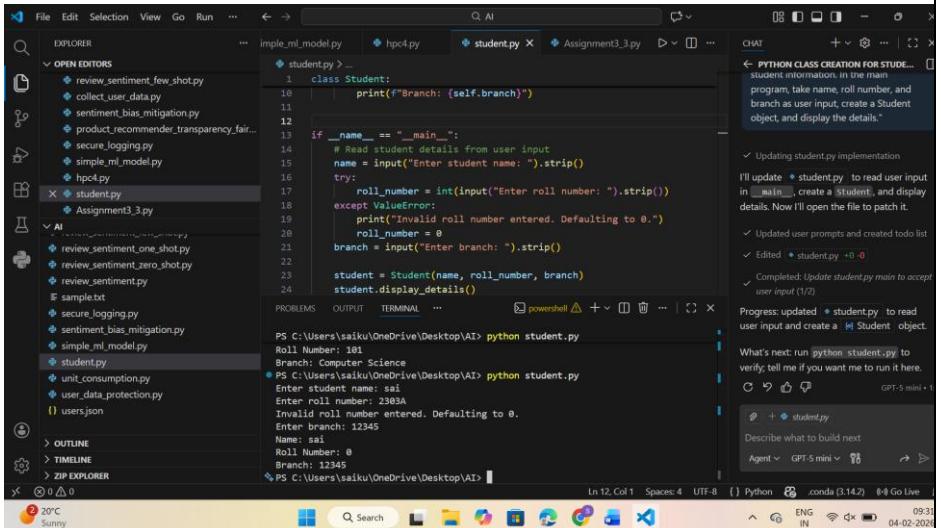


SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	
Course Coordinator Name		Dr. Rishabh Mittal	
Instructor(s) Name		Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju	
Course Code	23CS002PC304	Course Title	AI Assisted Coding
Year/Sem	III/II	Regulation	R23
Date and Day of Assignment	Week3 – Wednesday	Time(s)	23CSBTB01 To 23CSBTB52
Duration	2 Hours	Applicable to Batches	All batches
AssignmentNumber: 6.3(Present assignment number)/24(Total number of assignments)			

Q.No.	Question	Expected Time to complete
1	Lab 6: AI-Based Code Completion – Classes, Loops, and Conditionals Lab Objectives <ul style="list-style-type: none"> 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. 	Week3 - Wednesday

	<p>Lab Outcomes (LOs)</p> <p>After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> • 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. 	
	<p>Task Description #1: Classes (Student Class)</p> <p>Scenario You are developing a simple student information management module.</p> <p>Prompt: Write a Python program that defines a class named Student with attributes name, roll number, and branch. Use a constructor to initialize these values. Add a method display_details() to print student information. In the main program, take name, roll number, and branch as user input, create a Student object, and display the details.</p>  <pre> 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(f"Name: {self.name}") print(f"Roll Number: {self.roll_number}") print(f"Branch: {self.branch}") if __name__ == "__main__": name = input("Enter student name: ").strip() roll_number = int(input("Enter roll number: ").strip()) branch = input("Enter branch: ").strip() student = Student(name, roll_number, branch) student.display_details() </pre>	
	<p>Task Description #2: Loops (Multiples of a Number)</p> <p>Scenario You are writing a utility function to display multiples of a given number.</p> <p>Prompt:</p> <p>1 – For Loop Write a Python function that asks the user to enter a number and prints the first 10 multiples of that number using a for loop.</p> <p>2-While Loop Rewrite the same program that prints the first 10 multiples of a number, but use a while loop instead of a for loop. Keep user input dynamic.</p> <p>3- Analysis Explain the difference between the for loop and while loop implementations for printing multiples of a number.</p>	

Screenshot 1: Python Class Creation for Student Attributes

The screenshot shows a Visual Studio Code (VS Code) interface. The Explorer sidebar lists files including `multiples.py` and `multiples_while.py`. The `multiples.py` file contains a `for` loop implementation:

```

1 def print_first_10_multiples():
2     """Ask the user for a number and prints its first 10 multiples."""
3     try:
4         n = int(input("Enter a number: ").strip())
5     except ValueError:
6         print("Invalid number entered.")
7         return
8
9     for i in range(1, 11):
10         print(n * i)
11
12
13 if __name__ == "__main__":
14     print_first_10_multiples()
15

```

The terminal shows the output of running `multiples.py` with input '5'.

