Grade Count

Concepts Covered:

- Collections
- Functions

Problem Description:

The RCC Cricket Club wants to grade their players and count the players in each grade based on their average score for the entire year. Perform the following task to implement this scenario using a Python program.

Get the players' average score from the user as a colon-separated string.

Requirement 1: Define a function with the name **'create_list()'**

Requirement	**Methods**	**Responsibilities**
Create a list of scores from the input string.	create_list(input_string)	This method takes a string of values separated with a colon as an argument. The function should split the string based on the colon separator and generate it as a list. Convert each element into integer values and return this list of values to the caller method. Refer to the sample input and output for more clarifications.

Requirement2: Define a function with the name '**find_player_count()'**

Requirement	**Methods**	**Responsibilities**	
Find the players count in each grade.	find_player_count(score_list)	This method should take the **list of scores** as its argument. Iterate this list and find the grade based on the below-mentioned criteria:	
		Score	**Grade**
		score>50	A

		score>40 and score<=50 | B score>25 and score<=40 | C score<=25 | D Once identify the grade for each player, count the no. of players in each grade, and return this information as a dictionary in which set the **'Grade' as the key** and **count as the value.**

Process flow:

- In the **'main'** method, get the average scores separated with colon(':').
- Call the '**create_list**' method and pass the input string as its argument and capture the list returned by the method
- Then call the method **'find_player_count'** and pass the list of scores returned by the method 'create_list' as its parameter.
- Capture the dictionary returned by this method, and display it as specified in the sample input and output statements.

The main method is excluded from the evaluation. You are free to write your own code in the main method to invoke the business methods to check its correctness.

Note:

- In the sample input/output provided, the highlighted text in bold corresponds to the input given by the user, and the rest of the text represents the output.
- Get the average scores from the user as a single string separated by a colon: Example: 34:45:56:23:55:12:33
- The average scores should be of integer type.
- Do not alter the given code template. Write your code in the necessary places alone.
- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input 1:

Enter the average scores (as colon-separated values):**22:45:65:33:48:34:51:32**

Sample Output 1:

Grade count:

A : 2

B : 2

C : 3

D : 1

Sample Input 2:

Enter the average scores (as colon-separated values):**45:34:26:42:37:48:35**

Sample Output 2:

Grade count:

A : 0

B : 3

C : 4

D : 0

Archers' Scores

Concepts Coverage:

- Functions
- Collections

Problem Description:

A new archery academy in the city is in need of an automated system that should store the archers' scores and find the next round's selected archers and IDs. Develop a Python application towards this requirement.

Requirement: Create archers' score details, select them, and display them using functions.

Requirement 1: Define a function with the name '**select_archer**'

Requirement	**Method Name**	**Responsibilities**
Create the archers' score details list of those archers who are found eligible to play round 2.	select_archer(archers_score)	This method should take a list of archer Ids with scores (string) as its argument. Iterate through this list and split each string based on **','** (comma). split the id and scores alone. Convert the archers score into an integer. If the sum of their scores>=50 then appends that into a new list then creates a list of dictionaries that should contain archer id and scores. Not all 10 scores must get placed inside the list; **but only those scores(starting from index 0) that add up to make >=50.** That is, if the score is [9,6,6,8,8,8,8,8,9,9], then the new_list should be[9, 6, 6, 8, 8, 8, 8] The function should return this list of dictionary values of selected archers. **For example:** If the input is: ['ARC101,0,0,0,5,5,5,7,7,9,9', 'ARC102,9,6,6,8,8,8,8,8,9,9', 'ARC103,6,6,6,6,6,8,8,9,9,9'] then it should return a list of dictionaries as:

		[{'ARC102': [9, 6, 6, 8, 8, 8, 8]}, {'ARC103': [6, 6, 6, 6, 6, 8, 8, 9]}] If no one is selected for the next round then it should **return an empty list.**

Requirement 2: Define a function with the name '**extract_id**'

Requirement	**Method Name**	**Responsibilities**
Find the selected archers' Ids as a list.	extract_id(selected_list)	This method should take the (list of dictionaries) selected archer details as its argument. The method should return the selected archer's Id as a list. For example, if the list of dictionaries is: : [{'ARC102': [9, 6, 6, 8, 8, 8, 8]}, {'ARC103': [6, 6, 6, 6, 6, 8, 8, 9]}] then, it should return ['ARC102', 'ARC103'].

Process flow:

- In the '**main**' method, get the no of archers, and then the archer id and 10 scores as comma-separated values. Append that values to the archers_score list.
- Call the '**select_archer**' method by passing archers_score. Capture the list returned by this method.
- If the '**select_archer**' method returned the **empty list** then it should display the message: "**No one selected**".
- Else call the '**extract_id**' method and pass the selected list which is returned by the '**select_archer**' method as its argument and capture the list of selected archers IDs and display it as specified in the sample output.

Note:

- In the sample input/output provided, the highlighted text in bold corresponds to the input given by the user, and the rest of the text represents the output.

- The code for creating a menu for displaying various options to create and display details is provided along with the code template. You have to implement the functionalities alone.
- Strictly follow the naming conventions for variables and functions as specified in the problem description**.**

Sample Input 1:

Enter the no of archers:

3

Enter the archer id and 10 scores of round1 as comma-separated values:

ARC101,0,0,0,5,5,5,7,7,9,9

ARC102,9,6,6,8,8,8,8,8,9,9

ARC103,6,6,6,6,6,8,8,9,9,9

Sample Output 1:

Selected Archers Id's:

ARC102

ARC103

Sample Input 2:

Enter the no of archers:

2

Enter the archer id and 10 scores of round1 as comma-separated values:

ARC111,0,0,0,5,5,5,7,7,9,9

ARC122,9,0,0,0,8,8,8,8,3,4

Sample Output 2:

No one selected

Encrypted Message

Concepts Covered:

- Strings
- Collections
- Functions

Problem Description:

Encrypt a message entered by the user based on various criteria specified.

Criteria 1: Remove the duplicate words in the given sentence or paragraph without considering spaces or punctuation. Space is the delimiter for words.

Criteria 2: Once the duplicate words are removed then encrypt that message using the key and generate the caesar cipher text.

Write a Python program to simulate these criteria and generate the encrypted message.

Requirement 1: Define the method: **remove_duplicates()**

Requirement	**Methods**	**Responsibilities**
Remove the duplicate words in the given sentence.	remove_duplicates(sentence)	This method takes an argument which is a sentence and identifies the duplicated words in it and removes the same. This method should return a list of unique words in that string. **Example:** If the entered string is: "**It is okay not to be**

		okay", then the returned list should be: ['It','is','okay','not','to','be']

Requirement 2: Define the function: **caesar_cipher()**

Requirement	**Methods**	**Responsibilities**
Generate the cipher text	caesar_cipher(words_list, key)	This method takes two arguments, one is the list of unique words returned from the 'remove_duplicates' method and the other is the numerical key and generates the cipher text based on the key. This function should convert the list of unique words to a single string separated by space and then replace every alphabet in that string with another alphabet that is found after key places. Example: if the string is "**AaBb CcD**" and the key is **2** then the cipher text is "**CcDd EeF**". **Note:** • Make use of 'chr()' and 'ord()' methods. • Leave the spaces, punctuations, and numbers as it is.

Process flow:

- In the **'main'** method, the user has to get input as a sentence.
- The sentence is then passed to the '**remove_duplicate'** function and captures the list of unique words returned from the function.

- Then invoke the function '**caesar_cipher**' and pass the list of unique words to this function and capture the encrypted text returned by the function.

Note: Please do not include ',' (comma) in the input.

The main method is excluded from the evaluation. You are free to write your own code in the main method to invoke the business methods to check its correctness.

Note:

- In the sample input/output provided, the highlighted text in bold corresponds to the input given by the user, and the rest of the text represents the output.
- Do not alter the given code template. Write your code only in the necessary places alone
- Strictly follow the naming conventions for variables and functions as specified in the problem description.

Sample Input and Output 1:

Enter a sentence: **The Brown FOX**
The unique words in the sentence is/are**: ['The', 'Brown', 'FOX']**

Enter the key: **1**

The ciphertext is: Uif Cspxo GPY

Sample Input and Output 2:

Enter a sentence: **Hello world hello friend**
The unique words in the sentence is/are: ['Hello', 'world', 'hello', 'friend']

Enter the key: **2**
The ciphertext is: Jgnnq yqtnf jgnnq htkgpf

Sample Input and Output 3:

Enter a sentence: **Hello world Hello friend**
The unique words in the sentence is/are: ['Hello', 'world', 'friend']

Enter the key: **3**
The ciphertext is/are: Khoor zruog iulhqg

Premium Amount

Concepts Covered:

- Collections
- Functions

Problem Description:
A vehicle insurance agency needs to gather information about its customers. The information comprises the customer name, customer id, vehicle number, vehicle type, and premium amount. There are two types of vehicles they used to handle: **Two wheeler** and **Four wheeler**.

They also need to generate a report for the total premium amount collected for a vehicle type. Upon giving the vehicle type, it should display total premium amount paid by the vehicle number, the insurance premium amount, and the total premium amount collected for that particular vehicle type.

Write a Python program to carry out these tasks.

Requirement 1: Define the function with the name **'validate_customer_id()'**

Requirement	**Methods**	**Responsibilities**
Validate the customer id.	validate_customer_id (customer_list)	This function should take the customer list(list of tuples) as its argument, and validate each customer id in the list. Store the details of each customer with a valid customer id as a list of lists. Each list in the list of lists should represent each

		customer's details. Then the function should return this list of lists. **For example:** [[Jack,INS-123,TN43AD1778,Two wheeler,3000], [Jill,INS-125,TN43CR1976,Four wheeler,8000]] If no valid customer IDs are found then it has to return an empty list. **Validation rule:** Customer id should start with the letters: "INS" followed by "-" and followed by 3 integer numbers. Length should be 7.

