AI Assignment-5.1

Name: A. MANIDEEPIKA

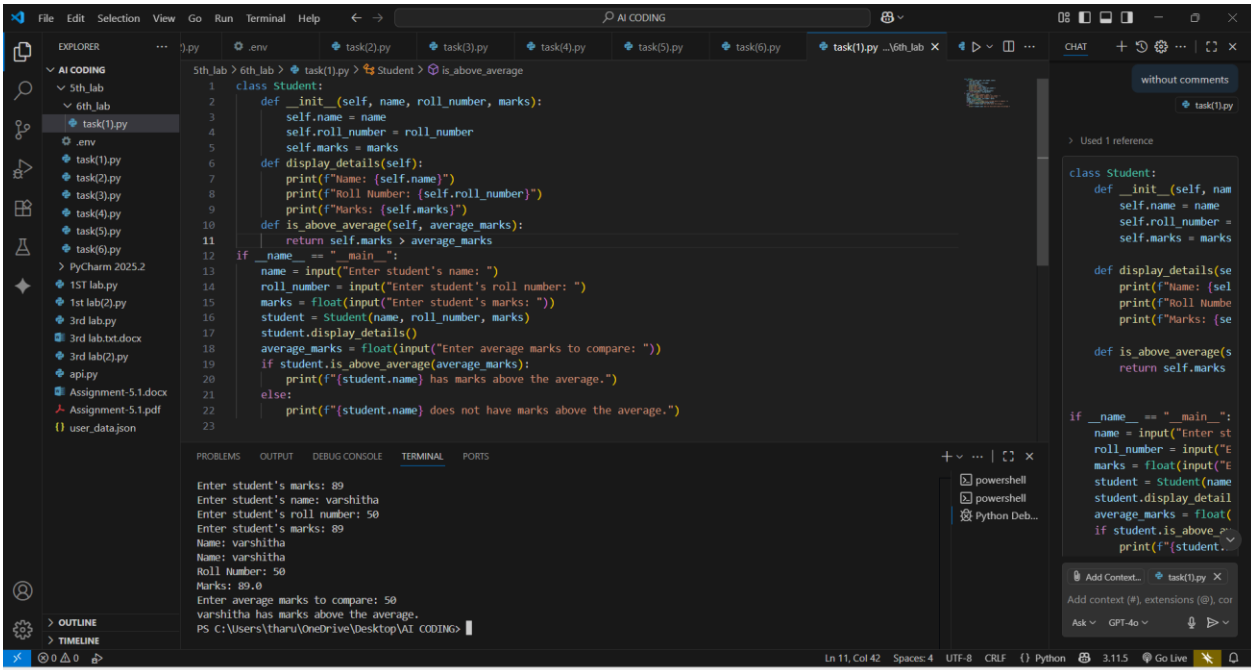
HT.NO: 2403A52052

Batch: AIB03

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: Start a Python class named Student with attributes name, roll number, and marks.

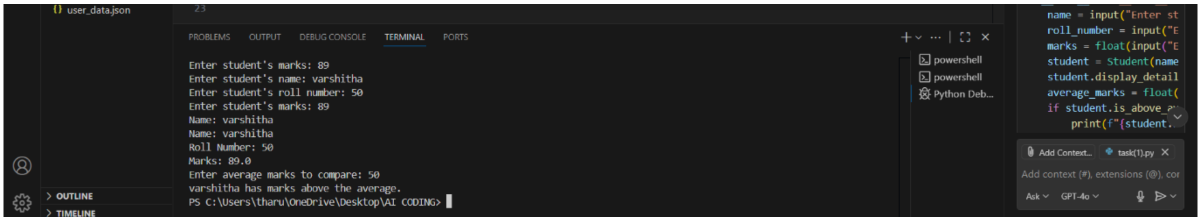
CODE:



```python
class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks
    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Marks: {self.marks}")
    def is_above_average(self, average_marks):
        return self.marks > average_marks
if __name__ == "__main__":
    name = input("Enter student's name: ")
    roll_number = input("Enter student's roll number: ")
    marks = float(input("Enter student's marks: "))
    student = Student(name, roll_number, marks)
    student.display_details()
    average_marks = float(input("Enter average marks to compare: "))
    if student.is_above_average(average_marks):
        print(f"{student.name} has marks above the average.")
    else:
        print(f"{student.name} does not have marks above the average.")
```

```
Enter student's marks: 89
Enter student's name: varshitha
Enter student's roll number: 50
Enter student's marks: 89
Name: varshitha
Name: varshitha
Roll Number: 50
Marks: 89.0
Enter average marks to compare: 50
varshitha has marks above the average.
PS C:\Users\tharu\OneDrive\Desktop\AI CODING>
```

