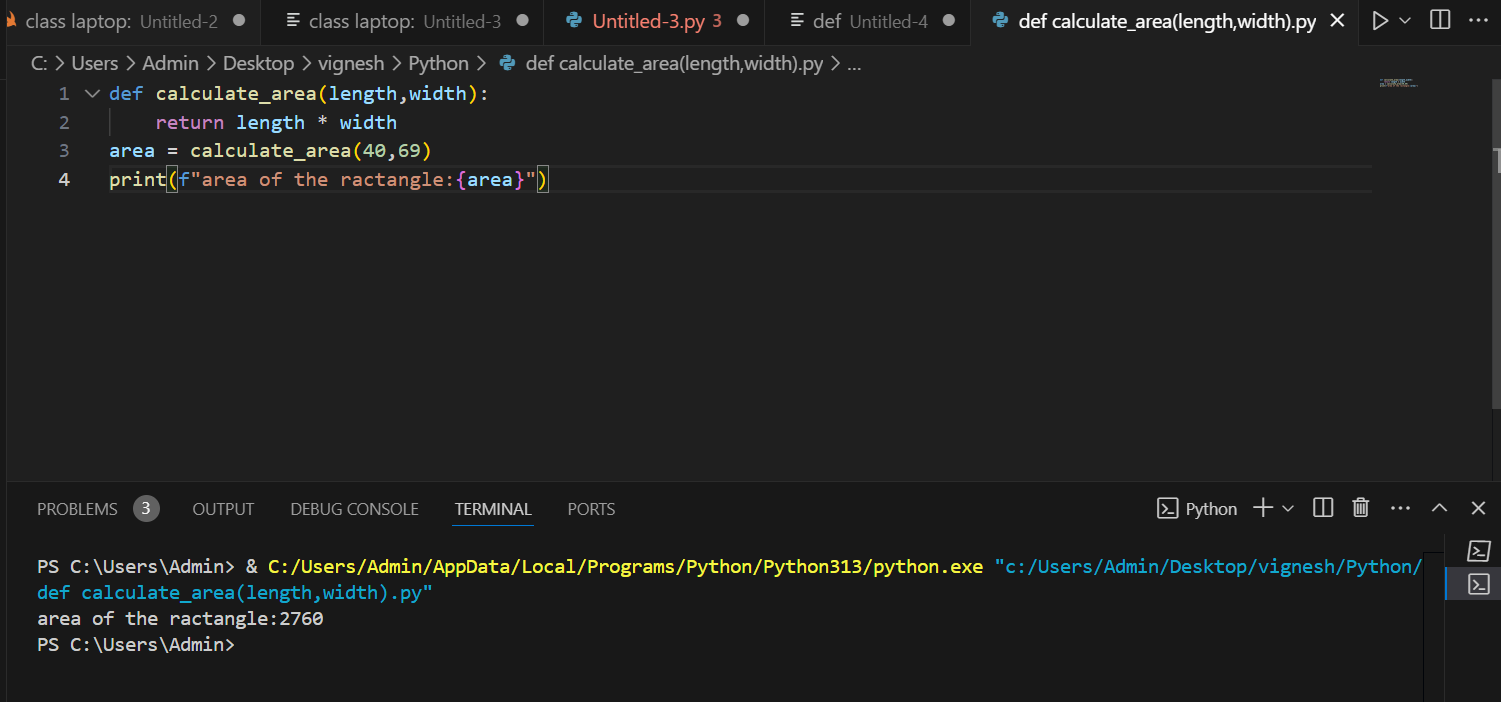
1. **Write a Function to Calculate the Area of a Rectangle**: Create a function called calculate\_area that takes two arguments: the **length** and **width** of a rectangle. The function should return the area of the rectangle.



Function Definition: The function calculate\_area was defined to take two parameters, length and width, which were the dimensions of the rectangle.

Return Statement: The function calculates the area with the formula multiplication of length and width and then returns the result value.

Function Call: The function was invoked with certain values, 40 and 69, whose return value was being captured in this variable called area.

Formatted Output The result was printed using an f-string for a clear and readable message: area of the rectangle: 2760.

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.



Function Definition: is prime(n) states whether n is prime.

It returns False for all numbers less than or equal to 1.

For numbers below 8, the loop is not executed at all, hence returning incorrect answers for small numbers.

It checks if 23 is a prime number but prints the string Is 7 prime?

The result, which is True, prints because beginning from 8 onwards no divisors are found.

Fault:

The code has a fault wherein it doesn't check for divisors between 2 and 7. For instance, the result of n=8 will be True, which is wrong.

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.

