**LAB ASSIGNMENT-6.4**

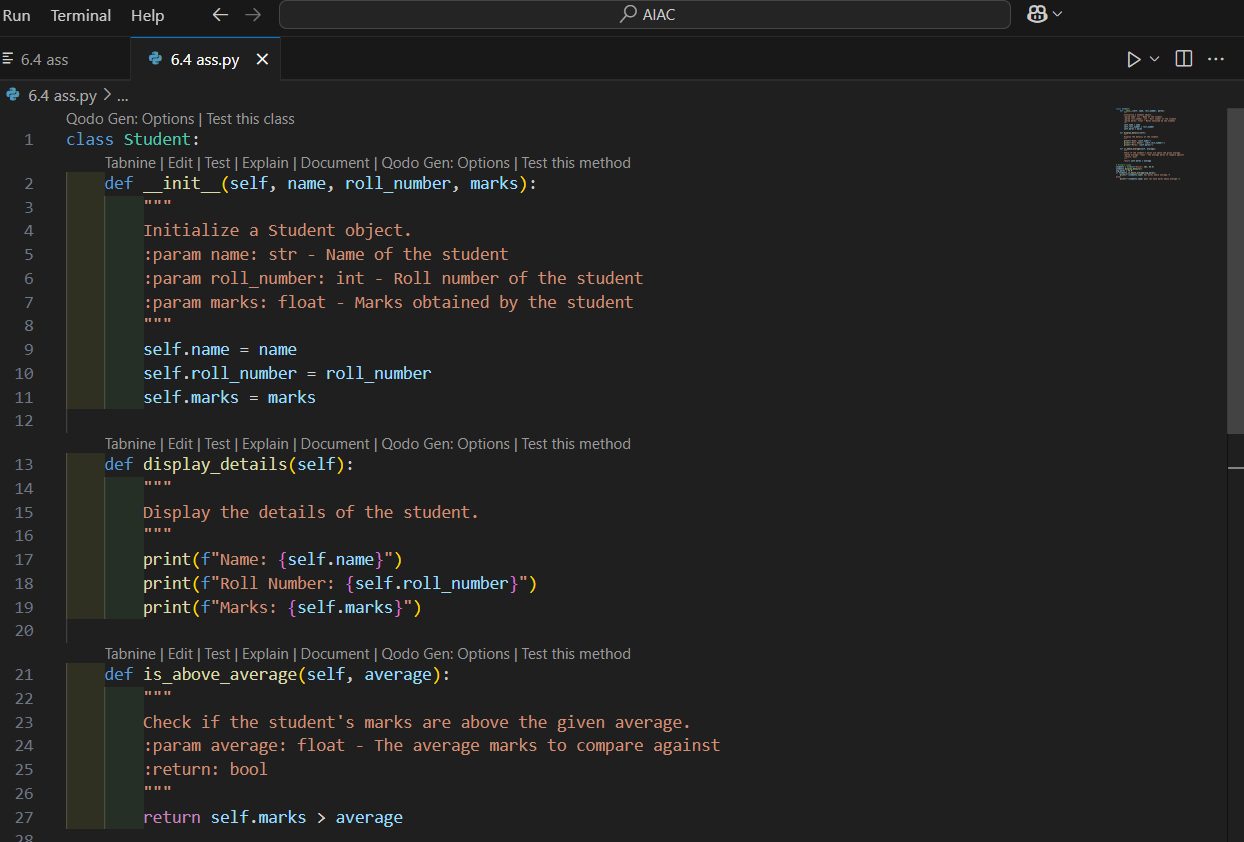
**Name:P.Navya**

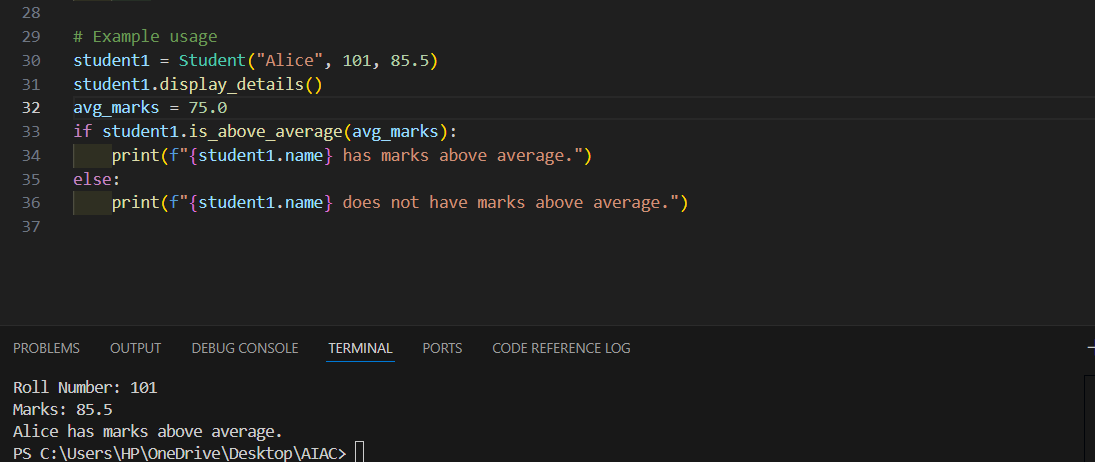
**HTNO:2403A51331**

**Batch:13**

**Task-1**

Start a Python class named Student with attributes name, roll\_number, and marks. Prompt  
GitHub Copilot to complete methods for displaying details and checking if marks are above  
