Q1. Validating Alliteration

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Validating Alliteration(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB
Concepts Covered:

- Strings
- Functions

Problem Description:
Build a logic in python to check if the given sentence is alliterative or not.

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

Requirement	**Methods**	**Responsibilities**
Check whether the given sentence is alliterative or not	is_alliterative(sentence)	This method takes the sentence as the argument. • If the no. of words in the sentence is less than 2 words, return **False** to the caller method. • If there are more than 2 words in the sentence, and yet they begin with vowels (including upper case), return **False** to the caller method. • If there are more than 2 words in the sentence, and yet they all begin with different consonants (non-alliterative), return **False** to the caller method. • If there are more than 2 or equal to 2 words in the sentence, and if they all begin with the same consonant, **irrespective of the case**(i.e., alliterative), return **True** to the caller method. **Example 1:** If the sentence is: 'She sells sea shells', then the function should return True. **Example 2:** If the sentence is: 'Ann sells sea shells', then the function should return False.

Process flow:

- In the '**main**' method, get the sentence from the user.
- Call the '**is_alliterative**' and pass this sentence as an argument and capture the boolean value returned by the method.
- If the method '**is_alliterative**' returns True then display the message as "**The sentence is alliterative**" else display the message as "**The sentence is not alliterative**"

The main method is excluded from the evaluation. You are free to write your own code in the main method to invoke the business method and 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.
- Strictly follow the naming conventions for functions, as specified in the problem description

Sample Input 1:
Enter the sentence to be validated for alliteration: **She sells sea shells**

Sample Output 2:
The sentence is alliterative

Sample Input 2:
Enter the sentence to be validated for alliteration: **Ann sells sea shells**

Sample Output 2:
The sentence is not alliterative

Q2 Find Relationship

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Find Relationship(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB
Concepts Covered:

- Strings
- Functions

Problem Description:

Mr. George is a software developer. He wants to develop software that can identify the relationships between the users if they entered their names. According to his requirement, the program should get two names from the users and, based on the total length of the names entered, it should find out the relationship between the persons with that name. Help him develop the software using Python.

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

<table>
<tr><th>Requirement</th><th>Methods</th><th>Responsibilities</th></tr>
<tr><td>Find the relationship between users</td><td>find_relationship(name1,name2)</td><td>This method takes two names as arguments. Find the total length of the names, and get the reminder value by dividing the total length of the name by 6. Based on this reminder value, decide the relationship.

Refer to the table below for deciding the relationship:

<table>
<tr><th>Reminder</th><th>Return value</th></tr>
<tr><td>0</td><td>Soulmates</td></tr>
<tr><td>1</td><td>Colleagues</td></tr>
<tr><td>2</td><td>Friends</td></tr>
<tr><td>3</td><td>Good friends</td></tr>
<tr><td>4</td><td>Best friends</td></tr>
<tr><td>5</td><td>Close friends</td></tr>
</table>

Once the relationship is identified, then return the relationship value to the caller method.

Note:

To find out the length of the name, do not consider the space in the name.</td></tr>
</table>

Process flow:

- In the '**main**' method, get two names from the user.
- Invoke the method '**find_relationship**' and pass the two names as arguments.
- Capture the relationship value returned from the method, and display it as specified in the sample input and output.

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.
- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input 1:

Enter the name 1:**Glenn**

Enter the name 2:**Kim**

Sample Output 1:

Friends

Sample Input 2:

Enter the name 1:**Lilly**

Enter the name 2:**Lenny**

Sample Output 2:

Best friends

Sample Input 3:

Enter the name 1:**Michael**

Enter the name 2:**Vince**

Sample Output 3:

Soulmates

Sample Input 4:

Enter the name, 1:**Glenn Martin**

Enter the name, 2:**Liv Morgan**

Sample Output 4:

Friends

Q3 Sentence Palindrome

- Strings
- Functions

Problem Description:

The program is to find out if a sentence is a palindrome or not ignoring punctuation and whitespaces

The program must get a sentence as input and pass this sentence to an is_palindrome() function which checks if the given sentence is a palindrome or not and returns back a boolean value to the main

Requirement 1: Define the function: **is_palindrome()**

Requirement	Methods	Responsibilities
Check if the input sentence is a palindrome or not.	is_palindrome(sentence)	This method takes an argument which is a sentence and checks if the sentence is a palindrome or not (omitting the

		punctuations and white space) and returns back a boolean value to the main function. Return 'True' if the sentence is a palindrome, else return 'False'.

Process flow:

- In the **'main'** method, the user has to get input as a sentence
- The sentence is then passed to the '**is_palindrome'** function which checks if the given input sentence is palindrome or not after omitting the white space and punctuation
- The function '**is_palindrome'** returns back a **boolean value,** '**True**' if the sentence is a palindrome and **'False'** if the sentence is not a palindrome
- In the **'main'** method if the **'is_palindrome'** method returns true then we have to print that the sentence is a palindrome else the sentence is not a palindrome, and replace the sentence with the given actual input sentence.

Sample Input and Output 1:

```
Enter a sentence: liril mom liril
liril mom liril is a palindrome.
```

Sample Input and Output 2:

```
Enter a sentence: Sun rises in the east
Sun rises in the east is not a palindrome.
```

Sample Input and Output 3:

```
Enter a sentence: liril, mom liril
liril, mom liril is a palindrome
```

Q4 Blood Pressure Status

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Blood Pressure Status(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Strings

- Functions

Problem Description:

Kate was under treatment for High Blood Pressure (BP) for the past 6 months. He used to self-check his BP reading. He wants to know the status of his BP level. Help him by writing a program in Python to find the BP status for him. Get Systolic and Diastolic as single input separated with '/' (90/70) from the user and tell the status of the BP level.

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

<table>
<tr><th>Requirement</th><th>Methods</th><th>Responsibilities</th></tr>
<tr><td>Generate the status of BP</td><td>generate_status (BP_level)</td><td>This method takes the BP level as its argument, finds the status, and returns the same.

To generate the status use the below conditions:

<table>
<tr><th>Systolic</th><th>Diastolic</th><th>Status</th></tr>
<tr><td><90</td><td><60</td><td>Low BP</td></tr>
<tr><td>>=90 and <=120</td><td>>=60 and <=80</td><td>Normal</td></tr>
<tr><td>>=121 and <=140</td><td>>=81 and <=90</td><td>Pre-High BP</td></tr>
<tr><td>>=141 and <=190</td><td>>=91 and <=100</td><td>High BP</td></tr>
<tr><td>>190</td><td>>100</td><td>Hyper Tension</td></tr>
</table>

If the input does not meet the above criteria, it should return "Invalid Input".</td></tr>
</table>

Process flow:

- In the '**main**' method, get the BP level from the user ("/" separated)
- Call the '**generate_status**' method and pass this input string as its argument and capture the string returned by the method
- Display the status returned by the function 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.
- 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 1:

Enter the BP level:

90/70

Sample Output 1:

Normal

Sample Input 2:

Enter the BP level:

145/98

Sample output 2:

High BP

Q5 **Flat Discount**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Flat Discount(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Strings
- Functions

Problem Description:

'WeBuild' construction is a famous construction company that sells apartments at low prices. All the flats have 2BHK and 3BHK houses. Based on the sum of house number and house type they had decided to give a discount for their customers from the house amount. Help the company to calculate the discount for their customer based on given conditions.

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

<table>
<tr><th>Requirement</th><th>Methods</th><th>Responsibilities</th></tr>
<tr><td>Find the discount amount.</td><td>calculate_discount (input_string)</td><td>This method takes a string as its argument.

Split the input string based on the colon (':') with colon-separated.

Find the sum of the house number and calculate the discount amount based on the below conditions:

<table>
<tr><th>House Type</th><th>Cost</th><th>House Number sum</th><th>Discount</th></tr>
<tr><td>2BHK</td><td>3900000</td><td>Odd Number</td><td>4</td></tr>
<tr><td>3BHK</td><td>5100000</td><td>Odd Number</td><td>8</td></tr>
<tr><td>2BHK</td><td>3700000</td><td>Even Number</td><td>5</td></tr>
<tr><td>3BHK</td><td>4900000</td><td>Even Number</td><td>7</td></tr>
</table>

Calculate the discount amount and return the same.</td></tr>
</table>

Process flow:

- In the '**main**' method, get the house number and house type from the user as colon-separated values.
- Invoke the '**calculate_discount**' method and pass the input_string as an argument to capture the discount amount and display it as specified in the sample input and output.

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.
- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input 1:

Enter the details:

123:2BHK

Sample Output 1:

185000.0

Sample Input 2:

Enter the details:

435:3BHK

Sample Output 2:

343000.0

Q6 **Find Lukcy Number**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Find Lukcy Number(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Strings
- Functions

Problem Description:

Mr. Lenny wants to create an application to find a lucky number based on the date of birth. When this app receives the user's date of birth, it sums each value of the date of birth to determine the lucky number. Help him develop software using Python.

