Contents

Python - First Problem Statement

1.1 Question:

Write a python code to find numbers and alphabets from the given string and put it in one list.
The list will have two elements – the first element is a type of string which contains the alphabet and the second element is a type of integer. The input string will not have any special characters. The input string can have either the combination of Alphabets and Numbers or only Alphabets or only Numbers or it can be an empty string.

Define a function to build a logic which returns a list. This list will contains a string first followed by numbers (must be in an integer format).

Cases to be handled:
1. If the string contains alphabets and numbers both then the list will contain two elements. First will be a string and the second will be a number(s) [Alphabets, Numbers].
2. If the string contains only alphabets then the list will contain only one element in the list [Alphabets].
3. If the string contains only numbers then the list will contain only one element in the list [Numbers].
4. If the string is empty, then it will return an empty list [].

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

Instructions to write the main section of the code.

You would be required to write the main section completely, hence follow the below instructions
1. First read a string from the user.
2. Next, call a function str_num_list(string) with string as an argument.
3. str_num_list(string) function will separate the alphabet and numbers and return the list.
4. Print the returned list from the str_num_list(string)

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

Sample Input:
Hyper12345Lower100
Sample Output:
['HyperLower', 12345100]

1.2 Solution:

```
def str_num_list(s1):
    str_num = []
    alpha = ''.join([i for i in s1 if i.isalpha()])
    str_num.append(alpha) if len(alpha) != 0 else str_num
    num = ''.join([i for i in s1 if i.isdigit()])
    str_num.append(int(num)) if len(num) != 0 else str_num
    return str_num


inp_s = input()
ans = str_num_list(inp_s)
print(ans)
```

1.3 Test Cases:

Test Case 2:

Sample Input 2:
Sachin24Apr

Sample Input 2:
['SachinApr', 24]

Test Case 3:

Sample Input 3:
LionelMessi

Sample Input 3:
['LionelMessi']

Test Case 4:

Sample Input 4:
100210

Sample Input 4:
[100210]

Python - Second Problem Statement

2.1 Question:

Create a class **Scholar** with attributes:
ScholarId - Number (representing a unique id for each student)
ScholarName - String (represents the name of the student)
State - String (representing the state of the student)
Marks - List (storing the marks of 3 subjects, obtained marks from each subject is out of 100)

Define the __init__ method which takes parameters in the above sequence and sets the values for attributes.

Create another class **ScholarResult** with following attribute and methods
ScholarGrade - list of dictionaries (It will contain the record for each student which will have ScholarId, ScholarName, TotalMarks, Grade and State)

Define the __init__ method which sets the values for above mentioned attribute.

Define the methods in the **ScholarResult** class as below
1. **First method** will take the list of Scholar objects and Grade as an input parameters. Firstly, you need to calculate the percentage for each student (based on the marks obtained in 3 subjects).

To calculate the percentage use the following formula:
Percentage = (Subject1Marks + Subject2Marks + Subject3Marks) / 3)
Note: Percentage must be an integer value and rounds the value. That means, if Percentage >=40.0% or <40.5% then it will be considered as a 40% and,
if percentage >=40.5% or <=41.0% then it will be considered as a 41%.

After calculating the percentage, you need to assign the grades as per the following given conditions:

If student's percentage >=80, then assign Grade "A"
If student's percentage >= 60 and < 80 then assign Grade "B"
If student's percentage >= 50 and < 60 then assign Grade "C"
If student's percentage <50 then assign Grade "D"

After calculating the grades, make a dictionary which contains ScholarId, ScholarName, TotalMarks, Grade and State and append it to the **ScholarGrade (**list of dictionaries) which is an attribute of **ScholarResult** class.

Secondly, your method will return the list of dictionaries according to the Grade passed as parameter. This list of dictionaries will be sorted based on their TotalMarks in descending order.

For Example:
[{'ScholarId': 106, 'ScholarName': 'palak', 'TotalMarks': 282, 'Grade': 'A', 'State': 'gujarat'},
{'ScholarId': 104, 'ScholarName': 'ramesh', 'TotalMarks': 275, 'Grade': 'A', 'State': 'rajasthan'},
'ScholarId': 102, 'ScholarName': 'ram', ' TotalMarks': 270, 'Grade': 'A', 'State': 'delhi'}]
Assumption: No scholars would have same TotalMarks

Method should return **None**, in the following cases:
if search is unsuccessful
if no records are found, or
Grade is any other string apart from “A”, “B”, ”C” or “D”

2. **Second method** will calculate the pass:fail ratio for each state and will return the list of states with pass:fail ratio. This list will be sorted in ascending order based on state's name.
For example: We have 2 states i.e. delhi and rajasthan. Output would be delhi first and then rajasthan.
A student who has acquired either Grade “A”, “B” or “C” will be considered as a Pass and a student with Grade “D” will be considered as a Fail.

Formula to calculate the Passing and Failing Percentage:
PassingPercentage = (Number of Scholars Passed / Total Scholars) * 100)
FailingPercentage = (Number of Scholars Failed / Total Scholars) * 100)
Note: Percentage must be an integer value and rounds the value. That means, if Percentage >=40.0% or <40.5% then it will be considered as a 40% and,
if percentage >=40.5% or <=41.0% then it will be considered as a 41%.

For Example-
Your returning list must follow the format as per the below sample example:
[['delhi', '60:40'], ['gujarat', '100:0'],['rajasthan', '70:30']]

If no records are found then method should return **None**.

Instructions to write main section of the code:

1. You would require to write the main section completely, hence please follow the below instructions for the same.
2. You would be required 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.
3. Create the respective objects(**Scholar** and **ScholarResult**) with the given sequence of arguments to fulfill the __init__ method requirement defined in the respective classes referring to the below instructions.
    i. Create a list of Scholar Objects.

To create the list of objects:

    a. Read the count of Scholar objects you want to create.
    b. Create a **Scholar** object after reading the data related to it and add the object to the list of **Scholar** objects which will be provided to the **ScholarResult** object. This point repeats for the number of Scholar objects (considered in the first line of input, point #3.i.a) .

4. Read the Grade as an input from the user.
5. Call the **first method** which has a list of Scholar's objects and Grade (either "A", "B", "C" or "D") as the arguments and return the list of dictionaries as required.

Format of the output would be:

"ScholarId ScholarName TotalMarks Grade State" (excluding the quotes).

If the first method is returning **None**, then display "No Record Found" (excluding the quotes).

For more clarity please refer the sample testcase.

6. Call the **second method** which will return a list of states with pass:fail ratio.

Format of the output would be:

"State PassingPercentage:FailingPercentage" (excluding the quotes).

If the second method is returning **None**, then display "No Record Found" (excluding the quotes).

For more clarity please refer the sample testcase.

Note: *All the inputs and searches will be case insensitive. Display State and ScholarName in the lowercase.*

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

Input Format:

1. First input is the integer that reads values for n Scholar objects.
2. The next set of inputs are ScholarId, ScholarName, State, Subject1Marks, Subject2Marks and Subject3Marks.
3. For each Scholar object repeat point#2 and this point is repeated for n number of Scholar objects given in the first line of input.
4. Last line of input represents a string which is a Grade. It can be either “A”, “B”, “C” or “D”.

You can consider below sample input and output to verify your implementation before submitting.

Sample Input1:

```
5
101
Tanmay
delhi
90
88
93
102
Sunil
delhi
90
95
90
103
Karvi
maharashtra
70
45
50
104
monika
tamilnadu
20
35
40
```

105
Ram
tamilnadu
90
65
50
a

Sample Output1:
102 sunil 275 A delhi
101 tanmay 271 A delhi
delhi 100:0
maharashtra 100:0
tamilnadu 50:50

2.2 Solution:

```
class Scholar:
    def __init__(self, scholar_id, scholar_name, state, marks):
        self.scholar_id = scholar_id
        self.scholar_name = scholar_name
        self.state = state
        self.marks = marks



class ScholarGrade:
    def __init__(self):
        self.scholar_grade = []

    def calculate_grade_return(self, scholar_list, grade):
        grade_list = ['A', 'B', 'C', 'D']
        key_list = ['scholar_id', 'scholar_name', 'total_marks', 'grade', 'state']
        if len(scholar_list) > 0:
            for i in scholar_list:
                assign_grade = ''
                total_marks = sum(i.marks)
                percentage = int(round(total_marks / 3, 0))
                if percentage >= 80:
                    assign_grade = grade_list[0]
                elif 60 <= percentage < 80:
                    assign_grade = grade_list[1]
```

```
            elif 50 <= percentage < 60:
                assign_grade = grade_list[2]
            else:
                assign_grade = grade_list[3]
            student_record_list = [i.scholar_id, i.scholar_name,
                                   total_marks, assign_grade, i.state]
            student_record = dict(zip(key_list, student_record_list))
            self.scholar_grade.append(student_record)
        find_scholar = [rec for rec in self.scholar_grade if rec['grade'] == grade]
        find_scholar = sorted(find_scholar, key=lambda x: (x['total_marks']), reverse=True)
        return find_scholar
    else:
        return None

def display_statewise_result(self):
    if len(self.scholar_grade) == 0:
        return None
    else:
        state_pass_fail_ratio = []
        states = list(set([each_record['state'] for each_record in self.scholar_grade]))
        states.sort()
        for item in states:
            count_scholars = 0
            count_fail = 0
            for each_record in self.scholar_grade:
                if each_record['state'] == item:
                    count_scholars += 1
                    if each_record['grade'] == 'D':
                        count_fail += 1
            fail_percentage = int(round((count_fail / count_scholars) * 100, 0))
            count_pass = count_scholars - count_fail
            pass_percentage = int(round((count_pass / count_scholars) * 100, 0))
            one_state_record = [item, f'{pass_percentage}:{fail_percentage}']
            state_pass_fail_ratio.append(one_state_record)
        return state_pass_fail_ratio


scholar_count = int(input())
scholar_list = []
for i in range(scholar_count):
    scholar_id = int(input())
    scholar_name = input().lower()
    state = input().lower()
```

```
    sub1_marks = int(input())
    sub2_marks = int(input())
    sub3_marks = int(input())
    marks = [sub1_marks, sub2_marks, sub3_marks]
    scholar_obj = Scholar(scholar_id, scholar_name, state, marks)
    scholar_list.append(scholar_obj)


grade = input().upper()
scholar_grade_obj = ScholarGrade()
student_list_with_grade = scholar_grade_obj.calculate_grade_return(scholar_list, grade)
if student_list_with_grade is None or len(student_list_with_grade) == 0:
    print("No Record Found")
else:
    for record in student_list_with_grade:
        for k, v in record.items():
            print(v, end=" ")
        print()

pass_fail_ratio_statewise = scholar_grade_obj.display_statewise_result()
if pass_fail_ratio_statewise is None:
    print("No Record Found")
else:
    for record in pass_fail_ratio_statewise:
        for item in record:
            print(item, end=" ")
        print()
```

2.3 Test cases:

Test Case 2:

Sample Input 2:
3
101
Timir
delhi
91
84
78
102
Satyam
delhi
29
31
38
103
Karan
maharashtra
73
65
50
X

Sample Output 2:
No Record Found
delhi 50:50
maharashtra 100:0

Test Case 3:

Sample Input 3:
4
101
Samyak
Gujarat
88
95
80

102
KAUSHA
GUJARAT
77
66
75
103
samir
Rajasthan
35
40
38
104
rajiv
rajaSTHan
70
55
80
b

Sample Output 3:
102 kausha 218 B gujarat
104 rajiv 205 B rajasthan
gujarat 100:0
rajasthan 50:50

Test Case 4:

Sample Input 4:
0
A

Sample Output 4:
No Record Found
No Record Found