average





**Explanation:**

The Student class represents a student with three attributes: name, roll\_number, and marks.

The \_init\_ method initializes these attributes when a new Student object is created.

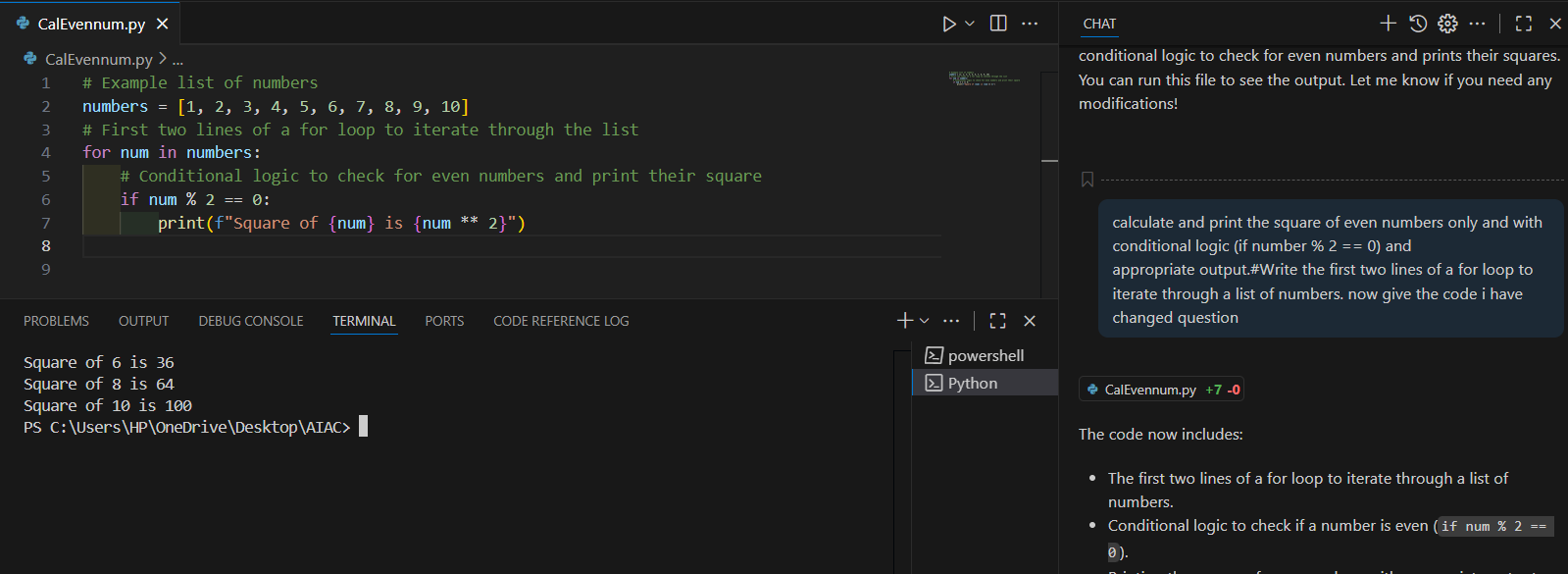
The display\_details method prints the student's name, roll number, and marks.

