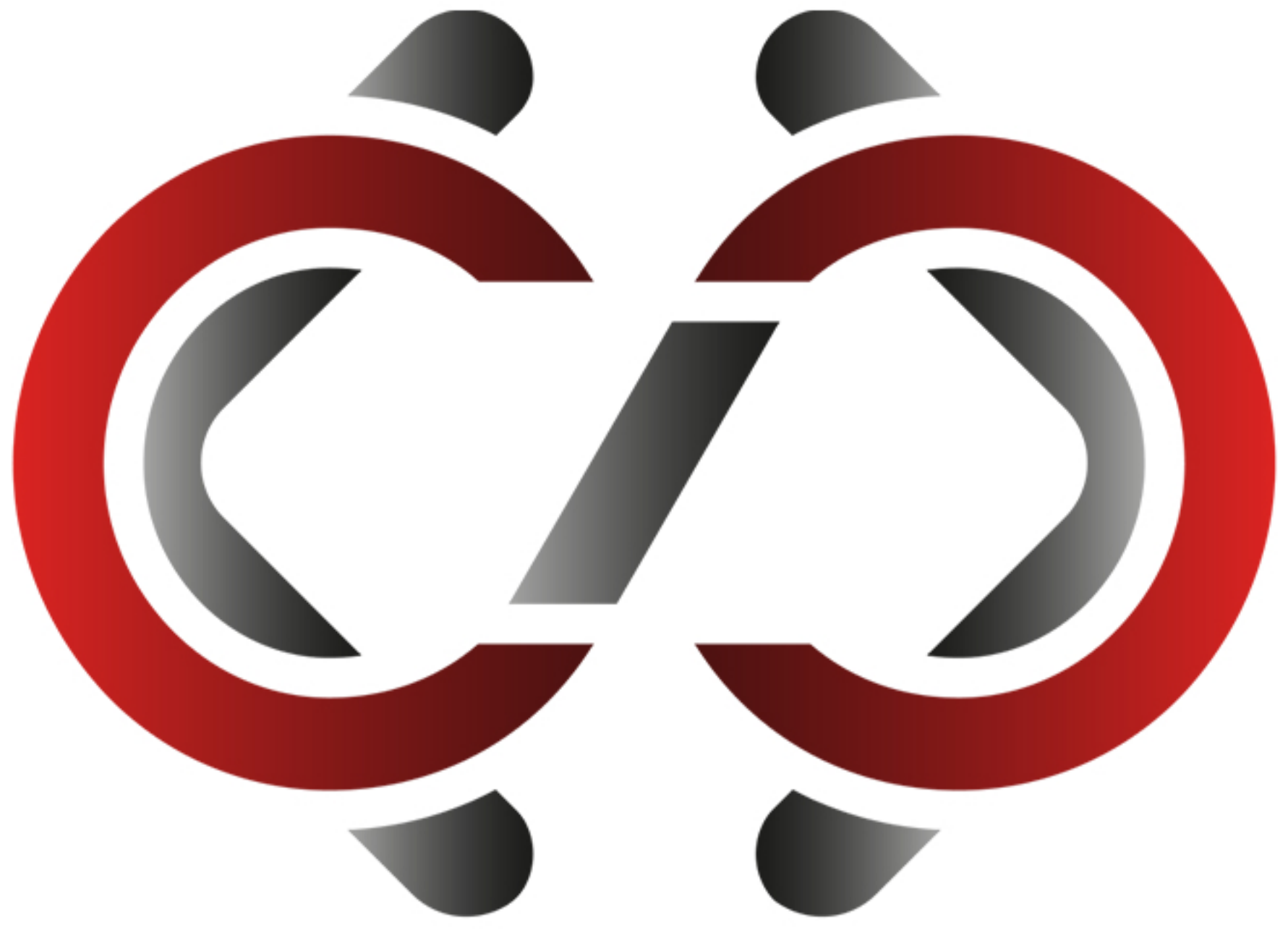




CODE CHALLENGE
CHAMPIONSHIP



Python

s e n i o r

H a n d b o o k



Code Challenge Championship



Mission 1

Customer Satisfaction Analyzer

Customer Satisfaction Analyzer

Overview:

A large fast-food restaurant, "**CrispyTrips**", regularly collects customer satisfaction data through surveys. They want to automate the analysis of this data to understand trends in customer satisfaction and identify areas for improvement.

