**AI ASSISTED CODING LAB ASSIGNMENT 6.4**

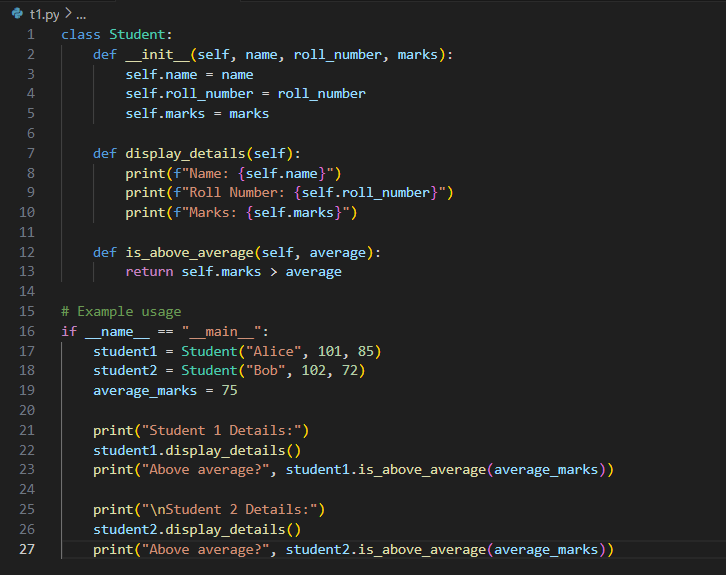
**ENROLLMENT NO :** 2503A51L21

BATCH NO: 19

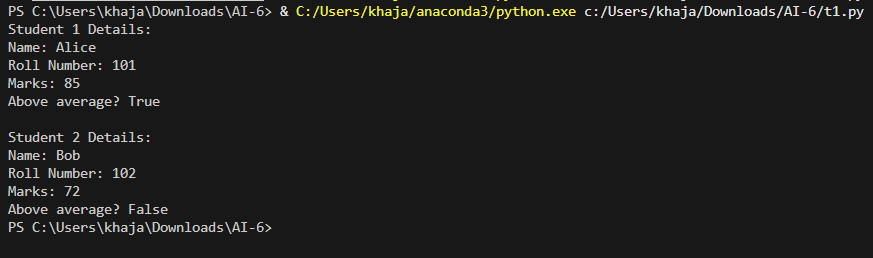
**NAME: LUNGHIMPOU KAMEI**

**TASK1 DESCRIPTION:**  
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:**

Create a Python class Student with attributes name, roll\_number, marks and methods to display details and check if marks are above average (assume average=50). ****

**OUTPUT:**

****

**Observation:**

1. The Student class has three attributes: name, roll\_number, and marks.

2. display\_details() prints all the student information.

3. is\_above\_average() checks if marks > 50 and prints a message accordingly.

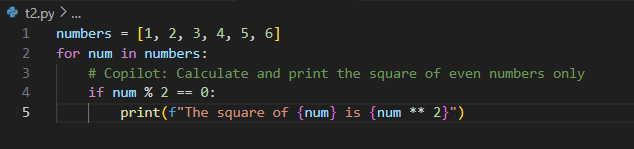
4. Example usage creates two students:

* student1 has marks above average → prints above average message.
* student2 has marks below average → prints below average message.

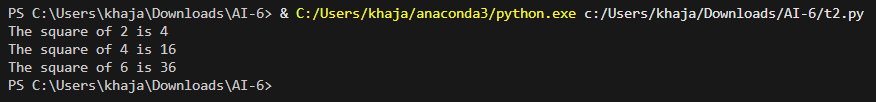
**TASK2 :**

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, and add a comment to suggest how to calculate and print the square of even numbers only.

