**HALL TICKET NUMBER: 2403A51365**

**NAME: RAYAPUDI HARITHA**

**BATCH: 24BTCAICSB14**

**AssignmentNumber:6.4**

**Lab 6: AI-Based Code Completion – Classes, Loops, and Conditionals  
Lab Objectives:  
• To explore AI-powered auto-completion features for core Python constructs.  
• To analyze how AI suggests logic for class definitions, loops, and conditionals.  
• To evaluate the completeness and correctness of code generated by AI assistants.  
Lab Outcomes (LOs):  
Week3 -  
Thursday**

**After completing this lab, students will be able to:  
• Use AI tools to generate and complete class definitions and methods.  
• Understand and assess AI-suggested loops for iterative tasks.  
• Generate conditional statements through prompt-driven suggestions.  
• Critically evaluate AI-assisted code for correctness and clarity**

**Task Description #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.**

**Prompt:-**

**• 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.**

**Code And Output:-**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**Code Explanation:-**

1. The Student class is defined to represent a student with basic attributes.
2. The \_\_init\_\_ method initializes the student's name, roll number, and marks.
3. self.name, self.roll\_number, and self.marks store individual student data.
4. The display\_details method prints the student's information in a readable format.
5. is\_above\_average checks if the student's marks exceed a default average of 50.
6. A Student object named student1 is created with sample data.
7. display\_details() is called to show the student's name, roll number, and marks.
8. The is\_above\_average() method evaluates if the student scored above 50.
9. If true, it prints that the student has above average marks.
10. Otherwise, it prints that the student has below average marks.

Task Description #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.

Prompt:-

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.

