Lab Assignment # 05

Course Title : AI Assistant Coding
Name of Student : B.Sai Varsha
Enrollment No. : 2303A54065
Batch No. : 48

Lab 6: AI-Based Code Completion – Classes, Loops, and Conditionals

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.

Prompt:

Create a Python Student class with attributes name, roll number, and branch.

Add a method display_details() to print student information.

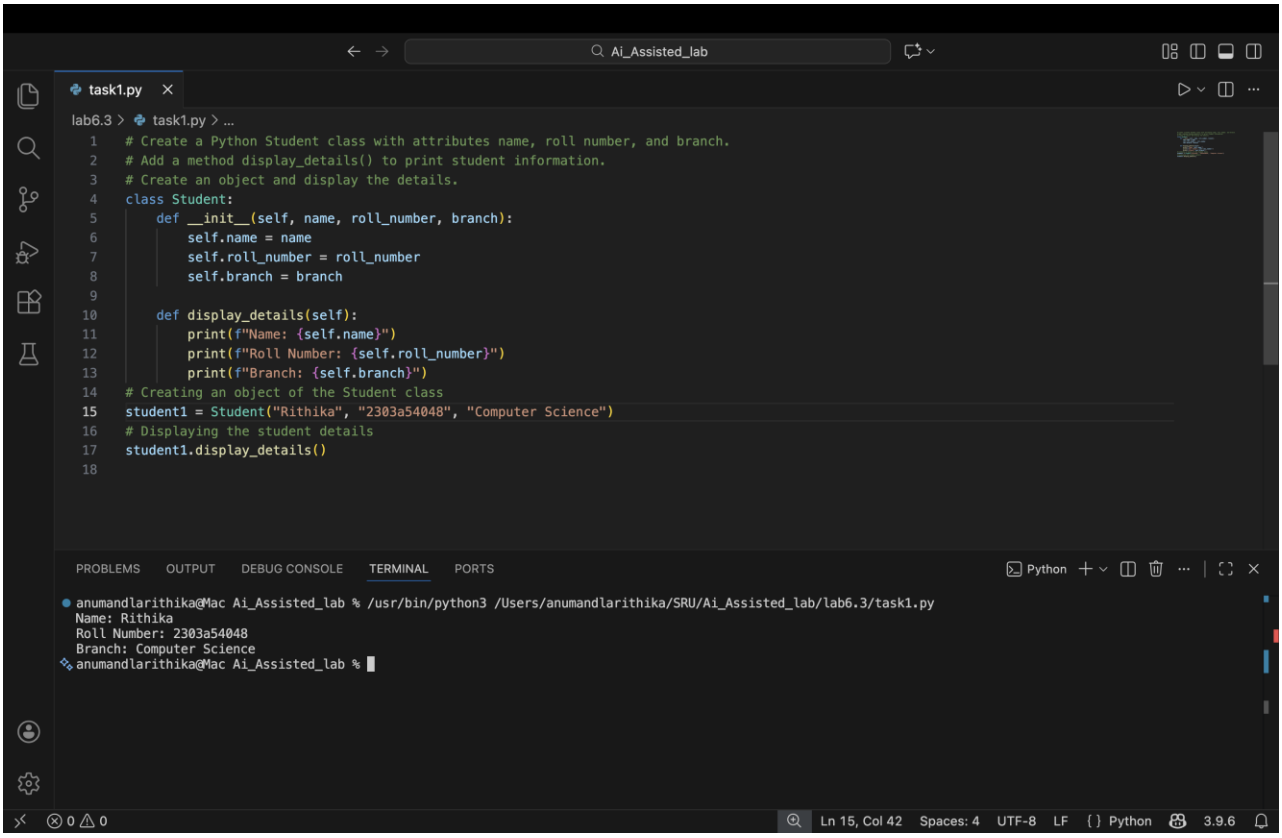
Create an object and display the details.

Code :

```
class Student:
def init(self, name, roll_number, branch):
self.name = name
self.roll_number = roll_number
self.branch = branch
def display_details(self):
        print("Name:", self.name)
        print("Roll Number:", self.roll_number)
        print("Branch:", self.branch)

student1 = Student("Rithika", "23CS101", "CSE")
student1.display_details()
```

Expected Output #1



Code Explanation

- The __init__() method initializes student details.
- display_details() prints all attributes.
- An object is created and the method is executed.

Comments / Analysis

- AI generated a correct class structure.
- Code is clear, readable, and follows OOP principles.