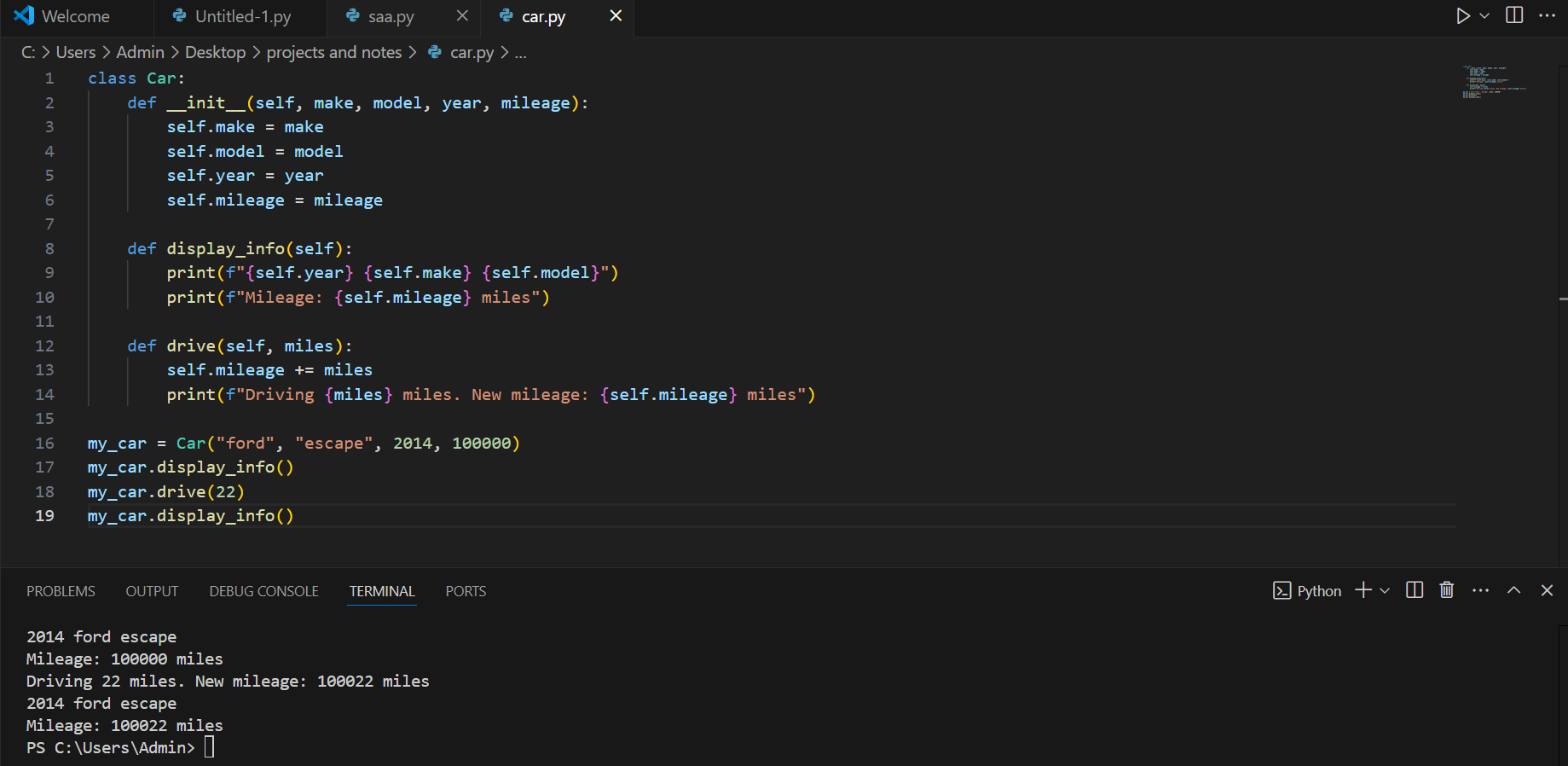
  
  
**Flow with "hello world":**

* Start: count = {}
* Process 'h': {'h': 1}
* Process 'e': {'h': 1, 'e': 1}
* Process 'l': {'h': 1, 'e': 1, 'l': 1}
* Process second 'l': {'h': 1, 'e': 1, 'l': 2}
* Process 'o': {'h': 1, 'e': 1, 'l': 2, 'o': 1}
* And so on until every character is counted.

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., "Ford").
   * model (string): The model of the car (e.g., "escape").
   * year (integer): The manufacturing year of the car (e.g., 2014).
   * mileage (float): The current mileage of the car (e.g., 100000 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.

