**Exercise 4: Creating Functions and Classes in Python**

**Objective:**

Learn how to create and use functions and classes in Python to organize code, encapsulate behavior, and implement object-oriented principles.

**Part 1: Creating Functions in Python**

A screenshot of a computer

Description automatically generated

**1.Write a Function to Calculate the Area of a Rectangle**: Create a function called calculate area that takes two arguments: the **length** and **width** of a rectangle. The function should return the area of the rectangle.

**Code Summary:**

For this challenge I created a Python function, calculate area that takes two parameters, length and width and uses those two to calculate the area of a rectangle.

**Key Takeaways:**

Learned how to declare a function with parameters. Understand how to utilize the return statement with a calculation. How to pass values to a function and print the output.

**Output:**

That program calculated the area of a rectangle with dimensions 9 by 3 as 27.

**2.Write a Function to Check if a Number is Prime**: Create a function called is prime that takes one argument, a positive integer n, and returns True if the number is prime, otherwise False.

**Code Summary:**

In this exercise, I implemented the is prime function to check whether a given number is prime. The function first eliminates numbers less than or equal to 1 and uses a loop to check divisibility. The loop iterates only up to the square root of the number for improved efficiency. If any divisor is found, the function returns False; otherwise, it returns True.

**Key Takeaways:**

The mathematical concept learned was about prime numbers and divisors. Optimized the algorithm by considering the square root method to avoid unnecessary iteration. Practiced flow control with the help of loops and if statements.

**Output:** The program confirmed that the number 9 is prime by returning False.

**3.Write a Function to Count the Occurrences of Each Character in a String**: Create a function called char count that takes a string s and returns a dictionary where the keys are the characters, and the values are the number of times each character appears in the string.

**Summary:**

This exercise involved writing a function to count the occurrences of each character in a string. Using a dictionary, the function efficiently stores and updates counts for each character while iterating through the string. I learned:

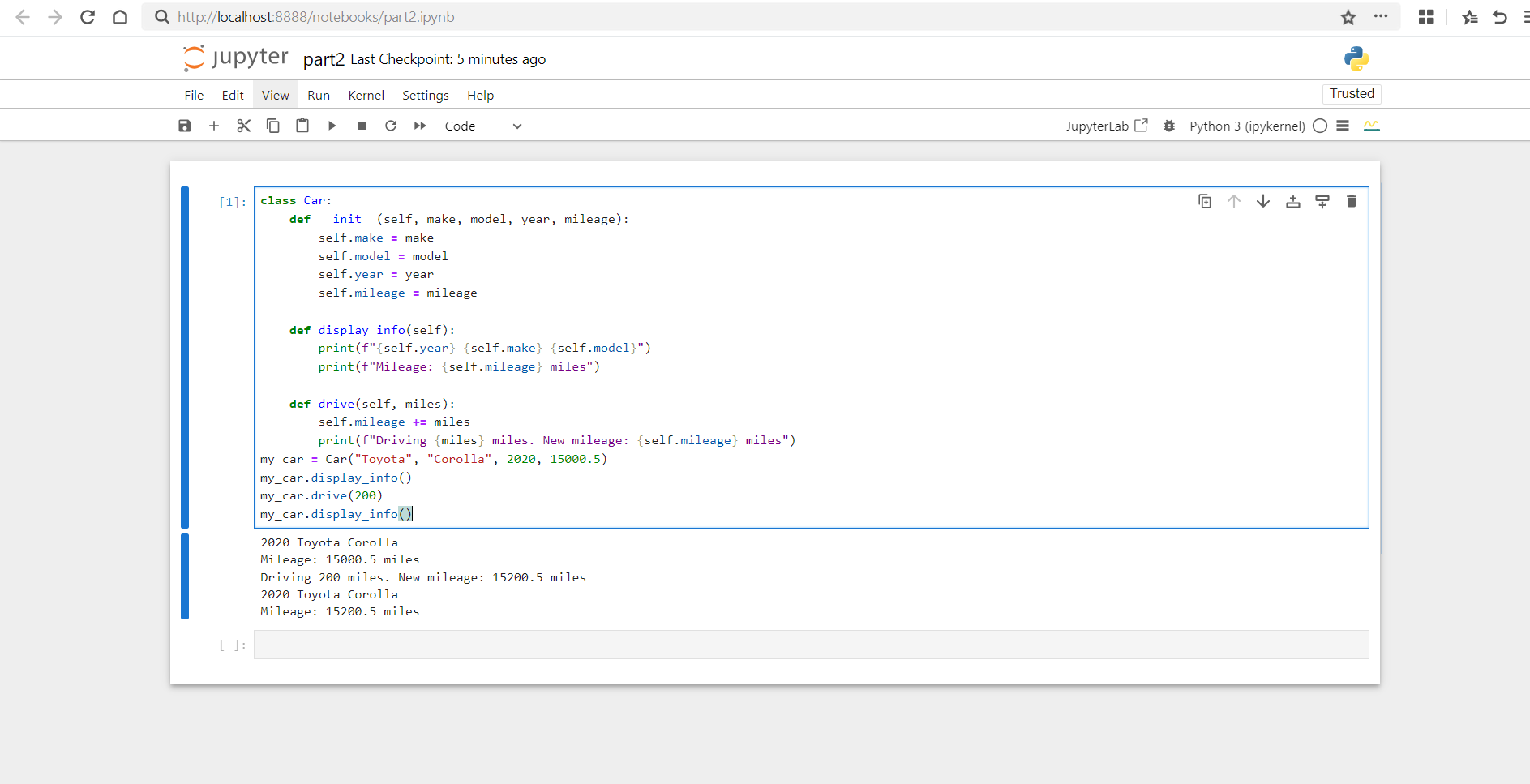
* How to use dictionaries for mapping keys (characters) to values (counts).
* How to handle repetitive tasks using conditionals and loops.
* The importance of accounting for spaces and case sensitivity in strings.

