# <u>Dashboard</u> / <u>My courses</u> / <u>PSPP/PUP</u> / <u>Experiments based on Dictionary and its operations.</u> / <u>Week8 Coding</u>

Started on	Wednesday, 19 June 2024, 9:50 PM
State	Finished
Completed on	Wednesday, 19 June 2024, 10:09 PM
Time taken	19 mins
Marks	3.00/5.00
Grade	<b>60.00</b> out of 100.00

Give a <u>dictionary</u> with value lists, sort the keys by summation of values in value <u>list</u>.

**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

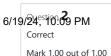
# Answer: (penalty regime: 0 %)

```
1 def sort_dict_by_sum(test_dict):
        # Calculate sums for each key
3
        sums = {key: sum(values) for key, values in test_dict.items()}
 4
 5
        # Sort keys based on their sums in descending order
 6
        sorted_keys = sorted(test_dict.keys(), key=lambda x: sums[x], reverse=True)
 7
8
        # Create the sorted dictionary
9
        sorted_dict = {key: sums[key] for key in sorted_keys}
10
11
        return sorted_dict
12
13
    test_dict1 = { 'best' [7, 6, 5],sep=''\'Gfg' [6, 7, 4],}
14
15
    result1 = sort_dict_by_sum(test_dict1)
    print(result1)
16
17
18
```

SyntaxError: unexpected character after line continuation character

Incorrect

Marks for this submission: 0.00/1.00.



In the game of Scrabble<sup>™</sup>, 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<sup>m</sup> score for a word. Create a <u>dictionary</u> that maps from letters to point values. Then use the <u>dictionary</u> 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.

```
1 v def scrabble_score(word):
 2
         # Define the points mapping according to Scrabble rules
 3
         points = {
             'A': 1, 'E': 1, 'I': 1, 'L': 1, 'N': 1, 'O': 1, 'R': 1, 'S': 1, 'T': 1, 'U': 1,
 4
 5
             'D': 2, 'G': 2,
             'B': 3, 'C': 3, 'M': 3, 'P': 3, 
'F': 4, 'H': 4, 'V': 4, 'W': 4, 'Y': 4,
6
 7
             'K': 5,
8
             'J': 8, 'X': 8,
9
10
             'Q': 10, 'Z': 10
11
12
13
         # Convert the word to uppercase to handle both lowercase and uppercase inputs
14
         word = word.upper()
15
16
         # Initialize total score
17
         total_score = 0
18
19
         # Calculate score for each letter in the word
20
         for letter in word:
21
             total_score += points.get(letter, 0) # Use get() to handle non-existent letters (returning 0)
22
23
         return total_score
24
25
    # Example usage:
    word = input()
26
27
    score = scrabble_score(word)
28
    print(word, 'is worth', score, 'points.')
29
```

6/19/24, 10:09 PM Week8\_Coding: Attempt review | REC-PS

	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.



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:**

Output: John

We have four Candidates with name as 'John', 'Johnny', 'jamie', 'jackie'. The candidates John and Johny get maximum votes. Since John is alphabetically smaller, we print it. Use <u>dictionary</u> to solve the above problem

# Sample Input:

10

John

John

Johny

Jamie Jamie

Johny

Jack

Johny

Johny

Jackie

# **Sample Output:**

Johny

```
1 ▼ def find_winner(votes):
2
        # Dictionary to store vote counts
3
        vote_counts = {}
4
        # Counting votes for each candidate
5
6
        for candidate in votes:
7 ,
            if candidate in vote_counts:
8
                vote_counts[candidate] += 1
9
            else:
10
                vote_counts[candidate] = 1
```

```
6/19/24, 10/209 PM # Finding the candidate with maximum votes Week8_Coding: Attempt review | REC-PS
                 winner = ""
        14
        15
        16
                 for candidate, count in vote_counts.items():
        17
                     if count > max_votes:
                         max_votes = count
        18
        19
                         winner = candidate
        20
                     elif count == max_votes:
        21
                         # Handle tie by choosing lexicographically smaller name
        22
                         if candidate < winner:</pre>
                             winner = candidate
        23
        24
        25
                 return winner
        26
        27
        28 v if __name__ == "__main__":
        29
                 # Reading input from user
                 n = int(input())
        30
        31
                 votes = []
                 for i in range(n):
        32 •
        33
                     vote = input().strip()
        34
                     votes.append(vote)
        35
                 # Finding the winner
        36
        37
                 winner = find_winner(votes)
        38
                 # Printing the winner
        39
        40
                 print( winner)
        41
```

