

[Dashboard](#) / [My courses](#) / [CD19411-PPD-2022](#) / [WEEK 10-Dictionary](#) / [WEEK-10\\_CODING](#)

<b>Started on</b>	Friday, 17 May 2024, 8:31 PM
<b>State</b>	Finished
<b>Completed on</b>	Friday, 17 May 2024, 8:36 PM
<b>Time taken</b>	4 mins 43 secs
<b>Marks</b>	7.00/7.00
<b>Grade</b>	<b>50.00</b> out of 50.00 ( <b>100%</b> )
<b>Name</b>	<a href="#">NAVEEN RAJ B 2022-CSD-A</a>

## Question 1

Correct

Mark 1.00 out of 1.00

A teacher wants to evaluate her class results for the subject she handles. She want to do the following analysis:

1. Display Class average
2. Display Maximum mark Roll no
3. Display Minimum mark Roll no

Kindly help her out. Use dictionary for storing the student details.

Input Format:

In line 1 no of students will be given

Followed by n lines containing student rollno and marks

Output Format:

Line 1 Class average

Line 2 Maximum mark Roll no

Line 3 Minimum mark Roll no

Sample Input:

```
4
01 87
02 99
03 45
04 77
```

Output:

```
77
02
03
```

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

```
1 def evaluate_class_results(student_data):
2     # Calculate class average
3     total_marks = sum(student_data.values())
4     class_average = total_marks / len(student_data)
5
6     # Find maximum and minimum marks
7     max_mark = max(student_data.values())
8     max_roll_no = max(student_data, key=student_data.get)
9     min_mark = min(student_data.values())
10    min_roll_no = min(student_data, key=student_data.get)
11
12    return class_average, max_mark, max_roll_no, min_mark, min_roll_no
13
14 if __name__ == "__main__":
15     # Read the number of students
16     num_students = int(input())
17
18     # Read student roll numbers and marks into a dictionary
19     student_data = {}
20     for _ in range(num_students):
21         roll_no, mark = input().split()
```

22

	Input	Expected	Got	
✓	4	77	77	✓
	01 87	02	02	
	02 99	03	03	
	03 45			
	04 77			

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



## Question 2

Correct

Mark 1.00 out of 1.00

Two words are anagrams if they contain all of the same letters, but in a different order. For example, "evil" and "live" are anagrams because each contains one "e", one "i", one "l", and one "v". Create a program that reads two strings from the user, determines whether or not they are anagrams, and reports the result.

Sample Input 1

evil

live

Sample Output 1

Those strings are anagrams.

Sample Input 2

meet

met

Sample Output 2

Those strings are not anagrams.

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

```

1 def are_anagrams(word1, word2):
2     # Convert both words to lowercase
3     word1 = word1.lower()
4     word2 = word2.lower()
5
6     # Count occurrences of each character in both words
7     count_word1 = {}
8     count_word2 = {}
9
10    for char in word1:
11        count_word1[char] = count_word1.get(char, 0) + 1
12
13    for char in word2:
14        count_word2[char] = count_word2.get(char, 0) + 1
15
16    # Check if the counts of characters are the same
17    if count_word1 == count_word2:
18        return True
19    else:
20        return False
21
22 if __name__ == "__main__":

```

	Input	Expected	Got	
✓	evil live	Those strings are anagrams.	Those strings are anagrams.	✓
✓	meet met	Those strings are not anagrams.	Those strings are not anagrams.	✓
✓	rec cer	Those strings are anagrams.	Those strings are anagrams.	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

## Question 3

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.

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

```

1 def scrabble_score(word):
2     # Define the dictionary mapping letters to points
3     letter_points = {
4         'A': 1, 'B': 3, 'C': 3, 'D': 2, 'E': 1, 'F': 4, 'G': 2,
5         'H': 4, 'I': 1, 'J': 8, 'K': 5, 'L': 1, 'M': 3, 'N': 1,
6         'O': 1, 'P': 3, 'Q': 10, 'R': 1, 'S': 1, 'T': 1, 'U': 1,
7         'V': 4, 'W': 4, 'X': 8, 'Y': 4, 'Z': 10
8     }
9
10    # Calculate the score of the word
11    score = sum(letter_points.get(letter.upper(), 0) for letter in word)
12
13    return score
14
15 if __name__ == "__main__":
16     word = input()
17     score = scrabble_score(word)
18     print("{} is worth {} points.".format(word, score))
19

```

	Input	Expected	Got	
✓	REC	REC is worth 5 points.	REC is worth 5 points.	✓
✓	RAJALAKSHMI	RAJALAKSHMI is worth 27 points.	RAJALAKSHMI is worth 27 points.	✓

Passed all tests! ✓

**Correct**

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Create a program that determines and displays the number of unique characters in a string entered by the user. For example, Hello, World! has 10 unique characters while zzz has only one unique character. Use a dictionary or set to solve this problem.

