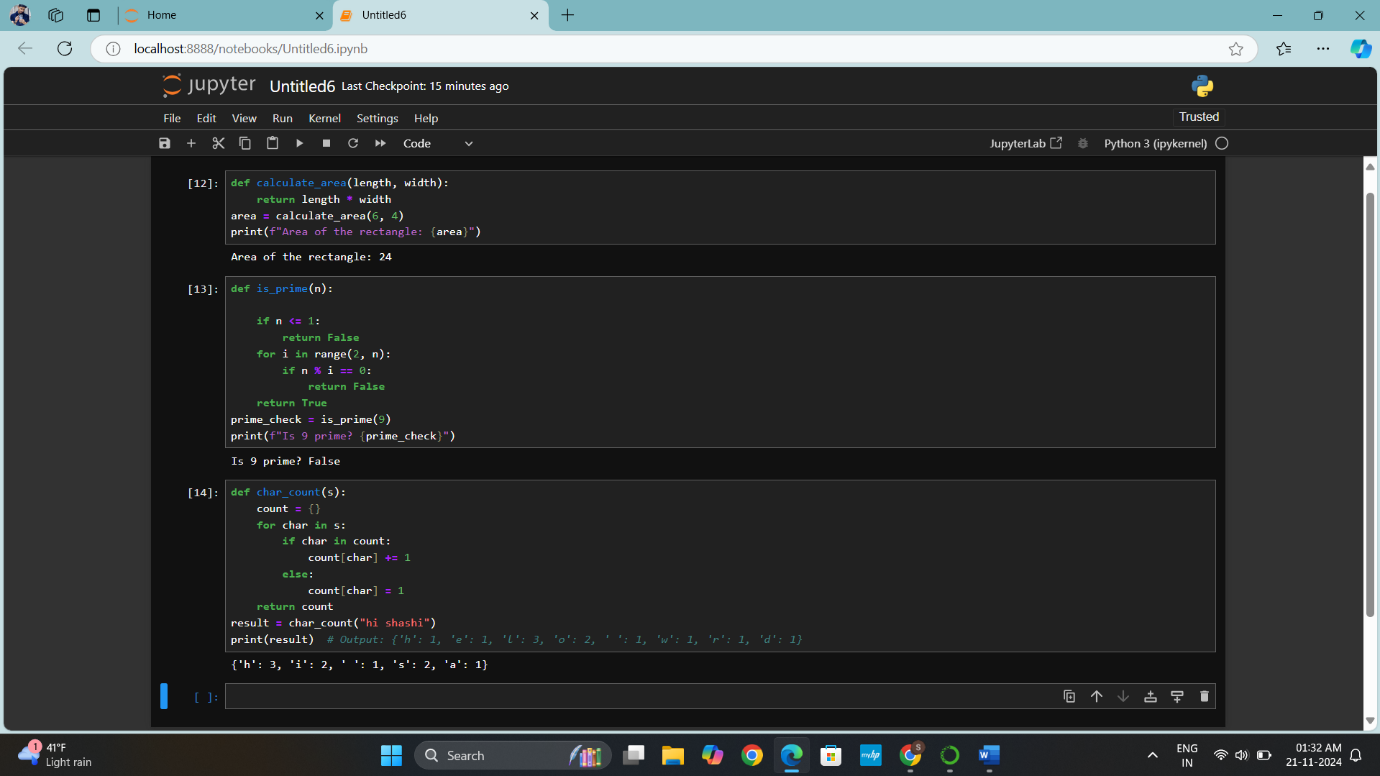
**Exercise 4: Creating Functions and Classes in Python**

**Objective:**

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

**Part 1: Creating Functions in Python**



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.

**Summary**

Calculate Area of a Rectangle Function

**Code Summary:**

This exercise involves the writing of a Python function, calculate area, to find the area of a rectangle. The function accepts two arguments, length and width, multiplies both, and returns the result.

**Key Takeaways:**

How to declare a function with parameters.

How to use the return statement for a calculated outcome.

Passed values to a function to print their values.

**Output:**

Program calculated the area of a rectangle with 6x4 dimensions as 24.

1. **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.

**Summary**

Title: Prime Number Check Function

**Summary of the code:**

For this exercise, I have implemented the function is prime which checks whether a number is prime or not. It first eliminates numbers less than equal to 1 and then uses a loop for checking divisibility. The loop runs up to the square root of the number for efficiency. In case any divisor is found, the function will return False; otherwise, True.

