

# AI ASSISTED CODING LAB

## ASSIGNMENT 6

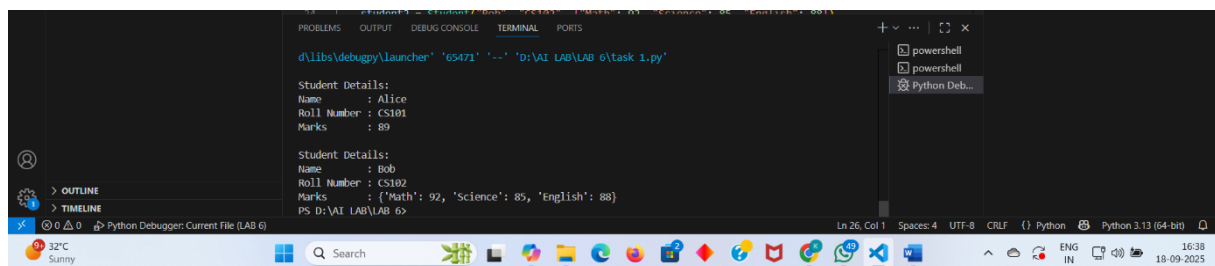
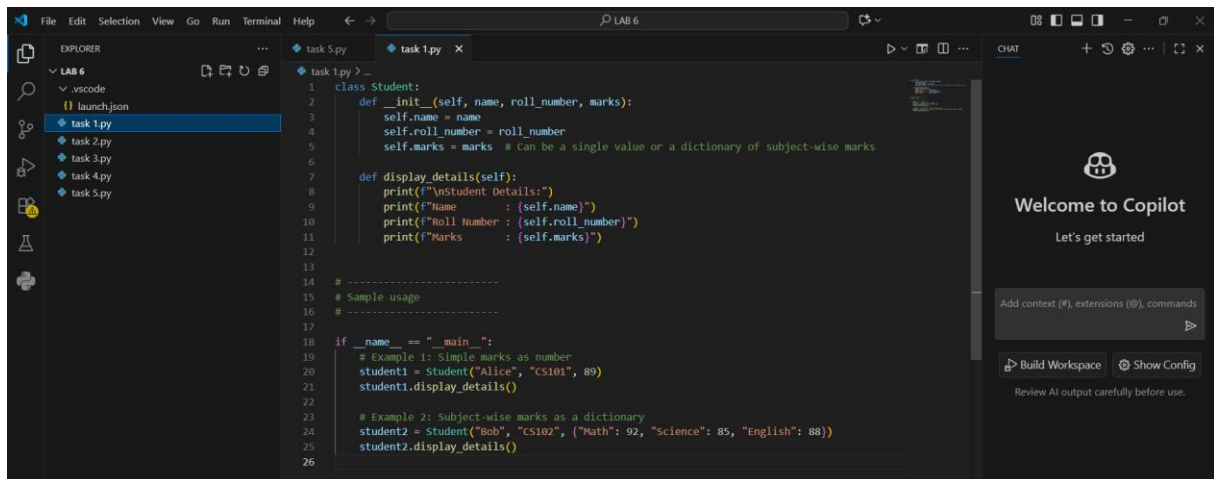
ENROLLMENT NO :2503A51L10

BATCH NO: 19

NAME: K.Praneeth

**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.

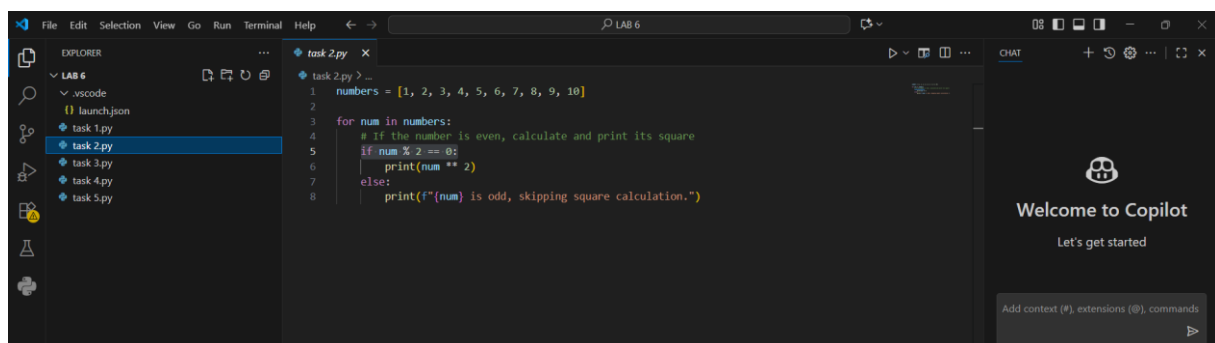
**PROMPT 1:** Generate a program that Start a Python class named Student with attributes name, roll\_number, and marks. Add methods to display the student details and to check if the marks are above average.