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

Requirement	**Methods**	**Responsibilities**
Find the lucky number from the input string.	find_lucky_number(dob)	This method takes the date of birth (as a string) in the format dd/mm/yyyy as an argument. Validate the string according to the following condition: In the input string, the first two characters representing a day should be between 01 and 31, the next character should be a slash ('/'), and the following two characters representing a month should be between 01 and 12, the next character should be a slash ('/'), and the following character representing a year should be less than 2023. **Eg:28/08/1999** If the string is valid, then add the day, month, and year. Then sum each digit of the added values and return the same. If the input string is not valid, the function should return the message as '**Invalid format'.** **For example**: If the string (date of birth) entered is 28/08/1999, then the lucky number is calculated as, 28 + 08 + 1999= 2035 2+0+3+5=10 The lucky number is 10. **Note: Do not use date functions. Consider the entered date format value as a string and do the specified manipulations.**

- In the **'main'** method, get the input string date of birth from the user. The input string date of birth should be in the following format: dd/mm/yyyy.

- Invoke the method '**find_lucky_number'** and pass the input string as arguments.

- Capture the value returned from the method, and display it as specified in the sample input and output.

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.
- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input 1:

Enter the date of birth

24/04/1990

Sample Output 1:

The lucky number is 11

Sample Input 2:

Enter the date of birth

11-11-2001

Sample Output 2:

Invalid format

Sample Input 3:

Enter the date of birth

33/14/2999

Sample Output 3:

Invalid format

Q7 **Toll Check**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Toll Check(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Strings
- Functions

Problem Description:

The National Highways Department has announced a discount scheme for four-wheelers at the toll gate as part of its 50th-year celebration.

If the vehicle number plate's last four character sum is an odd number, then they will receive an even place sum discount. They will receive the odd place sum discount if the last four-character sum is an even number. Using the function below, assist them in creating an application in Python for the aforementioned purposes.

Requirement: Define a function with the name '**check_number()"**

Requirement	Methods	Responsibilities
Find the toll discount percentage	check_number (vehicle_number **)**	This method takes the vehicle number string as its argument. The length of the vehicle number should be 10; otherwise, it should return the message "**Invalid vehicle number."** If the length of the vehicle number is 10, then find the sum of the last four numbers in the vehicle number. If the vehicle number's last four numbers sum is an odd number, then they will get the even place sum discount. **For example**: If the vehicle number is TN43CD1112 then have to return the message with a discount as "**Your discount percentage is 3**". Here, the sum of the last four numbers is 5 which is an odd number. So have to add even place values(1+2)=3. So the percentage is 3. If the vehicle number's last four numbers sum is an even number, then they will get the odd place sum discount. **For example:** If the vehicle number is TN43CD1311 then have to return the message with a discount as "**Your discount percentage is 2**". Here, the last 4 numbers sum is 6 which is an even number, so we should take the odd number sum, that is 1+1=2. So the discount percentage is 2.

Process flow:

- In the '**main**' method, get the vehicle number as the user input string.
- Then call the '**check_number**' method and pass the vehicle number as an argument.
- Display the values returned by the function.

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.
- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input 1:

Enter Vehicle Number: **TN35CR1112**

Sample Output 1:

Your discount percentage is 3

Sample Input 2:

Enter Vehicle Number: **TN43AD1311**

Sample Output 2:

Your discount percentage is 2

Sample Input 3:

Enter Vehicle Number: **TN43AD23**

Sample Output 3:

Invalid vehicle number

Q8 **Find Grade**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Find Grade(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Strings
- Functions

Problem Description:

The NNN Academy conducts an exam for its students. They want to develop software to find their students' exam results. As per their software, the student gets 2 marks for correct answers and 1 negative mark for wrong answers. Then calculate the students' mark percentage. Based on the percentage of students, the result should be determined. Help them develop software using Python.

Get the correct and incorrect answers to count from the user. then invoke the following function to find the exam result.

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

<table>
<tr><th>Requirement</th><th>Methods</th><th>Responsibilities</th></tr>
<tr><td>Find the exam result of the student.</td><td>find_exam_result(correct,Incorrect)</td><td>This method takes the correct and incorrect counts of a question as arguments. Then, add both questions counts together to get the total number of questions. If the total question is 120, then find the total marks. Otherwise, this function should return the message 'Invalid number of questions

Total marks are calculated by assigning two marks to each correct answer and one negative mark to each incorrect answer.

Then calculate the percentage as:

Percentage=(total marks/120)*100

Based on the mark percentage students' grades should be determined and returned the same.
<table>
<tr><th>Percentage</th><th>Return string value</th></tr>
<tr><td>Greater than equal to 75</td><td>You have received A grade</td></tr>
<tr><td>Greater than equal to 60 and less than 75</td><td>You have received B grade</td></tr>
</table>
</td></tr>
</table>

		Greater than equal to 50 and less than 60	You have received C grade
		Less than 50	Sorry! You have failed

Process flow:

- In the '**main**' method, get the correct and incorrect answer counts from the user.
- Invoke the method '**find_exam_result**' and pass the correct and incorrect answer counts as arguments.
- Capture the value returned from the method, and display it as specified in the sample input and output.

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.
- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input 1:

Enter the count of correct answers:**90**

Enter the count of incorrect answers:**30**

Sample Output 1:

You have received A grade

Sample Input 2:

Enter the count of correct answers:**40**

Enter the count of incorrect answers:**80**

Sample Output 2:

Sorry! You have failed

Sample Input 3:

Enter the count of correct answers:**60**

Enter the count of incorrect answers:**50**

Sample Output 3:

Invalid number of questions

Q9 **Product Code**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Product Code
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 512 MiB

Concepts Covered:

- Strings
- Functions

Problem Description:

A manufacturing company ships the product to a specific location. For that, they need a code for each product. The code is generated based on the product name, destination location, and manufactured month and year. Help them generate code in Python.

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

Requirement	**Methods**	**Responsibilities**
Generate the code for the product	generate_code(product_details)	This method takes product_details as arguments. Product details contain product name, destination, month, and year (productName:destination:month: year) as colon-separated values. The function should split the product_details based on the (':') colon separator and then validate the details. The validation rules are: The length of the product name and the length of the destination should be greater than 3. The month should be between 1 to 12 (inclusive), and the length of the year should be 4. If all the details are valid, then generate the product code. The product code format should be **Product_name/destination/month_year**. The **product_name** in the product code should be formed as: **1.** If the length of the product name is an odd number, the product code will be the first 3 characters, For example, The product code for **mango** will be **MAN** 2. If the length of the product name is an even number, the code will be the last 3 characters. For example, the product code for **grapes** will be **PES.** 3. Generated product name should be in upper case. The **destination** in the product code should be the first and the last characters of the destination in upper case. For example, if the destination is **Florida**, then the destination in the product code should be: **FA** The **month with the yea**r in the product code should be formed as the month followed by the last 2 digits in the year. For example: if the month and year are **9 and 2019,** then the month_year should be '**919**'. **Example:** If the input string is: **Sanitizer:**

		Florida:9:2019 Then the generated code will be **SAN/FA/919** **Note:** Do not consider the space in the product name and destination. If the entered product detail is not valid, then display the message: **Invalid product details.**

Process flow:

- In the '**main**' method, get the sentence and a word from the user.
- Call the '**generate_code**' method and pass the sentence and the word as its arguments.
- Capture the string returned by the function and display it.

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.
- Strictly follow the naming conventions for variables and functions as specified in the problem description.

Sample Input 1:

Enter the details:

Mask: Pune:11:2019

Sample Output 1:

ASK/PE/1119

Sample Input 2:

Enter the details:

Sanitizer: Florida:9:2019

Sample Output 2:

SAN/FA/919

Sample Input 3:

Enter the details:

Sanitizer: Bo:13:2019

Sample Output 3:

Invalid product details

Q10 **Replace the Word**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Replace the Word
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Strings
- Functions

Problem Description:

A social media company wants to process users' posts to replace any unwanted words before they are published on the platform. Help him by writing a Python program to replace the unwanted words.

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

Requirement	Methods	Responsibilities

Replace all the occurrences of the specified word.	replace_word(sentence, word)	This method takes a sentence and the word to be replaced as the arguments. If the sentence contains the specified word, then this function should replace all the occurrences of that word from the sentence (case-insensitively) with **asterisks** and return it to the caller method. **Note:** number of asterisks should be the length of the specified word. If the specified word is not there in the sentence, then return the message as "**The given word is not found in the sentence".** Refer to the sample input and output statements for more clarifications.

Process flow:

- In the '**main**' method, get the sentence and a word from the user.
- Call the '**replace_word**' method and pass the sentence and the word as its arguments.
- Capture the string returned by the function and display it.

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.
- Strictly follow the naming conventions for variables and functions as specified in the problem description.