Mission:

- 1-** Make a function named "**analyze_satisfaction**" that has one parameter named "**ratings**" of type list, where each rating is an integer between 1 and 5.
 - 2-** This function should utilize this rating value to analyze and print the following:
 - The **average** customer satisfaction rating.
 - The **median** customer satisfaction rating.
 - 3-** If the user doesn't enter "**stop**" to exit the program, the program should be continuously asking to enter the rating value to be inserted in a **list** before calculating the requested insights.
 - 4-** Additionally, you should handle any invalid entries made by the restaurant admin, such as negative values or ratings outside the accepted range of 1 to 5.
- As a bonus part**, extend the function to categorize the ratings into three groups which are; "**Low**" (1-2), "**Medium**" (3), "**High**" (4-5) Count the number of ratings in each category and print out the percentages.

Useful Hints:

- ✓ Utilize the **built-in statistics module** functions like `mean()`, and `median()` to perform the calculations.
- ✓ Use the **`range()`** class to easily categorize the ratings.
- ✓ Create an empty list and use the **`append()`** method to add each rating to this list in order to perform the requested calculations.

Expected Output:

Example of 10 customers ratings entered with the bonus part

```
Enter rating 1 or type 'stop' to end: 5
Enter rating 2 or type 'stop' to end: 4
Enter rating 3 or type 'stop' to end: 1
Enter rating 4 or type 'stop' to end: 6
Invalid rating. Please enter a value between 1 and 5.
Enter rating 4 or type 'stop' to end: 5
Enter rating 5 or type 'stop' to end: 4
Enter rating 6 or type 'stop' to end: 2
Enter rating 7 or type 'stop' to end: -4
Invalid rating. Please enter a value between 1 and 5.
Enter rating 7 or type 'stop' to end: 3
Enter rating 8 or type 'stop' to end: 3
Enter rating 9 or type 'stop' to end: 5
Enter rating 10 or type 'stop' to end: 2
Enter rating 11 or type 'stop' to end: stop
```

```
Average rating: 3.40
Median rating: 3.50
Low ratings (1-2): 3 (30.00%)
Medium ratings (3): 2 (20.00%)
High ratings (4-5): 5 (50.00%)
```

Bonus part



Mission 2

Managing Library Books

Managing Library Books

Overview:

A public library named "**BookNook**" is looking to modernize its book management system which is currently done manually. They aim to create a basic system using OOP where users can easily add, view, search, and remove books from the library's collection.

Mission:

1- Create a class named "**LMS**" which is the abbreviation of the library management system. This class will contain the constructor method:

- **__init__()**: containing an empty list to store the library books.

2- Furthermore, this class should contain the following methods:

- **add_book()**: Adds a new book to the library with the specified title and author.
- **display_books()**: Displays all the books currently in the library, showing the title and author for each book.
- **search_book()**: Searches for a book by its title. If found, it displays the book's details; if not, it informs the user that the book is not in the library.
- **delete_book()**: Deletes a book from the library by its title, after confirming the deletion with the user.
- **update_book()**: Updates the details of an existing book, allowing changes to the title or author.

3- Finally, create a simple user interface to interact with the "**LMS**" class, allowing you to add, display, search, and delete books by using the previous methods. The program will continuously prompts the user for actions until they choose to exit.

As a **bonus part**, make another method named **update_book()** which updates the details of an existing book, allowing changes to the title or author.

Useful Hints:

- ✓ Use a **dictionary** to store each book's title and author, with keys like {"title": "Book Title", "author": "Author Name"}.
- ✓ Utilize a **while loop** to keep the program running and continue prompting the user for actions until they choose to exit.
- ✓ Use **match statement** to effectively match the user options using numbering cases.

Expected Output:

Example of adding and displaying a new book

```
Library Menu:
1. Add a Book
2. Display Books
3. Search for a Book
4. Delete a Book
5. Update Book Data
6. Exit
Select an option (1-6): 1
Enter the book title: Zen of Python
Enter the book author: Tim Peters
Book added successfully.

Library Menu:
1. Add a Book
2. Display Books
3. Search for a Book
4. Delete a Book
5. Update Book Data
6. Exit
Select an option (1-6): 2

Library Books:
Title: Zen of Python, Author: Tim Peters
```

Example of searching or deleting an unfounded book

Library Menu:

1. Add a Book
2. Display Books
3. Search for a Book
4. Delete a Book
5. Update Book Data
6. Exit

Select an option (1-6): 3

Enter the title of the book to search: Python for Beginners

Book not found in the library.

