**AI\_ASSISTED\_CODING**

**2403A52100**

**Md.Shanawaz**

**B-04**

* **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.

**Prompt:**

Write a Python class named Student with the following:

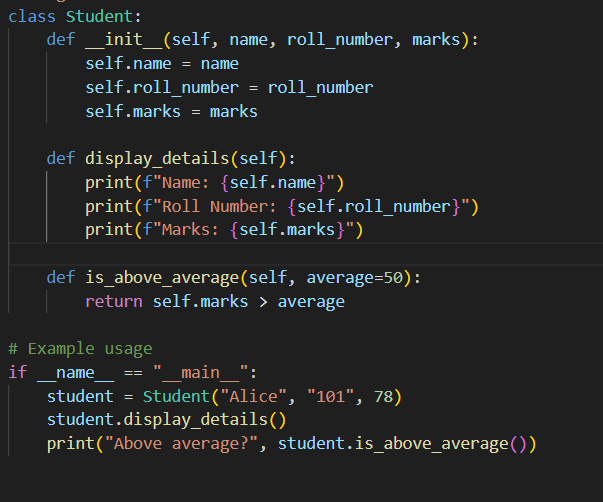
- Attributes: name, roll\_number, and marks

- A method display\_details() to print student details

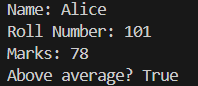
- A method is\_above\_average(average) that returns True if the student’s marks are greater than the given average, otherwise False

Also, show an example usage by creating at least two Student objects and demonstrating both methods.

**Code:**



**Output:**

****

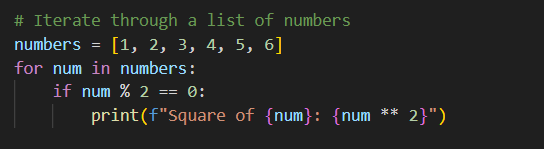
**Observation:**

This code defines a Student class with attributes (name, roll number, marks), a method to display details, and another method to check if marks are above average. Then it demonstrates usage with an example student "Alice"**.**

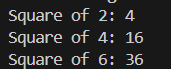
**Task 2:** Write the first two lines of a for loop to iterate through a list of numbers. Use a comment prompt to let Copilot suggest how to calculate and print the square of even numbers only.

**Prompt:** Write a Python program that iterates through a list of numbers from 1 to 6. For each even number, print its square in the format: 'Square of X: Y'

**Code:**



**Output:**



**Observation:**

* + This program iterates through numbers 1–6, selects only even numbers, and prints their squares.

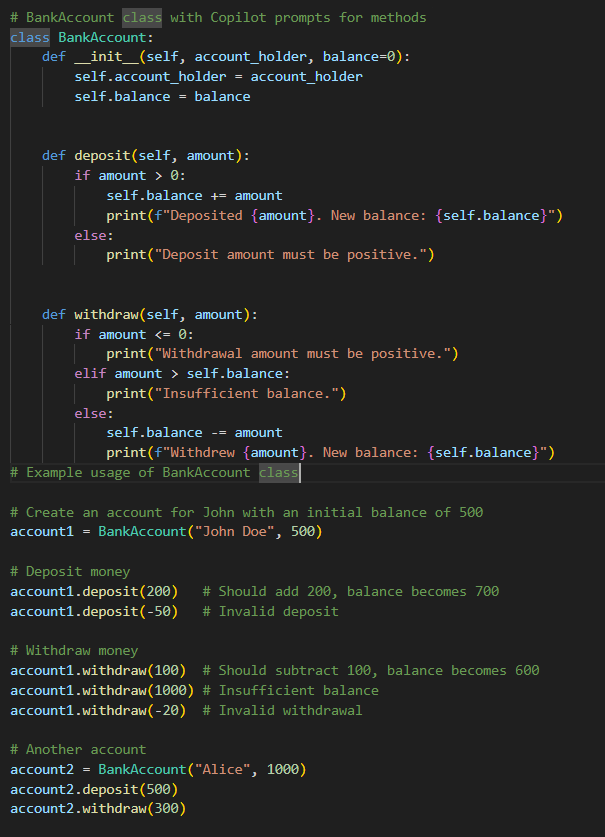
**Task 3:** Create a class called BankAccount with attributes account\_holder and balance. Use Copilot to

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

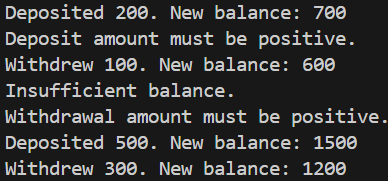
**Prompt:**

Create a Python class named BankAccount with attributes account\_holder and balance (default 0). Add methods deposit(amount) to add money if the amount is positive, and withdraw(amount) to subtract money only if the amount is positive and does not exceed the balance. Print appropriate messages for deposits, withdrawals, invalid amounts, and insufficient balance. Also provide example usage by creating accounts, depositing money, and withdrawing money.