OP:



```
Enter student's marks: 89
Enter student's name: varshitha
Enter student's roll number: 50
Enter student's marks: 89
Name: varshitha
Name: varshitha
Roll Number: 50
Marks: 89.0
Enter average marks to compare: 50
varshitha has marks above the average.
PS C:\Users\tharu\OneDrive\Desktop\AI CODING> 
```

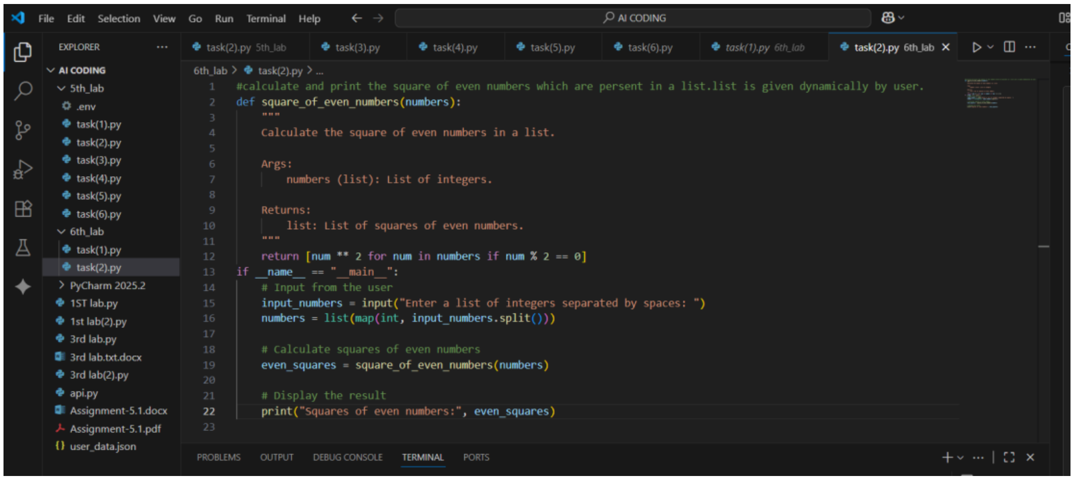
```
name = input("Enter st
roll_number = input("E
marks = float(input("E
student = Student(name
student.display_detail
average_marks = float(
if student.is_above_a
    print(f"{student.
```

Observation: The AI-generated code was accurate and complete. It produced correct methods for displaying details and checking above-average marks. The suggestions matched Python standards and required little or no correction. The lab helped me understand how AI can assist in coding, but also the need to carefully check the code.

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.

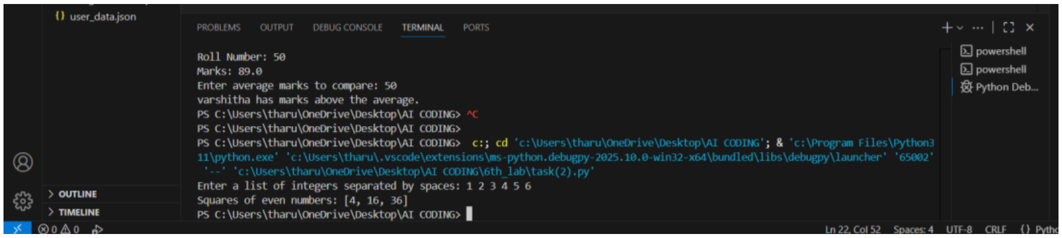
Prompt: calculate and print the square of even numbers which are persent in a list.list is given dynamically by user

CODE:



```python
#calculate and print the square of even numbers which are persent in a list.list is given dynamically by user.
def square_of_even_numbers(numbers):
    """
    Calculate the square of even numbers in a list.

    Args:
        numbers (list): List of integers.

    Returns:
        list: List of squares of even numbers.
    """
    return [num ** 2 for num in numbers if num % 2 == 0]
if __name__ == "__main__":
    # Input from the user
    input_numbers = input("Enter a list of integers separated by spaces: ")
    numbers = list(map(int, input_numbers.split()))

    # Calculate squares of even numbers
    even_squares = square_of_even_numbers(numbers)

    # Display the result
    print("Squares of even numbers:", even_squares)
```