Code And Output:-

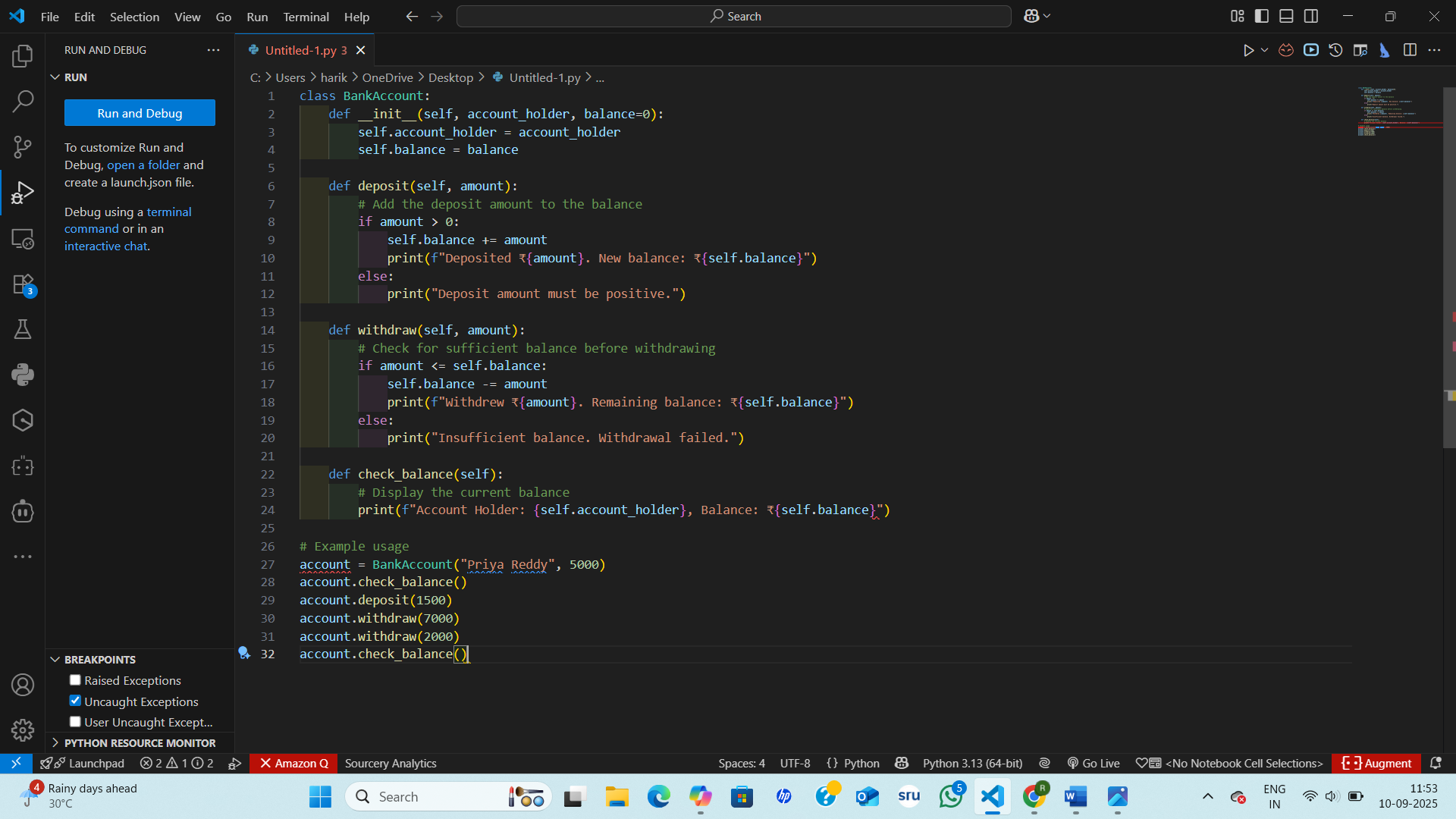
A screenshot of a computer screen

AI-generated content may be incorrect.