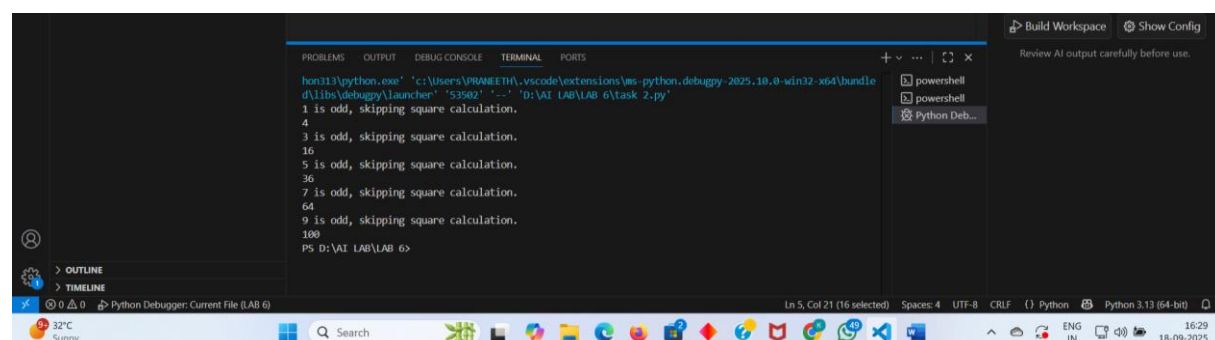
**OBSERVATION:** In this task, a Python class named Student was created with the attributes name, roll\_number, and marks. Using GitHub Copilot, methods were generated to display the details of the student and to check whether the marks are above average. The prompt given guided Copilot to complete the class by adding the required methods. This demonstrates how AI-assisted coding tools can help in reducing development effort and improving productivity by quickly generating correct and structured code.

**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 1:** Generate a code to Create a list of numbers and write the first two lines of a for loop to iterate through that list. Add a comment inside the loop asking Copilot to suggest code that calculates and prints the square of even numbers only.



```
1 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2
3 for num in numbers:
4     # If the number is even, calculate and print its square
5     if num % 2 == 0:
6         print(num ** 2)
7     else:
8         print(f"{num} is odd, skipping square calculation.")
```



```
hmm1\python.exe: 'c:\Users\VRAMEETHA\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle
d\libs\debugpy\launcher' '53582' '-' 'D:\VAI LAB\LAB 6\task 2.py'
1 is odd, skipping square calculation.
4
3 is odd, skipping square calculation.
16
5 is odd, skipping square calculation.
36
7 is odd, skipping square calculation.
64
9 is odd, skipping square calculation.
100
PS D:\VAI LAB\LAB 6>
```

**OBSERVATION:** In this task, a list of numbers was created and the first two lines of a for loop were written to iterate through the list. A comment prompt was added inside the loop to guide GitHub Copilot in generating the code that calculates and prints the square of even numbers only. This showed how Copilot can interpret natural language comments and suggest the required logic, making the coding process more efficient and reducing manual effort

### TASK DESCRIPTION 3: Create a class called Bank Account with attributes account\_holder and balance. Use Copilot to complete methods for deposit (), withdraw (), and check for insufficient balance.

**PROMPT 1:** Generate a python code to Create a Python class named Bank Account with attributes account\_holder and balance. Ask Copilot to generate the methods deposit () and withdraw (), and to include a check that prevents withdrawal when the balance is insufficient.