The is\_above\_average method checks if the student's marks are greater than a given average and returns True or False.

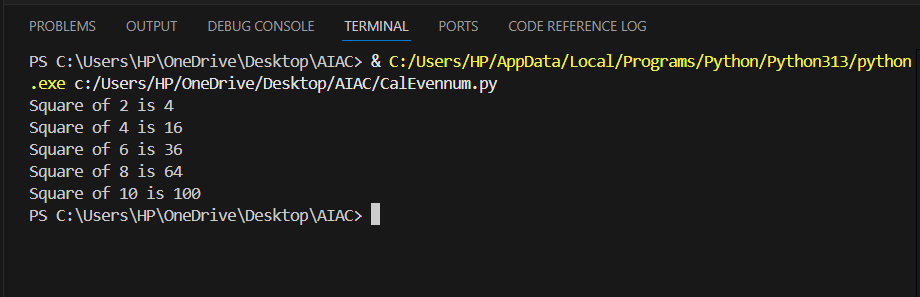
In the example usage, a Student object named student1 is created. The details are displayed, and the code checks if the student's marks are above the average (75.0), printing a message accordingly**.**

**TASK-2**

Write the first two lines of a for loop to iterate through a list of numbers. Use a comment  
prompt to let Copilot suggest how to calculate and print the **square of even numbers only**