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).

Prompt 1:

Write a Python function to print the first 10 multiples of a given number using a for loop.

Code :

```
def print_multiples(num):
    for i in range(1, 11):
        print(num * i)
print_multiples(5)
```

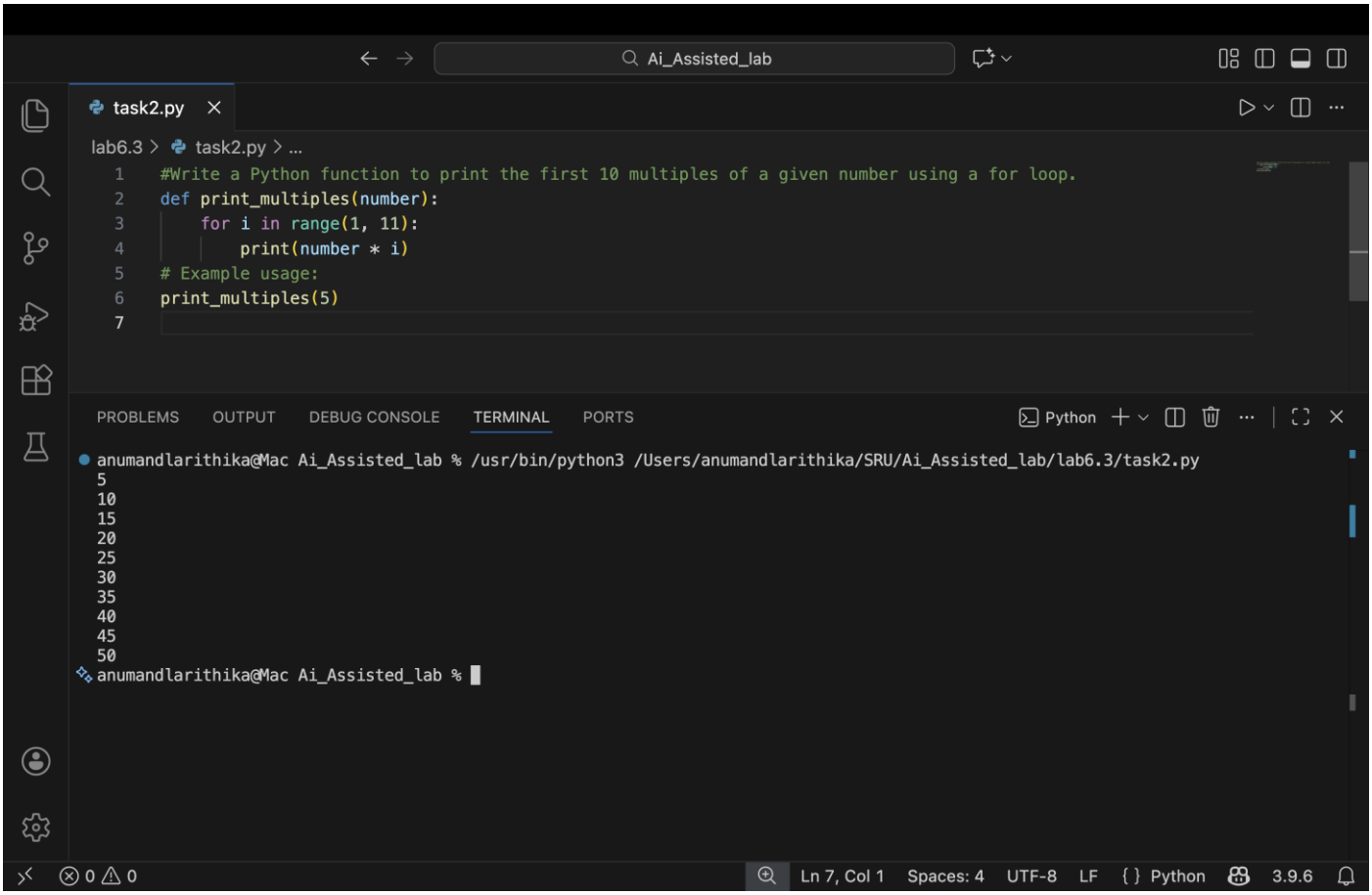
Prompt 2:

Write the same functionality using a while loop.

Code :

```
def print_multiples_while(num):
    i = 1
    while i <= 10:
        print(num * i)
        i += 1
print_multiples_while(5)
```

Expected Output #2



Code Explanation

- for loop iterates fixed number of times.
- while loop uses a condition to control execution.