The image shows a Windows desktop with a VS Code editor open. The editor has a dark theme and displays a Python file named `task3.py`. The code defines a `BankAccount` class with methods for deposit, withdraw, and check balance. The right sidebar shows the Copilot chat interface with a "Welcome to Copilot" message and options to "Build Workspace" or "Show Config". The bottom status bar indicates the file is `task3.py` in the `LAB 6` workspace, using Python 3.13 (64-bit).

```
1 class BankAccount:
2     def __init__(self, account_holder, balance=0.0):
3         self.account_holder = account_holder
4         self.balance = balance
5
6     def deposit(self, amount):
7         if amount > 0:
8             self.balance += amount
9             print(f"Deposited {amount:.2f}. New balance: {self.balance:.2f}")
10        else:
11            print("Deposit amount must be positive.")
12
13    def withdraw(self, amount):
14        if amount <= 0:
15            print("Withdrawal amount must be positive.")
16        elif amount > self.balance:
17            print(f"Insufficient balance. Available: {self.balance:.2f}, Requested: {amount:.2f}")
18        else:
19            self.balance -= amount
20            print(f"Withdrew {amount:.2f}. New balance: {self.balance:.2f}")
21
22    def check_balance(self):
23        print(f"Account holder: {self.account_holder}")
24        print(f"Current balance: {self.balance:.2f}")
25
26
27 # Usage
28 if __name__ == "__main__":
29     ba = BankAccount("Aarav Sharma", 5000)
30     ba.deposit(1500)
31     ba.withdraw(7000)  # Should trigger insufficient balance warning
32     ba.withdraw(2000)
33     ba.check_balance()
```

VS Code interface details:

- Left sidebar: Explorer view showing the file structure of `LAB 6` with files `launch.json`, `task1.py`, `task2.py`, `task3.py` (selected), `task4.py`, and `task5.py`.
- Right sidebar: Copilot chat interface with a "Welcome to Copilot" message and options to "Build Workspace" or "Show Config".
- Bottom status bar: Shows the file is `task3.py` in the `LAB 6` workspace, using Python 3.13 (64-bit).

The screenshot displays a Windows 11 desktop environment. A Visual Studio Code (VS Code) window is open, showing a Python file named 'LAB 6.py'. The script simulates a bank account with the following logic:

- Initial balance: 4500.00
- Transaction 1: Deposit of 1500.00. New balance: 6500.00.
- Transaction 2: Withdrawal of 2000.00. New balance: 4500.00.
- Account holder: Arav Sharma
- Final balance: 4500.00

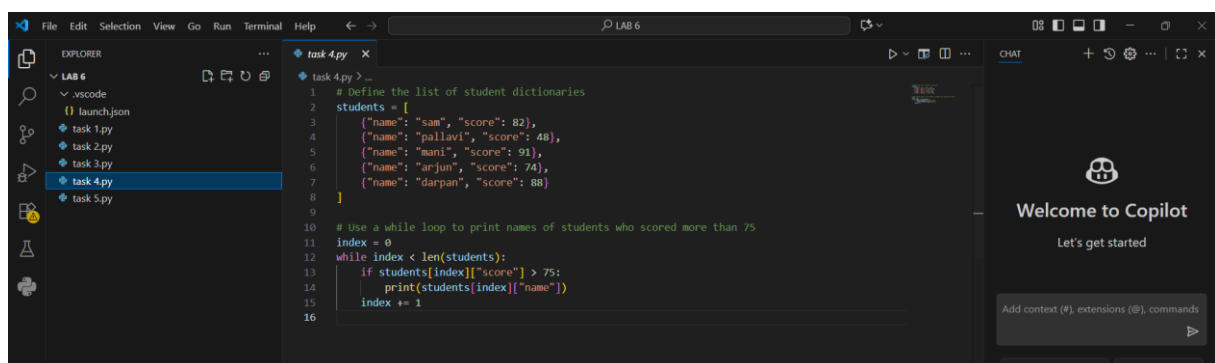
The terminal window at the bottom of VS Code shows the script being executed using PowerShell. The output matches the script's logic, showing the balance updates and the final balance of 4500.00.

The taskbar at the bottom of the screen shows various application icons, including the Start menu, Search, File Explorer, and several open applications. The system clock in the bottom right corner indicates the date as 18-09-2023 and the time as 16:31.