**Key Points to Remember:**

Knew prime numbers and the concept of divisors.

Optimized the algorithm using square root to avoid extra iterations

Practice with control flow loops and if statements.

**Output:**

The program confirmed that the number 9 is prime by returning True.

1. **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**

Title: Character Count Function

**Summary of Code**:

The char count function takes a string and counts the frequency of its characters into a dictionary. Every time a character is encountered, it is added to the dictionary as a key with its frequency as its value. It iterates over a given string and updates the dictionary accordingly.

**Key Takeaways:**

Learned how to store data dynamically by using a dictionary.

Improved iteration over strings.

Understood how conditionals may be used for repetitive tasks.

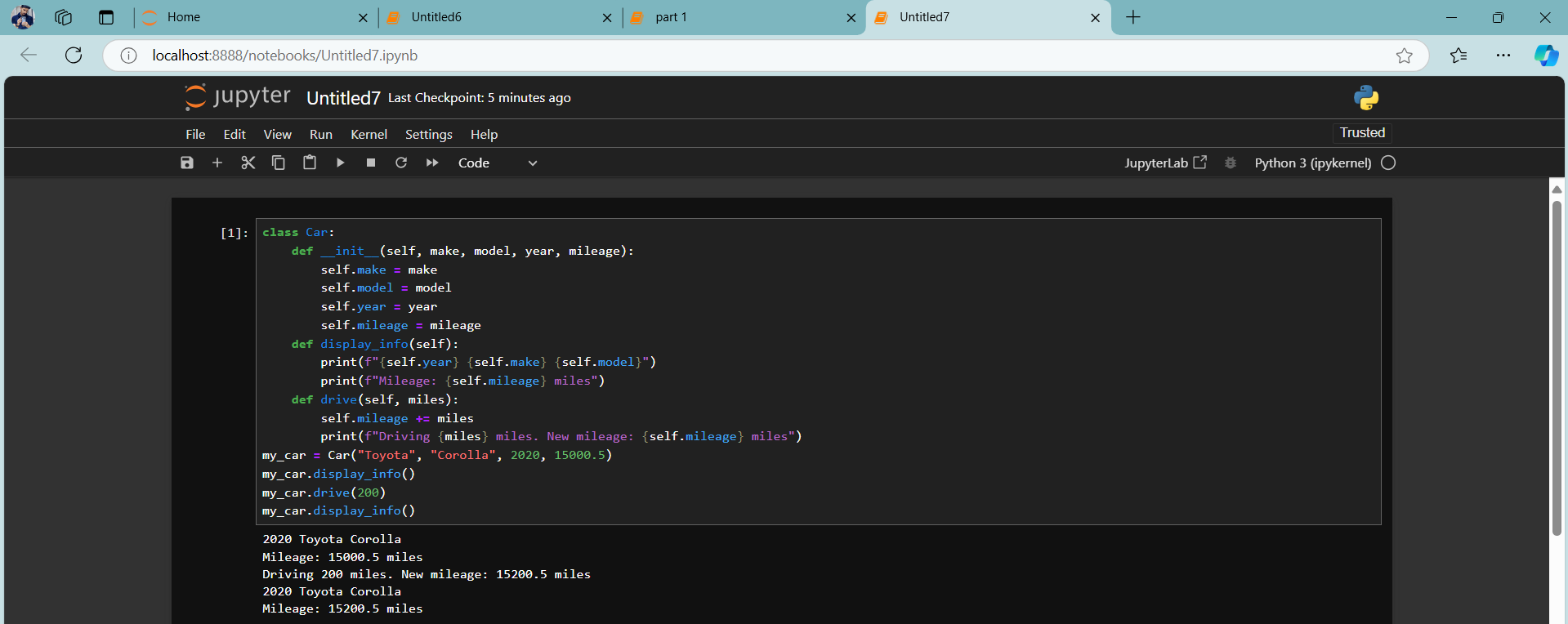
The function correctly counted the occurrences of each character in "hi Shashi", as was expected with the **output:**