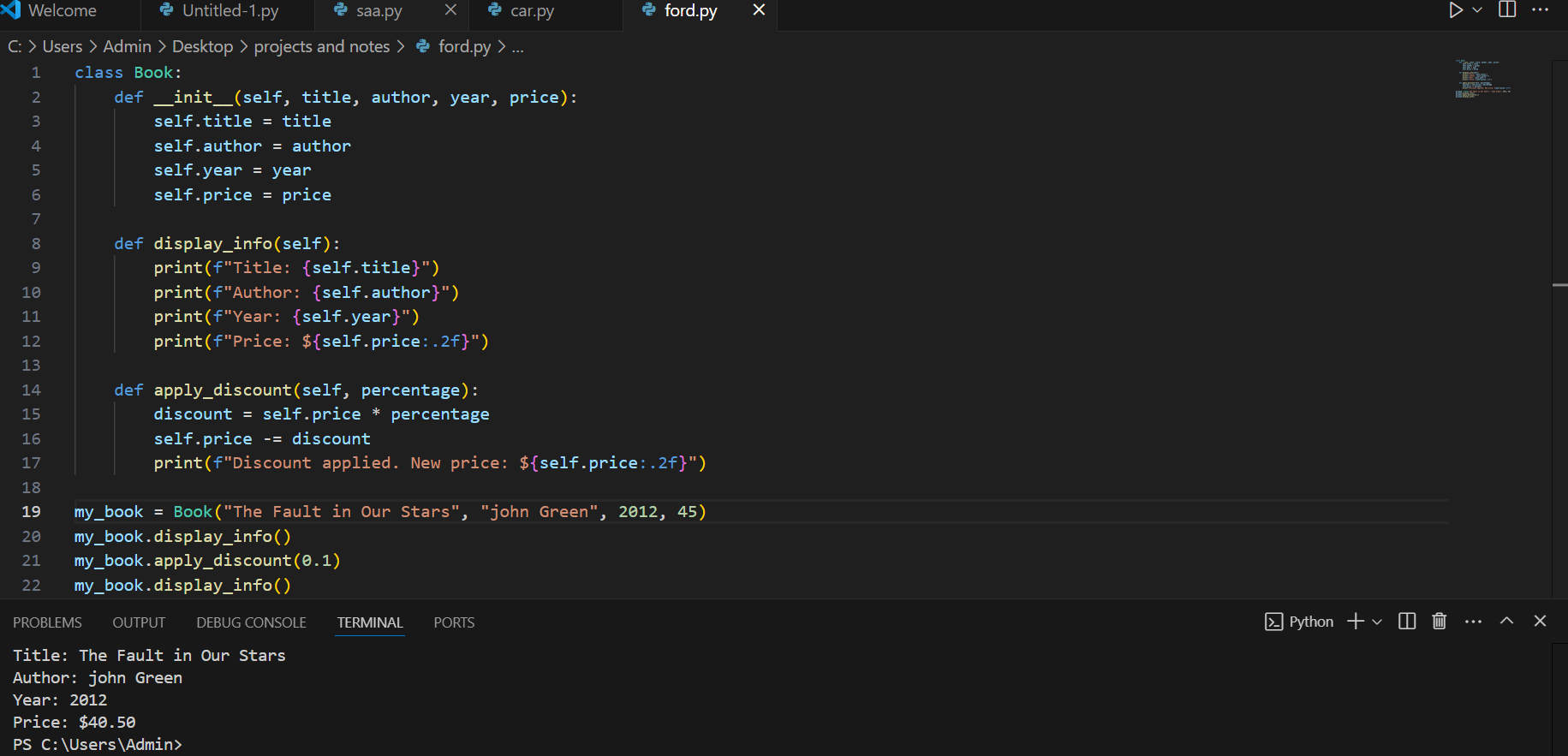


This code defines a Car class with attributes (make, model, year, mileage) and methods to display the car's details and simulate driving by updating its mileage. An instance of the class is created, its details are displayed, it is driven 22 miles (updating the mileage), and the updated details are displayed again.

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., "The fault in our star").
   * author (string): The author of the book (e.g., "john green").
   * 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.



Code Explanation: The above code defines a Book class, comprising attributes such as title, author, year, and price, and two methods, display\_info and apply\_discount. The former method displays the details of a book, namely the title, author, year, and price. The latter applies a discount to a book, depending on the percentage given as an argument. In the usage example, the Book object is instantiated with the title "The Fault in Our Stars", written by John Green, published in 2012, and priced at $45. The display\_info method first displays the initial information, then a 10% discount is given, which reduces its price to $40.50, and the updated information is displayed again. With this code, a simple mimic of maintaining a book's information and giving discounts on it is simulated.