OP:



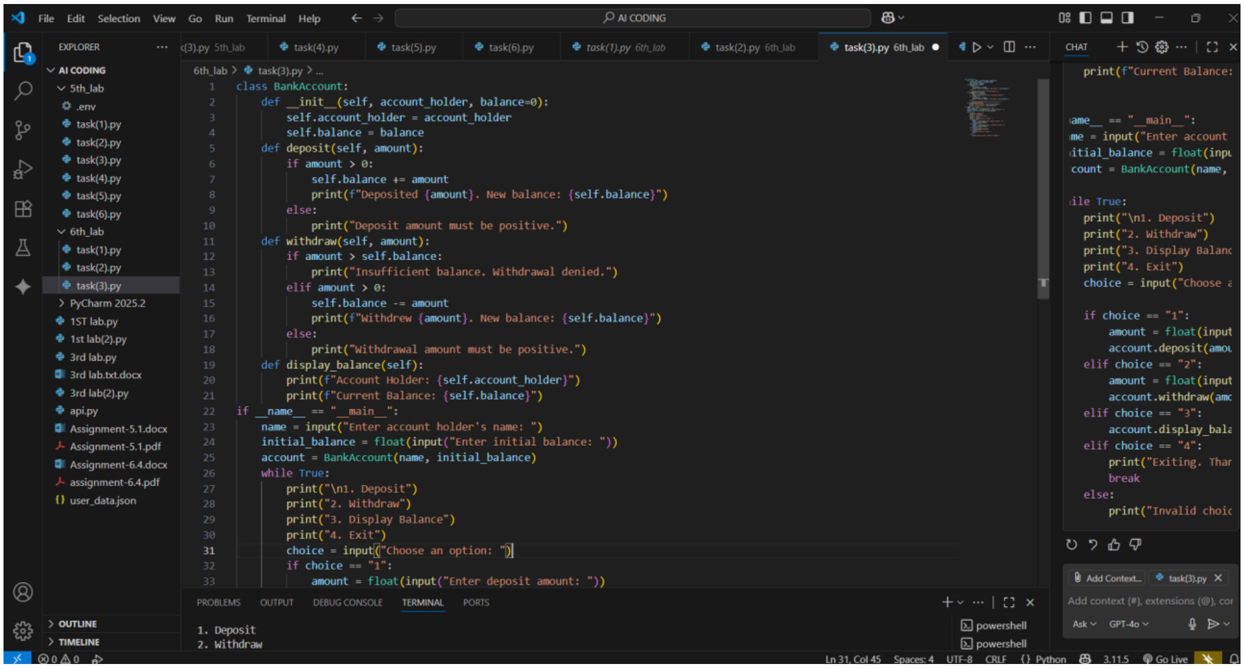
```
Roll Number: 50
Marks: 89.0
Enter average marks to compare: 50
varshitha has marks above the average.
PS C:\Users\tharu\OneDrive\Desktop\AI CODING> ^C
PS C:\Users\tharu\OneDrive\Desktop\AI CODING>
PS C:\Users\tharu\OneDrive\Desktop\AI CODING>  c:; cd 'c:\Users\tharu\OneDrive\Desktop\AI CODING'; & 'c:\Program Files\Python3
11\python.exe' 'c:\Users\tharu\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '65002'
 '--' 'c:\Users\tharu\OneDrive\Desktop\AI CODING\6th_lab\task(2).py'
Enter a list of integers separated by spaces: 1 2 3 4 5 6
Squares of even numbers: [4, 16, 36]
PS C:\Users\tharu\OneDrive\Desktop\AI CODING> 
```

Observation: I just wrote the comment to calculate squares of even numbers and asked AI to complete the logic. The AI used a list comprehension to check even numbers and square them, which was both correct and efficient. The program took input from the user, processed it, and displayed the result without errors. The code worked as expected, and the AI's completion was accurate and simple to understand.

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:Complete the code using deposit(), withdraw() methods. Even check for the insufficient balance.