**Code:**

****

**Output:**

****

**Observation:**

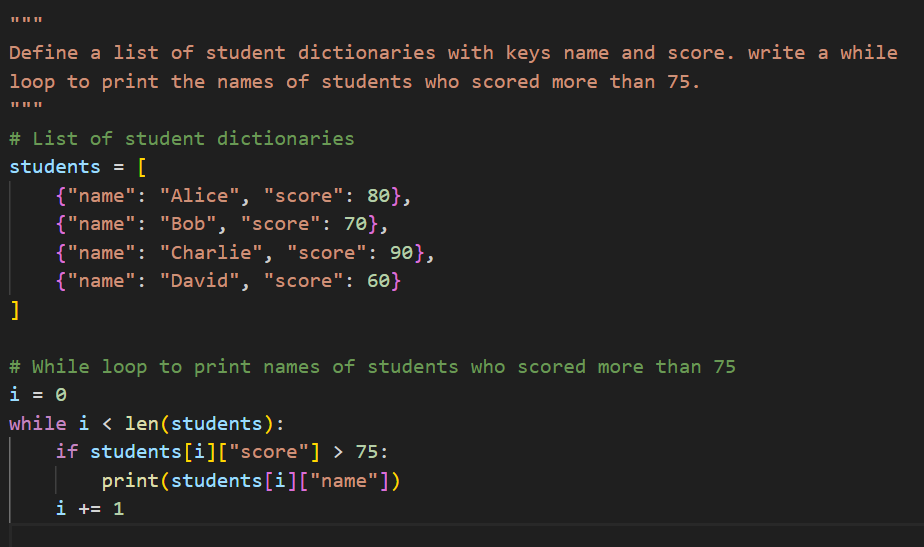
This program defines a BankAccount class with methods to deposit and withdraw money. It includes safety checks (like invalid deposits, insufficient balance, and invalid withdrawals). Two accounts (John Doe and Alice) are created, and operations are performed on them.

**Task 4:** Define a list of student dictionaries with keys name and score. write a while loop to print the names of students who scored more than 75**.**

**Prompt:**

Define a list of student dictionaries where each dictionary contains name and score. Write a Python program using a while loop to print the names of students whose score is greater than 75.

**Code:**

****

**Output:**

****

**Observation:**

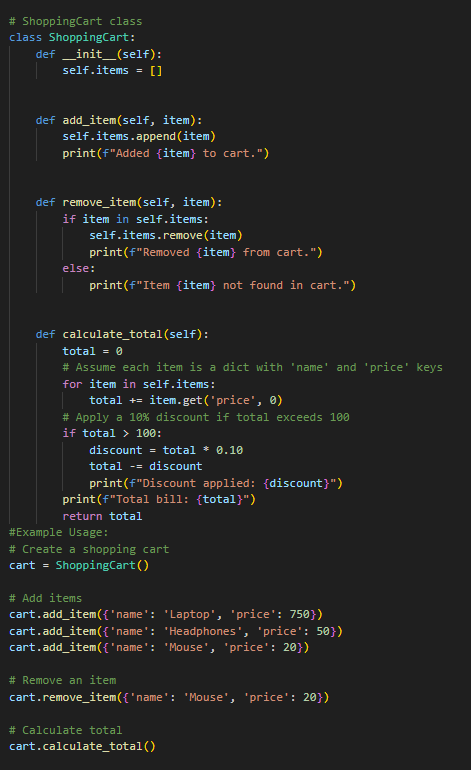
This program stores student details in a list of dictionaries and uses a while loop to check each student’s score. If the score is greater than 75, it prints the student’s name.

**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.

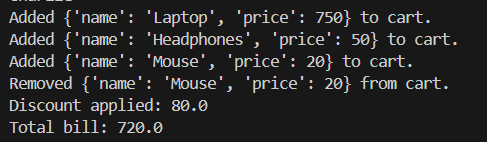
**Prompt:**

Create a Python class named ShoppingCart that allows adding items, removing items, and calculating the total bill. Each item should be stored as a dictionary with keys name and price. The add\_item method should append an item to the cart, remove\_item should remove it if present, and calculate\_total should compute the sum of prices. If the total exceeds 100, apply a 10% discount and display both the discount and final bill. Also, provide example usage by creating a cart, adding items, removing an item, and calculating the total.

**Code:**

****

**Output:**

****

**Observation:**

The code defines a ShoppingCart class with methods to add, remove, and calculate the total price of items. It applies a 10% discount if the total exceeds $100. The example usage demonstrates creating a cart, adding items, removing one, and then calculating the final bill. The final output will be a bill of $720, which is the sum of the remaining items ($800) minus the $80 discount.