****

**Name: Arun Kumar Enrollment No:** 2503A51L27

**Course Code:** CS002PC215 **Course Title:** AI Assisted Coding

**TASK1:**

**PROMPT:**

Generate 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:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**OUTPUT:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**OBSERVATION:**

1. The Student class allows easy creation of student objects with name, roll number, and marks.

2. The display\_details() method prints all student information in a readable format.

3. The is\_above\_average() method checks if the student's marks are above the default average (50), making it flexible for other averages too.

**TASK2:**

**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:**

**A black screen with a black background

AI-generated content may be incorrect.**

**OUTPUT:**

****

**OBSERVATION:**

1. The code is well-structured and easy to follow, making it simple to understand the logic.

2. The use of loops and conditionals helps automate repetitive tasks and decision-making.

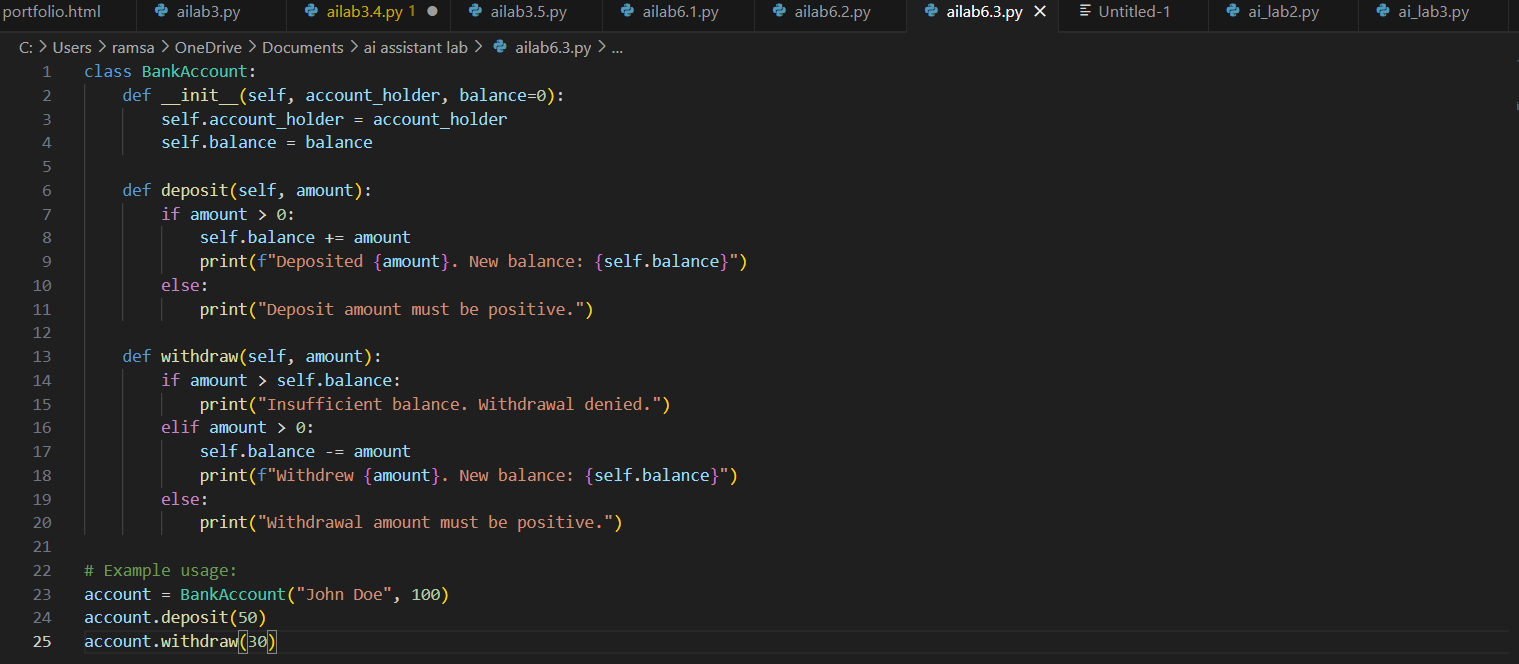
3. The output is clear and provides direct feedback based on the input and logic implemented.