Comments / Analysis

- Both loops give correct results.
- for loop is simpler and more readable.
- while loop offers flexible control.

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).

Prompt :

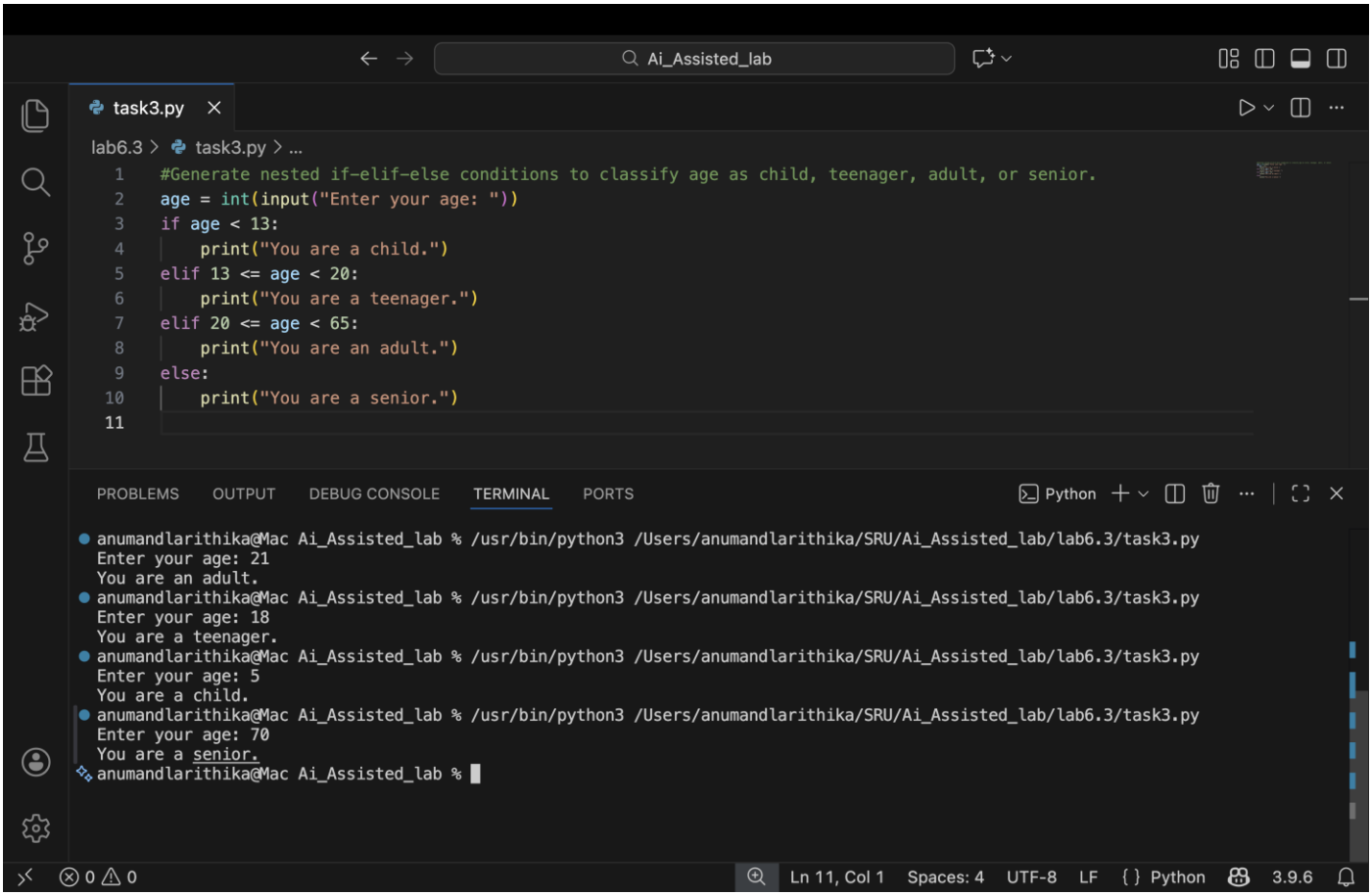
Generate nested if-elif-else conditions to classify age as child, teenager, adult, or senior.

Code :

```
def classify_age(age):
    if age < 13:
        return "Child"
    elif age < 20:
        return "Teenager"
    elif age < 60:
        return "Adult"
    else:
        return "Senior"

print(classify_age(18))
```

Expected Output #3



Code Explanation

- Conditions are checked in sequence.
- The correct age category is returned.