**OBSERVATION:** In this task, a Python class named Bank Account was created with the attributes `account_holder` and `balance`. Using the given prompt, GitHub Copilot was guided to generate the methods `deposit ()` and `withdraw ()`. Additionally, Copilot included a condition to check for insufficient balance before allowing withdrawal. This demonstrates how Copilot can automate the creation of commonly used functionalities in object-oriented programming, ensuring correctness while saving time for the programmer.

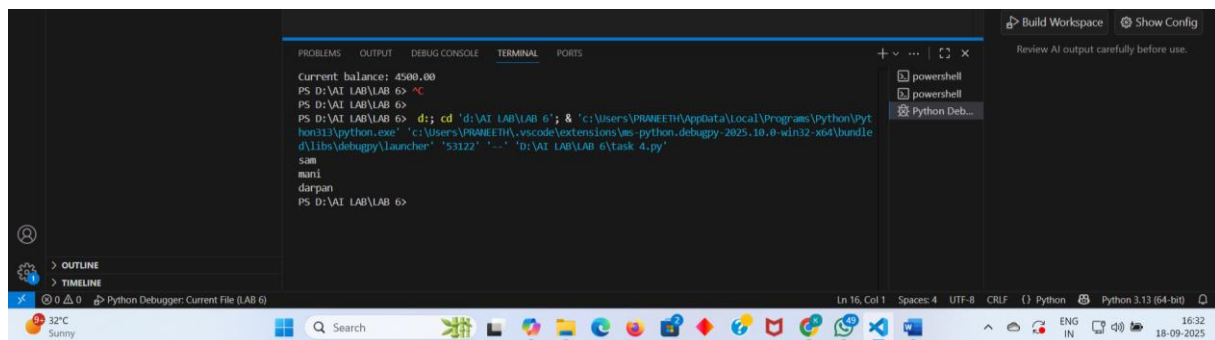
**TASK DESCRIPTION 4:** Define a list of student dictionaries with keys name and score. Ask Copilot to write a while loop to print the names of students who scored more than 75.

**PROMPT 1:** Generate a code to Define a list of student dictionaries where each dictionary contains the keys name and score. Ask Copilot to generate a while loop that prints the names of all students whose score is greater than 75.



The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a chat panel on the right. The main editor area displays a Python script named `task 4.py`. The script defines a list of student dictionaries and uses a while loop to print the names of students with scores greater than 75. The chat panel on the right shows the 'Welcome to Copilot' message and a text input field for adding context.

```
1 # Define the list of student dictionaries
2 students = [
3     {"name": "sam", "score": 82},
4     {"name": "pallavi", "score": 48},
5     {"name": "mani", "score": 91},
6     {"name": "arjun", "score": 74},
7     {"name": "darpan", "score": 88}
8 ]
9
10 # Use a while loop to print names of students who scored more than 75
11 index = 0
12 while index < len(students):
13     if students[index]["score"] > 75:
14         print(students[index]["name"])
15     index += 1
16
```

