary.

The above point repeats for the number of employee objects, read in point #a.

c. The next line of input (i.e. the second last line of input data) refers to the overtime threshold to be passed as the argument to the method to check if the employee is eligible for overtime bonus or not

d. The last line of input refers to the rate per hour to be passed as argument to the method to calculate the total bonus amount to be paid by the Organization for all the employees in the Organization.

Sample Test-case 1:

****

Input :

******

5

Sunita

Faculty

23000

2

jan

4

March

6

Arun

Admin

30000

3

Feb

4

March

12

June

10

Dipak

Admin

25000

3

Jan

12

July

5

Aug

3

Balen

HR

12000

3

Jan

12

July

5

Aug

3

Tarun

HR

78000

3

Jan

12

July

5

Aug

3

18

100

Output:

****

Sunita False

Arun True

Dipak True

Balen True

Tarun True

8600

Sample Test-case 2 :

****

Input :

****

5

Sunita

Faculty

23000

4

jan

4

March

6

apr

6

June

3

Arun

Admin

30000

3

jan

4

March

6

apr

6

Dipak

Admin

25000

3

jan

4

March

6

apr

6

Balen

HR

12000

3

jan

4

March

6

apr

6

Tarun

HR

78000

3

jan

4

March

6

apr

6

30

100

Output :

******

Sunita False

Arun False

Dipak False

Balen False

Tarun False

0

Solution

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
class Employee:
    def __init__(self,emp_name,desgi,salary,overTimeContribution,overTimeStatus="False"):
        self.emp_name = emp_name
        self.desgi = desgi
        self.salary = salary
        self.overTimeContribution = overTimeContribution
        self.overTimeStatus = overTimeStatus

class Organization(Employee):
    def __init__(self,employee_list):
        self.employee_list = employee_list

    def overtimestatus(self,overtimethreshold):
        for i in self.employee_list:
            if(int(sum(i.overTimeContribution.values())) >= int(overtimethreshold)):
                i.overTimeStatus = "True"
        return self.employee_list
```

```
    def totalamount(self,overtimehour):
        total = 0
        for i in self.employee_list:
            if(i.overTimeStatus == "True"):
                total += int(sum(i.overTimeContribution.values()))*overtimehour
        return total


a = int(input())
lst = []
for i in range(a):
    emp_name = input()
    desgi = input()
    salary = int(input())
    b = int(input())
    dct = {}
    for j in range(b):
        mon = input()
        bon = int(input())
        dct[mon.lower()] = bon
    obj = Employee(emp_name,desgi,salary,dct)
    lst.append(obj)
obj = Organization(lst)
overtimethreshold = int(input())
rateperhour = int(input())
lst = obj.overtimestatus(overtimethreshold)
```

```
for i in lst:
    print(i.emp_name,i.overTimeStatus)
total = obj.totalamount(rateperhour)
print(total)
```

Test cases

TestCase1:

****

Input:

*****

5

Sunita

Faculty

23000

2

jan

4

March

6

Arun

Admin

30000

3

Feb

4

March