Requirement 2: Define the function with the name **'calculate_total_amount()'**

Requirement	**Methods**	**Responsibilities**
Calculate the total amount of premium collected for the specified vehicle type.	calculate_total_amount (valid_customers_list, vehicle_type)	This function should take a list of valid customer details (list of lists) and vehicle type as its argument and should find the vehicle number and premium cost, and calculate the total amount of premium collected for the specified vehicle type. This function should return a list of dictionaries and the total amount of premium collected. (**For example:** [{'TN37AA1234': 3000,'TN37TR2897': 2000}, 5000]

		In the list of dictionaries, each dictionary should contain the vehicle number as the key and the premium amount as the value.

Process flow:

- In **the 'main'** method get the number of customers and customer details as user input(**comma separated)** and store it in a list(list of tuples).
- Call the '**validate_customer_id**' method and pass the customer_list and capture the list returned by the method.
- If the '**validate_customer_id**' returns an empty list then display the message as **"No valid customer details found"** and terminate the program**.**
- If the '**validate_customer_id**' returns a valid customer details list display it as specified in the sample input and output statements and then get the vehicle type from the user and call the '**calculate_total_amount**' method and pass the valid customer details list and vehicle type as arguments and capture the list of dictionary returned by the method and display as specified in the sample input and output statements
- If the list of dictionaries returned by the '**calculate_total_amount**' method is empty then display the message **"No vehicle found"**.

The main method is excluded from the evaluation. You are free to write your own code in the main method to invoke the business methods to check its correctness.

Note:

- In the sample input/output provided, the highlighted text in bold corresponds to the input given by the user, and the rest of the text represents the output.
- Do not alter the given code template. Write your code only in the necessary places alone
- Strictly follow the naming conventions for variables and functions as specified in the problem description.

Sample Input and Output 1:

No of customers: **5**

Jack,INS-123,TN43CD1778,Two wheeler,3000

Jill,IN-125,TN43AD1976,Four wheeler,8000

Amir,INS-156,TN37TR2897,Two wheeler,2000

Kevin,INS-155,TN35BR1001,Four wheeler,7000

Mathew,INS-198,TN37SC2424,Four wheeler,6000

Valid customers list:

['Jack', 'INS-123', 'TN43CD1778', 'Two wheeler', '3000']

['Amir', 'INS-156', 'TN37TR2897', 'Two wheeler', '2000']

['Kevin', 'INS-155', 'TN35BR1001', 'Four wheeler', '7000']

['Mathew', 'INS-198', 'TN37SC2424', 'Four wheeler', '6000']

Enter the vehicle type: **Two wheeler**

Vehicle No. Amount

TN43CD1778 3000

TN37TR2897 2000

Total amount 5000

Sample Input and Output 2:

No of customers: **4**

Jack,INS-123,TN43AS1778,Two wheeler,3000

Jill,IN-125,TN43CD1976,Four wheeler,8000

Amir,INS-156,TN37RR2897,Two wheeler,2000

Kevin,INS-155,TN35BD1001,Two wheeler,3500

Valid customers list:

['Jack', 'INS-123', 'TN43AS1778', 'Two wheeler', '3000']

['Amir', 'INS-156', 'TN37RR2897', 'Two wheeler', '2000']

['Kevin', 'INS-155', 'TN35BD1001', 'Two wheeler', '3500']

Enter the vehicle type: **Four wheeler**

No vehicle found

Car Details

Write a query to display the car name and the sum of the fare amount of all the cars that picked and returned in the same month and has the sum of total fare amount less than 20000.

Give the alias name for the sum of the fare amount as Total_Fare.

Sort the records based on the Total_Fare in ascending order.

Maximum rented Car Name

Write a query to display the car name and total number of days the car was rented, if and only if the car was rented for the maximum number of days in the month of 'May'.

Give an alias name as 'No_of_Days' for the number of days the car was rented.

Contact Information

Write a query to display the id, name, address, and fare amount of all the customers who use 'gmail' as their email. If the address is not available, then display the phone number. If the phone number is unavailable, display 'Not Available'.

Give an alias name as 'Contact_Info' for the address.

Sort the details based on the customer name in ascending order.

(Note: Data is case-sensitive.)

Passcode Generation

Write a query to display the customer id, name, address and pass_code for all the customers who have 'gmail' account.

Pass code is generated by concatenating the first 3 characters of the customer id and the first 3 characters of the email id and give alias name as 'pass_code'.

Sort the records based on the customer id in descending order.

(**Hint** : Data is case sensitive.)