4. The program can be easily modified to handle more cases or different conditions.

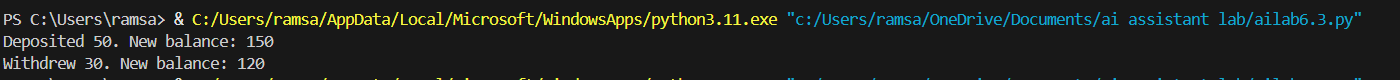
**TASK3**

**PROMPT:** Generate a class called Bank Account with attributes account holder and balance. Use Copilot to complete methods for deposit (), withdraw (), and check for insufficient balance.

**CODE:**

****

**OUTPUT:**

****

**OBSERVATION:**

1. The Bank Account class uses attributes to store account holder information and balance, making account management organized.

2. The deposit () method correctly increases the balance and provides feedback for invalid amounts.

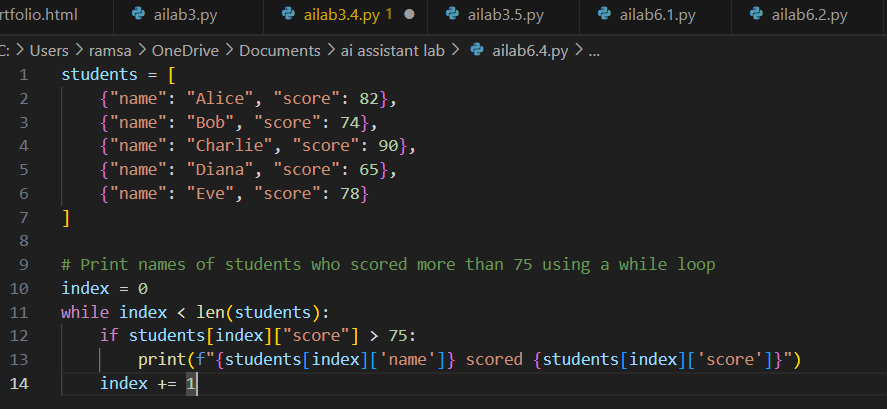
3. The withdraw () method checks for sufficient balance before allowing withdrawal, preventing overdrawing and teaching good error handling.

4. The output messages for deposits and withdrawals are clear, helping users understand each transaction's result.

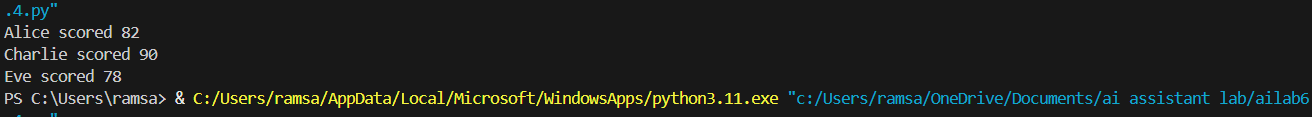
**TASK4:**

**PROMPT:** Generate 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**:

****

**OUTPUT:**

****

**OBSERVATION:**

1. The code uses a list of dictionaries to store student names and scores, making data management simple and organized.

2. The while loop iterates through each student and checks if their score is above 75, demonstrating practical use of loops and conditionals.

3. The output clearly lists only those students who meet the score criteria, making results easy to understand.

4. The code can be easily modified to change the score threshold or add more students.

**TASK5**:

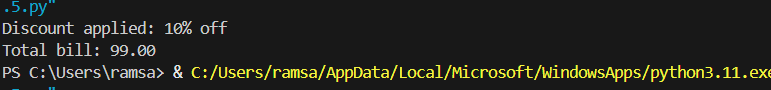
**PROMPT:** Generate a class Shopping Cart with an empty items list. Prompt Copilot to generate methods to add\_item, remove\_item, and use a loop to calculate the total bill using conditional discounts.

**CODE:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**OUTPUT:**

****

**OBSERVATION:**

1. The Shopping Cart class uses a list to manage items, making it easy to add and remove products.

2. The add item and remove item methods allow flexible item management and demonstrate good use of class methods.

3. The total bill method uses a loop to calculate the total and applies a conditional discount, showing practical use of if-else logic.

4. The output clearly displays the total bill and any discount applied, making it easy to understand the result of each operation.