****

**Output:**

****

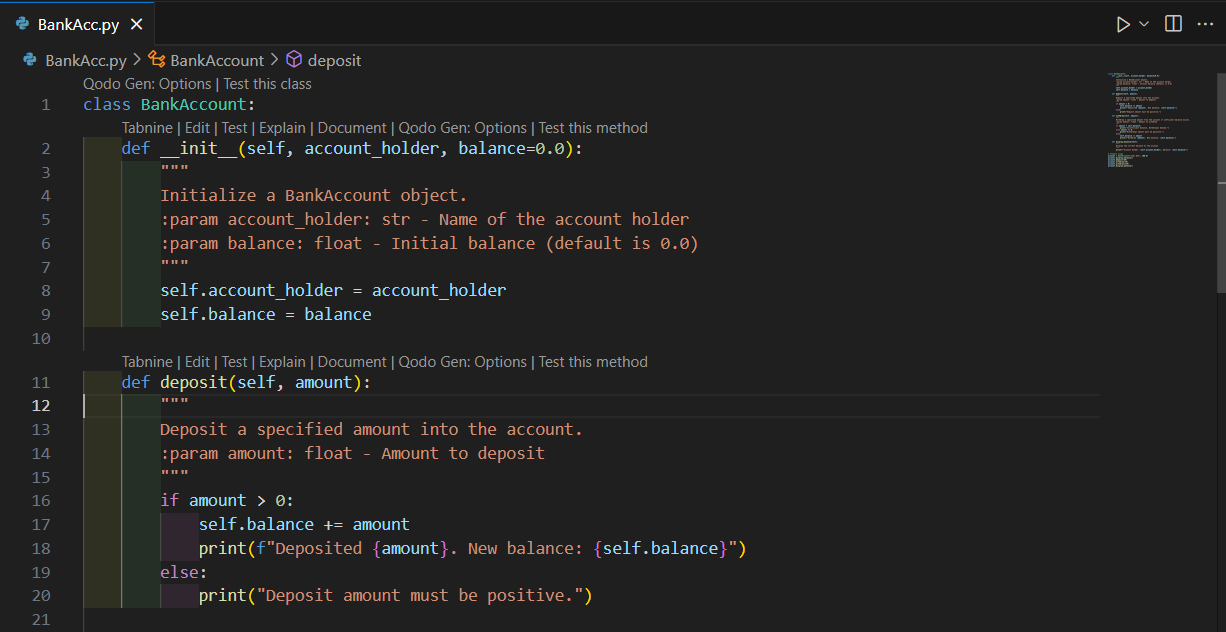
**Explanation:**

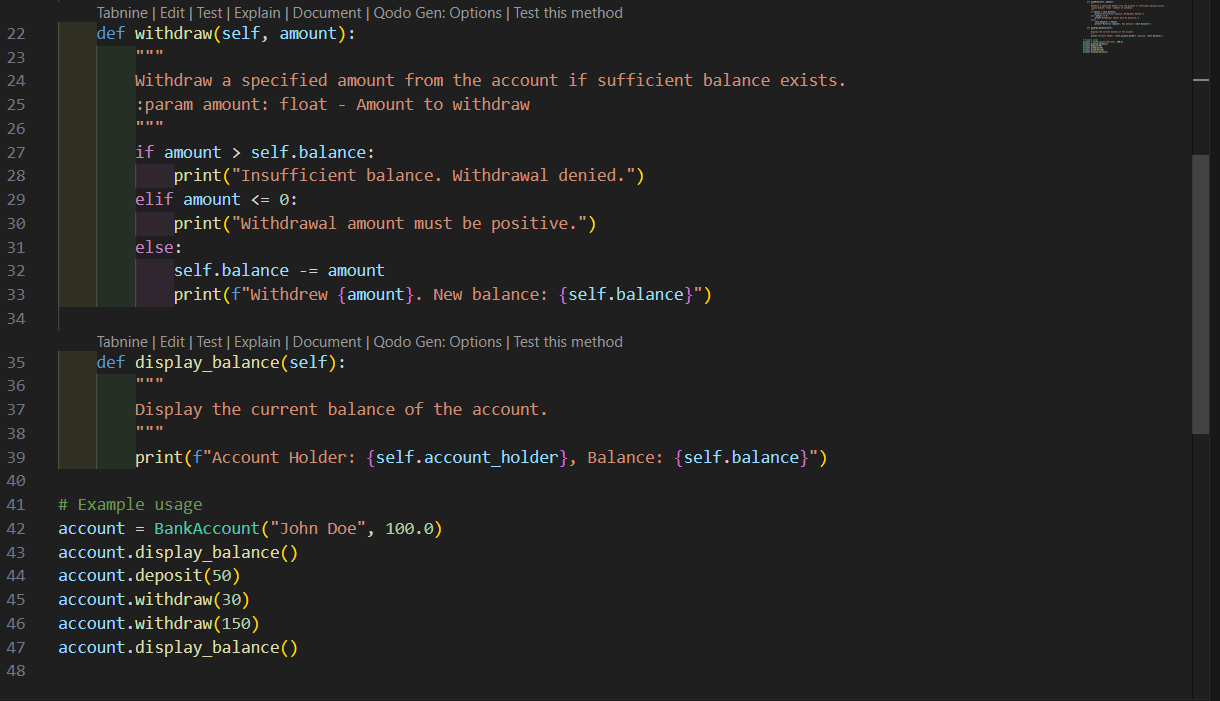
* Define a list of numbers, for example: numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].
* A for loop iterates through each number in the list: for num in numbers:.
* Inside the loop, an if statement checks if the number is even: if num % 2 == 0.
* If the condition is true (the number is even), the code calculates its square (num \*\* 2) and prints a formatted message showing the number and its square.
* Odd numbers are ignored; only even numbers have their squares printed

**TASK-3**

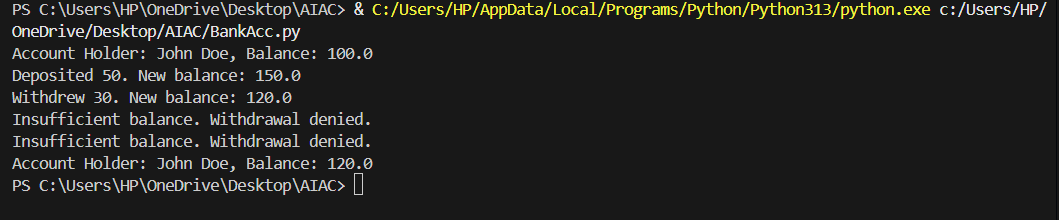
Create a class called BankAccount with attributes account\_holder and balance. Use Copilot

To complete methods for deposit(), withdraw(), and check for insufficient balance**.**

****

****

**Output:**

****

**Explanation:**

The BankAccount class models a simple bank account.

