# Project: - Car Showroom

* **AIM: -** To Automate Car Manufacturing Plant for Their Automotive Industry-Based Company

## **INTRODUCTION: -**

This code defines a program that helps users explore and learn about different types of cars based on their transmission preference. There are four classes: **M01, M02, M03, and M04** each representing a specific type of car.

**M01, M02, M03, and M04** These four classes **Inherited** by **Car class.**

The **Car class** have two attributes like **Airbags and Number of wheels**

The four classes have various attributes like **model name, engine type, seating capacity, horsepower, safety features, and more**. Users can choose between Manual and Automatic transmission types.

If they select Manual, they can then choose the color of the car and get details about a **Manual Gasoline Car**. If they select **Automatic**, they can further choose between **Gasoline, Electric, and Hybrid engines**, pick a **color**, and learn about the specific car's features.

The code uses exception handling **to ensure that user inputs are valid**. If a user provides an invalid transmission type or engine type, the program will guide them to enter correct values.

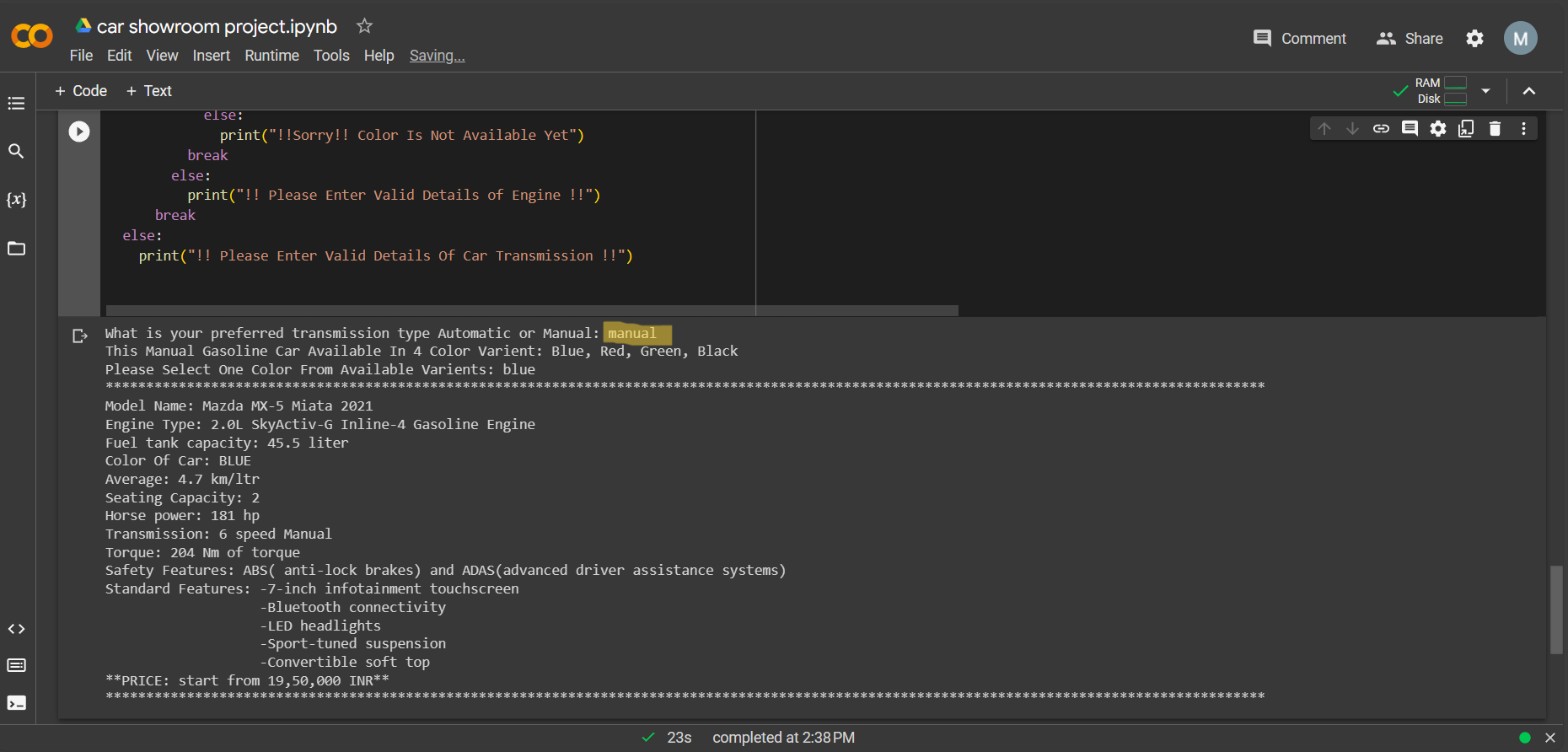
Similarly, when selecting a car color, the code checks if the chosen color is valid; otherwise, it prompts the user to select a valid color. Exception handling ensures that the program doesn't crash due to unexpected inputs and provides a smooth user experience.

