# ASSIGNMENT_6.3

## NAME : V.NITHIN

## ROLL NO. : 2303A51652

## BATCH NO. : 09

Lab 6: AI-Based Code Completion – Classes, Loops, and Conditionals
Lab Objectives
- To explore AI-powered auto-completion features for core Python constructs such as classes, loops, and conditional statements.
- To analyze how AI tools suggest logic for object-oriented programming and control structures.
- To evaluate the correctness, readability, and completeness of AI-generated Python code.

Lab Outcomes (LOs)
After completing this lab, students will be able to:
- Use AI tools to generate and complete Python class definitions and methods.
- Understand and assess AI-suggested loop constructs for iterative tasks.
- Generate and evaluate conditional statements using AI-driven prompts.
- Critically analyze AI-assisted code for correctness, clarity, and efficiency.

## Task 1:

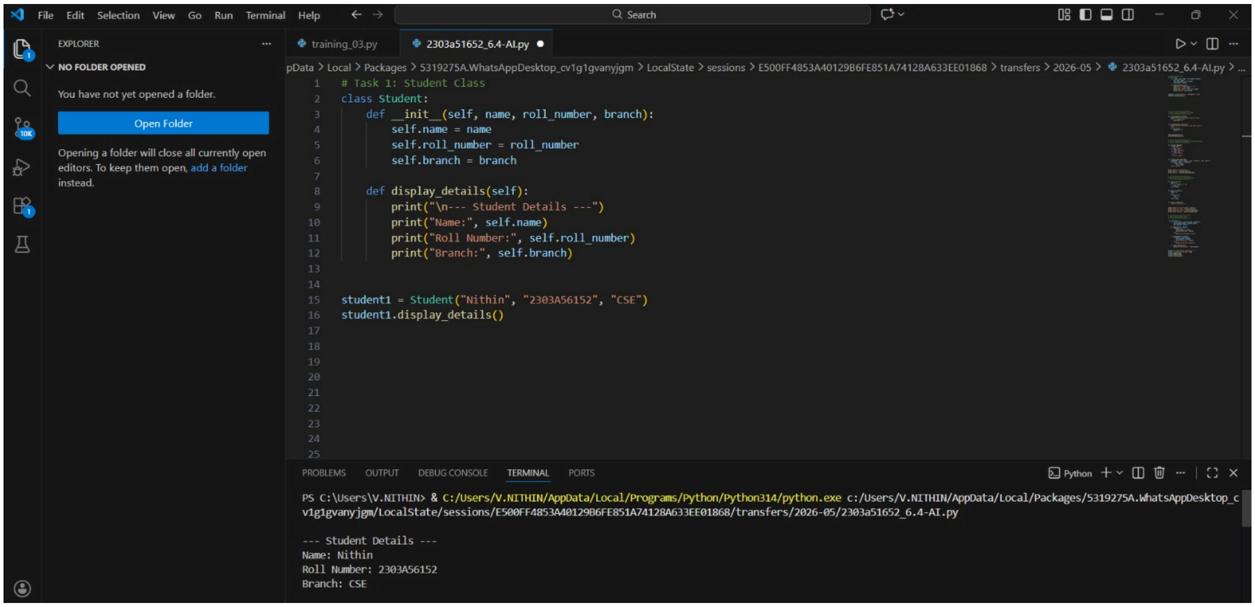
## Classes (Student Class)

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.

# CODE & OUTPUT:



File Edit Selection View Go Run Terminal Help
Search
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
training_03.py
2303a51652_6.4-AI.py
pData > Local > Packages > 5319275A.WhatsAppDesktop_cv1g1gvanyjgm > LocalState > sessions > E500FF4853A40129B6FE851A74128A633EE01868 > transfers > 2026-05 > 2303a51652_6.4-AI.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python

```python
# Task 1: Student Class
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("\n--- Student Details ---")
        print("Name:", self.name)
        print("Roll Number:", self.roll_number)
        print("Branch:", self.branch)


student1 = Student("Nithin", "2303A56152", "CSE")
student1.display_details()
```

```
PS C:\Users\V.NITHIN> & C:/Users/V.NITHIN/AppData/Local/Programs/Python/Python314/python.exe c:/Users/V.NITHIN/AppData/Local/Packages/5319275A.WhatsAppDesktop_c
v1g1gvanyjgm/LocalState/sessions/E500FF4853A40129B6FE851A74128A633EE01868/transfers/2026-05/2303a51652_6.4-AI.py

--- Student Details ---
Name: Nithin
Roll Number: 2303A56152
Branch: CSE
```

# Analysis:

The class is created properly. All student details are stored in variables. The display function prints correct information. The program runs without errors.

