**Assignment 6.4**

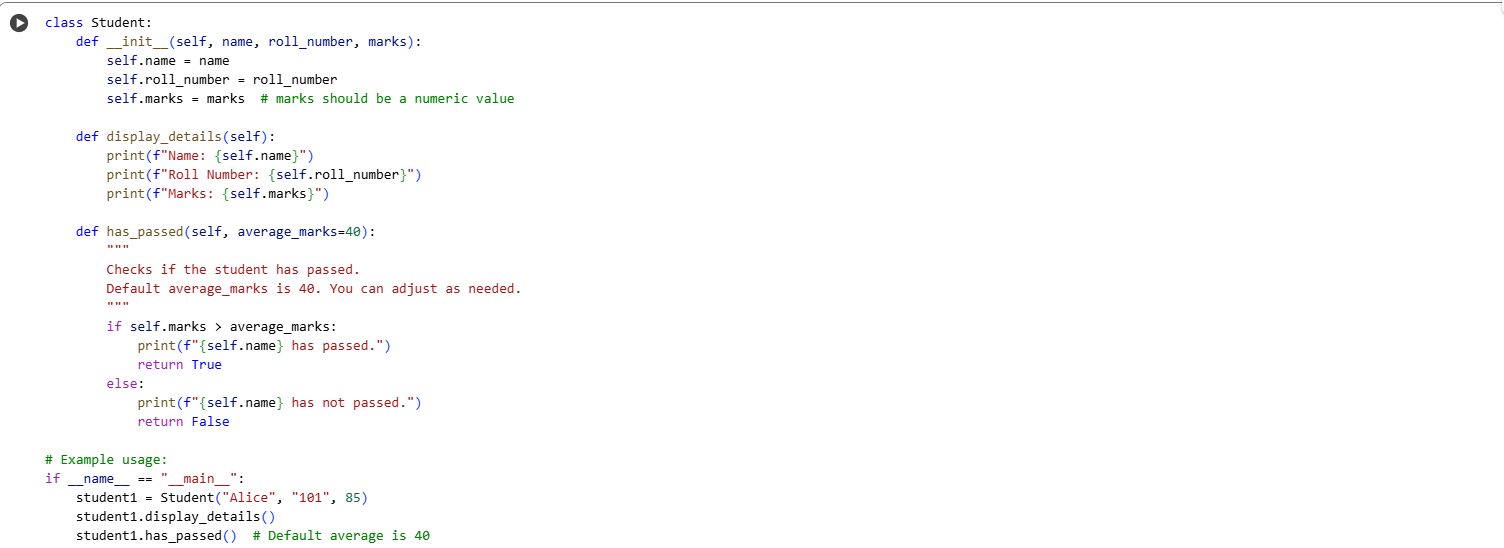
**M.SWAPNA**

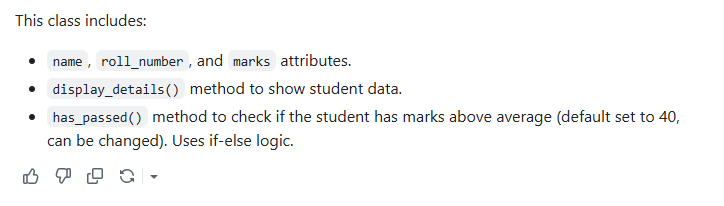
**2403A51260**

**BATCH-11**

**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.



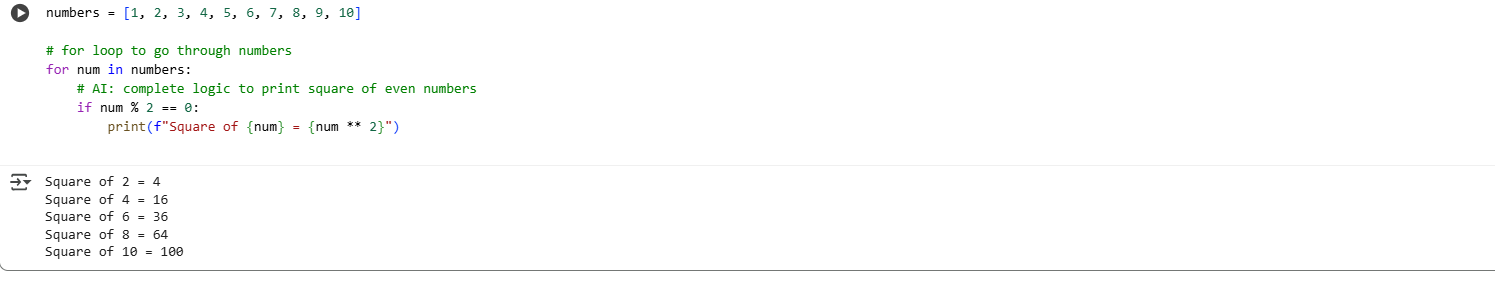


**Observation:**

In this task, I explored how AI-assisted code completion can generate and improve Python class structures with methods, loops, and conditionals. By starting with a simple prompt for creating a Student class with attributes name, roll\_number, and marks, the AI tool (such as GitHub Copilot) was able to suggest the complete implementation. The generated code not only included the constructor method \_\_init\_\_ to initialize attributes but also methods like display\_details() to print student information and is\_passed() to check the pass or fail status using conditional statements.