\_init\_ initializes the account with the holder’s name and starting balance.

deposit(amount) adds money to the account if the amount is positive; otherwise, it prints an error.

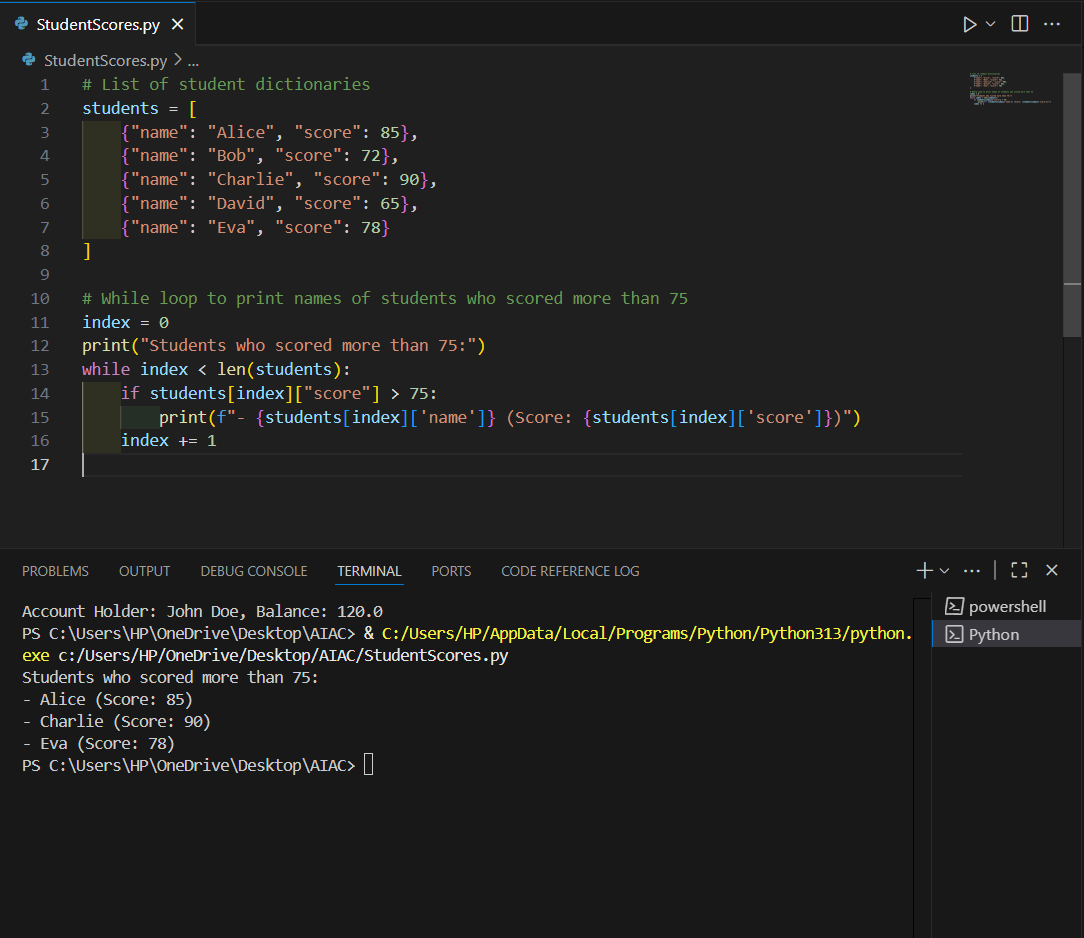
withdraw(amount) subtracts money from the account only if there is enough balance and the amount is positive; otherwise, it prints an error or a warning for insufficient funds.

display\_balance() prints the account holder’s name and current balance.

The example usage creates an account, displays the balance, makes deposits and withdrawals, and shows how the class prevents overdrawing.

**TASK-4:**

Define a list of student dictionaries with keys name and score. Ask Copilot to write a while  
loop to print the names of students who scored more than 75.



**Explanation:**

A list named students is defined, where each element is a dictionary containing a student's name and score.

The code initializes an index variable (index = 0) to start iterating from the first student.

It prints a header: "Students who scored more than 75:".

A while loop runs as long as index is less than the length of the students list.