Library Menu:

1. Add a Book
2. Display Books
3. Search for a Book
4. Delete a Book
5. Update Book Data
6. Exit

Select an option (1-6): 4

Enter the title of the book to delete: Python book

Book not found in the library.

Example of updating the stored book (bonus part)

Library Menu:

1. Add a Book
2. Display Books
3. Search for a Book
4. Delete a Book
5. Update Book Data
6. Exit

Select an option (1-6): 5

Enter the title of the book to update: Zen of Python

Enter the new title (or press Enter to keep current title): Python Zens

Enter the new author (or press Enter to keep current author): Tim Peters

Book updated successfully.



Mission 3

Paint Coverage Estimator

Paint Coverage Estimator

Overview:

A paint store named "**ColorWorks**" wants to offer a helpful tool for customers planning to paint their walls. To enhance their shopping experience and provide a cost estimate, they need a program that displays different paint types and helps customers calculate the total cost based on their paint choice and wall dimensions.

Mission:

1- Create a function named "**paint_estimator**" that takes 3 parameters: the "**paint_type**" of type string, the "**width**" and "**height**" of type float for the wall dimensions in meters.

2- For the paint type, it can be one of the following with its corresponding price in EGP/gallon:

- Regular Paint (Covers 50 m²/gallon) → **1210 EGP/gallon**
- Premium Paint (Covers 80 m²/gallon) → **1490 EGP/gallon**
- Deluxe Paint (Covers 120 m²/gallon) → **1580 EGP/gallon**

3- The function should **calculate the area of the wall(s)** that the customer wants to paint. Then, **determine the number of gallons used**, and finally **calculate the total cost**. So, the final estimated cost will be calculated as the following:

- (a) Calculating the area of the wall(s) → **width x height**
- (b) Identifying the gallons used → **area / paint coverage**
- (c) Calculating the estimated cost → **gallons used x price per gallon**

4- Afterwards, display the paint options and **continuously ask customers** to enter their choice of paint type and wall dimensions as long as they don't enter "**done**" to calculate the total cost.

5- Handle invalid cases as checking for **invalid paint types, missing inputs, non-positive wall dimensions**.

As a **bonus part**, make a counter that **counts the number of customers** that will use the program until the program is exited.

Useful Hints:

- ✓ Use a **dictionary** to store the **paint types as keys** and their **respective prices as values** to easily access prices in the function.
- ✓ Initialize the **counter variable** outside the loop to avoid resetting its value after each iteration.
- ✓ Use the **lower()** or **upper()** string methods to handle any string input entered by the user.

Expected Output:

Example of 2 customers used the program

```
Welcome to the Paint Coverage Estimator!
Regular Paint -> 1210.00 EGP/gallon
Premium Paint -> 1490.00 EGP/gallon
Deluxe Paint -> 1580.00 EGP/gallon

Enter (Regular/Premium/Deluxe) or "done" to quit: Regular
Enter the width of the wall in meters: 5
Enter the height of the wall in meters: 8
You will need 0.80 gallons of Regular Paint to cover a wall of 40.00m^2 The estimated cost is 968.00 EGP

Enter (Regular/Premium/Deluxe) or "done" to quit: Deluxe
Enter the width of the wall in meters: 12.5
Enter the height of the wall in meters: 15.25
You will need 1.59 gallons of Deluxe Paint to cover a wall of 190.62m^2 The estimated cost is 2509.90 EGP

Enter (Regular/Premium/Deluxe) or "done" to quit: done
2 customers used the program before exiting..
Thank you for using the Paint Coverage Calculator!
```

Handling invalid cases as choosing invalid paint type, etc.

```
Welcome to the Paint Coverage Estimator!
Regular Paint -> 1210.00 EGP/gallon
Premium Paint -> 1490.00 EGP/gallon
Deluxe Paint -> 1580.00 EGP/gallon

Enter (Regular/Premium/Deluxe) or "done" to quit: Ultra
Invalid paint type, please try again!

Enter (Regular/Premium/Deluxe) or "done" to quit:
You should choose an option or done to quit!

Enter (Regular/Premium/Deluxe) or "done" to quit: Premium
Enter the width of the wall in meters: 12.9
Enter the height of the wall in meters: -6.5
Width and height must be greater than 0

Enter (Regular/Premium/Deluxe) or "done" to quit: Done
0 customers used the program before exiting..
Thank you for using the Paint Coverage Calculator!
```