Comments / Analysis

- AI generated logical and efficient conditions.
- Easy to understand and maintain.

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.

Prompt 1:

Create a function sum_to_n(n) using a for loop.

Code :

```
def sum_to_n(n):
    total = 0
    for i in range(1, n + 1):
        total += i
    return total
print(sum_to_n(10))
```

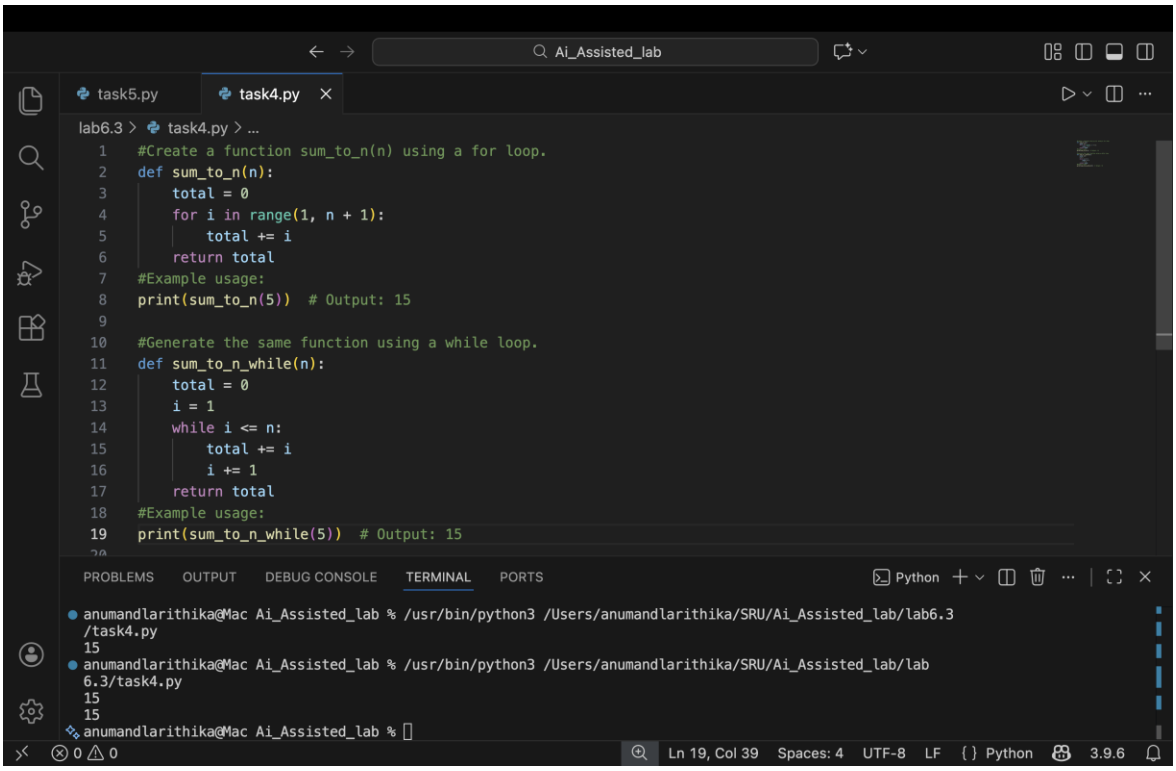
Prompt 2 :

Generate the same function using a while loop.

Code :

```
def sum_to_n_while(n):
    total = 0
    i = 1
    while i <= n:
        total += i
        i += 1
    return total
print(sum_to_n_while(10))
```

Expected Output #4



Code Explanation

- Loops accumulate the sum step-by-step.
- Both methods produce the same result.

Comments / Analysis

- for loop is concise.
- while loop is useful for dynamic conditions.

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.

Prompt :

Create a BankAccount class with deposit(), withdraw(), and check_balance() methods.

Add comments and demonstrate usage.