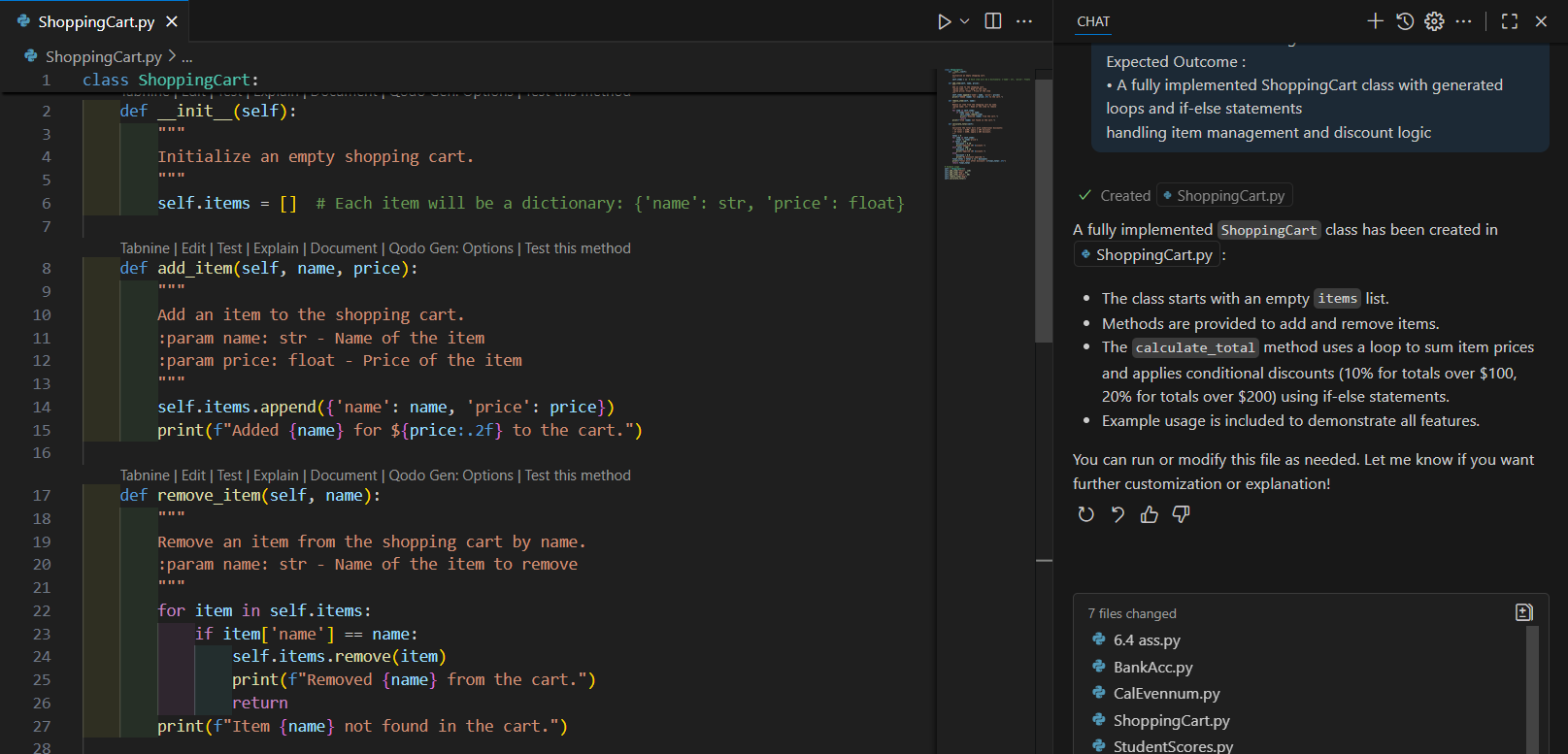
Inside the loop, it checks if the current student's score is greater than 75.