Screenshot 2: Python Class Creation for Student Attributes

This screenshot shows the same VS Code environment with the `multiples_while.py` file open. It contains a `while` loop implementation:

```

1 def print_first_10_multiples():
2     i = 1
3     while i <= 10:
4         print(n * i)
5         i += 1
6
7
8 if __name__ == "__main__":
9     print_first_10_multiples()
10

```

The terminal shows the output of running `multiples_while.py` with input '5'.

Screenshot 3: Python Class Creation for Student Attributes

This screenshot shows the VS Code interface with the `multiples_while.py` file open. A Chat window provides an explanation of the difference between `for` and `while` loops:

PYTHON CLASS CREATION FOR STUDENT ATTRIBUTES

Explain the difference between the for loop and while loop implementations for printing multiples of a number.

✓ Explaining Loop Differences

Summary

- Iteration control: `for:range(1,11)` drives exactly 10 iterations; `while`: you manually control the loop variable and condition (`i = 1; while i <= 10:`).
- Setup & Increment: `for` bundles initialization, condition, and increment in one concise expression; `while` requires separate init and explicit `i += 1`.
- Readability: `for` is more concise and clearer for a known fixed count (first 10 multiples); `while` is more explicit and can be clearer when the loop condition depends on external/state changes.
- Safety: `for` reduces risk of infinite loops since iteration bounds are explicit; `while` can accidentally loop forever if the counter or condition is wrong.
- Flexibility: `while` is better when you need a condition that isn't a simple numeric range (e.g., stop on user signal); `for` is ideal for fixed-length sequences.
- In our files: `*multiples.py` uses `for` loop (concise for 10 multiples); `*multiples_while.py` uses `while` (shows manual control and increment).

The terminal shows the output of running `multiples_while.py` with input '5'.

Task Description #3: Conditional Statements (Age Classification)

Scenario

You are building a basic classification system based on age.

Prompt

1 – if-elif-else

Write a Python program that takes age as user input and classifies the person into Child (0–12), Teenager (13–19), Adult (20–59), or Senior (60+) using nested if-elif-else statements.

2 – Alternative Approach

Rewrite the age classification program using a different approach such as dictionary-based range mapping or simplified logical conditions. Take age as user input.

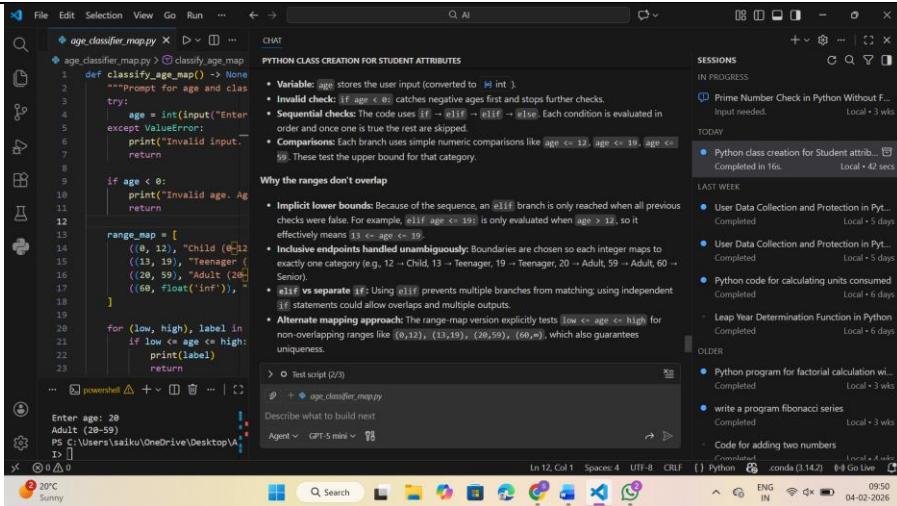
3 – Explanation

Explain how the conditions in the age classification program work and why the ranges do not overlap.

```

File Edit Selection View Go Run ... ← → Q AI
OPEN EDITORS
  EXPLORER
    OPEN EDITORS
      collect_user_data.py
      sentiment_bias_mitigati...
      product_recommender...
      secure_logging.py
      simple_ml_model.py
      hpc.py
      student.py
      age_classifier.py > classify.age
      collect_user_data.py
      sentiment_bias_mitigati...
      product_recommender...
      secure_logging.py
      simple_ml_model.py
      hpc.py
      student.py
      AI
        conda
        vscode
        age_classifier.py
        Assignment3_3.py
        A123.pdf
        AI-4.docx
        Assignment14.py
        Assignment14.txt
        Assignment3_3.py
        Assignment3_4.py
        assignment4_4.py
        cm_to_inches.py
        collect_user_data.py
        student.py
        age_classifier.py
        Assignment3_3.py
        OUTLINE
        TIMELINE
        ZIP EXPLORER
        20°C Sunny
File Edit Selection View Go Run ... ← → Q Search
OPEN EDITORS
  EXPLORER
    OPEN EDITORS
      collect_user_data.py
      sentiment_bias_mitigati...
      product_recommender...
      secure_logging.py
      simple_ml_model.py
      hpc.py
      student.py
      age_classifier.map.py > classify_age_map
      collect_user_data.py
      sentiment_bias_mitigati...
      product_recommender...
      secure_logging.py
      simple_ml_model.py
      hpc.py
      student.py
      AI
        conda
        vscode
        age_classifier.map.py
        age_classifier.py
        Assignment3_3.py
        A123.pdf
        AI-4.docx
        Assignment14.py
        Assignment14.txt
        Assignment3_3.py
        Assignment3_4.py
        assignment4_4.py
        cm_to_inches.py
        collect_user_data.py
        student.py
        age_classifier.map.py
        Assignment3_3.py
        OUTLINE
        TIMELINE
        ZIP EXPLORER
        20°C Sunny

```



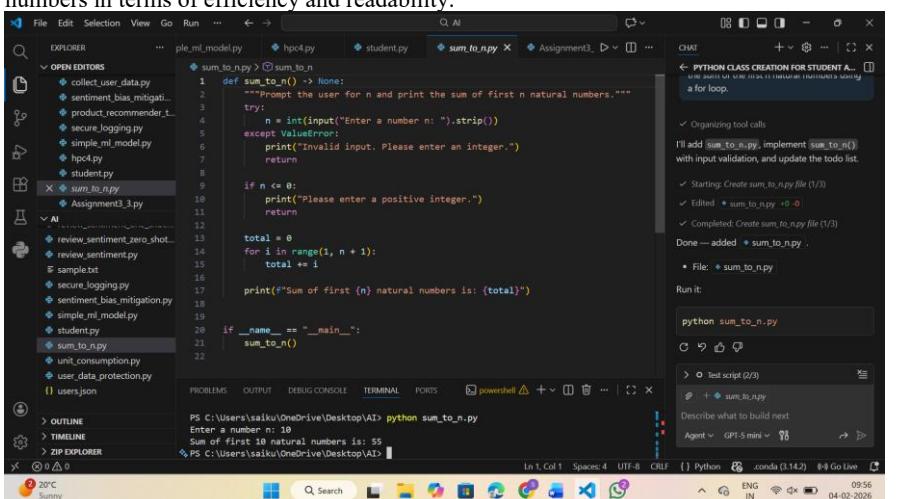
Task Description #4: For and While Loops (Sum of First n Numbers)

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

Prompt
1 – For Loop
 Write a Python function `sum_to_n()` that asks the user to enter a number n and calculates the sum of the first n natural numbers using a for loop.

Prompt 2 – Alternatives
 Provide two more versions of the `sum_to_n()` program: one using a while loop and another using the mathematical formula $n(n+1)/2$. Use user input.

Prompt 3 – Comparison
 Compare the loop-based and formula-based approaches for calculating the sum of n natural numbers in terms of efficiency and readability.



The image displays three vertically stacked screenshots of the Microsoft Visual Studio Code (VS Code) interface, each showing a different version of a Python script for calculating the sum of natural numbers.

Top Window:

- File Explorer:** Shows files like `simple_ml_model.py`, `hpc4.py`, `student.py`, and two versions of `sum_to_n_formula.py` (one named `sum_to_n_formula.py` and another named `sum_to_n_formula.py`).
- Code Editor:** Displays the code for `sum_to_n_formula.py` using a formula approach. It includes error handling for non-positive integers and prints the total sum.
- Terminal:** Shows the command `PS C:\Users\saiku\Desktop\AI> python sum_to_n_formula.py` and its output: "Enter a positive integer n: 10" and "Sum of first 10 natural numbers is: 55".
- Status Bar:** Shows the date (04-02-2026), time (05:59), and system status (20°C, Sunny).

Middle Window:

- File Explorer:** Similar to the top window, showing the same file structure.
- Code Editor:** Displays the code for `sum_to_n_formula.py` using a while loop approach. It includes error handling and prints the total sum.
- Terminal:** Shows the command `PS C:\Users\saiku\Desktop\AI> python sum_to_n_formula.py` and its output: "Enter a positive integer n: 10" and "Sum of first 10 natural numbers is: 55".
- Status Bar:** Shows the date (04-02-2026), time (10:00), and system status (20°C, Sunny).

Bottom Window:

- File Explorer:** Similar to the top window, showing the same file structure.
- Code Editor:** Displays the code for `sum_to_n_formula.py` using a while loop approach. This version includes a detailed comparison of the formula and loop methods.
- Terminal:** Shows the command `PS C:\Users\saiku\Desktop\AI> python sum_to_n_formula.py` and its output: "Enter a positive integer n: 10" and "Sum of first 10 natural numbers is: 55".
- Status Bar:** Shows the date (04-02-2026), time (10:02), and system status (20°C, Sunny).

Task Description #5: Classes (Bank Account Class)

Scenario

You are designing a basic banking application.

Prompt : Write a complete Python program that defines a class `BankAccount` with attributes `account_holder_name` and `balance`. Use a constructor to initialize these values. Include methods `deposit(amount)`, `withdraw(amount)`, and `check_balance()`, ensuring withdrawal is not allowed if the balance is insufficient.

In the main program, take user input for account holder name and initial balance. Then run a looped menu system where the user can repeatedly choose to deposit money, withdraw money, check balance, or exit.

For deposit and withdrawal operations, ask the user to enter the amount dynamically each time. Validate inputs to prevent negative deposits and overdrawing. Display the updated balance after every transaction. Add comments explaining the logic.

The screenshot shows a Python IDE interface with the following details:

- File Explorer:** Shows files in the current workspace, including student.py, bank_account.py, Assignment3_3.py, collect_user_data.py, sentiment_bias_mit.py, product_recommender.py, secure_logging.py, simple_ml_model.py, hpc4.py, student.py, and bank_account.py.
- Code Editor:** The main editor window displays the `bank_account.py` file. The code implements a bank account class with deposit and withdraw methods, including validation logic and user interaction via a command-line menu.
- Terminal:** A terminal window titled "python_bank_account.py" is open, showing the message "Want me to add unit tests or save transaction history to a file next?"
- Status Bar:** Shows the current file is `Assignment3_3.py`, the line number is 81, and the status bar includes icons for file operations like Save, Undo, Redo, and Help.

The screenshot shows a terminal window within a code editor interface, displaying a Python application for managing bank accounts. The application uses a menu-driven interface with options for deposit, withdraw, check balance, and exit. It also includes a command-line input for account holder names and transaction amounts.

```
PS C:\Users\saikumar\OneDrive\Desktop\AI> python bank_account.py
Withdraw 1000.00. New balance: 100.00
Choose an option:
1. Deposit
2. Withdraw
3. Check balance
4. Exit
Enter choice (1-4): 3
Account holder: sai. Balance: 100.00

Choose an option:
1. Deposit
2. Withdraw
3. Check balance
4. Exit
Enter choice (1-4): 3
Account holder: sai. Balance: 100.00

Choose an option:
1. Deposit
2. Withdraw
3. Check balance
4. Exit
Enter choice (1-4): 3
Account holder: sai. Balance: 100.00
```

At the bottom right of the terminal window, there is a message from the developer: "I've updated + bank_account.py with a full interactive program that matches your requirements."

On the right side of the code editor, there is a sidebar with tabs for CHAT, Todos (3/3), and Describe what to build next. The Todos tab lists "python bank_account.py".

Note: Report should be submitted as a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.