**Output:**

The function was successfully tested with the string "Python Program".

{'P': 2, 'y': 1, 't': 1, 'h': 1, 'o': 2, 'n': 1, ' ': 1, 'r': 2, 'g': 1, 'a': 1, 'm': 1}

**Part 2: Creating Classes in Python**



**1.Create a Car Class**: Define a class called Car. The Car class should have the following attributes:

* + make (string): The make of the car (e.g., "Toyota").
  + model (string): The model of the car (e.g., "Corolla").
  + year (integer): The manufacturing year of the car (e.g., 2020).
  + mileage (float): The current mileage of the car (e.g., 15000.5 miles).

The class should also have the following methods:

* + display\_info: Prints out the car's details (make, model, year, mileage).
  + drive: Takes an argument miles (the number of miles driven) and increases the car's mileage by that amount.

**Summary:** This exercise demonstrated how to create and use classes in Python to represent objects. By defining a Car class. I learned below things.

**Encapsulation:** Using attributes to store object data (make, model, year, mileage).

**Methods:** Implementing actions (display\_info, drive) that operate on the object’s attributes.

**Object-Oriented Programming:** How to create and manipulate objects using constructors (\_init\_).

**Practical Use:** How class methods allow us to change an object's state, something like updating the mileage itself: It successfully tested a Toyota Corolla, driven for 200 miles and then printed out the updated mileage.

A screenshot of a computer

Description automatically generated

**2.Create a Book Class**: Define a class called Book. The Book class should have the following attributes:

* + title (string): The title of the book (e.g., "To Kill a Mockingbird").
  + author (string): The author of the book (e.g., "Harper Lee").
  + year (integer): The year the book was published.
  + price (float): The price of the book.

The class should also have the following methods:

* + display\_info: Prints the book's title, author, year, and price.
  + apply\_discount: Takes a percentage (e.g., 0.1 for a 10% discount) and applies it to the price of the book.

**Overview:** The practice taught me how to declare and work with classes in Python to model objects and their behaviors. In the Book class, I was taught the following:

**Object Initialization:** How to set up attributes like title, author, year, and price.

**Methods for Functionality:**

**display\_info:** To show details of the object

**apply\_discount:** To change the state of an object (price) at runtime

**Encapsulation and Reusability:** Encapsulating logic within a class makes it reusable and maintainable.

Successfully tested book titled To Kill a Mockingbird, applied a discount of 10%, and saw its updated price.