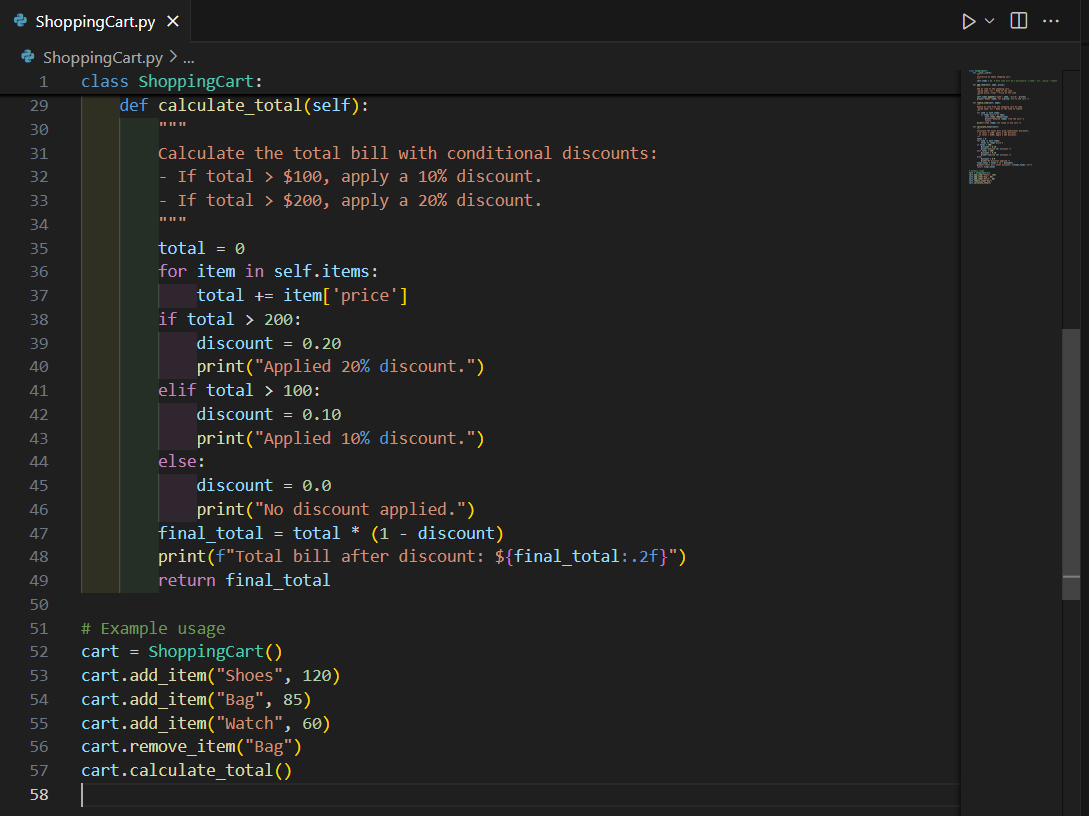
If the condition is true, it prints the student's name and score in a formatted way.

The index is incremented by 1 in each iteration to move to the next student.

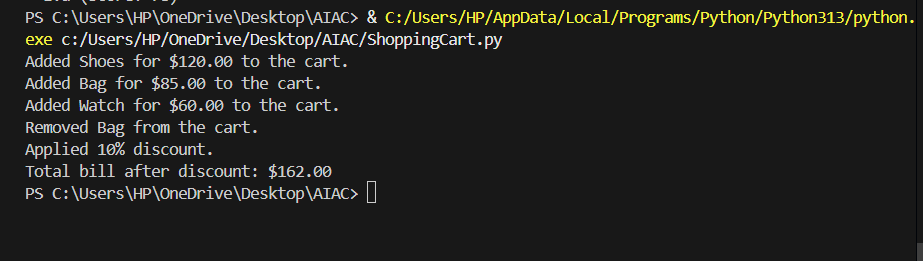
**TASK-5:**

Begin writing a class ShoppingCart with an empty items list. Prompt Copilot to generate  
methods to add\_item, remove\_item, and use a loop to calculate the total bill using conditional discounts.





**Output:**



**Explanation:**

* The ShoppingCart class starts with an empty list called items to store products.
* The add\_item method lets you add an item by providing its name and price. Each item is stored as a dictionary.
* The remove\_item method searches for an item by name and removes it from the cart if found.
* The calculate\_total method loops through all items to sum their prices. It then checks the total:
  + If the total is over $200, a 20% discount is applied.
  + If the total is over $100, a 10% discount is applied.
  + Otherwise, no discount is given.
* The method prints the discount applied and the final bill.
* Example usage at the end shows how to add, remove items, and calculate the total with discounts.