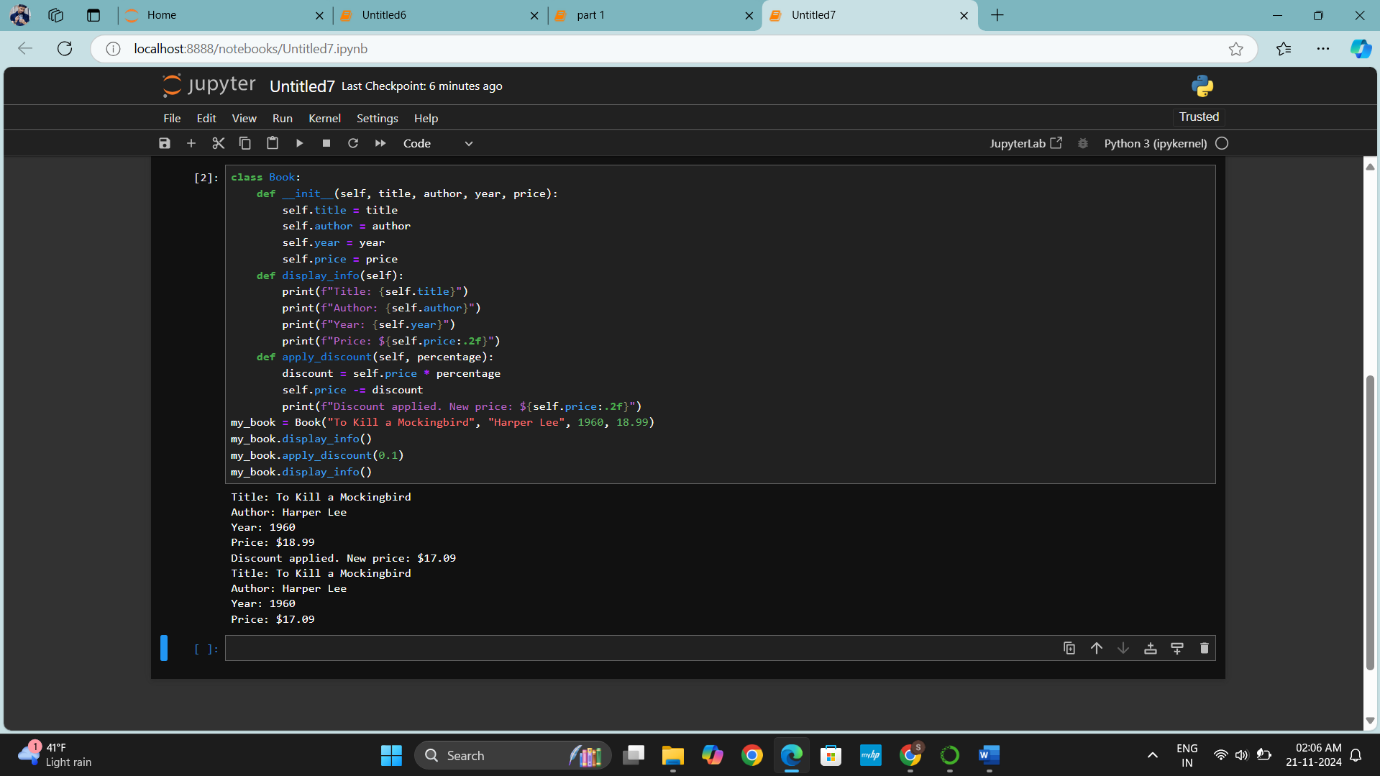
{'h': 3, 'i': 2, ' ': 1, 's': 2, 'a': 1}.

**Part 2: Creating Classes in Python**

**Objective:**

Learn how to create and use classes, define attributes, and implement methods to represent objects and their behaviours.





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**

Title: Car Class Implementation

**Summary:**

This lab taught how to create and utilize classes in Python programming to model objects. I have defined a Car class in this lab and learned the following:

Encapsulation: Attributes are used to retain information about the object (make, model, year, mileage).

Methods: Actions, such as display info and drive, that manipulate the object's attributes.

Object-Oriented Programming: The use of constructors (\_\_init\_\_) while creating and manipulating objects.

Practical Use: Demonstrate how class methods can modify an object's state and update the mileage.

Program successfully tested on Toyota Corolla driven for 200 miles, with updated mileage displayed.

1. **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.

**Summary**

Title: Book Class Implementation

**Summary:**

This lab taught me how to define and manipulate classes in Python to model objects and their actions. Using the Book class, I learned the following:

Object Initialization: How to establish attributes such as title, author, year, and price.

Methods for Functionality:

display info: Shows details of the object.

apply discount: Alters the state of the object (price) at runtime.

Encapsulation+: The encapsulation of logic within a class makes it reusable and maintainable.

I have tested a book named To Kill a Mockingbird, applied a 10% discount on it, and checked its updated price.