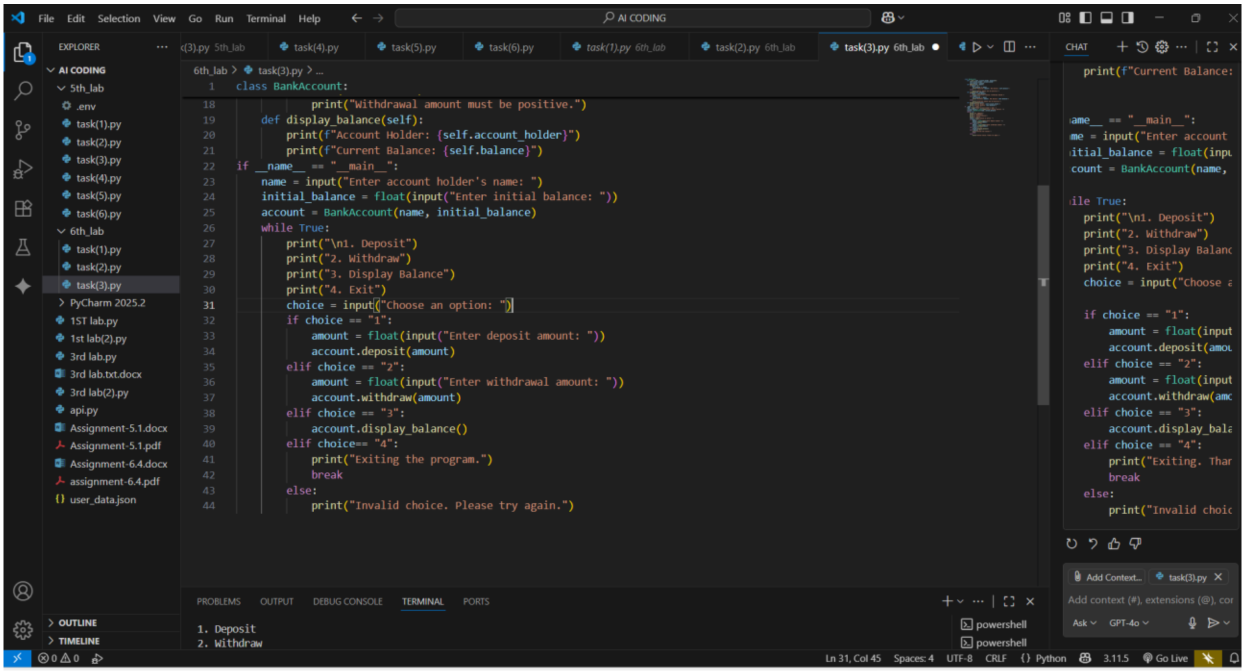
CODE:



```python
class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited {amount}. New balance: {self.balance}")
        else:
            print("Deposit amount must be positive.")
    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance. Withdrawal denied.")
        elif amount > 0:
            self.balance -= amount
            print(f"Withdrew {amount}. New balance: {self.balance}")
        else:
            print("Withdrawal amount must be positive.")
    def display_balance(self):
        print(f"Account Holder: {self.account_holder}")
        print(f"Current Balance: {self.balance}")
if __name__ == "__main__":
    name = input("Enter account holder's name: ")
    initial_balance = float(input("Enter initial balance: "))
    account = BankAccount(name, initial_balance)
    while True:
        print("\n1. Deposit")
        print("2. Withdraw")
        print("3. Display Balance")
        print("4. Exit")
        choice = input("Choose an option: ")
        if choice == "1":
            amount = float(input("Enter deposit amount: "))
```

```
1. Deposit
2. Withdraw
```