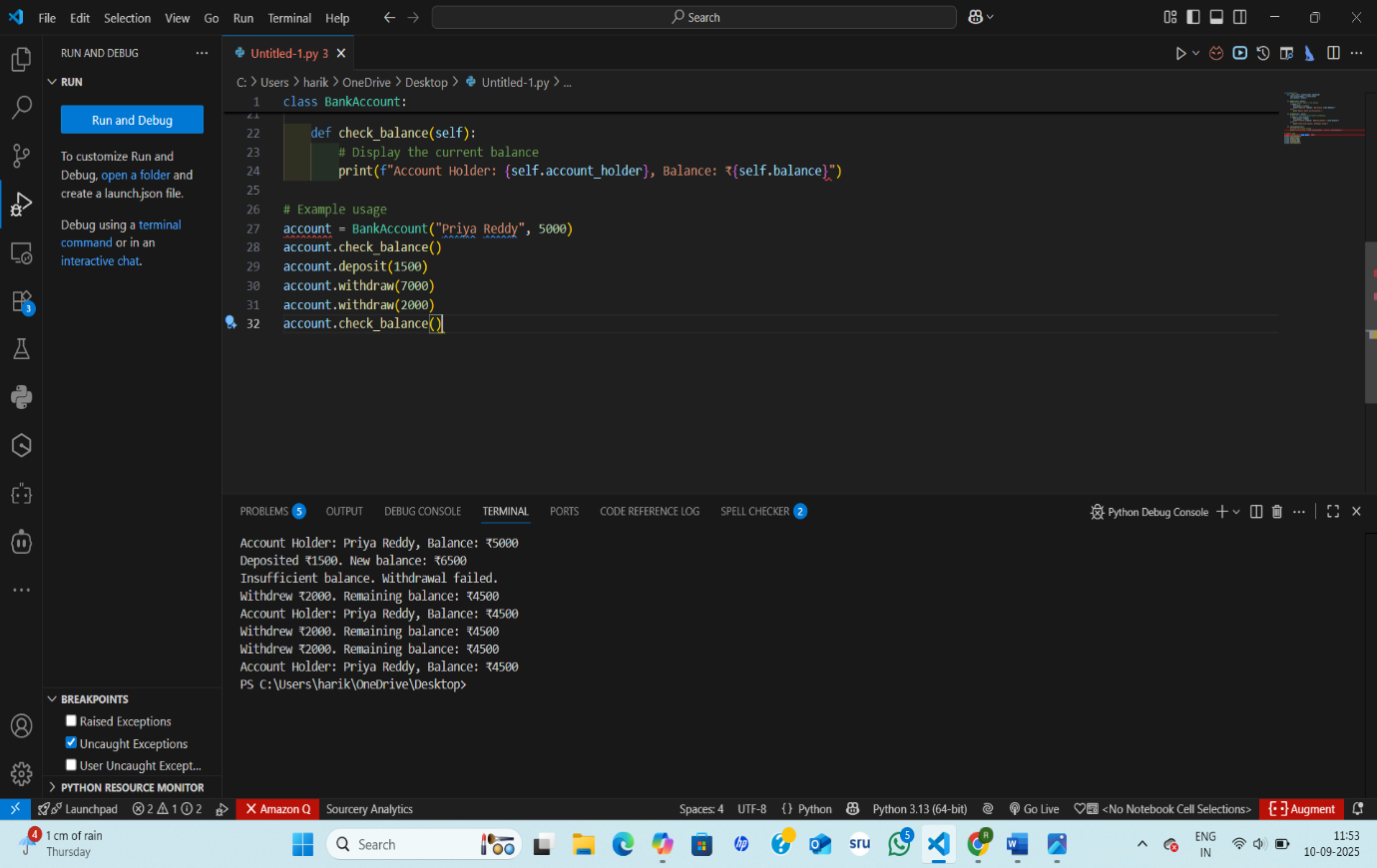
**Code Explanation:-**

1. A list named numbers is created containing integers from 1 to 10.
2. A for loop is used to iterate through each number in the list.
3. The loop assigns each number to the variable num one by one.
4. Inside the loop, an if statement checks if num is even using num % 2 == 0.
5. The modulo operator % returns the remainder when num is divided by 2.
6. If the remainder is 0, the number is even and the condition is true.
7. When the condition is true, the code calculates the square using num \*\* 2.
8. The print() function displays the result in a formatted string.
9. Only even numbers trigger the print statement; odd numbers are skipped.
10. The output shows the square of each even number in the list clearly.