****

**OUTPUT:**

****

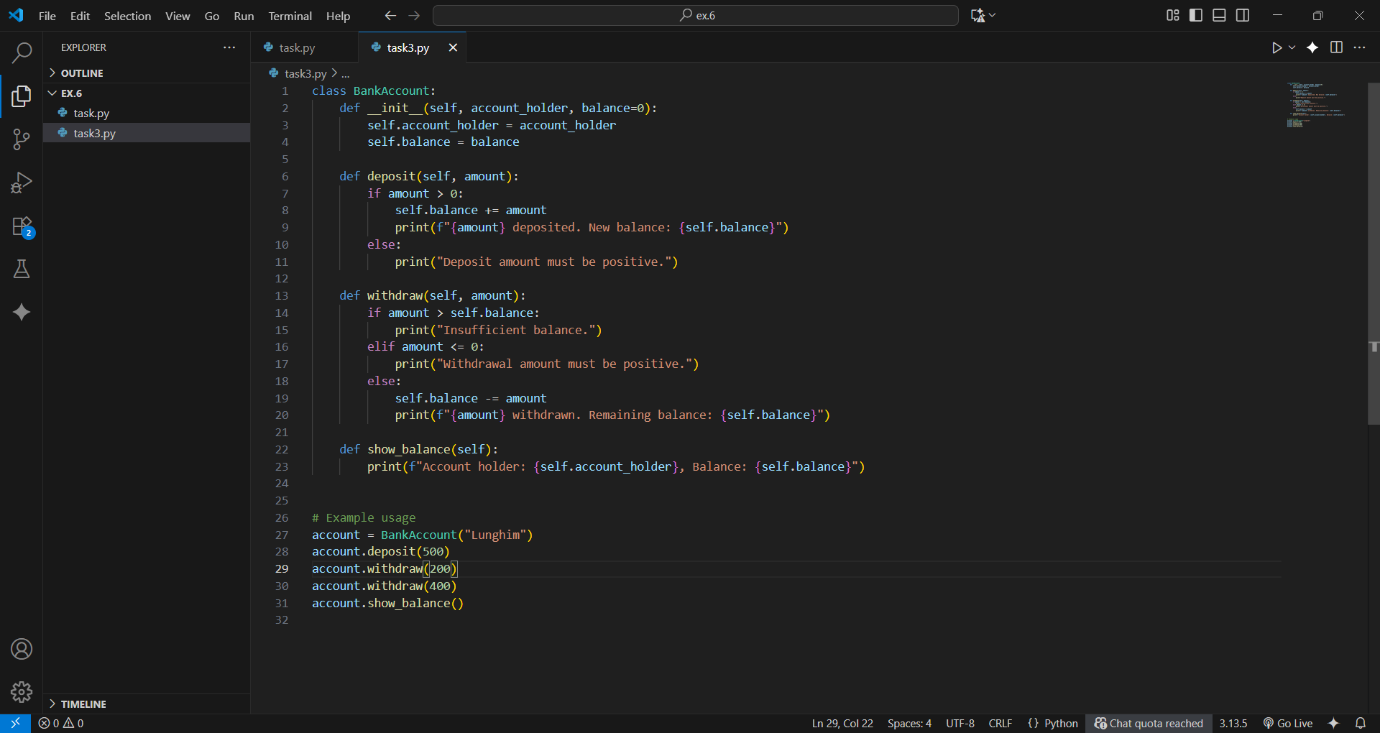
**Observation:**

1. A list numbers is defined containing integers [1, 2, 3, 4, 5, 6].
2. A for loop iterates through each element in the list using the variable num.
3. Inside the loop, a conditional check if num % 2 == 0 is used to determine **even numbers only**.
4. If the number is even, its square is calculated using num \*\* 2 and printed.
5. Odd numbers are ignored and not printed.
6. The program efficiently filters and processes only the numbers that satisfy the condition.

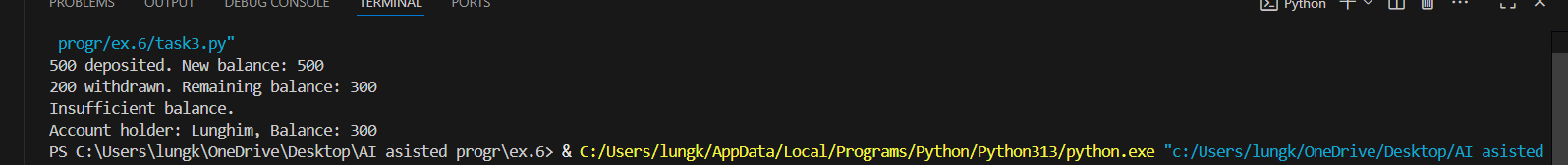
**TASK3 :**

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:** Write a Python class BankAccount with account\_holder and balance attributes, and methods to deposit money, withdraw money, and show appropriate messages. Include example usage.

****

**OUTPUT:**

****

**Observation:**

1. **\_\_init\_\_** initializes account\_holder and balance (default 0).

2. **deposit(amount)** adds money and prints the new balance, validates positive amount.

3. **withdraw(amount)** subtracts money if enough balance, validates positive amount, prints messages.

