AI Assisted Coding

Name:D.udayan

Hall.no:2303A54017

Assignment: 6.3

Task1:

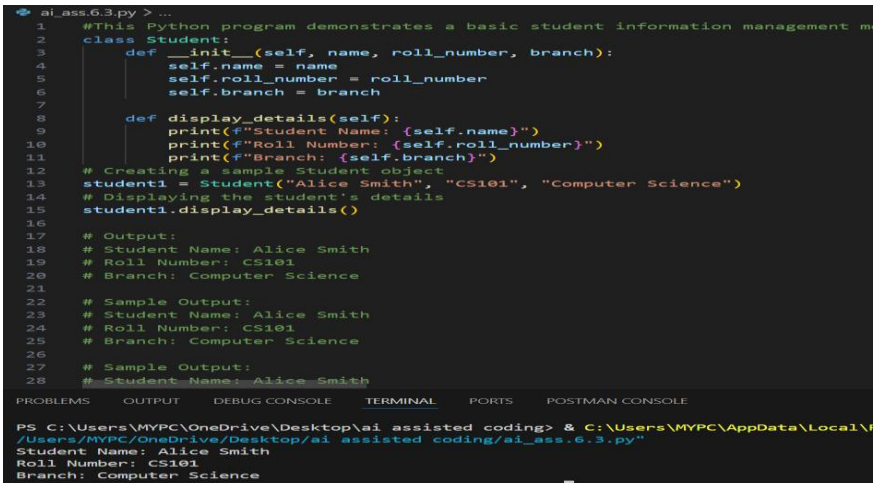
Task Description #1: Classes (Student Class)

Scenario
You are developing a simple student information management module.

Task
- Use an AI tool (GitHub Copilot / Cursor AI / Gemini) to complete a Student class.
- The class should include attributes such as name, roll number, and branch.
- Add a method display_details() to print student information.
- Execute the code and verify the output.
- Analyze the code generated by the AI tool for correctness and clarity.

Expected Output #1
- A Python class with a constructor (__init__) and a display_details() method.
- Sample object creation and output displayed on the console.
- Brief analysis of AI-generated code.

Ai generated code:



Prompt:

This Python program demonstrates a basic student information management module. It defines a class with attributes for the student's name, roll number, and branch. A constructor is used to initialize these values when a student object is created. The class also includes a method, which prints the student's information in a clear and readable format. After defining the class, a sample object is created, and the method is called to display the output on the console.

Task2:

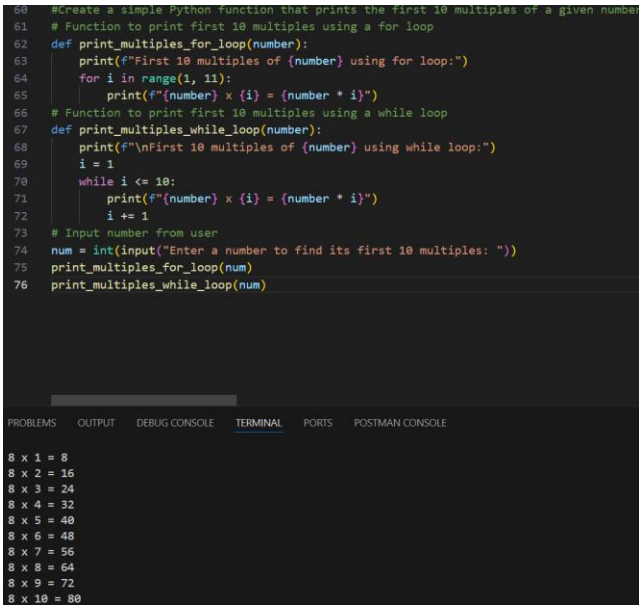
Task Description #2: Loops (Multiples of a Number)

Scenario
You are writing a utility function to display multiples of a given number.

Task
- Prompt the AI tool to generate a function that prints the first 10 multiples of a given number using a loop.
- Analyze the generated loop logic.
- Ask the AI to generate the same functionality using another controlled looping structure (e.g., while instead of for).

Expected Output #2
- Correct loop-based Python implementation.
- Output showing the first 10 multiples of a number.
- Comparison and analysis of different looping approaches.

Ai generated code:



Prompt:

Create a simple Python function that prints the first 10 multiples of a given number using a loop. First, implement the solution using a for loop and display the output clearly on the console. Then, write the same functionality again using a while loop instead of a for loop. Keep the code simple, beginner-friendly, and easy to understand. Make sure the program runs without errors and clearly shows the first 10 multiples of the number.

Task3:

Task Description #3: Conditional Statements (Age Classification)

Scenario
You are building a basic classification system based on age.

Task
- Ask the AI tool to generate nested if-elif-else conditional statements to classify age groups (e.g., child, teenager, adult, senior).
- Analyze the generated conditions and logic.
- Ask the AI to generate the same classification using alternative conditional structures (e.g., simplified conditions or dictionary-based logic).

Expected Output #3
- A Python function that classifies age into appropriate groups.
- Clear and correct conditional logic.
- Explanation of how the conditions work.