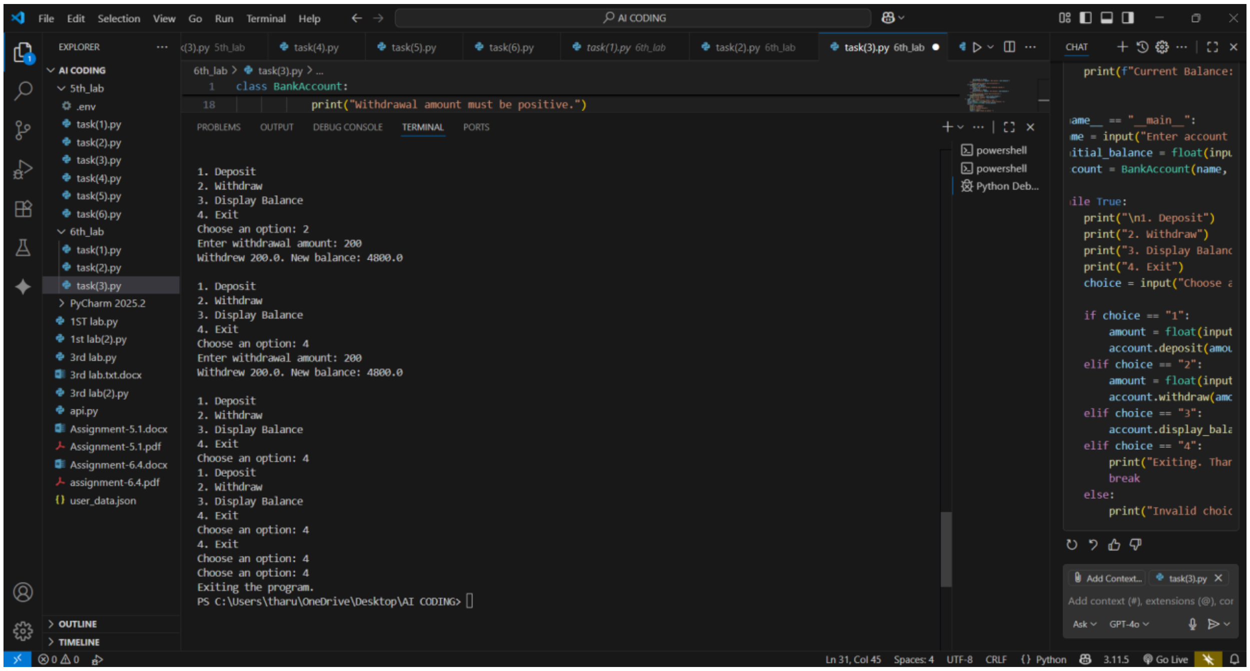
```python
    print(f"Current Balance:

ame__ == "__main__":
me = input("Enter account
itial_balance = float(inpu
count = BankAccount(name,

ile True:
    print("\n1. Deposit")
    print("2. Withdraw")
    print("3. Display Balanc
    print("4. Exit")
    choice = input("Choose a

    if choice == "1":
        amount = float(input
        account.deposit(amou
    elif choice == "2":
        amount = float(input
        account.withdraw(amc
    elif choice == "3":
        account.display_bala
    elif choice == "4":
        print("Exiting. Thar
        break
    else:
        print("Invalid choic
```



```python
class BankAccount:
            print("Withdrawal amount must be positive.")
    def display_balance(self):
        print(f"Account Holder: {self.account_holder}")
        print(f"Current Balance: {self.balance}")
if __name__ == "__main__":
    name = input("Enter account holder's name: ")
    initial_balance = float(input("Enter initial balance: "))
    account = BankAccount(name, initial_balance)
    while True:
        print("\n1. Deposit")
        print("2. Withdraw")
        print("3. Display Balance")
        print("4. Exit")
        choice = input("Choose an option: ")
        if choice == "1":
            amount = float(input("Enter deposit amount: "))
            account.deposit(amount)
        elif choice == "2":
            amount = float(input("Enter withdrawal amount: "))
            account.withdraw(amount)
        elif choice == "3":
            account.display_balance()
        elif choice== "4":
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please try again.")
```

```
1. Deposit
2. Withdraw
```

OP:



```python
class BankAccount:
            print("Withdrawal amount must be positive.")
```

```
    print(f"Current Balance:

ame__ == "__main__":
me = input("Enter account
itial_balance = float(inpu
count = BankAccount(name,

ile True:
    print("\n1. Deposit")
    print("2. Withdraw")
    print("3. Display Balanc
    print("4. Exit")
    choice = input("Choose a

    if choice == "1":
        amount = float(input
        account.deposit(amou
    elif choice == "2":
        amount = float(input
        account.withdraw(amc
    elif choice == "3":
        account.display_bala
    elif choice == "4":
        print("Exiting. Thar
        break
    else:
        print("Invalid choic
```