Sample Input 1:

Enter the sentence:

This is a great day but sometimes people can be so damn annoying

Enter the word:

Damn

Sample Output 1:

This is a great day but sometimes people can be so **** annoying

Sample Input 2:

Enter the sentence:

Welcome to our home. Our home is very nice

Enter the word:

our

Sample Output 2:

Welcome to *** home. *** home is very nice

Sample Input 3:

Enter the sentence:

It's really annoying when a train is late and there's no explanation

Enter the word:

delay

Sample Output 3:

The given word is not found in the sentence

Q11 **Key generation**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes

Based on: Key generation
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Strings
- Functions

Problem Description:

Shopjee is one of the famous online markets. The market intends to provide its customers with a secret code in order to facilitate their purchases. If the customer wants to purchase a product, they must provide their username and secret key. The secret key is a single character that is the average of the ASCII values of their username. Help them create an application in Python to create the secret key using the below-mentioned function.

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

Requirement	**Methods**	**Responsibilities**
generate the secret key	generate_secret_key(name)	This method should take the name as its argument and generate a secret key for the customer. The length of the string should be between 2 to 10, both inclusive and the username must have alphabets only. If not, the function should return "**Invalid Input**". For generating the secret key, convert the input string to lowercase and find the average of the equivalent ASCII values of all characters in the input string. Return the equivalent character to the average value as output For example: "Reverse" has a length of 7 and has only alphabets. The lowercase equivalent is "reverse". So, the sum of equivalent ASCII values is(r=114,e=101,v=118,e=101,r=114,s=115,e=101) 764. Now, the average of ASCII values is (764/7) 109. Therefore, the equivalent character of the average value, 109 is m. **Note:**

		Make use of **ord()** and **chr()** methods for getting the ASCII value (Unicode value) of a character and for converting an ASCII (Unicode) to the corresponding character. ord('e') will give 101 chr(101) will give 'e'.

Process flow:

- In the '**main**' method, get the name from the user.
- Call the '**generate_secret_key**' method and pass this input string as its argument and capture the key returned by the method.
- Display the value returned by the function 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.
- 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 1:

Enter the name: **Mathew**

Sample Output 1:

k

Sample Input 2:

A

Sample Output 2:

Invalid Input

Sample Input 3:

Ab123

Sample Output 3:

Invalid Input

Q12 **Trainer Ratings**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Trainer Ratings(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Collections
- Functions

Problem Description:

BestOne is a top training institute. The institute has a large number of trainers to train new candidates who join their institute. In the end, the candidates have to give a rating to the trainers based on the training they have taken. The institute wants to know how many candidates gave ratings between 0 - 5 (inclusive) and 6 and above so that they can give some incentives to the trainer. Write a program in Python to simulate this scenario.

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

Requirement	Methods	Responsibilities
Create a list of ratings from the input string.	create_ratings (input_string)	This method takes a string of values (trainer_id: ratings, colon separated) separated with a comma as an argument. The function should split the string based on the comma separator and generate it as a list. Then split each element in the list based on ':' (colon) and store the **trainer id as the key** to the dictionary and the **rating as a float value** for that key. Return this dictionary to the caller method.

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

Requirement	Methods	Responsibilities
Count the ratings	count_ratings(rating_dict)	This method takes the **dictionary of ratings** as the argument. Iterate this dictionary and store the id of the trainers in a list of the no. of ratings between 0 and 5 (inclusive). Likewise, if the trainer rating is 6 and above, then add the trainer id to another list. Finally, return both lists to the caller method. The return order should be: First, return the trainer list whose rating is between 0-5 and the other trainer list should be the second value **For example**, If the rating dictionary is: **{'ss12':2, 'rr34':6, 'ww21':3,'tr45': 7, 'yt:23':5},** then the function should return the values as : **['ss12', 'ww21','yt23'], ['rr34', 'tr45']** If there are no trainers with the range specified, then return the string **and empty list** instead of that specified list.

Process flow:

- In the **'main'** method, get a string of trainer details (trainer_id: ratings) separated with a comma (',').

E.g.: WW23:4,RR45:8,YY12:1

- Call the '**create_ratings**' method and pass this input string as its argument and capture the dictionary returned by the method.
- Then call the '**count_ratings**' method and pass the dictionary of ratings as its argument.
- Display the values returned by the function as specified in the sample input and output statements.
- If any or all of the list returned by the '**count_ratings**' function is empty then display the message 'Nil' as shown in the sample 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.
- 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 1:

Enter the ratings (as comma-separated values): **WW12:5, TT11:9, WQ34:7,YU60:3,BN01:7,VV55:8,PL23:6**

Sample Output 1:

The list of trainers with ratings between 0-5: ['WW12:', 'YU60']

The list of trainers with ratings between 6 and above: ['TT11', 'WQ34', 'BN01', 'BN01',' PL23']

Sample Input 2:

Enter the ratings (as comma-separated values): **WE01:0, TR02:1, PO02:4, IT05:2**

Sample Output 2:

The list of trainers with ratings between 0-5: ['WE01', 'TR02', 'PO02',' IT05']

The list of trainers with ratings between 6 and above: Nil

Sample Input 3:

Enter the ratings (as comma-separated values): **WE01:10, TR02:6, PO02:7, IT05:9**

Sample Output 3:

The list of trainers with ratings between 0-5: Nil

The list of trainers with ratings between 6 and above: : ['WE01', 'TR02', 'PO02',' IT05']

Q13 **Filter Participants**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Filter Participants(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Collections
- Functions

Problem Description:

Buddy's academy is one of the most famous gaming academies in our city. They conduct 3 types of games for the kids. As part of the game, their average points will be calculated. From their average points, they are selected for the next round. Help the academy select the participants for the next level using the Python program.

Requirement 1: Define a function ' **calculate_score()'**

Requirement	**Methods**	**Responsibilities**
Calculate the score for each participant	calculate_score(participants_list)	This method takes the **list of participants'** names and each round score as its argument. Iterate through this list, calculate the average score and store the name and the average in a dictionary. The dictionary keys should be: the name of the participant and the value should be average. **Example:** {'Kings":87, 'John':65, ...} Finally, return this dictionary to the caller method.

Requirement 2: Define a function '**filter_participants**()'

Requirement	**Methods**	**Responsibilities**
Filter the participants	Filter_participants (participants_dictionary, pass_score)	This method takes the dictionary that is returned by the **calculate_score()** method and the passing score (score for selecting the next level as an argument). Iterate the participants' dictionary and find out the qualifiers for the next level based on the passing score provided. If the participant's average score is greater than or equal to the specified pass score, then display the participant's name. **For example:** If the participants' dictionary is: {'Kevin': 84.5, 'Smith': 70, 'Benny':81'} and the pass score is 80, then the returned list should be: **["Kevin", "Benny"]** If no participants are selected for the next level, then return an empty list (**[]**).

Process flow:

- In the **'main'** method, get the participant name, and each round point from the user as colon (":") separated values (already given in the code template) and append it to a list.
- Call the '**calculate_score'** method and pass the list of participants' details and capture the list of dictionaries returned by the function and display it.
- Then call the **'filter_participants**' method and pass the participants' dictionary and pass the score to the method for filtering the participants and appends the name of the participants who are selected for the next level to a list and returned the same. If the function returns an empty list, display the message as " **No one selected".**

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.
- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input and Output1:

Enter the no. of participants**:4**

Enter the details:

Johan:77:87:69

Smith:44:65:56

George:99:73:56

Benny:67:67:33

Enter the pass score to select next level:**76**

Next level selected participants are:

Johan

George

Sample Input and Output 2:

Enter the no. of participants:**3**

Enter the details:

James:70:82:66

Livi:47:75:36

Ronn:96:73:56

Enter the pass score to select the next level:**80**

No one selected

Q14 **Room Rent**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Room Rent(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Strings
- Collections
- Functions

Problem Description:

AADISTA Hotel Management performs all its tasks, especially customer manipulations and rent calculations manually. Since they take an enormous time to

perform these tasks, they planned to automate them. In order to do so, they need an application in Python language that calculates the number of days a room is taken for rent and calculates room rent.

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

Requirement	**Methods**	**Responsibilities**
calculate the number of days	calculate_days(from_date, to_date)	This method should take the **from_date** and **to_date** as arguments and calculate the number of days the room was taken for rent and return the same **Note:** Without applying the date function find the number of days. Use the string split function to split the date. In from_date and to_date input, the first two digits represent the days and the next two digits represent the month. **Assume, the number of days in a month should be 30 days.**

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

Requirement	**Methods**	**Responsibilities**
calculate total amount	calculate_total_amount (customer_name, room_type, no_of_days):	This method should take customer_name, room_type, and no_of_days as arguments and calculate the total amount for the room. The room amount should be calculated based on the below conditions.

Room_type	**Room cost per day**	**No of days**	**Discount**
Single	3300	<=3	10%

			>3	15%
		Double	4000	<=3	10 %
				>3	17%
		Triple	4500	<=3	10%
				>3	20%
		This method should return a dictionary with the keys: "Customer Name", "No of days" and "Total amount" and the values should be the corresponding values of those keys.			