# Task 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).

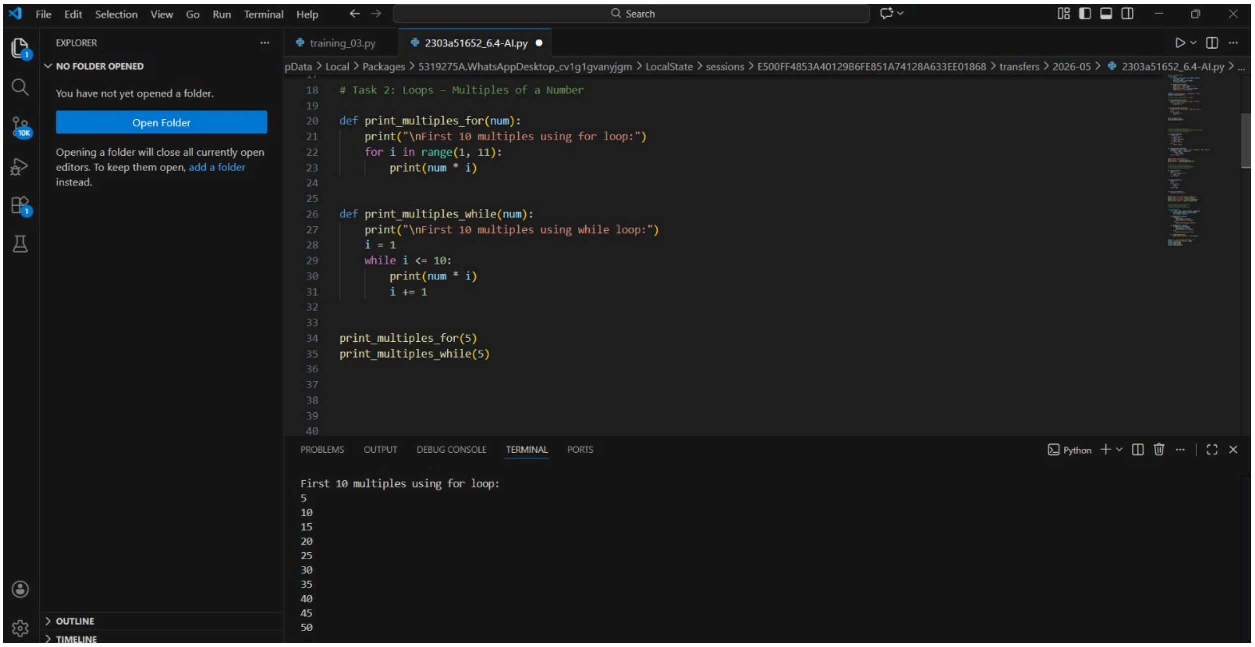
Expected Output #2

• Correct loop-based Python implementation.

• Output showing the first 10 multiples of a number.

• Comparison and analysis of different looping approaches.

## CODE & OUTPUT:


File Edit Selection View Go Run Terminal Help
Search
EXPLORER
NO FOLDER OPENED
You have not opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
OUTLINE
TIMELINE
training_03.py
2303a51652_6.4-AI.py
pData > Local > Packages > 5319275A.WhatsAppDesktop_cv1g1gvanyjgm > LocalState > sessions > E500FF4853A40129B6FE851A74128A633EE01868 > transfers > 2026-05 > 2303a51652_6.4-AI.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python

```python
# Task 2: Loops - Multiples of a Number

def print_multiples_for(num):
    print("\nFirst 10 multiples using for loop:")
    for i in range(1, 11):
        print(num * i)


def print_multiples_while(num):
    print("\nFirst 10 multiples using while loop:")
    i = 1
    while i <= 10:
        print(num * i)
        i += 1


print_multiples_for(5)
print_multiples_while(5)
```

```
First 10 multiples using for loop:
5
10
15
20
25
30
35
40
45
50
```

## Analysis:

The loop works step by step and prints correct multiples. Both for loop and while loop give the same output. There are no mistakes in the logic.