Code:-



Output:-



**Code Explanation:-**

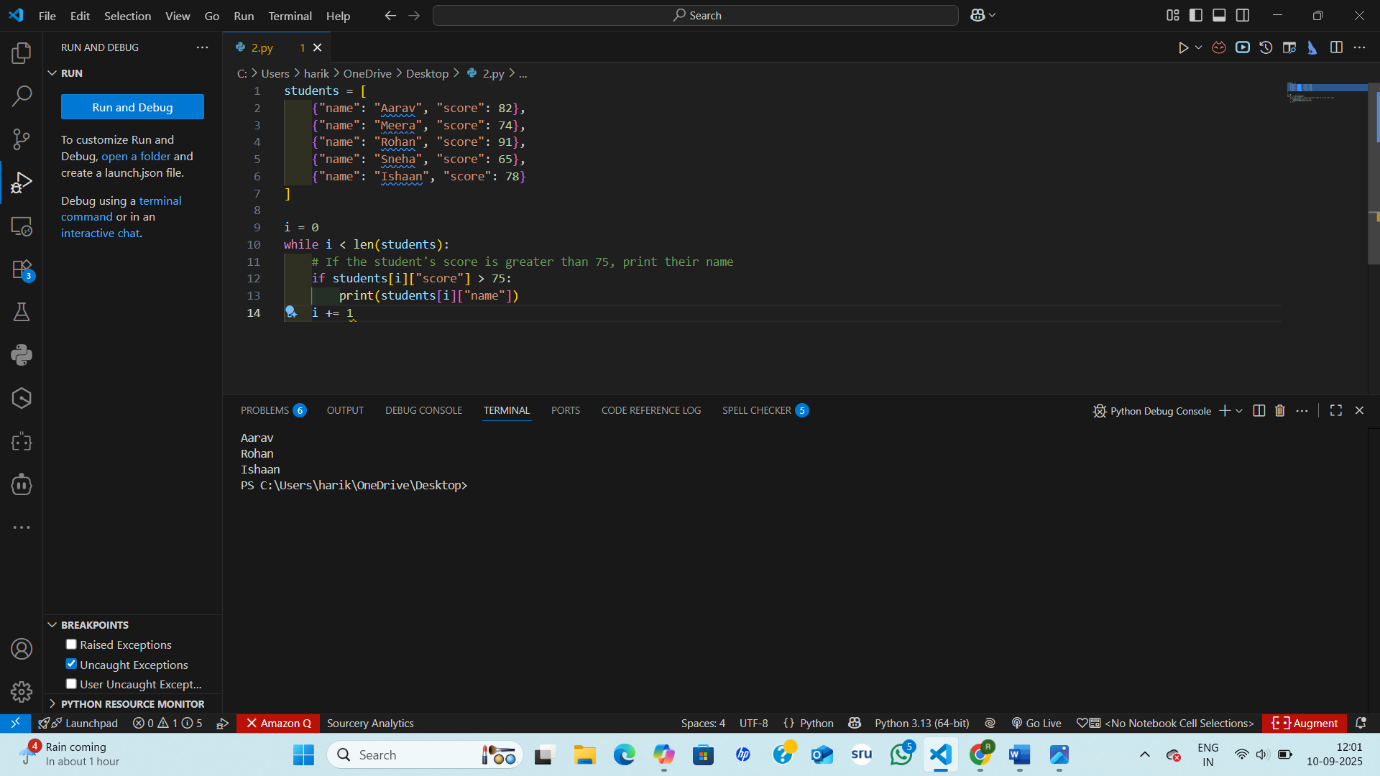
1. The BankAccount class models a simple bank account with basic operations.
2. The \_\_init\_\_ method initializes the account holder's name and starting balance.
3. The deposit() method adds a positive amount to the account balance.
4. It checks if the deposit amount is valid before updating the balance.
5. The withdraw() method subtracts money from the balance if funds are sufficient.
6. If the withdrawal amount exceeds the balance, it prints an error message.
7. The check\_balance() method displays the account holder's name and current balance.
8. An object account is created with the name "Priya Reddy" and ₹5000 balance.
9. The account receives a deposit, then attempts two withdrawals—one fails due to low funds.
10. The final balance is printed, showing the result of all transactions.