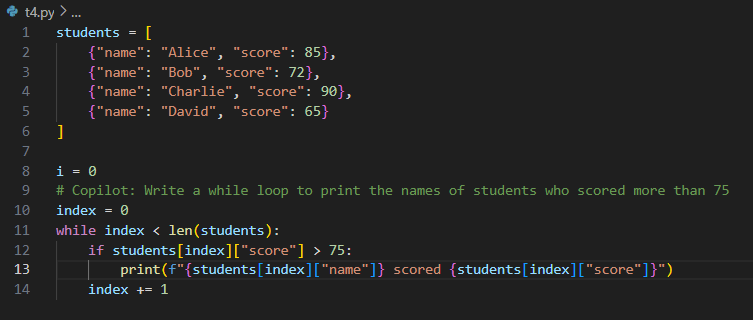
4. **show\_balance()** displays the account holder and current balance.

Example usage shows deposits, withdrawals, and balance check.

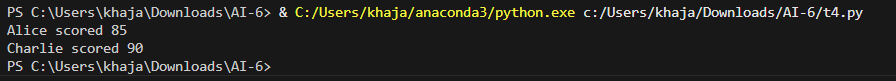
**TASK4 :**

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:** Write a Python while loop to go through a list of students (with name and score) and print the names and scores of students who scored more than 75.

****

**OUTPUT:**

****

**Observation :**

1.A list of dictionaries is created where each dictionary has a student’s name and score.

2.A while loop is used with an index i starting at 0.

3.The loop continues until i < len(students), meaning it checks each student one by one.

4.Inside the loop, it checks if students[i] [“score’] > 75.

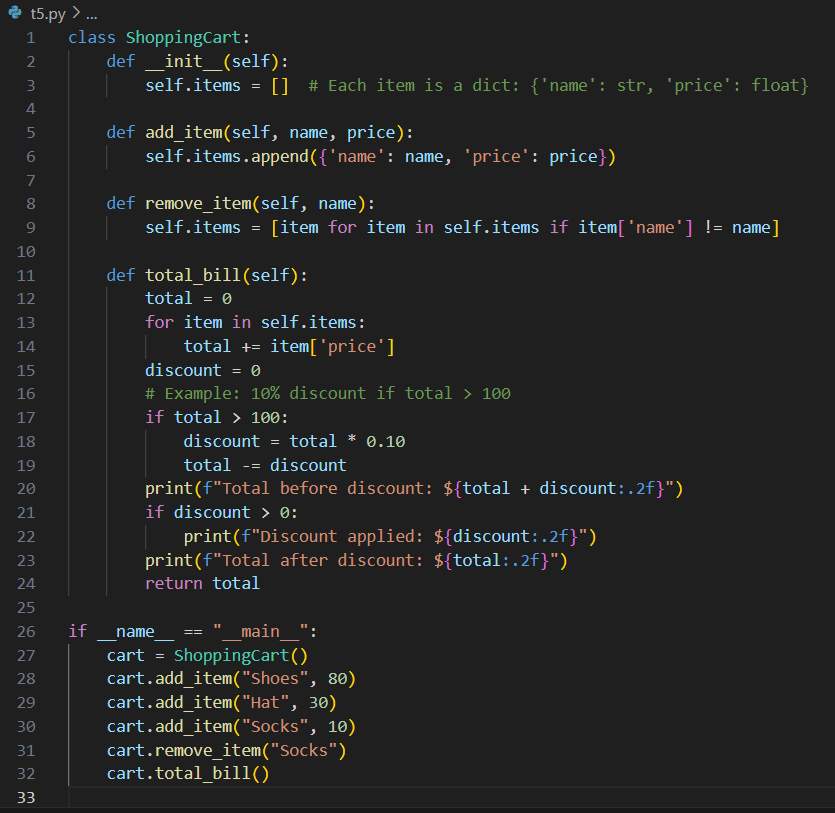
5.If the condition is true, the student’s name is printed.

6. The index i is then increase by 1 to move to the next student.

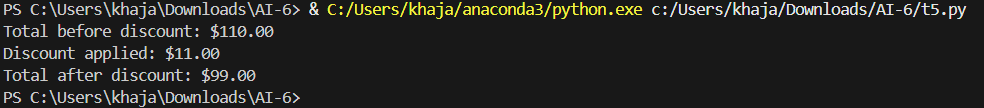
**TASK5** :

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 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.

****

**OUTPUT:**

****

**OBSERVATION:**

1. A class ShoppingCart is created with an empty list items to hold all products.
2. add\_item() appends a dictionary ({"name": name, "price": price}) to the list.
3. remove\_item() removes an item by filtering out the one with the given name.
4. calculate\_total() loops through all items, adds their prices, and applies
5. Finally, the discounted bill is returned.