

[Dashboard](#) / [My courses](#) / [PSPP/PUP](#) / [Experiments based on Dictionary and its operations](#) / [Week8_Coding](#)

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	60.00 out of 100.00

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 17
Gfg 6 7 4	Best 18
Best 7 6 5	

Answer: (penalty regime: 0 %)

```

1 def sort_dict_by_sum(test_dict):
2     # 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
14 test_dict1 = { 'best' [7, 6, 5],sep='\ 'Gfg' [6, 7, 4],}
15 result1 = sort_dict_by_sum(test_dict1)
16 print(result1)
17
18

```

Syntax Error(s)

File "__tester__.python3", line 14
 test_dict1 = { 'best' [7, 6, 5],sep='\ 'Gfg' [6, 7, 4],}
 ^

SyntaxError: unexpected character after line continuation character

Incorrect

Marks for this submission: 0.00/1.00.

Question 2
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 def scrabble_score(word):
2     # Define the points mapping according to Scrabble rules
3     points = {
4         'A': 1, 'E': 1, 'I': 1, 'L': 1, 'N': 1, 'O': 1, 'R': 1, 'S': 1, 'T': 1, 'U': 1,
5         'D': 2, 'G': 2,
6         'B': 3, 'C': 3, 'M': 3, 'P': 3,
7         'F': 4, 'H': 4, 'V': 4, 'W': 4, 'Y': 4,
8         'K': 5,
9         'J': 8, 'X': 8,
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:
26 word = input()
27 score = scrabble_score(word)
28 print(word, 'is worth', score, 'points.')
29

```

	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.

Question 3
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 def find_winner(votes):  
2     # Dictionary to store vote counts  
3     vote_counts = {}  
4  
5     # Counting votes for each candidate  
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, 10:09 PM

Finding the candidate with maximum votes

Week8_Coding: Attempt review | REC-PS

```
11
12
13 max_votes = -1
14 winner = ""
15
16 for candidate, count in vote_counts.items():
17     if count > max_votes:
18         max_votes = count
19         winner = candidate
20     elif count == max_votes:
21         # Handle tie by choosing lexicographically smaller name
22         if candidate < winner:
23             winner = candidate
24
25     return winner
26
27
28 if __name__ == "__main__":
29     # Reading input from user
30     n = int(input())
31     votes = []
32     for i in range(n):
33         vote = input().strip()
34         votes.append(vote)
35
36     # Finding the winner
37     winner = find_winner(votes)
38
39     # Printing the winner
40     print( winner)
41
```

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

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

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

```

6/19/24, 10:09 PM

21

highest_avg_students = [student]

22

elif average_score == highest_avg_score:

23

highest_avg_students.append(student)

24

25

Check for highest assignment mark

26

if assignment_mark > highest_assignment_score:

27

highest_assignment_score = assignment_mark

28

highest_assignment_students = [student]

29

elif assignment_mark == highest_assignment_score:

30

highest_assignment_students.append(student)

31

32

Check for lowest lab mark

33

if lab_mark < lowest_lab_score:

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

40

if average_score < lowest_avg_score:

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

47

results = {

48

"Highest average score": highest_avg_students,

49

"Highest assignment marks": highest_assignment_students,

50

"Lowest lab marks": lowest_lab_students,

51

"Lowest average score": lowest_avg_students

52

}

Week8_Coding: Attempt review | REC-PS

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