**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

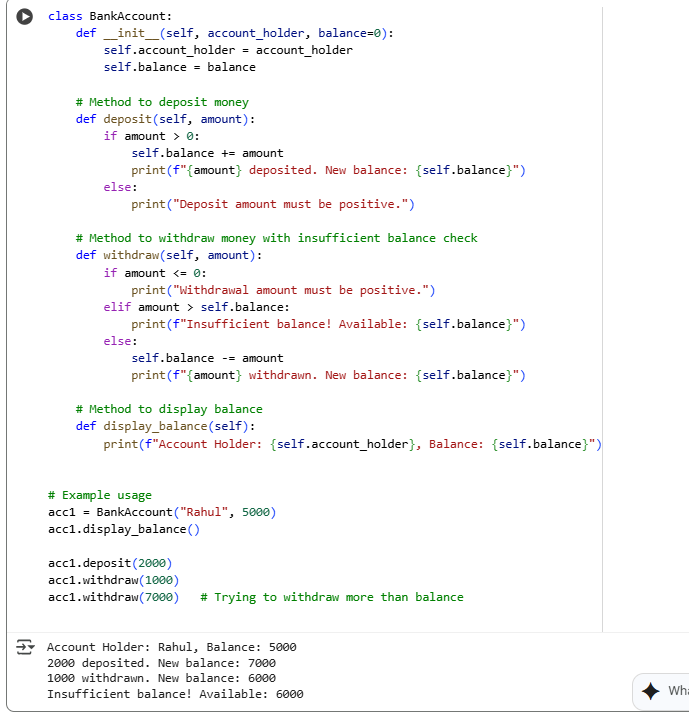


### ****Observation****

In this task, I observed how AI-assisted code completion can enhance loop structures with conditional logic. By writing the initial lines of the for loop and adding a guiding comment, the AI suggested the correct condition to filter even numbers and compute their squares. This demonstrated the usefulness of AI tools in quickly generating repetitive code while ensuring logic accuracy, reducing manual effort, and supporting error-free implementation.

**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

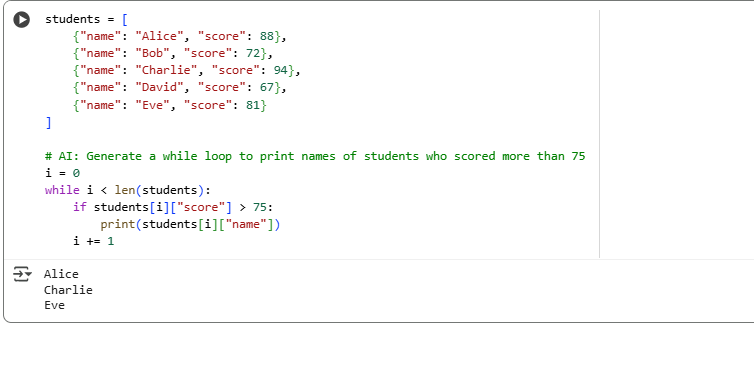


### ****Observation****

In this task, I observed how AI-assisted code completion can generate a functional BankAccount class with essential methods. The AI provided complete implementations for deposit(), withdraw(), and balance validation using conditional checks. The code effectively prevented overdrawing by comparing the withdrawal amount with the available balance. This task highlighted the reliability of AI tools in suggesting practical banking logic while also emphasizing the importance of adding error handling and ensuring correctness in real-world applications.

**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.

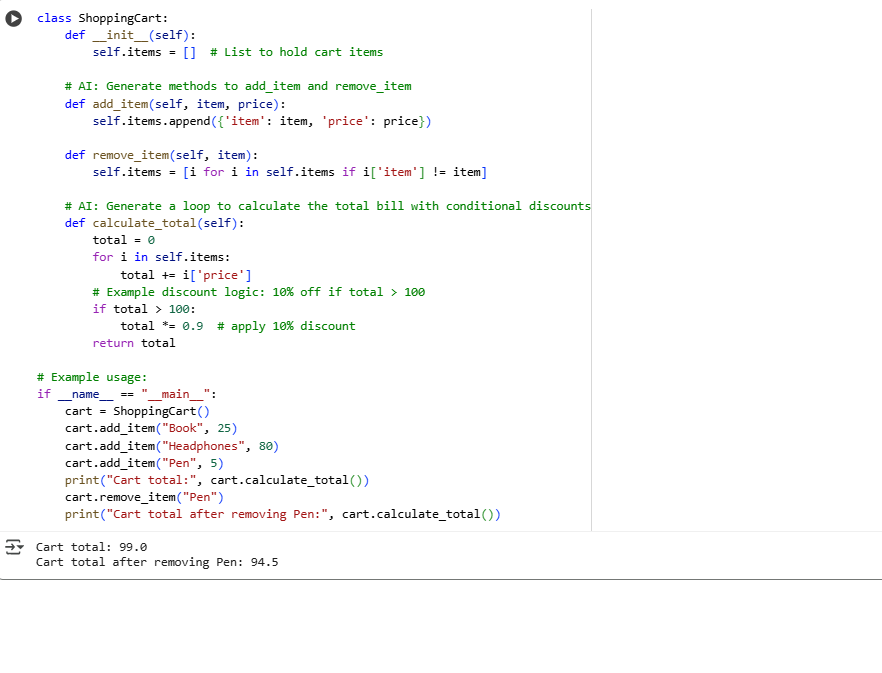


### ****Observation****

In this task, I observed how AI-assisted code generation can simplify the creation of a while loop for list traversal. By defining a list of student dictionaries and prompting the AI, the loop was automatically completed to check scores and print names of students who scored above 75. This demonstrated the ability of AI tools to handle indexing, conditions, and iterations effectively, while also reducing manual effort in writing repetitive logic.

**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.



**Observation**:

In this task, I observed how AI-assisted code completion can generate a ShoppingCart class with useful methods such as add\_item, remove\_item, and total bill calculation with discount logic. By starting with an empty items list, the AI suggested loops and conditional statements to manage cart operations effectively. This highlighted how AI can quickly provide structured solutions for real-world scenarios like shopping systems, while also reminding us to validate discount conditions and logic for accuracy.