For example:

Input	Result
Hello, World!	10

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

```

1 def count_unique_characters(string):
2     # Initialize an empty set to store unique characters
3     unique_characters = set()
4
5     # Iterate through each character in the string
6     for char in string:
7         # Add the character to the set
8         unique_characters.add(char)
9
10    # Return the number of unique characters
11    return len(unique_characters)
12
13 # Get input from the user
14 user_input = input()
15
16 # Calculate the number of unique characters
17 unique_count = count_unique_characters(user_input)
18
19 # Display the result
20 print(unique_count)
21

```

	Input	Expected	Got	
✓	Hello, World!	10	10	✓
✓	zzz	1	1	✓
✓	RECCSE	4	4	✓
✓	AAABBBCCC	3	3	✓

Passed all tests! ✓

**Correct**

Marks for this submission: 1.00/1.00.

## Question 5

Correct

Mark 1.00 out of 1.00

A sentence is a list of words that are separated by a single space with no leading or trailing spaces. Each word consists of lowercase and uppercase English letters.

A sentence can be shuffled by appending the 1-indexed word position to each word then rearranging the words in the sentence.

For example, the sentence "This is a sentence" can be shuffled as "sentence4 a3 is2 This1" or "is2 sentence4 This1 a3".

Given a shuffled sentence *s* containing no more than 9 words, reconstruct and return the original sentence.

Example 1:

**Input:**

is2 sentence4 This1 a3

**Output:**

This is a sentence

Explanation: Sort the words in *s* to their original positions "This1 is2 a3 sentence4", then remove the numbers.

Example 2:

**Input:**

Myself2 Me1 I4 and3

**Output:**

Me Myself and I

Explanation: Sort the words in *s* to their original positions "Me1 Myself2 and3 I4", then remove the numbers.

Constraints:

$2 \leq s.length \leq 200$

*s* consists of lowercase and uppercase English letters, spaces, and digits from 1 to 9.

The number of words in *s* is between 1 and 9.

The words in *s* are separated by a single space.

*s* contains no leading or trailing spaces.

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

```
1 def reconstruct_sentence(s):
2     # Split the sentence into words
3     words = s.split()
4
5     # Sort the words based on the appended numbers
6     words.sort(key=lambda x: int(x[-1]))
7
8     # Remove the numbers from each word
9     original_sentence = ' '.join(word[:-1] for word in words)
10
11     return original_sentence
12
13 # Test the function with example inputs
14 word1 = input()
15 print(reconstruct_sentence(word1)) # Output: "This is a sentence"
16 #print(reconstruct_sentence("Myself2 Me1 I4 and3")) # Output: "Me I
17
```





	Input	Expected	Got	
✓	is2 sentence4 This1 a3	This is a sentence	This is a sentence	✓
✓	Myself2 Me1 Vijay4 and3	Me Myself and Vijay	Me Myself and Vijay	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



Question **6**

Correct

Mark 1.00 out of 1.00

To Check if a Given Key Exists in a Dictionary or Not

Input: Any dictionary format input (Ex: d={'A':1,'B':2,'C':3})

Enter Key to check: A

Output:

Key is present and value of the key is: (location)

Present # True Statement

Not Present # False Statement

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

```
1 d = {'A':1, 'B':2, 'C':3}
2 a = input()
3 if a in d:
4     print("Present")
5 else:
6     print("Not Present")
```

	Input	Expected	Got	
✓	A	Present	Present	✓

Passed all tests! ✓

**Correct**

Marks for this submission: 1.00/1.00.

## Question 7

Correct

Mark 1.00 out of 1.00

## Multiply All the Items in a Dictionary

Input: Any input in Dictionary format (Ex: d={'A':10,'B':10,'C':239})

Output: multiplication of dictionary values (23900)

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

```

1 def multiply_dictionary_values(d):
2     result = 1
3     for value in d.values():
4         result *= value
5     return result
6
7 # Example input dictionary
8 d = {'A': 10, 'B': 10, 'C': 239}
9
10 # Calculate the multiplication of dictionary values
11 output = multiply_dictionary_values(d)
12
13 print(output)
14

```

	Input	Expected	Got	
✓	d={'A':10, 'B':10, 'C':239}	23900	23900	✓

Passed all tests! ✓

**Correct**

Marks for this submission: 1.00/1.00.

[◀ Week-10\\_MCQ](#)

Jump to...

[WEEK-10-Extra ▶](#)