	Input	Expected	Got	
~	10 John Johny Jamie Jamie Johny Jack Johny Johny Johny Jackie	Johny	Johny	~
<b>~</b>	6 Ida Ida Ida Kiruba Kiruba Kiruba	Ida	Ida	~

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

10



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

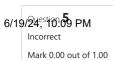
```
1 v def uncommon_words(s1, s2):
 2 ,
        def count_words(sentence):
 3
            word_count = {}
 4
            for word in sentence.split():
                if word in word_count:
 5 ,
 6
                     word_count[word] += 1
 7
                 else:
                     word_count[word] = 1
 8
 9
            return word_count
10
11
        count1 = count_words(s1)
        count2 = count_words(s2)
12
13
14
        uncommon_words = []
15
16
        # Check words in s1
        for word, count in count1.items():
17
18
            if count == 1 and word not in count2:
                uncommon_words.append(word)
19
20
        # Check words in s2
21
22
        for word, count in count2.items():
23
            if count == 1 and word not in count1:
24
                uncommon_words.append(word)
25
        return uncommon_words
26
27
28
    if __name__ == "__main__":
29
30
        s1 = input().strip()
31
        s2 = input().strip()
32
33
        result = uncommon_words(s1, s2)
34
        print(*result)
```

	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.



Create a student <u>dictionary</u> 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 as 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	

```
1 v def compute_statistics(students):
 2
        # Initialize variables to store results
 3
        highest_avg_score = -1
 4
        highest_avg_students = []
 5
        highest_assignment_score = -1
 6
        highest_assignment_students = []
 7
        lowest_lab_score = float('inf')
8
        lowest_lab_students = []
9
        lowest_avg_score = float('inf')
10
        lowest_avg_students = []
11
12
        # Calculate statistics
13
        for student, marks in students.items():
14
            test_mark, assignment_mark, lab_mark = marks
15
            # Calculate average score
16
            average_score = (test_mark + assignment_mark + lab_mark) / 3
17
18
            # Check for highest average score
19
            if average_score > highest_avg_score:
20
                highest_avg_score = average_score
```

```
21
                         highest_avg_students = [student]
                     elif average_score == highest_avg_scoreWeek8_Coding: Attempt review | REC-PS
6/19/24, 10209 PM
        23
                         highest_avg_students.append(student)
        24
        25
                     # Check for highest assignment mark
                     if assignment_mark > highest_assignment_score:
        26
        27
                         highest_assignment_score = assignment_mark
        28
                         highest_assignment_students = [student]
        29
                     elif assignment_mark == highest_assignment_score:
                         highest_assignment_students.append(student)
        30
        31
        32
                     # Check for lowest lab mark
                     if lab_mark < lowest_lab_score:</pre>
        33
        34
                         lowest_lab_score = lab_mark
        35
                         lowest_lab_students = [student]
        36
                     elif lab_mark == lowest_lab_score:
        37
                         lowest_lab_students.append(student)
        38
        39
                     # Check for lowest average score
                     if average_score < lowest_avg_score:</pre>
        40
        41
                         lowest_avg_score = average_score
        42
                         lowest_avg_students = [student]
        43
                     elif average_score == lowest_avg_score:
        44
                         lowest_avg_students.append(student)
        45
        46
                 # Prepare results
                 results = {
        47
        48
                     "Highest average score": highest_avg_students,
                     "Highest assignment marks": highest_assignment_students,
        49
        50
                     "Lowest lab marks": lowest_lab_students,
                     "Lowest average score": lowest_avg_students
        51
        52
```

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

Your code must pass all tests to earn any marks. Try again.

Incorrect

Marks for this submission: 0.00/1.00.

### ■ Week8\_MCQ

Jump to...

Functions ►

10