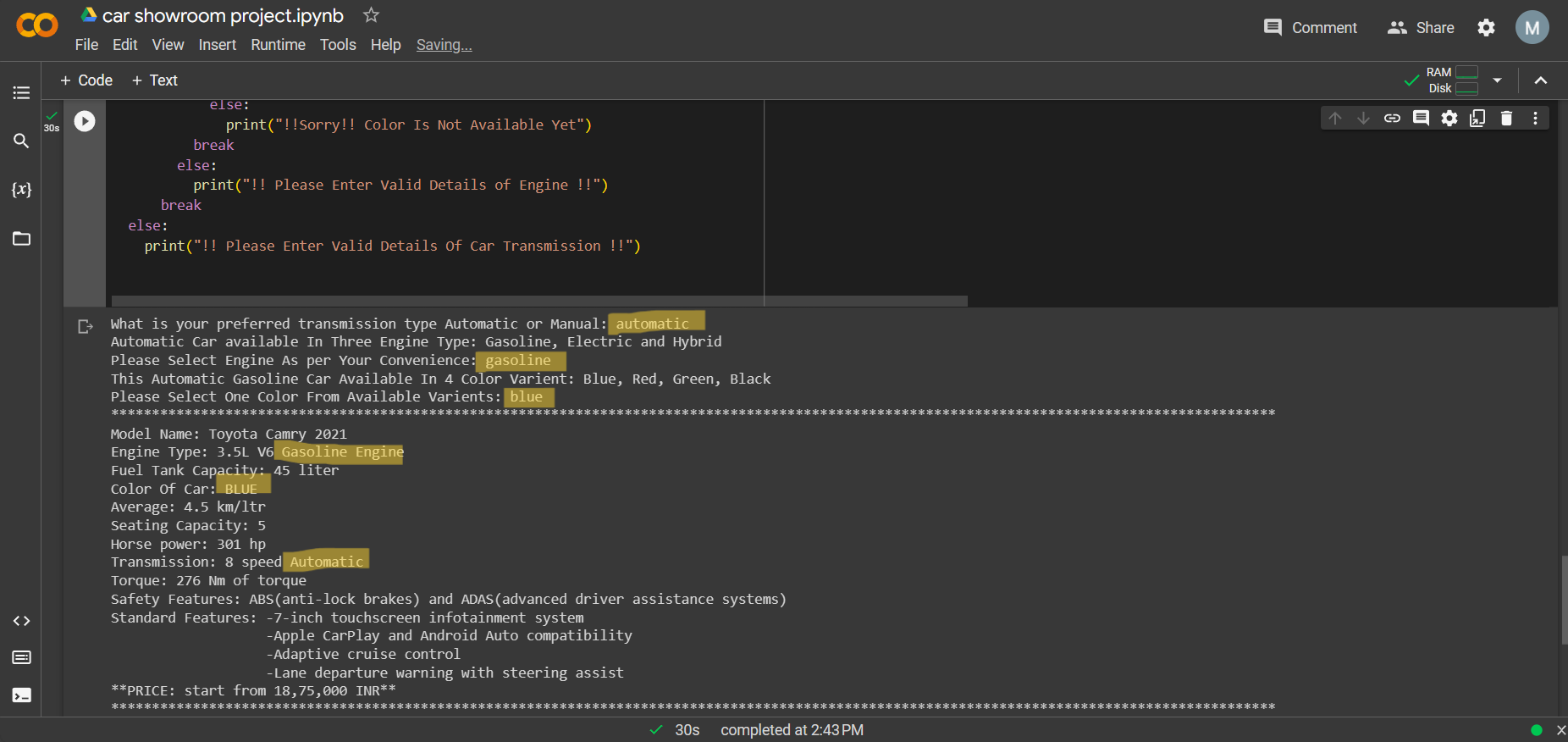
Overall, the code is a user-friendly tool to explore different car options based on transmission preferences. It offers informative details about various cars and handles user inputs gracefully through exception handling, making the interaction seamless and enjoyable

## **Unit Tests: -**

**Unit Test 1: -** When a user inputs the Transmission and Color (in the case of an Automatic Car's Engine) of a Car, even if the user enters 'manual' or 'MAnual' instead of 'Manual', the code should still allow the user to proceed to the next steps.

The same flexibility should apply to the Color input and the Automatic Engine input. as illustrated in the figure.





**Unit Test 2: -** When a user inputs the Transmission and Color (in the case of an Automatic Car Engine) of a Car, the input should be correct in terms of spelling.

For example, if the user enters 'mnaual' instead of 'manual', the code should not allow the user to proceed to the next steps. Instead, the code should attempt to correct the spelling and provide a statement guiding the user to **input valid data** or indicating that the **input is not available**.

The same process should apply to Automatic Cars (Transmission, Color, and Engine). The code will print statements for user understanding, as illustrated in the figure.