Task Description #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

Prompt:-

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

Code And Output:-



**Code Explanation:-**

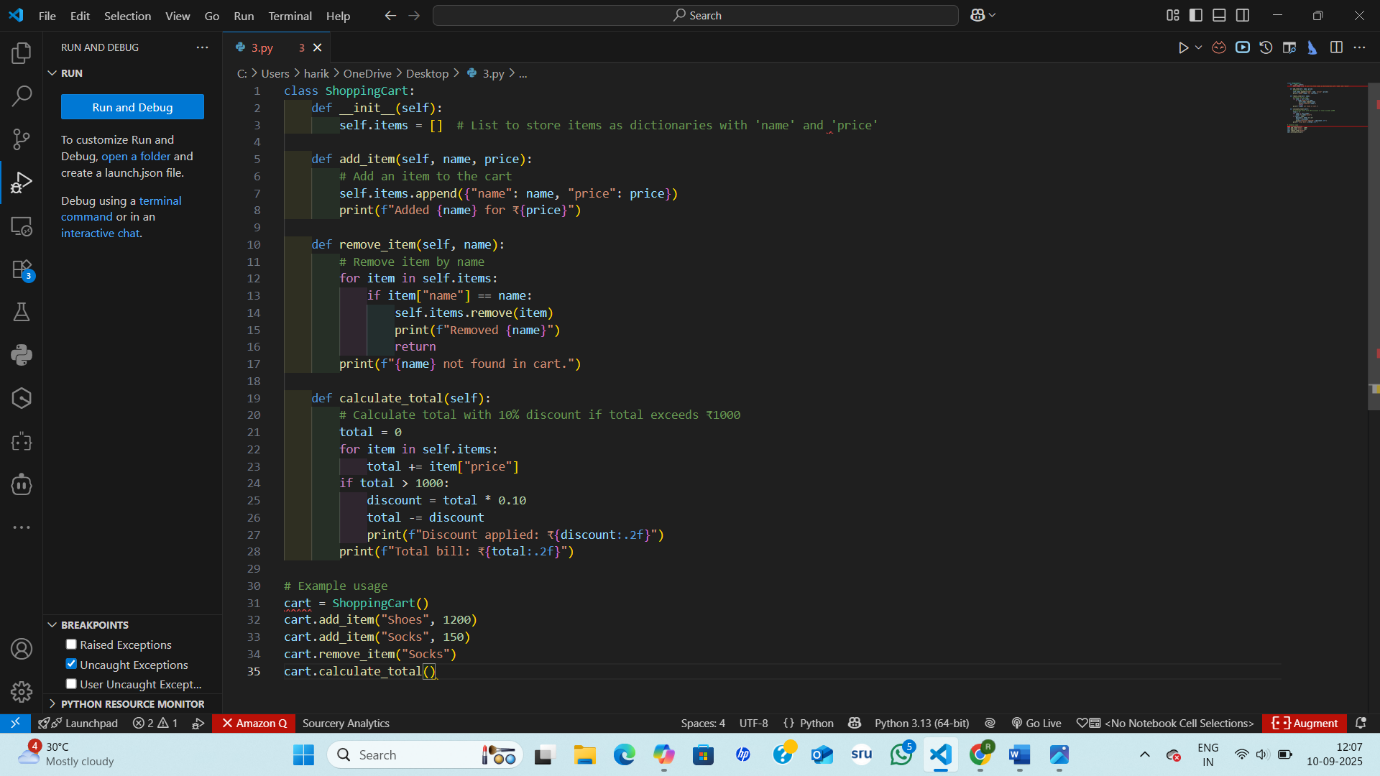
1. A list named students is created, containing dictionaries with each student's name and score.
2. Each dictionary has two keys: "name" for the student's name and "score" for their marks.
3. A variable i is initialized to 0 to serve as the loop counter.
4. A while loop runs as long as i is less than the length of the students list.
5. Inside the loop, students[i] accesses the current student's dictionary.
6. The condition students[i]["score"] > 75 checks if the student scored above 75.
7. If the condition is true, the student's name is printed using students[i]["name"].
8. If the score is 75 or below, nothing is printed for that student.
9. The counter i is incremented by 1 to move to the next student.
10. The loop continues until all students have been checked.

Task Description #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.

Prompt:-

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.

Code:-



Output:-

A screenshot of a computer screen

AI-generated content may be incorrect.

**Code Explanation:-**

1. The ShoppingCart class is created to manage items in a shopping cart.
2. The constructor \_\_init\_\_ initializes an empty list called items to store products.
3. The add\_item() method takes a name and price, then appends a dictionary to the cart.
4. It prints a confirmation message showing the item added and its price.
5. The remove\_item() method searches for an item by name and removes it if found.
6. If the item isn’t found, it prints a message saying so.
7. The calculate\_total() method loops through all items and adds up their prices.
8. If the total exceeds ₹1000, a 10% discount is applied and shown.
9. The final bill amount is printed after applying any discount.
10. The example usage shows adding items, removing one, and calculating the total with discount.