Process flow:

- In the **'main'** method, the input for the program is given along with the code template. The input format should be:

  room_no:customer_name:room_type:from_date:to_date (colon separated).

  In **the from_date** and **to_date** input values, the first two digits represent the days and the next two digits represent the month.

Example: **ARO123:Smith:Double:12/05:13/05**

- Call the **'calculate_days'** method and pass **from_date and to_date** as its arguments. Capture the days returned by the method.
- Call the '**calculate_total_amount'** method and pass **customer_name, room_type**, and **no_of_days** as parameters.
- Display the values returned by the function 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, 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:

ARO123:Smith:Double:12/05:13/05

Sample Output:

Customer Name: Smith

No of days: 1

Total amount: 3600.0

Q15 **Bike Race**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Bike Race(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Functions
- Collections

Problem Description:

F2 conducts a bike race event. At the end of the event, they want to know how many participants are qualified for the next level. Get the total number of participants for

the race and the biker's Id, name, and speed from the user. Based on the speed, find out the time taken to complete the race, and display the participants who are all qualified for the next level based on the given time.

Write a Python program to find the count of participants qualified for the next level.

Requirement 1: Define a function 'calculate_time()'

Requirement	**Methods**	**Responsibilities**
Calculate the completion time of each racer.	calculate_time(race_details)	This method takes a **list of bikers' details (string) as its argumen** Iterate through this list and split each string based on ':' (colon) and completion time of each player. The completion time should be calculated as : **Time taken= distance/speed. (where distance=200)** **Note: Round off the completion time for one (1) decimal place.** Then store the id, name, and completion time in a dictionary in wh 'Id', 'Name', and 'Time' and the values should be the corresponding attribute. Then add each of these dictionaries to a list. For example, if the input for the function is : **['BK12:Keane:2.5','BK23:Smith:3.3','BK03:Maxi:2.4'],** then the generated dictionary should be: **[{'Id':'BK12','Name':'Keane','Time': 2.5}, {'Id':'BK23','Name':'Smith','Time':3.3},{'Id':'BK03','Nar** Finally, return this list of dictionaries to the caller method.

Requirement 2: Define a function '**find_qualifiiers()**'

Requirement	**Methods**	**Responsibilities**
Find the qualifiers.	find_qualifiers(race_details,time)	This method takes the **list of dictionaries** that are returned by the calculate_time() method and t**he time to qualify** for the next level. Iterate through this list, and find out the qualifiers for the next level based on the qualifying time provided.

		Any participants who have taken less than or equal to the time specified by the user are qualified for the next level. Append these qualified racer's names to a list and return this list to the caller function. For example: If the list of dictionaries is like **[{'Id':'BK12','Name':'Keane',' Time': 2.5}, {'Id':'BK23','Name':'Smith',' Time':3.3},{'Id':'BK03','Name':'Maxi',' Time':3}]** and the qualifying time is **3**, then, the returned list should be: **["Keane", "Maxi"]** If no participants are qualified for the next level, then return an **empty list ([]).**

Process flow:

- In the **main()** method, get the no. of racers and their details as specified in the sample input statements. (already given in the code template) and append it to a list.
- Call the '**calculate_time'** method and pass the list of race details and capture the list of dictionaries returned by the function.
- Then call the **'find_qualifiers**' method and pass this list of dictionaries to the method for identifying the qualifiers and display the list of qualifier names returned. If the function returns an empty dictionary, display the message as "**No one is qualified".**

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 number of racers and the racers' details (biker id, name, and speed) from the user. Get the racer details as a single string separated by '**:**' (colon). Refer to the sample input and output statements for more clarifications.

Example:

Enter the no. of race participants:3

BK12:Keane:80

BK23:Smith:60

BK03:Maxi:85

- Do not alter the given code template. Write your code only in the necessary places.
- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input and Output 1:

Enter the no. of race participants:**3**

Enter the details:

BK12:Keane:80

BK23:Smith:60

BK03:Maxi:85

Enter the time to qualify for the next level:**2.5**

The qualified participants are:

Keane

Maxi

Sample Input and Output 2:

Enter the no.of race participants**:2**

BK12:Glenn:56

BK45:Ruby:45

Enter the time to qualify for the next level:**3**

No one is qualified

Q16 **Deducing Blood Group**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Deducing Blood Group(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Collections
- Functions

Problem Description:

SNAS laboratory decided to develop software to deduce the blood group of a person based on two factors.
i. antigens and antibodies type present
ii. presence or absence of the RhD antigen

Help them to develop software using Python.

Get the input from the user as a comma-separated string like 4 y/n (yes or no) corresponding to the presence or absence of A antigens, B antigens, anti-A antibodies, anti-B antibodies, and + or - corresponding to the presence or absence of the Rh factor.

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

Requirement	**Methods**	**Responsibilities**
Create a list from the given input string	create_list(factors)	This method should take a string value separated by commas as an argument. Then split the string based on the comma, generate a new_list, and then return this list to the caller method.

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

Requirement	**Methods**	**Responsibilities**

<table>
<tr>
<td>Deduce the blood group</td>
<td>deduce_blood_group(blood_details)</td>
<td>This method takes the list containing 5 characters: 4 y/n corresponding to the presence or absence of A antigens, B antigens, anti-A antibodies, anti-B antibodies, and + (plus) or - (minus) corresponding to the presence or absence of the Rh factor as the argument.

Deduce the blood group by matching the input obtained with the help of the table given below and then return the blood group.

Refer to the table given below.

<table>
<tr><th>Antigens/ Antibodies type 1</th><th>Antigens/ Antibodies type 2</th><th>RhD</th><th>Blood Group</th></tr>
<tr><td>A antigens</td><td>anti-B antibodies</td><td>positive (+)</td><td>A+</td></tr>
<tr><td>B antigens</td><td>anti-A antibodies</td><td>positive (+)</td><td>B+</td></tr>
<tr><td>A antigens</td><td>B antigens</td><td>positive</td><td>AB+</td></tr>
<tr><td>anti-A antibodies</td><td>anti-B antibodies</td><td>positive (+)</td><td>O+</td></tr>
<tr><td>A antigens</td><td>anti-B antibodies</td><td>negative</td><td>A-</td></tr>
<tr><td>B antigens</td><td>anti-A antibodies</td><td>negative (-)</td><td>B-</td></tr>
<tr><td>A antigens</td><td>B antigens</td><td>negative (-)</td><td>AB-</td></tr>
<tr><td>anti-A antibodies</td><td>anti-B antibodies</td><td>negative (-)</td><td>O-</td></tr>
</table>

Example: If the inputs are like y,y,n,n,+. then the function should return the value: AB+

Note:

The presence of A antigens and anti-A antibodies is an incorrect combination.
The presence of B antigens and anti-B antibodies is an incorrect combination.

</td>
</tr>
</table>

		In either case, the function should return **False**.

Process flow:

- In the '**main**' method, get the inputs separated with a comma(',').
- Call the '**create_list**' and pass this input string as its argument and capture the list returned by the method.
- Then call the '**deduce_blood_group**' and pass the new_list and capture the value returned by the function. If the function returns **True** display the output as specified in the sample input and output statements.
- If the '**deduce_blood_group**' method returns **False**, display the message as "**Incorrect combination of antigens/antibodies entry**".

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 and to check for their 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 inputs from the user as a single string separated by commas.
- 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 1:
Enter y/n for A antigens, y/n for B antigens, y/n for anti-A antibodies, y/n for anti-B antibodies, and +/- for Rh factor (as comma separated values):**y,y,n,n,-**

Sample Output 1:
Deduced blood group: AB-

Sample Input 2:
Enter y/n for A antigens, y/n for B antigens, y/n for anti-A antibodies, y/n for anti-B antibodies, and +/- for Rh factor (as comma separated values): **y,n,y,n,+**

Sample Output 2:
Incorrect combination of antigens/antibodies entry

Skip Remaining time

Remaining time

Live Proctoring

Qualifier Assessment

Participants

Grades

Python Basics

Python Collection

Python

ANSISQL Joins

ANSISQL

Help Desk

Q17 **Cricket Academy**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Cricket Academy(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Coverage:

- Functions
- Collections

Problem Description:

A leading cricket selection academy in the state is in need of an automated system that should manipulate the player details provided. Help them to develop a Python application that can be used by the administrator for the mentioned requirement.

Requirement: Create player details, store them, and display them using functions.

Requirement 1: Create the player details

Requirement	**Method Name**	**Responsibilities**
Create the player details and store this information in a list as dictionary	create_player(player_id, player_name, matches_played, runs_scored)	This method should take player id, name,matches_played, and runs_scored as its argument and should save inside a dictionary as : {"Id":<id of the player>,"Name":<name of the player>,"Matches_Played":<no. of matches played>, "Runs_Scored":<total no. of run taken>} The function should **return this dictionary** to the caller method.