Ai generated code:

```python
def classify_age_nested(age):
    if age < 0:
        print("Invalid age")
    elif age <= 12:
        print("Child")
    elif age <= 19:
        print("Teenager")
    elif age <= 59:
        print("Adult")
    else:
        print("Senior")


def classify_age_simplified(age):
    categories = {
        range(0, 13): "Child",
        range(13, 20): "Teenager",
        range(20, 60): "Adult",
        range(60, 150): "Senior"
    }

    for age_range, category in categories.items():
        if age in age_range:
            print(category)
            return

    print("Invalid age")


user_age = int(input("Enter your age: "))
```

```python
94    def classify_age_simplified(age):
107       print("Invalid age")
108
109
110   user_age = int(input("Enter your age: "))
111
112   print("Classification using nested if-elif-else:")
113   classify_age_nested(user_age)
114
115   print("\nClassification using simplified conditions:")
116   classify_age_simplified(user_age)
117
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

```
/Users/MYPC/OneDrive/Desktop/ai assisted coding/ai_ass.6.3.py"
Enter your age: 9
Classification using nested if-elif-else:
Child

Classification using simplified conditions:
Child
```

Prompt:

Create a simple Python program that classifies a person into an age group based on their age. Use nested if-elif-else statements to categorize the age as child, teenager, adult, or senior. The program should include a function that takes age as input and prints the appropriate category. After that, implement the same age classification logic using an alternative approach, such as simplified conditions or a dictionary-based method. Keep the code clear, beginner-friendly, and easy to understand, and ensure it runs without errors.

Task4:

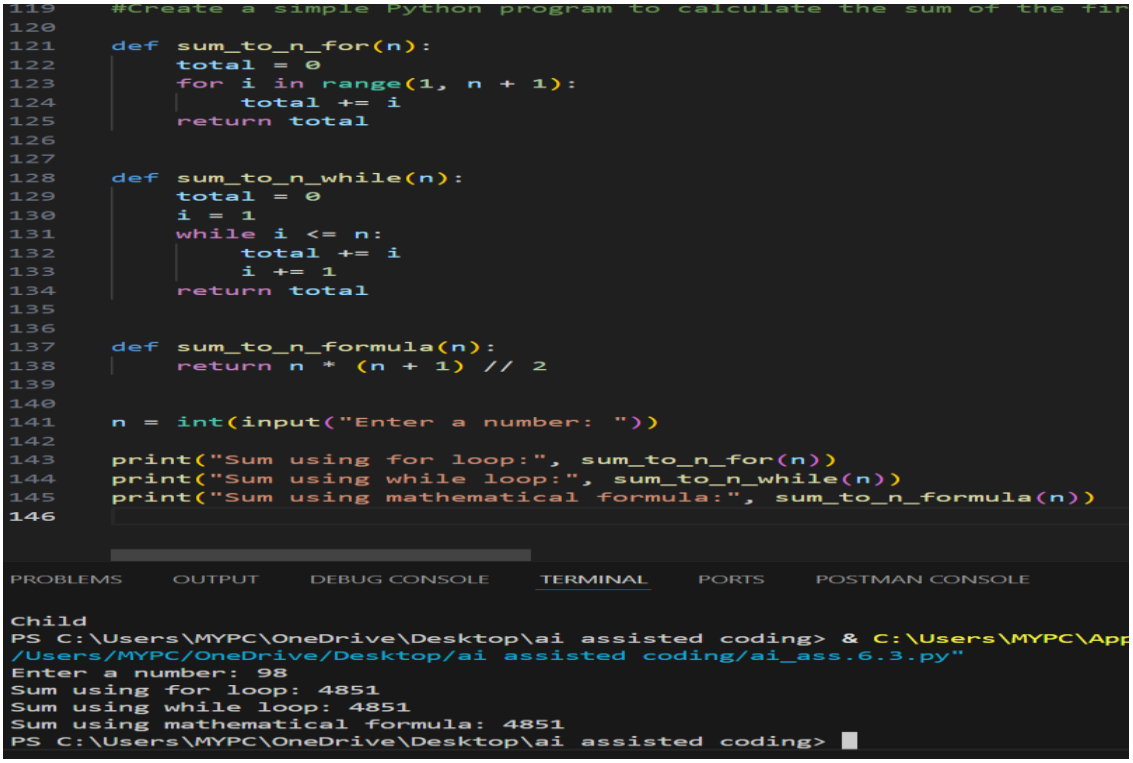
Task Description #4: For and While Loops (Sum of First n Numbers)

Scenario
You need to calculate the sum of the first n natural numbers.

Task
- Use AI assistance to generate a sum_to_n() function using a for loop.
- Analyze the generated code.
- Ask the AI to suggest an alternative implementation using a while loop or a mathematical formula.

Expected Output #4
- Python function to compute the sum of first n numbers.
- Correct output for sample inputs.
- Explanation and comparison of different approaches.

Ai generated code:



```
119     #Create a simple Python program to calculate the sum of the fir
120
121     def sum_to_n_for(n):
122         total = 0
123         for i in range(1, n + 1):
124             total += i
125         return total
126
127
128     def sum_to_n_while(n):
129         total = 0
130         i = 1
131         while i <= n:
132             total += i
133             i += 1
134         return total
135
136
137     def sum_to_n_formula(n):
138         return n * (n + 1) // 2
139
140
141     n = int(input("Enter a number: "))
142
143     print("Sum using for loop:", sum_to_n_for(n))
144     print("Sum using while loop:", sum_to_n_while(n))
145     print("Sum using mathematical formula:", sum_to_n_formula(n))
146
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    POSTMAN CONSOLE

Child
PS C:\Users\MYPC\OneDrive\Desktop\ai assisted coding> & C:\Users\MYPC\App
/Users/MYPC/OneDrive/Desktop/ai assisted coding/ai_ass.6.3.py"
Enter a number: 98
Sum using for loop: 4851
Sum using while loop: 4851
Sum using mathematical formula: 4851
PS C:\Users\MYPC\OneDrive\Desktop\ai assisted coding>
```

Prompt:

```
#Create a simple Python program to calculate the sum of the first n natural
numbers. First, write a function named sum_to_n(n) that uses a for loop to
compute the sum. Then, implement an alternative solution using a while loop or a
mathematical formula. Display the result for sample inputs and keep the code
clear, beginner-friendly, and easy to understand. Ensure the program runs without
errors and clearly shows the output
```

Tack5:

Task Description #5: Classes (Bank Account Class)

Scenario
You are designing a basic banking application.

Task
- Use AI tools to generate a Bank Account class with methods such as deposit(), withdraw(), and check_balance().
- Analyze the AI-generated class structure and logic.
- Add meaningful comments and explain the working of the code.

Expected Output #5
- Complete Python Bank Account class.
- Demonstration of deposit and withdrawal operations with updated balance.
- Well-commented code with a clear explanation.

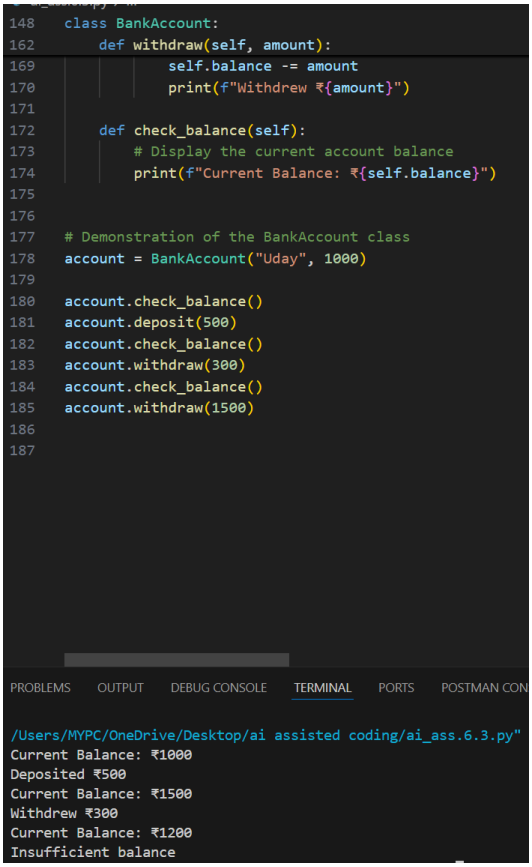
Ai generated code:

```python
class BankAccount:
    def __init__(self, account_holder, balance=0):
        # Initialize account holder name and starting balance
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        # Add money to the account
        if amount > 0:
            self.balance += amount
            print(f"Deposited ₹{amount}")
        else:
            print("Invalid deposit amount")

    def withdraw(self, amount):
        # Withdraw money if sufficient balance is available
        if amount <= 0:
            print("Invalid withdrawal amount")
        elif amount > self.balance:
            print("Insufficient balance")
        else:
            self.balance -= amount
            print(f"Withdrew ₹{amount}")

    def check_balance(self):
        # Display the current account balance
        print(f"Current Balance: ₹{self.balance}")
```



```
148   class BankAccount:
162       def withdraw(self, amount):
169               self.balance -= amount
170               print(f"Withdrew ₹{amount}")
171
172       def check_balance(self):
173           # Display the current account balance
174           print(f"Current Balance: ₹{self.balance}")
175
176
177   # Demonstration of the BankAccount class
178   account = BankAccount("Uday", 1000)
179
180   account.check_balance()
181   account.deposit(500)
182   account.check_balance()
183   account.withdraw(300)
184   account.check_balance()
185   account.withdraw(1500)
186
187
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CON

/Users/MYPC/OneDrive/Desktop/ai assisted coding/ai_ass.6.3.py"
Current Balance: ₹1000
Deposited ₹500
Current Balance: ₹1500
Withdrew ₹300
Current Balance: ₹1200
Insufficient balance
```

Prompt:

Create a simple Python program for a basic banking application. Define a BankAccount class that includes methods such as deposit(), withdraw(), and check_balance(). The class should store account holder details and the current balance. Demonstrate the working of the class by creating an account object and performing deposit and withdrawal operations while displaying the updated balance. Keep the code beginner-friendly, well-structured, and include meaningful comments explaining how each part of the code works.