## Task 3:

## Conditional Statements (Age Classification)

Scenario

You are building a basic classification system based on age.

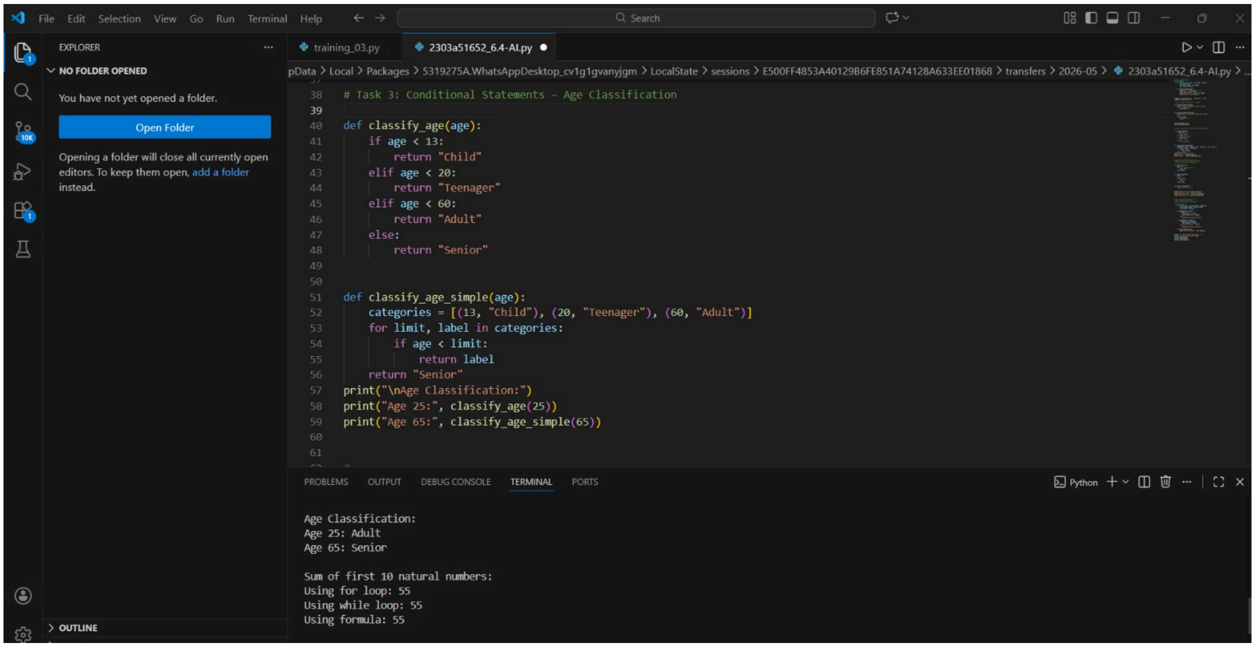
Task

- 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.

## CODE & OUTPUT:


File
Edit
Selection
View
Go
Run
Terminal
Help
Search
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
OUTLINE
training_03.py
2303a51652_6.4-AI.py
pData > Local > Packages > 5319275A.WhatsAppDesktop_cv1g1gvanyjgm > LocalState > sessions > E500FF4853A40129B6FE851A74128A633EE01868 > transfers > 2026-05 > 2303a51652_6.4-AI.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python

```python
# Task 3: Conditional Statements - Age Classification

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


def classify_age_simple(age):
    categories = [(13, "Child"), (20, "Teenager"), (60, "Adult")]
    for limit, label in categories:
        if age < limit:
            return label
    return "Senior"
print("\nAge Classification:")
print("Age 25:", classify_age(25))
print("Age 65:", classify_age_simple(65))
```

```
Age Classification:
Age 25: Adult
Age 65: Senior

Sum of first 10 natural numbers:
Using for loop: 55
Using while loop: 55
Using formula: 55
```

## Analysis:

The conditions are written in correct order. Each age group is checked properly. The program gives the correct group for each age.

# Task 4:

## For and While Loops (Sum of First n Numbers)

Scenario

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

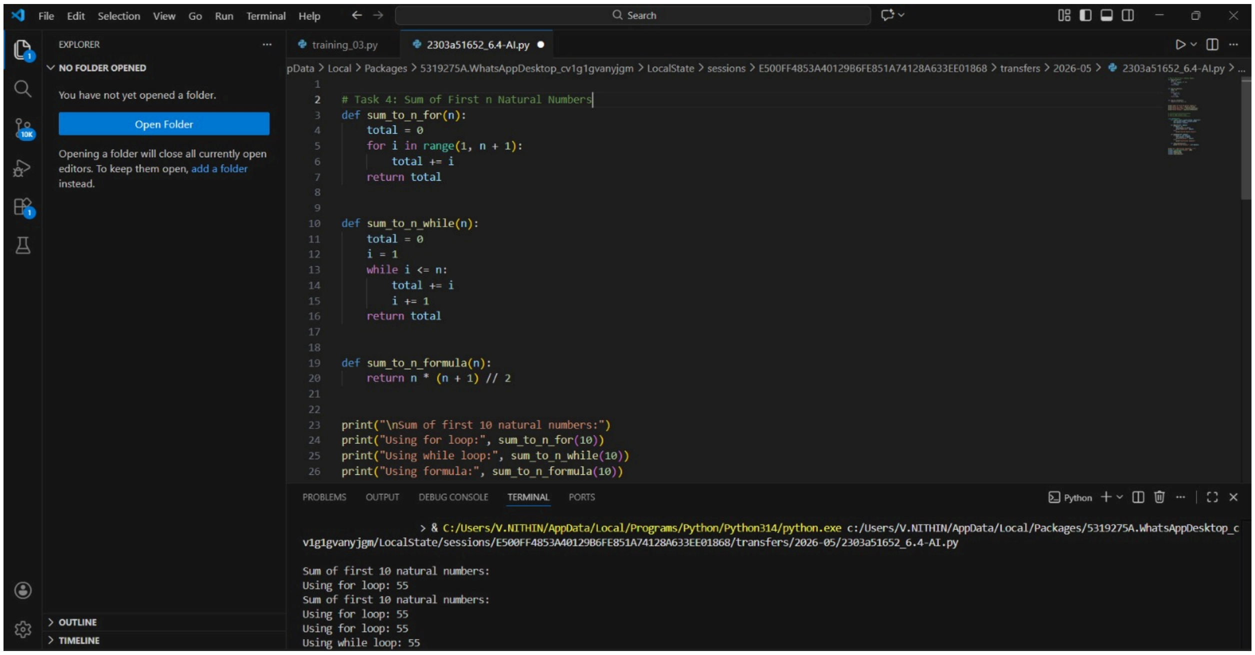
Task

- 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.

# CODE & OUTPUT:


File
Edit
Selection
View
Go
Run
Terminal
Help
Search
EXPLORER
NO FOLDER OPENED
Open Folder
OUTLINE
TIMELINE
training_03.py
2303a51652_6.4-AI.py
pData > Local > Packages > 5319275A.WhatsAppDesktop_cv1g1gvanyjgm > LocalState > sessions > E500FF4853A40129B6FE851A74128A633EE01868 > transfers > 2026-05 > 2303a51652_6.4-AI.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python

You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

```python

# Task 4: Sum of First n Natural Numbers
def sum_to_n_for(n):
    total = 0
    for i in range(1, n + 1):
        total += i
    return total


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


def sum_to_n_formula(n):
    return n * (n + 1) // 2


print("\nSum of first 10 natural numbers:")
print("Using for loop:", sum_to_n_for(10))
print("Using while loop:", sum_to_n_while(10))
print("Using formula:", sum_to_n_formula(10))
```

```
                > & C:/Users/V.NITHIN/AppData/Local/Programs/Python/Python314/python.exe c:/Users/V.NITHIN/AppData/Local/Packages/5319275A.WhatsAppDesktop_c
v1g1gvanyjgm/LocalState/sessions/E500FF4853A40129B6FE851A74128A633EE01868/transfers/2026-05/2303a51652_6.4-AI.py

Sum of first 10 natural numbers:
Using for loop: 55
Sum of first 10 natural numbers:
Using for loop: 55
Using for loop: 55
Using while loop: 55
```

# Analysis:

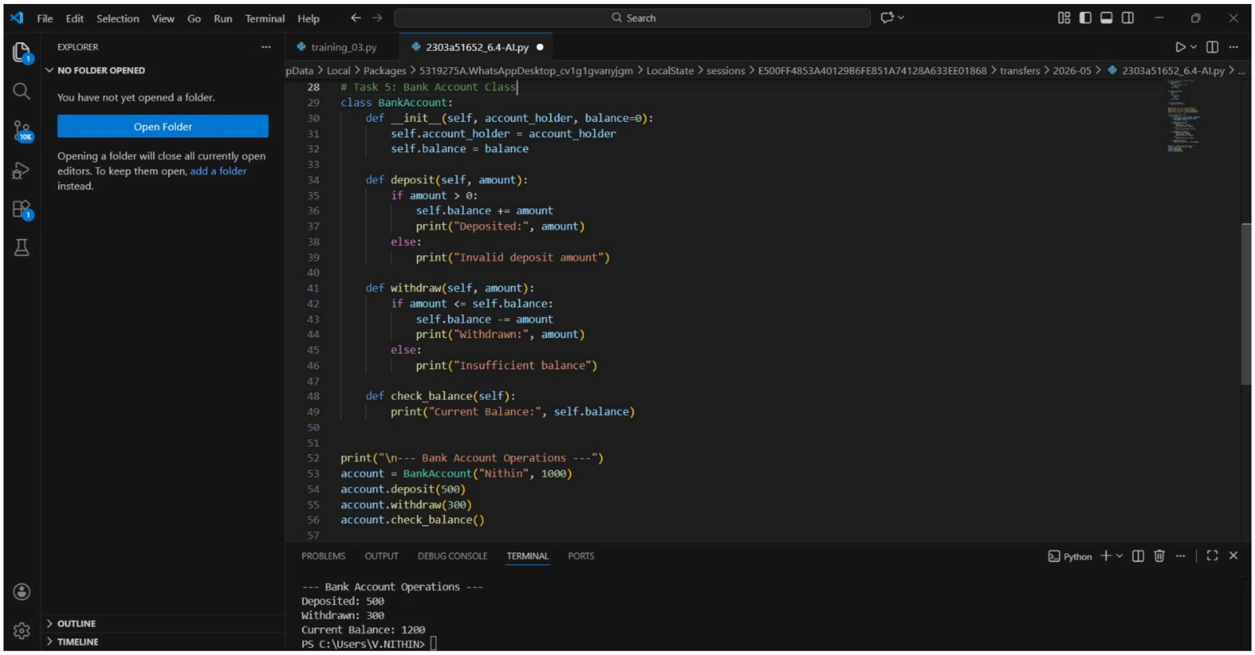
The program adds numbers correctly. All methods give the same answer. The formula method is the fastest and easiest.

# Task 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.

## CODE & OUTPUT:


File
Edit
Selection
View
Go
Run
Terminal
Help
Search
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
OUTLINE
TIMELINE
training_03.py
2303a51652_6.4-AI.py
pData > Local > Packages > 5319275A.WhatsAppDesktop_cv1g1gvanyjgm > LocalState > sessions > E500FF4853A40129B6FE851A74128A633EE01868 > transfers > 2026-05 > 2303a51652_6.4-AI.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python

```python
# Task 5: Bank Account Class
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("Deposited:", amount)
        else:
            print("Invalid deposit 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)


print("\n--- Bank Account Operations ---")
account = BankAccount("Nithin", 1000)
account.deposit(500)
account.withdraw(300)
account.check_balance()
```

```
--- Bank Account Operations ---
Deposited: 500
Withdrawn: 300
Current Balance: 1200
PS C:\Users\V.NITHIN> 
```

## Analysis:

The class methods work properly. Deposit adds money and withdraw removes money correctly. Balance is updated correctly after each action.