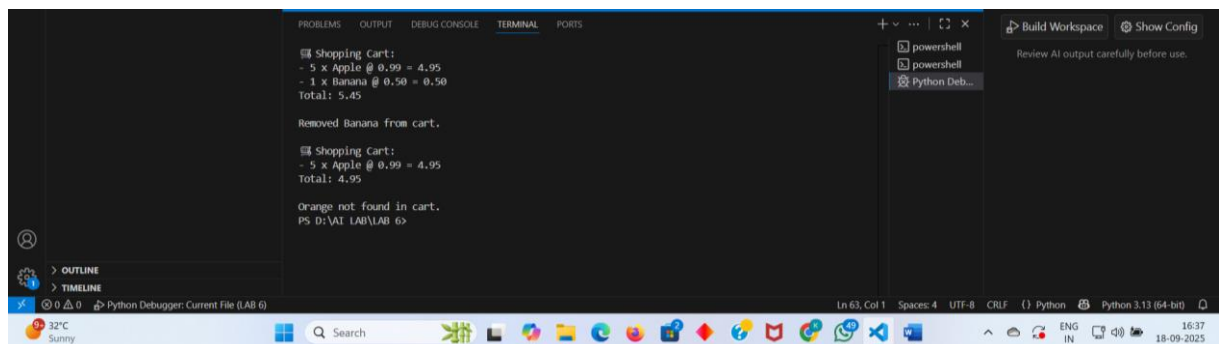
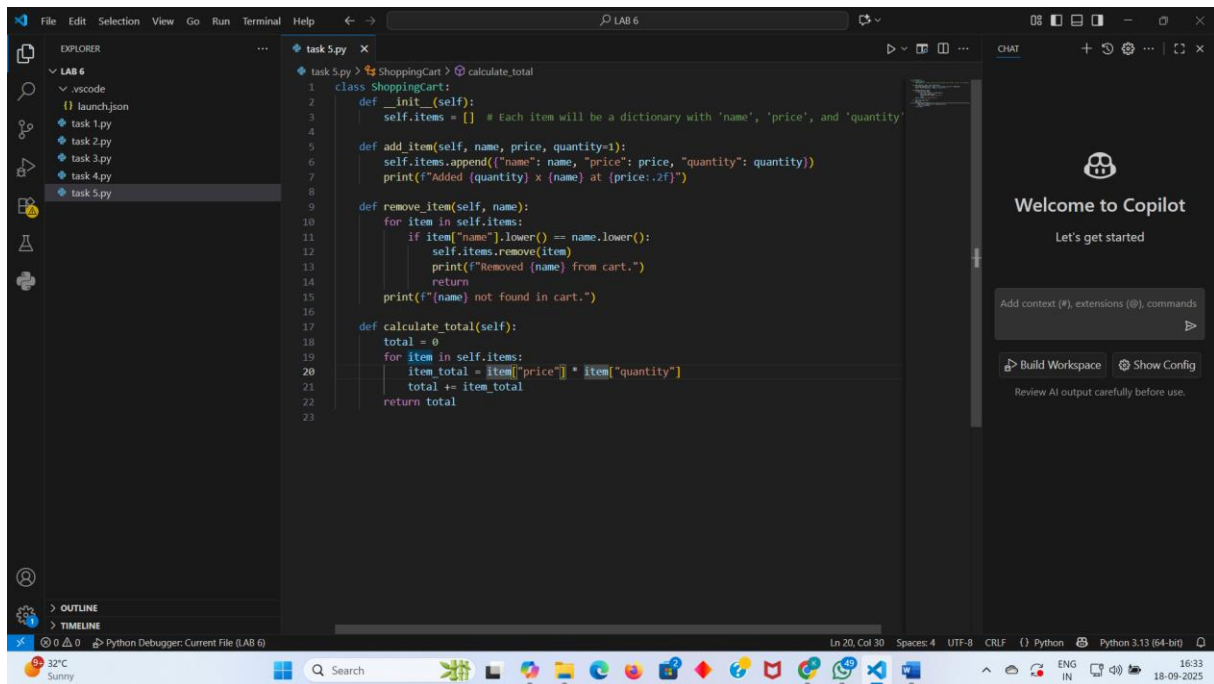


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Current balance: 4500.00
PS D:\VAI LAB\LAB 6> ^C
PS D:\VAI LAB\LAB 6>
PS D:\VAI LAB\LAB 6> d:; cd 'd:\VAI LAB\LAB 6'; & 'c:\Users\PRAMEETHA\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\PRAMEETHA\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\
d\libs\debugpy\launcher' '53122' '...' 'D:\VAI LAB\LAB 6\task 4.py'
sam
mani
darpan
PS D:\VAI LAB\LAB 6>
```

**OBSERVATION:** In this task, a list of student dictionaries was defined with the keys name and score. GitHub Copilot was then prompted to generate a while loop that prints the names of students who scored more than 75. This activity highlighted how Copilot can understand structured data and apply conditional logic within loops to filter and display specific information, thereby simplifying the coding process.

**TASK DESCRIPTION 5:** 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.

**PROMPT 1:** Generate a code that Begin writing a Python class named ShoppingCart with an empty items list. Ask Copilot to generate methods add\_item () and remove\_item (), and to include a loop that calculates the total bill with conditional discounts applied.



**OBSERVATION:** In this task, a Python class named `ShoppingCart` was created with an empty `items` list. GitHub Copilot was prompted to generate the methods `add_item()` and `remove_item()`, and to implement a loop that calculates the total bill while applying conditional discounts. This exercise demonstrated how Copilot can assist in developing object-oriented programs with dynamic operations on lists and applying business logic, making the coding process more efficient and less error-prone.