Code :

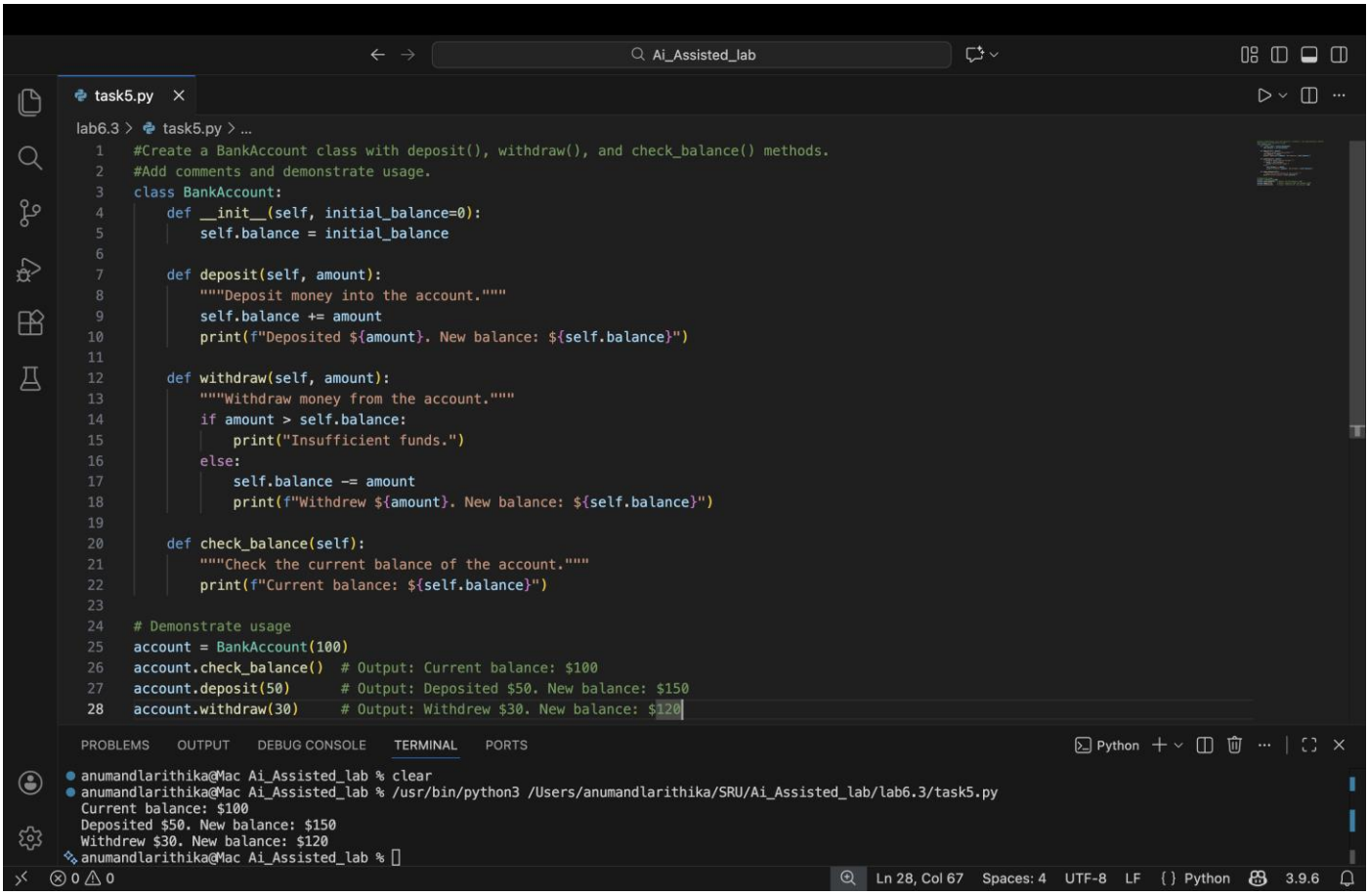
```
class BankAccount: def init(self, holder, balance=0): self.holder = holder self.balance = balance
def deposit(self, amount):
    self.balance += amount
    print("Deposited:", amount)

def withdraw(self, amount):
    if amount <= self.balance:
        self.balance -= amount
        print("Withdrawn:", amount)
    else:
        print("Insufficient balance")

def check_balance(self):
    print("Current Balance:", self.balance)

account = BankAccount("Rithika", 1000) account.deposit(500) account.withdraw(300)
account.check_balance()
```

Expected Output #5



Code Explanation

- Class models real-world bank operations.
- Methods update and display account balance.

Comments / Analysis

- AI generated a realistic and correct class.
- Proper validation is included for withdrawal.

Conclusion

GitHub Copilot effectively assisted in generating correct and readable Python code for classes, loops, and conditionals. The AI tool improved coding speed while maintaining accuracy and clarity.