Requirement 2: Display player details

Requirement	**Method Name**	**Responsibilities**
Iterate the list and display the player details.	display_player(players_details)	This method should take the '**players_details**' list, as an argument and iterate this list and display details of the players who

		have taken centuries. If no player has taken centuries, then display the message as: "No player details found"

Process flow:

1. In the main method, if the user enters option 1, get the player details such as player id, name, matches played, and runs scored, from the user and pass those details to the function ' **create_player'**. This method should return a dictionary of player details and append this dictionary to the list '**players_details**'.

2. If the user enters option2, pass list 'players_details' as an argument to function '**display_player**'

3. Option 3 is to stop the program execution. When the user chooses this option, **display the message** "**Thank you"** and exit. Please do not use 'sys. exit()' . Instead, use the '**break'** statement.

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.
- 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 and Output 1:

1. Create Player

2. Display Player details

3. Exit

Enter the option: **2**

No player details found

1. Create Player

2. Display Player details

3. Exit

Enter the option: **1**

Player id: **ICC345**

Player name: **Dhoni**

Matches played: **400**

Runs scored: **6789**

1. Create Player

2. Display Player details

3. Exit

Enter the option: **1**

Player id: **ICC890**

Player name: **Rohit**

Matches played: **20**

Runs scored: **568**

1. Create Player

2. Display Player details

3. Exit

Enter the option: **2**

Player 1 :

Id: ICC345

Name: Dhoni

Matches_Played : 400

Runs_Scored : 6789

Player 2 :

Id: ICC890

Name: Rohit

Matches_Played : 20

Runs_Scored : 568

1. Create Player

2. Display Player details

3. Exit

Enter the option **: 3**

Thank you

Sample Input and Output 2:

1. Create Player

2. Display Player details

3. Exit

Enter the option: **1**

Player id: **ICC877**

Player name: **Ronn**

Matches played: **2**

Runs scored: **56**

1. Create Player

2. Display Player details

3. Exit

Enter the option: **2**

No player details found

1. Create Player

2. Display Player details

3. Exit

Enter the option: **3**

Thank you

Q18 **Game Event**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Game Event(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Collections

- Functions

Problem Description:

At SRS College, "Rainbow FM" has arranged a game among college students. Mr. Joe, who is the organizer of this event, divided some students into two teams and conducted the events between them. Help the organizer identify the winning team in each round and each team's win count in the game by using a Python program.

Get the number of rounds and each round's points for team 1 and team 2 from the user and store them in two lists separately.

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

Requirement	**Methods**	**Responsibilities**
Find the winner of each round	find_each_round_winner(team1,team2)	This method should take two lists (team 1's points 2's points) as its arguments. To find out the winner of each round, the function iterate these two lists and should compare the cor points in both lists. Define an empty list to store t team. When comparing the scores in each round, • If Team 1 points are more than Team 2, append the string '**Team 1**' to the new li • If Team 2 points are more than Team 1, append the string '**Team 2**' to the new li • If both team points are equal, then **"Equ** be appended to the list. For example, if Team1 = [2, 3, 4], and Team2 = [4, 3, 2], The new list should be ['Team2','Equal','Team1'] Finally, returned this new list to the caller method

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

Requirement	Methods	Responsibilities
Count the winning details of each team	count_winners(winner_list)	This method takes the winner list as its argument. Find the number of rounds won by each team, and add it to the dictionary with "**Team1,**" "**Team2**," and "**Equal**" as the keys and the corresponding **count** as the value. Then return this dictionary to the caller method.

Process flow:

- In the '**main**' method, get the number of rounds and each round's points from the user and store them in a list.
- Invoke the method '**find_each_round_winner**' by passing the list of team1 and team2 as its argument. Capture the winner details returned by the method.
- Then invoke the method '**count_winners**' by passing the winner's details list returned by the previous method as its argument. Store the dictionary that contains the winning count of each team returned by the function 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.
- If the number of rounds is less than or equal to zero then display the message as "**Invalid rounds".**
- Do not alter the given code template. Write your code only in the necessary places.
- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input 1:

Enter the no of rounds :

5

Enter the team1 points :

9

3

7

5

4

Enter the team2 points :

9

6

5

5

9

Sample Output 1:

[Equal, Team2, Team1, Equal, Team2]

Team1: 1

Team2: 2

Equal: 2

Sample Input 2:

Enter the number of rounds:

3

Enter the team1 points:

8

6

5

Enter the team2 points:

7

3

2

Sample Output 2:

['Team1', 'Team1', 'Team1']

Team1 : 3

Team2 : 0

Equal : 0

Sample Input 3:

Enter the no of rounds :

-4

Sample Output 3:

Invalid rounds

Q19 **Task Manager**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Task Manager(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Collections
- Functions

Problem Description:

The program is a to-do list manager that allows a user to add tasks to a list and mark them as completed once the task is completed. The program should store the tasks in a list that maintains the order in which the tasks were added and allows tasks to be efficiently removed by their index once it is completed.

To implement the to-do list manager, we could use a list from the collections to store the tasks.

Requirement 1: Define a function '**add_task()**'

Requirement	**Methods**	**Responsibilities**
Add the tasks to a list	add_task(task, todo_list)	This method takes a task and todo_list as arguments and adds the task to the end of the todo_list and returns the list.

Requirement 2: Define a function **'mark_task_complete()'**

Requirement	**Methods**	**Responsibilities**
Remove the task from a list	mark_task_complete(index, todo_list)	This method takes the index position and the todo_list as its arguments, removes the task at a given index from the list, and then displays the list with the remaining tasks. If the index is greater than the length of the list or the list is empty then display the message as "**Invalid input**".

Process flow:

- In the **'main'** method, if the user enters "1", get the task, from the user and pass the task and todo_list to the function '**add_task'**.

- If the user enters "2", get the index, from the user and pass the index and todo_list to the function '**mark_task_complete'** and then display the todo_list as specified in the sample input and output.
- If the user enters "3" then stop the program execution. Please do not use 'sys. exit()' . Instead, use the '**break'** statement.
- If the user enters other than "1", "2" or "3" then display the message "**Invalid command"** and exit. Please do not use 'sys. exit()' . Instead, use the '**break'** statement.

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.
- The code for creating a menu for displaying various options to add and remove 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 and Output 1:

Enter a command (1 to add a task, 2 to mark a task complete, 3 to quit): **1**

Enter the task to add: **Schedule Meetings**

Enter a command (1 to add a task, 2 to mark a task complete, 3 to quit): **1**

Enter the task to add: **Order stationery**

Enter a command (1 to add a task, 2 to mark a task complete, 3 to quit): **1**

Enter the task to add: **Maintain Office Items**

Enter a command (1 to add a task, 2 to mark a task complete, 3 to quit): **2**

Enter the index of the task to mark as complete: **1**

Schedule Meetings

Maintain Office Items

Enter a command (1 to add a task, 2 to mark a task complete, 3 to quit): **3**

Sample Input and Output 2:

Enter a command (1 to add a task, 2 to mark a task complete, 3 to quit): **1**

Enter the task to add: **Order Items**

Enter a command (1 to add a task, 2 to mark a task complete, 3 to quit): **1**

Enter the task to add: **Schedule Meetings**

Enter a command (1 to add a task, 2 to mark a task complete, 3 to quit): **2**

Enter the index of the task to mark as complete: **3**

Invalid Input

Enter a command (1 to add a task, 2 to mark a task complete, 3 to quit): **6**

Invalid command

Sample Input and Output 3:

Enter a command (1 to add a task, 2 to mark a task complete, 3 to quit): **2**

Invalid Input

Enter a command (1 to add a task, 2 to mark a task complete, 3 to quit): **3**

Q20 **Immunization Record**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Immunization Record(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 512 MiB

Concepts Coverage:

- Functions
- Collections

Problem Description:

A new vaccination center in the city wants an automated system that will help them to create and maintain the details of children who have booked for vaccination. Develop a Python application to meet this requirement.

Requirement: Create the records of children who have booked for vaccination, store them, and display them using functions.

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

Requirement	**Method Name**	**Responsibilities**
Create the details of children, and store this information in a list of dictionaries.	create_record (children_records)	This method should take a list of children's names, gender, weeks, and contact details (colon-separated values) as its argument. Iterate through this list and split each string based on ':' (colon). Convert weeks into integers. If the week >=1 and <=24 then save the details inside a dictionary as: {"Name":<name of the child>,"Gender":<gender of the child>,"Weeks":<no. of weeks since birth>, "Contact":<contact number of the guardian or parent>} Then append that dictionary to the list and return the same. If all the children's details do not meet the above condition, then this function should return an empty list..