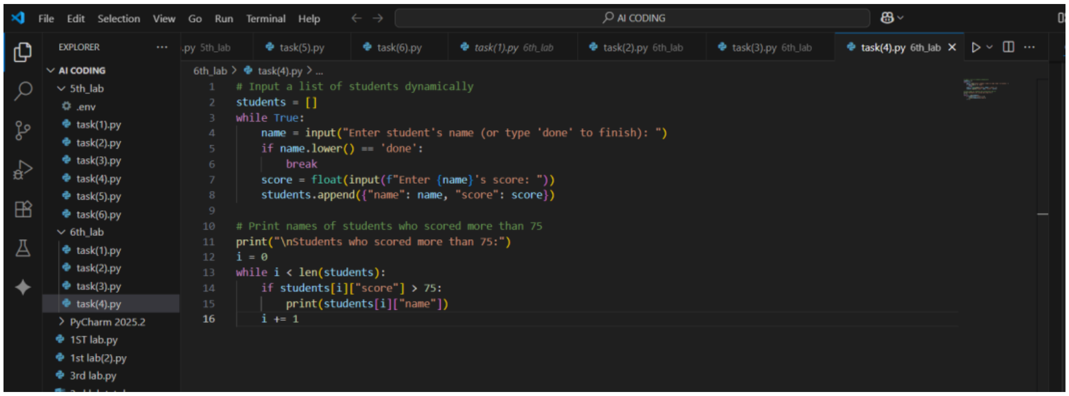
```
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Choose an option: 2
Enter withdrawal amount: 200
Withdrew 200.0. New balance: 4800.0

1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Choose an option: 4
Enter withdrawal amount: 200
Withdrew 200.0. New balance: 4800.0

1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Choose an option: 4
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Choose an option: 4
4. Exit
Choose an option: 4
Choose an option: 4
Exiting the program.
PS C:\Users\tharu\OneDrive\Desktop\AI CODING> 
```

Observation: I wrote the structure of the Bank Account class and let AI complete the methods for deposit, withdraw, and balance display. The AI also suggested a loop for menu options, which worked correctly. The code ran without errors and produced the expected results. The AI's suggestions were accurate, clear, and required no major corrections. This lab helped me understand how AI can simplify writing logic with loops, conditionals, and methods

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.

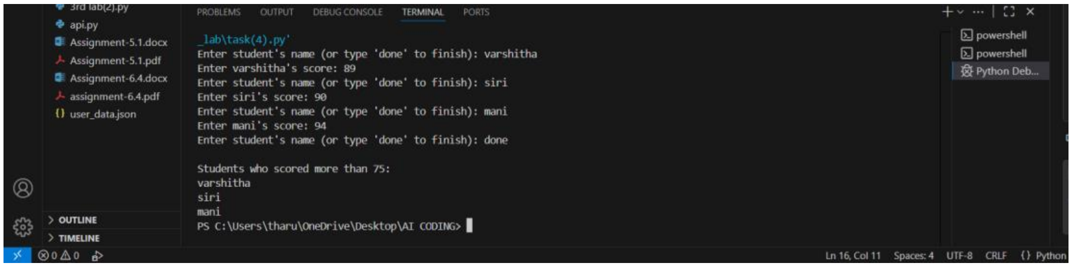
Prompt: write a while loop to print the names of students who scored more than 75

CODE:



```python
# Input a list of students dynamically
students = []
while True:
    name = input("Enter student's name (or type 'done' to finish): ")
    if name.lower() == 'done':
        break
    score = float(input(f"Enter {name}'s score: "))
    students.append({"name": name, "score": score})

# Print names of students who scored more than 75
print("\nStudents who scored more than 75:")
i = 0
while i < len(students):
    if students[i]["score"] > 75:
        print(students[i]["name"])
    i += 1
```

OP:



```
_lab\task(4).py'
Enter student's name (or type 'done' to finish): varshitha
Enter varshitha's score: 89
Enter student's name (or type 'done' to finish): siri
Enter siri's score: 90
Enter student's name (or type 'done' to finish): mani
Enter mani's score: 94
Enter student's name (or type 'done' to finish): done

Students who scored more than 75:
varshitha
siri
mani
PS C:\Users\tharu\OneDrive\Desktop\AI CODING> 
```

Observation: wrote the input structure for student details, and AI completed the loop logic to check which students scored above 75. It used a list of dictionaries to store the data and a while loop with conditions to filter results. The program worked correctly when tested, and the output matched the requirement. The AI's code was accurate, complete, and easy to follow

TASK 5:

Prompt:

CODE:

OP:

Observation: