

[Dashboard](#) / [My courses](#) / [PSPP/PUP](#) / [Experiments based on Dictionary and its operations.](#) / [Week8\\_Coding](#)

Started on	Sunday, 2 June 2024, 12:00 AM
State	Finished
Completed on	Wednesday, 5 June 2024, 8:41 PM
Time taken	3 days 20 hours
Marks	5.00/5.00
Grade	100.00 out of 100.00

## Question 1

Correct

Mark 1.00 out of 1.00

Create a student [dictionary](#) for n students with the student name as key and their test mark assignment mark and lab mark as values. Do the following computations and display the result.

1. Identify the student with the highest average score
2. Identify the student who has the highest Assignment marks
3. Identify the student with the Lowest lab marks
4. Identify the student with the lowest average score

Note:

If more than one student has the same score display all the student names

Sample input:

4

James 67 89 56

Lalith 89 45 45

Ram 89 89 89

Sita 70 70 70

Sample Output:

Ram

James Ram

Lalith

Lalith

**For example:**

Input	Result
4	Ram
James 67 89 56	James Ram
Lalith 89 45 45	Lalith
Ram 89 89 89	Lalith
Sita 70 70 70	

**Answer:** (penalty regime: 0 %)

```

1 n = int(input())
2 input_lines = [input() for i in range(n)]
3 lines_split = [input_lines[i].split(" ") for i in range(len(input_lines))]
4
5 student_names = []
6
7 for i in range(len(lines_split)):
8     student_names.append(lines_split[i][0])
9
10 test_mark = []
11 assignment_mark = []
12 lab_mark = []
13
14 for i in range(len(lines_split)):
15     test_mark.append(int(lines_split[i][1]))
16     assignment_mark.append(int(lines_split[i][2]))
17     lab_mark.append(int(lines_split[i][3]))
18 average = []
19 for i in range(len(lines_split)):
20     average.append((test_mark[i]+assignment_mark[i]+lab_mark[i])/3)

```

```

21
22 max_average = max(average)
23 max_assignment = max(assignment_mark)
24 min_lab_mark = min(lab_mark)
25 min_average = min(average)
26
27 index_max_average = []
28 index_max_assignment = []
29 index_min_lab_mark = []
30 index_min_average = []
31
32 for i in range(len(average)):
33     if average[i] == max_average:
34         index_max_average.append(i)
35     if average[i] == min_average:
36         index_min_average.append(i)
37
38 for i in range(len(assignment_mark)):
39     if assignment_mark[i] == max_assignment:
40         index_max_assignment.append(i)
41 for i in range(len(lab_mark)):
42     if lab_mark[i] == min_lab_mark:
43         index_min_lab_mark.append(i)
44 ans1 = []
45 ans2 = []
46 ans3 = []
47 ans4 = []
48
49 for i in index_max_average:
50     ans1.append(student_names[i])
51 for i in index_max_assignment:
52     ans2.append(student_names[i])

```

	Input	Expected	Got	
✓	4 James 67 89 56 Lalith 89 45 45 Ram 89 89 89 Sita 70 70 70	Ram James Ram Lalith Lalith	Ram James Ram Lalith Lalith	✓
✓	3 Raja 95 67 90 Aarav 89 90 90 Shadhana 95 95 91	Shadhana Shadhana Aarav Raja Raja	Shadhana Shadhana Aarav Raja Raja	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

Given an array of names of candidates in an election. A candidate name in the array represents a vote cast to the candidate. Print the name of candidates received Max vote. If there is tie, print a lexicographically smaller name.

**Examples:**

```
Input : votes[] = {"john", "johnny", "jackie",  
                  "johnny", "john", "jackie",  
                  "jamie", "jamie", "john",  
                  "johnny", "jamie", "johnny",  
                  "john"};
```

Output : John

We have four Candidates with name as 'John', 'Johnny', 'jamie', 'jackie'. The candidates John and Johnny get maximum votes. Since John is alphabetically smaller, we print it. Use [dictionary](#) to solve the above problem

**Sample Input:**

```
10  
John  
John  
Johnny  
Jamie  
Jamie  
Johnny  
Jack  
Johnny  
Johnny  
Jackie
```

**Sample Output:**

```
Johnny
```

**Answer:** (penalty regime: 0 %)

```
1 | try:  
2 |     n=int(input())  
3 |     votes_dict={}  
4 |     for _ in range(n):  
5 |         candidate = input()  
6 |         if candidate in votes_dict:  
7 |             votes_dict[candidate] += 1  
8 |         else:  
9 |             votes_dict[candidate] = 1  
10 |     max votes = max(votes dict.values())
```

```
11     winners = [candidate for candidate, votes in votes_dict.items() if votes == max_votes]
12     winner = min(winners)
13     print(winner)
14 except EOFError:
15     print("No input provided.")
```

	Input	Expected	Got	
✓	10 John John Johny Jamie Jamie Johny Jack Johny Johny Jackie	Johny	Johny	✓
✓	6 Ida Ida Ida Kiruba Kiruba Kiruba	Ida	Ida	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

A sentence is a string of single-space separated words where each word consists only of lowercase letters. A word is uncommon if it appears exactly once in one of the sentences, and does not appear in the other sentence.

Given two sentences s1 and s2, return a [list](#) of all the uncommon words. You may return the answer in any order.

Example 1:

Input: s1 = "this apple is sweet", s2 = "this apple is sour"

Output: ["sweet","sour"]

Example 2:

Input: s1 = "apple apple", s2 = "banana"

Output: ["banana"]

Constraints:

1 <= s1.length, s2.length <= 200

s1 and s2 consist of lowercase English letters and spaces.

s1 and s2 do not have leading or trailing spaces.

All the words in s1 and s2 are separated by a single space.

Note:

Use [dictionary](#) to solve the problem

For example:

Input	Result
this apple is sweet this apple is sour	sweet sour

Answer: (penalty regime: 0 %)

```
1 a = input().split()
2 b = input().split()
3 s = []
4
5 if a[0]!=b[0]:
6     for i in b:
7         print(i,end=" ")
8 else:
9     for i in a:
10        if i not in b:
11            s.append(i)
12    for i in b:
13        if i not in a:
14            s.append(i)
15    for i in s:
16        print(i,end=" ")
```

	Input	Expected	Got	
✓	this apple is sweet this apple is sour	sweet sour	sweet sour	✓
✓	apple apple banana	banana	banana	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Give a [dictionary](#) with value lists, sort the keys by summation of values in value [list](#).

**Input** : test\_dict = {'Gfg' : [6, 7, 4], 'best' : [7, 6, 5]}

**Output** : {'Gfg': 17, 'best': 18}

**Explanation** : Sorted by sum, and replaced.

**Input** : test\_dict = {'Gfg' : [8,8], 'best' : [5,5]}

**Output** : {'best': 10, 'Gfg': 16}

**Explanation** : Sorted by sum, and replaced.

Sample Input:

2

Gfg 6 7 4

Best 7 6 5

Sample Output

Gfg 17

Best 18

**For example:**

Input	Result
2 Gfg 6 7 4 Best 7 6 5	Gfg 17 Best 18

**Answer:** (penalty regime: 0 %)

```

1 try:
2     T = int(input())
3     result_dict = {}
4     for _ in range(T):
5         key, *values = input().split()
6         values = list(map(int, values))
7         sum_values = sum(values)
8         result_dict[key] = sum_values
9     sorted_result = dict(sorted(result_dict.items(), key=lambda item: item[1]))
10    for key, value in sorted_result.items():
11        print(key, value)
12 except EOFError:
13    print("No input provided,")

```

	Input	Expected	Got	
✓	2 Gfg 6 7 4 Best 7 6 5	Gfg 17 Best 18	Gfg 17 Best 18	✓



	Input	Expected	Got	
✓	2 Gfg 6 6 Best 5 5	Best 10 Gfg 12	Best 10 Gfg 12	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

## Question 5

Correct

Mark 1.00 out of 1.00

In the game of Scrabble™, each letter has points associated with it. The total score of a word is the sum of the scores of its letters. More common letters are worth fewer points while less common letters are worth more points. The points associated with each letter are shown below:

Points Letters

1 A, E, I, L, N, O, R, S, T and U

2 D and G

3 B, C, M and P

4 F, H, V, W and Y

5 K

8 J and X

10 Q and Z

Write a program that computes and displays the Scrabble™ score for a word. Create a [dictionary](#) that maps from letters to point values. Then use the [dictionary](#) to compute the score.

A Scrabble™ board includes some squares that multiply the value of a letter or the value of an entire word. We will ignore these squares in this exercise.

[Sample Input](#)

REC

[Sample Output](#)

REC is worth 5 points.

**For example:**

Input	Result
REC	REC is worth 5 points.

**Answer:** (penalty regime: 0 %)

```

1 letter_values = {
2     'A': 1, 'E': 1, 'I': 1, 'L': 1, 'N': 1, 'O': 1, 'R': 1, 'S': 1, 'T': 1, 'U': 1,
3     'D': 2, 'G': 2,
4     'B': 3, 'C': 3, 'M': 3, 'P': 3,
5     'F': 4, 'H': 4, 'V': 4, 'W': 4, 'Y': 4,
6     'K': 5,
7     'J': 8, 'X': 8,
8     'Q': 10, 'Z': 10
9 }
10 word = input()
11 score = sum(letter_values.get(letter.upper(), 0) for letter in word)
12 print(f"{word} is worth {score} points.")

```

	Input	Expected	Got	
✓	GOD	GOD is worth 5 points.	GOD is worth 5 points.	✓
✓	REC	REC is worth 5 points.	REC is worth 5 points.	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

[◀ Week8\\_MCQ](#)

Jump to...

[Functions ▶](#)