Requirement 2: Define a function with the name **display_record()**

Requirement	**Method Name**	**Responsibilities**
Iterate the list and display the details of the children.	display_record(valid_records, weeks)	This method should take the '**valid_records**' (list of dictionaries) and the week as arguments, iterate this list of dictionaries, and display details of all the children whose **no. of weeks since birth**

		is equal to or less than the weeks specified in the argument list. 1. If there's no child <= the no. of weeks specified, display: "**No child under <weeks> weeks has booked for the vaccination".** 2. If there's just 1 child under the no of weeks specified, display: **"There is 1 child under <weeks> weeks who have booked for the vaccination".** 3. Else, display : "**There are <count> children under <weeks> weeks who have booked for the vaccination".**

Process flow:

- In the main method, get the number of children and children details name, gender, week, and contact) as a string of **colon-separated values** from the user and append it to children_record.
- Invoke the "**create_record**' method, and pass the list of children's details as the argument. If this method returns an empty list to the caller, then display the message "**No records available**"
- If the list is not empty, get the weeks from the user and invoke the **'display_record'** method and pass the list of the dictionary (valid_records) and the weeks as arguments, and display the message based on the description.

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 and Output 1:

Enter the no of children:

5

Enter name, gender, weeks, and contact as colon-separated values:

Bobs: M:8:8876556789

Joshna: F:6:9867577881

Math: M:24:8769567465

Ann: F:16:9095023412

Sweety:F:25:9798422137

To display the records based on weeks since birth - Enter the no of weeks(<=24):**10**

Record 1 :

Name: Bobs

Gender: M

Weeks: 8

Contact: 8876556789

Record 2 :

Name: Joshna

Gender: F

Weeks: 6

Contact: 9867577881

There are 2 children under 10 weeks who have booked for the vaccination

Sample Input and Output 2:

Enter the no of children:

2

Enter name, gender, weeks, and contact as colon-separated values:

Neo: M:12:9095023491

Josh: F:25:9798422137

To display the records based on weeks since birth - Enter the no of weeks(<=24):**10**

No child under 10 weeks has booked for the vaccination

Q21 **Stock Details**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Stock Details
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Collections
- Functions

Problem Description:

You are a software engineer at a sportswear leather products sales company, and you have been asked to build a program in Python that helps to check the stock details and place orders for customers. The program should allow users to check the current availability of a specific item, place an order with the cost of the specified item and display the remaining stock details.

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

Requirement	**Method Name**	**Responsibilities**
Check the availability of the item.	check_availability (item, quantity, stock)	This method should take item name, quantity, and stock as arguments. (stock dictionary is given already) You are provided with one dictionary **stock** of each item and quantity of each item respectively,

		Iterate the dictionary and find whether the given item (using the item name)is available or not. If the item is present and the quantity available is equal to or above the required quantity, then, the function has to return "**True".** If the Item is not present or the quantity is less than the required quantity, then the function has to return **"False".**

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

Requirement	**Method Name**	**Responsibilities**
Place the orders	place_order(item, quantity, stock, prices)	This method should take item name, quantity, stock, and prices as arguments and find the total amount for the item and remaining stock details. (The 'stock', and 'prices' dictionaries are given with code templates.) Iterate the stock dictionary and reduce the quantity specified by the user for the specified item quantity. Then iterate the prices dictionary to find the total amount and display it. **total_amount= qunatity*price** This function should also display the stock dictionary after reducing the required quantity. Refer to the sample input and output statements for more clarifications.

Process Flow:

- In the '**main**' method, get the item name and quantity from the user.

Note: You are provided with two dictionaries **stock and prices**. The '**stock**' the dictionary contains the **name of the items as the key** and the **available quantity**

as the value whereas the '**prices**' dictionary contains the **name of the item as**

the key and th**e price of the item as the value.**

- Call the '**check_availability**' method and pass item name, quantity, and stock dictionary as its argument and capture the boolean value returned by the method
- If the '**check_availability**' method returns **True** then call the method '**place_order**' and pass the item name, quantity, stock, and prices dictionaries as its arguments. The function should display the total amount and the stock dictionary after reducing the required quantity.
- If the '**check_availability**' method returns **False** then display the message as "**Item is not available**" and terminate the program.

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 1:

Enter an item: Gloves

Enter a quantity: 4

Sample Output 1:

Total amount: 1480

Remaining stock details

Sports Balls: 56

Shin guards: 50

Gloves: 56

Footwear: 15

Sample Input 2:

Enter an item: Masks

Enter a quantity: 3

Sample Output 2:

Item is not available

Q21 **Predict Disease Probability**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Predict Disease Probability
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Collections
- Functions

Problem Description:

You are a data scientist at a healthcare company, and you have been asked to build a program that helps doctors predict the likelihood of a patient developing a certain disease based on various risk factors. The program should allow doctors to enter patient data as a colon-separated string, such as age, gender, blood pressure, and cholesterol level, and output a probability score indicating the likelihood of the patient developing the disease.

To solve this problem, you could write a Python program that defines the following functions:

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

<table>
<tr><th>Requirement</th><th>Methods</th><th>Responsibilities</th></tr>
<tr><td>Find the risk score of the patient.</td><td>compute_risk_score(patient_data)</td><td>This method should take a list of patient details as an argument.

The function should find the total risk score by adding each factor's risk score based on the below condition and returning the same.

The risk score should be calculated as:

<table>
<tr><th>Factors</th><th>Conditions</th><th>Risk score</th></tr>
<tr><td rowspan="2">age</td><td>greater than 60</td><td>10</td></tr>
<tr><td>less than equal to 60</td><td>5</td></tr>
<tr><td rowspan="2">Gender</td><td>M</td><td>5</td></tr>
<tr><td>F</td><td>3</td></tr>
<tr><td rowspan="2">BP level</td><td>greater than 120</td><td>10</td></tr>
<tr><td>less than equal to120</td><td>5</td></tr>
<tr><td rowspan="2">cholesterol level</td><td>Greater than 200</td><td>15</td></tr>
<tr><td>less than equal to 200</td><td>10</td></tr>
</table>
</td></tr>
</table>

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

Requirement	Methods	Responsibilities
Find the probability score of the patient.	predict_probability(risk_score)	This method takes a risk score as an argument and returns a probability score indicating the likelihood of the patient developing the disease. The probability score is calculated as 1 minus the reciprocal of 1 plus the risk score. **Example:**

		probability = 1 - (1 / (1 + risk_score)) **Note:** • It should return the probability score as a float value in two decimal places.

Process flow:

- In the '**main**' method, get the input patient data separated with a colon(:,').
- Call the '**compute_risk_score**' and pass this patient data as its argument and capture the risk_score returned by the method.
- Then call the '**predict_probability**' and pass the risk_score as its argument.
- Display the probability score value returned by the function 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 and to check for 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 inputs from the user as a single string separated by colon (':').
- 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 1:

Enter the patient data

62:M:125:210

Sample Output 1:

The probability of the patient developing the disease is 0.98

Sample Input 2:

Enter the patient data

44:F:100:180

Sample Output 2:

The probability of the patient developing the disease is 0.96

Q22 **Parking Details**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Parking Details
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Strings
- Functions
- Collections

Problem Description:

Zee Cinema is one of the most famous theaters in the city. They have a large parking area to park all the vehicles. The parking area is divided into 2 blocks, A block for parking two-wheeler and B block for parking four-wheelers. When they enter vehicle details, first they need to validate the license number, and then they want to generate the parking id and amount based on the vehicle type.

You, as their software consultant, automate the above process by developing a program in Python.

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

Requirement	Method	Responsibilities

Validate license number	validate_license_number(vehicle_details)	This method takes vehicle details as its argument and should validate the license number. The validation criteria should be: • The length of the driving license number should be 15. • The first two letters should be AA. • The next two letters should be (3rd and 4th) 99. • The 5-8th characters should be license-issued years which is between (1990 - 2022)both inclusive. • The remaining characters should be any numbers. If the license number is valid then it should return **True.** If not it should return **False.**

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

Requirement	**Method**	**Responsibilities**
Generate parking id and amount	generate_parking_id (vehicle_details)	This method takes valid vehicle details as its argument. Using the vehicle details generate the parking id. The criteria for a valid parking id should be: • The first character in the parking id should be the block name. The parking block for **two-wheelers** is **A** and **for four-wheelers is B.** • The next two characters should be the sum of the last eight numbers of the license number • The last character should be the first letter of the customer's name. **E.g.:** If the entered string is: **TN37DE1034,Livi,AA9920126787787,**

		Two wheeler,2" then the parking id will be: **A50L** This method should **return t**he valid parking code, name of the customer, and amount for parking in the form of a **dictionary.** The keys of the dictionary should be: '**Name', 'Parking Id**', and '**Amoun**t', and the values should be the **corresponding values of each ke**y like customer name, generated parking number, and parking cost. Cost for Two wheelers: hour*20. Cost for Four wheelers: hour*30 **Example:** {'Name': 'Johan, 'Parking Id': 'A50L', 'Amount': 40.0} Note: Vehicle type should be **Two wheeler or Four wheeler**

Process flow:

- In the '**main**' method, get the vehicle details input from the user in the following format: vehicle number, customer name, license number, vehicle type, and duration in hours. (comma separated)
- Call the '**validate_license_number**' method and pass vehicle details to validate the license number. If the license number is valid then return **True** else return **False**.
- If the '**validate_license_number**' method returns **True** then call the method '**generate_parking_id**' and pass the parking details to generate the parking id and amount.
- Capture the dictionary returned by the method and display the values returned by the function as specified in the sample input and output statements.

- If the '**validate_license_number**' method returns **False** then display the message as "**Invalid Input**" and terminate the program.

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.

- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input 1:

Enter the Details:

TN37DE1034,Livi,AA9920126787787,Two wheeler,2

Sample Output 1:

{'Name': 'Livi', 'Parking Id': 'A50L', 'Amount': 40.0}

Sample Input 2:

Enter the Details:

TN40FW1034,Mathew,AA9919976733387,Four wheeler,2

Sample Output 2:

{'Name': 'Mathew', 'Parking Id': 'B37M', 'Amount': 60.0}

Sample Input 3:

Enter the Details:

TN40AA1034,Mathew,AG8720006733AA7,Two wheeler,2

Sample Output 3:

Invalid Input

Q23 **Finding the Ugly Numbers**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Finding the Ugly Numbers(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Functions
- Collections

Problem Description:
Bricks Public School decides to evaluate the mathematical skills of its students. They are given with a list of numbers and are asked to display the ugly numbers. Ugly numbers are numbers whose only prime factors are 2, 3 or 5.
Write a Python program to find an ugly number from a given list of numbers.

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

Requirement	**Methods**	**Responsibilities**
Find all the ugly numbers from the given list	is_ugly(number_list)	This method takes the list of numbers as the argument. Iterate this list and display the ugly number. Ugly numbers are numbers whose only prime factors are 2, 3 or 5 **For example,** if the list's numbers are 80, 81, 10, and 77, the ugly numbers are **80, 81, and 10**. If there are no ugly numbers identified from the given list, then the message will be displayed as "**No ugly numbers found**"

Process flow:

- In the '**main**' method, get the input string as comma separated specified in the sample input statements. (already given in the code template) and append it to the number list.
- Then call the **is_ugly**, pass this number list as an argument and display the ugly number as specified in the sample input and output.
- If there are no ugly numbers in the given list, then display the message "**No ugly numbers 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 method and 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.
- Strictly follow the naming conventions for functions, as specified in the problem description.

Sample Input 1:

Enter the numbers (as comma-separated values):**81,77,99,10**

Sample Output 1:

81
10

Sample Input 2:
Enter the numbers (as comma-separated values):**11,22,33,44,55**

Sample Output 2:
No ugly numbers found

Q24 **Registration Number**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Registration Number(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Coverage:

- Functions
- Collections

Problem Description:

Adler University has planned to conduct a webinar for both; their students and other University students, for which they collected the register numbers of all students. Now the University needs to filter the register numbers of other University students alone. **The Adler University student's register number starts with '7119' and any registration number starts apart from '7119' belongs to other University students.**

Write a program in Python to simulate this.

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

Requirement	**Methods**	**Responsibilities**
Calculate the total cash-back amount	filter_regno(reg_no)	This method takes the **list of register numbers** of all students registered for the webinar as the argument. Iterate this list and filter register numbers of other college students alone in a list. The function '**filter_regno()**' should return a list of register numbers of students who registered from Universities other than Adler University. if no students are found, then **return an empty list.**

Refer to the sample input and output statements for more clarification.

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 no. of students registered for the webinar from the user and then get the registration number of students one by one.
- The register number of students registered for the webinar should be stored in a list and passed this list as the parameter to the function '**filter_regno()**'.
- The register number's values should be of **string type**.

Sample Input 1:

Enter the no. of students registered for the webinar: **5**
Enter the register numbers:
710617104025
711217104086
711916104026
711917106007
717618104078

Sample Output 1:
Register numbers of students from other Universities:
[710617104025,711217104086,717618104078]

Sample Input 2:

Enter the no. of students registered for the webinar:**2**
Enter the register numbers:
7119087
7119e4

Sample Output 2:

Register numbers of students from other Universities: [ ]

Q25 **Scholarships**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Scholarships(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Collections
- Functions

Problem Description:

A student's survey results information is stored in two different strings as comma-separated values. The first string represents all students' ids who are having scholarships and not having scholarships. The second string represents only the students' ids who is having scholarships.

Now the university wants to store them in two different lists and needs to identify the students who do not have scholarships. Write a program in Python to simulate the same.

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

Requirement	**Method Name**	**Responsibilities**
Identify the students who do not have scholarships	'**check_scholarships**(string1, string2)	This method should take two strings (string 1 contains the entire student ids as comma (',') separated values and string 2 contains the students' id who have scholarships) as its arguments. The function should make it two different lists and if the length of the first list is less than the second list, then return the message:" **Invalid data".**

		Otherwise, iterate both the list and identify students who do not have scholarships and append their ids on a new list. After adding all the ids, **return that list** to the caller functions. If all the students have scholarships, then return the message **"All students have scholarships".**

Process Flow:

- In the '**main**' method, get the entire students' ids as a single string separated by ',' (comma) from the user.
- Then get the ids of the students who are having scholarships as a single string separated with a comma (',') from the user.
- Call the '**check_scholarships**' method and pass these two string inputs as arguments.
- Capture the list of values returned from the function and display it as specified in the sample 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.
- 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 1:

117,112,113,114,115,116,111

113,114,115

Sample Output 1:

Students without scholarships: 117,112,116,111

Sample Input 2:

113,114,115

117,112,113,114,115,116,111

Sample Output 1:

Invalid data

Sample Input 2:

117,112,113,114,115,116,111

117,112,113,114,115,116,111

Sample Output 1:

All students have scholarships

Q26 **Dan's Scorecard**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Dan's Scorecard(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Functions
- Collections

Problem Description:

Dan is playing a video game, in which, his character competes in a hurdle race by jumping over hurdles with heights. He used to maintain the maximum heights of units he jumps in each race in his scorecard. But in this scorecard, he can only

append the score one after another. He cannot insert it in the middle or in the beginning. Dan uses this scorecard to maintain the total no. of scores with a split of how many score values are equal to or above 50% of the average score value.

Requirement: Define a function with the name '**calculate_score()**"

Requirement	**Methods**	**Responsibilities**
Find out the total number of score values that are equal to or above 50% of the average score	calculate_score(score_values)	This method takes the **list of score values** as the argument. Iterate this list and find out the total number of score values that are equal to or above 50% of the average score value and return this value to the caller method. For example, if a score card with the size of 4 and with the score values: 3,1,7, and 2, then the total no. of score values that are equal to or above 50% of the average score value is 3 and the function should return this value. If all the score values are '0', then the function should return 0.

Process flow:

- In the '**main**' method, get the size of the scorecard from the user and then get the score values of float type one by one and append it to a list.
- Then call the '**calculate_score**' method and pass the list of score values.
- Display the values returned by the function.

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.

- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input 1:

Enter the size of the score card:**7**

Enter the score values:

10
5
1
4
7
2
4

Sample Output 1:

The score values that are equal to or above 50% of the average score: 5

Sample Input 2:

Enter the size of the score card:3
Enter the score values:
0.2
0.5
1.5

Sample Output 2:

The score values that are equal to or above 50% of the average score: 2

Sample Input 3:

Enter the size of the score card:3
Enter the score values:
0
0

0

Sample Output 3:

The score values that are equal to or above 50% of the average score: 0

Q27 **Analyze comments**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Analyze comments(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts covered:

- Collections
- Functions

Problem Description:

You are a software engineer at a social media company, and you have been asked to build a program that takes in a string containing a user's post and a list of keywords and displays texts that contains one or more of the keywords.

The comments are provided as a single string, with each comment separated by a newline and the username preceding the comment text, separated by a colon.

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

Requirement	Methods	Responsibilities
Analyze the users' comments and find the keywords in their comments	analyze_comments(input_string, keywords)	This method takes an input string and keywords as arguments. splits the input string into a list of lines, iterates over the lines in the list, and splits each line into a username and comment text. Then iterate over the keywords. If any keywords are found in the comment text, display the text.

Process flow:

- In the '**main**' method, input string and keywords are given already.
- Call the method '**analyze_comments**' and pass the input string and keywords as arguments, and display it as specified in the sample input and output.

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, and the rest of the text represents the output.
- Do not alter the given code template. Write your code only in the necessary places.
- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input:

"user1: I love this post!\nuser2: This is a great post!\nuser3: I totally agree with user1\nuser4: This post is amazing!"

["love", "great", "amazing"]

Sample Output:

I love this post!

This is a great post!

This post is amazing!

Q28 **Cash Back Offer**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Cash Back Offer(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Collections
- Functions

Problem Description:

TravelWithUs is a travel credit card issuer that provides various offers for travel-related spending. As a part of the festive season, they planned to give a cash-back offer to their customers based on the credit points they have earned so far. Find below the cash-back offer details:

- Credit point is **50 or above**, then the cash-back offer is **$5 per point.**
- If the credit point is 30 or above and **below 50,** then the cash-back is **$2 per point.**
- If the credit point is **below 30**, then the offer is **$1 per point.**

Based on this information, the online shop needs to find out the total cash-back amount they need to spend for their customers. Write a program in Python to simulate this scenario.

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

Requirement	**Methods**	**Responsibilities**
Calculate the total cash-back amount	calculate_amount(credit_points)	This method takes the **list of credit points** as the argument. Iterate this list and calculate the total cash-back amount for each credit point based on the criteria mentioned and finally returned the total cash-back amount to the caller function.

Process flow:

- In the '**main**' method, get the no. of credit card users and credit points of each user one by one and append it to a list.
- Call the '**calculate_amount**' method and pass the list of credit points as its arguments.
- Capture the cashback amount returned by the function and display it.

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 no. of customers from the user and then get the credit points one by one.
- The credit point values should be of integer type.
- 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 1:

Enter the no. of travel credit card users:**5**
Enter the credit points for user 1 :
10
Enter the credit points for user 2 :
20
Enter the credit points for user 3 :
30
Enter the credit points for user 4 :
40
Enter the credit points for user 5 :
50

Sample Output 1:

Total cash-back amount: 420

Sample Input 2:

Enter the no. of travel credit card users:**3**
Enter the credit points for user 1 :
0
Enter the credit points for user 2 :
-5
Enter the credit points for user 3 :
0

Sample Output 1:

Total cash-back amount: 0

Q29 **Movie Ratings**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Movie Ratings(---RETIRED---)
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Collections
- Functions

Problem Description:

ZEE Film Fare Association released a new movie. They decided to find out whether the movie received the highest or lowest rating feedback from viewers. Help them create an application in Python to find the highest rating using the below mentioned function.

Requirement: Define a function with the name '**check_rating**()"

Requirement	Methods	Responsibilities
Find out the highest rating feedback given by viewers.	check_rating (rating_list)	This method takes the rating list as its argument. If the highest number of ratings is between 0 to 5 (inclusive) then it has to return "**The highest rating is for 0-5**". If the highest number of ratings is between 6 to 10 (inclusive) then it has to return "**The highest rating is for 6-10**". If both ratings are equal then it has to return "**Ratings are equal**". The rating should be between 0 and 10.

Process flow:

- In the '**main**' method, get the number of viewers and the rating from the user and append it to a rating_list.
- If the ratings are not between 0 to 10 display the message as "Invalid Rating" and ignore those ratings.
- Then call the '**check_rating**' method and pass the rating list as the argument.
- Display the values returned by the function.

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.
- Strictly follow the naming conventions for functions as specified in the problem description.

Sample Input :

Enter the number of viewers: **5**

2

3

4

8

9

Sample Output :

The highest rating is for 0-5

Sample Input 2:

Enter the number of viewers: **5**

2

4

7

8

9

Sample Output 2:

The highest rating is for 6-10

Sample Input 3:

Enter the number of viewers: **6**

3

2

5

6

8

9

Sample Output 3:

Ratings are equal

Q30 **Sales Competition**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Sales Competition
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 512 MiB

Concepts Covered:

- Collections
- Functions

Problem Description:

Lotus Marketing Company is one of the best sales firms in the city.
They will select two representatives each quarter and compare their achieved sales targets over a few days to determine which representative will be promoted. for this, they collect the sales details of the two selected representatives, compare their daily sales, and determine the winner. Assist them in determining the winner using the Python program.

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

Requirement	**Methods**	**Responsibilities**
Find the winner of the competition.	find_winner(sales_rep1,sales_rep2)	This method takes two lists of integer numbers as arguments. The first list represents the daily sales achieved by the first representative and the second list represents the daily sales achieved by the second representative.

		Compare the sales achieved by the representatives each day and find count the no. of times each representative had the most sales. If sales_rep 1 has the most winning count, then return **"Sales Representative 1 is the winner"** If sales_rep 2 has the most winning count, then return **"Sales Representative 2 is the winner"** if both have the same number of the winning count then return "**Both are winners**" **Example:** If sales_rep1=**[45, 67, 89]**, sales_rep2=**[34, 56, 90]** The winning count of sales_rep1 is **2**, and the winning count of sales_rep2 is **1.** So it has to return "**Sales Representative 1 is the winner.**"

Process flow:

- In the '**main**' method, get the number of days, and, for each day, get the sales details of both representatives from the user and append that to a separate list.
- Call the '**find_winner**' method and pass the sales_rep1 and sales_rep2 sales lists as its arguments.
- Capture the string returned by the function and display it.

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.
- Strictly follow the naming conventions for variables and functions as specified in the problem description.

Sample Input 1:
Enter the number of days: **5**

Enter daily sales for Sales Representative 1:

21

34

45

67

54

Enter daily sales for Sales Representative 2:

67

43

54

67

56

Sample Output 1:

Sales Representative 2 is the winner

Q31 **Daily Temperature**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Daily Temperature

Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Collections
- Functions

Problem Description:

Mr. Helen decided to develop software to analyze the daily temperature. The software shows the average temperature and the day with the highest temperature from a list of daily temperatures. Help him develop software using a Python program.

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

Requirement	**Methods**	**Responsibilities**
Find the average temperature and the day with the highest temperature.	find_average_temperature(input_string)	This method takes a comma-separated input string as an argument. The string contains daily records of temperature. Split this string based on the "," (comma) separator and iterate the list to find the average temperature and also find the day with the highest temperature. The average temperature should be calculated as, **Average_temperature = total temperature/length of the list.** Finally, the function should return a list that should contain the average temperature and the day with the highest temperature. **For example:** if the temperatures are 24,32,31,28,35 then the average temperature is 30.0, and the highest temperature day is 5. So it has to return [30.0,5].

		Note: • The average temperature should be in 2 decimal places. • If two or more days contain the highest temperature, then consider the first day with the highest temperature.

Process flow:

- In the '**main**' method, get the temperature of each day as a string separated by commas (',').
- Call the '**find_average_temperature**' method and pass the input_string as its arguments.
- Capture the value returned by the function and display it as per the sample input and output statement.

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.
- Strictly follow the naming conventions for variables and functions as specified in the problem description.

Sample Input 1:
Enter the temperatures of each day, separated by commas:**31,33,35,37,28,33,41,36**

Sample Output 1:

The average temperature is: 34.25

The day with the highest temperature is: 7

Q32 **Calorie Requirement Calculation**

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Based on: Calorie Requirement Calculation
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum memory used**: 320 MiB

Concepts Covered:

- Collections
- Functions

Problem Description:

The health care department of XYZ hospital wants to provide its patients with information on their calorie requirement to maintain weight, based on age, gender, height, weight, and particularly their activity level. Write a Python program that will perform the calorie requirement calculation.

Get the gender and activity level from the user as a comma-separated string.
Get the age, height, and weight from the user as a comma-separated string.

Requirement: Define a function with the name '**calculate_calories()**'.

<table>
<tr><th>Requirement</th><th>Methods</th><th>Responsibilities</th></tr>
<tr><td>Calculate the calorie requirement.</td><td>calculate_calories (input_string1_list, input_string2_list)</td><td>This method takes input_string1_list (gender and activity level) and input_string2_list (age, height in cm, and weight in kg) as its arguments.

Calculate the calories required to maintain weight based on below mentioned information and return the same.

• Gender can take the value - male or female
• Activity level can take the value - sedentary or moderately active or extra active
• Age is given in number, height in cm, and weight in kg.

<table><tr><th>Activity level</th><th>Value</th></tr><tr><td>sedentary</td><td>1.2</td></tr></table></td></tr>
</table>

		<table><tr><td>moderately active</td><td>1.55</td></tr><tr><td>extra active</td><td>1.9</td></tr></table> **Formula :** For a "**male**", calorie= ((10*weight)+(6.25*height)-(5*age)**-161)***activity level value For a "**female**", calorie= ((10*weight)+(6.25*height)- (5*age)**+5**)*activity level value **For example,** a 23-year-old female, activity level sedentary with a height of 153 cm and a weight of 60 kg, will require 1735.5 calories per day. **calorie= ((10*60)+(6.25*153)-(5*23)+5)*1.2= 1735.5**

Process flow:

- In the '**main**' method, get the input_string1 (gender and activity level), and input_string2 (age, height, weight) as comma-separated values from the user and convert that into the list (Refer to the sample input statements).
- Then call '**calculate_calories**', pass the input_string1_list and input_string2_list as arguments, capture the calories returned from this method, and display the same as specified in the sample output.

The main method is excluded from the evaluation. You are free to write your own code in the main method to invoke the business method and 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.
- Strictly follow the naming conventions for functions, as specified in the problem description.

Sample Input 1:

Enter the gender and activity level (as comma-separated values): **female, sedentary**
Enter the age, height, and weight (as comma-separated values): **23,153,60**

Sample Output 1:

To maintain your current weight, you'll need 1735.5 calories per day