NAME:B.NITHIN H.NO:2303A51803 B-26

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE	DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Program Name: B. Tech	Assignment Type: Lab	Academic Year:2025-2026
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	

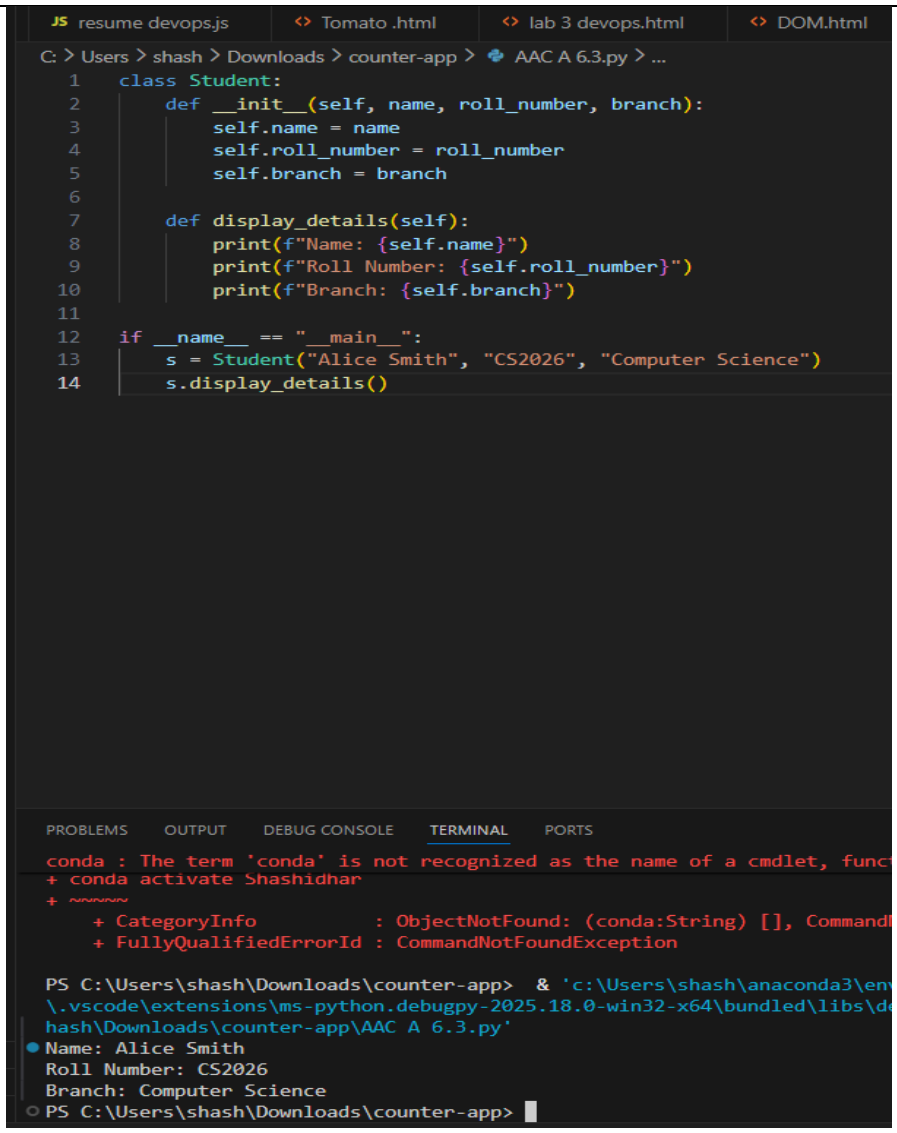
Course Code	23CS002PC304	Course Title	AI Assisted Coding
Year/Sem	III/II	Regulation	R23
Date and Day of Assignment	Week3 – Wednesday	Time(s)	23CSBTB01 To 23CSBTB52
Duration	2 Hours	Applicable to Batches	All batches

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** • 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

NAME:B.NITHIN H.NO:2303A51803 B-26

	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 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.	



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.

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.

NAME:B.NITHIN H.NO:2303A51803 B-26

```python
def classify_age_nested(age):
    if age < 0:
        return "invalid"
    if age <= 12:
        return "child"
    elif age <= 17:
        return "teenager"
    elif age <= 64:
        return "adult"
    else:
        return "senior"

def classify_age_simplified(age):
    if age < 0:
        return "invalid"
    if 0 <= age <= 12:
        return "child"
    if 13 <= age <= 17:
        return "teenager"
    if 18 <= age <= 64:
        return "adult"
    return "senior"

def classify_age_dict(age):
    if age < 0:
        return "invalid"
    thresholds = [(12, "child"), (17, "teenager"), (64, "adult"), (float('inf'), "senior")]
    for limit, label in thresholds:
        if age <= limit:
            return label

if __name__ == "__main__":
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\shash\Downloads\counter-app> c:; cd 'c:\Users\shash\Downloads\counter-app'; & 'c:\Users\sh
shidhar\python.exe' 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\li
49215' '--' 'c:\Users\shash\Downloads\counter-app\AAC A 6.3.py'
Nested if-elif-else:
-1 invalid
Dictionary-threshold approach:
3 child
15 teenager
30 adult
70 senior
-1 invalid
PS C:\Users\shash\Downloads\counter-app>
```

NAME:B.NITHIN H.NO:2303A51803 B-26

```
C: > Users > shash > Downloads > counter-app > AAC A 6.3.py > ...
  def classify_age_simplified(age):
      if 18 <= age <= 64:
          return "adult"
      return "senior"

  def classify_age_dict(age):
      if age < 0:
          return "invalid"
      thresholds = [(12, "child"), (17, "teenager"), (64, "adult"), (float('inf'), "senior")]
      for limit, label in thresholds:
          if age <= limit:
              return label

  if __name__ == "__main__":
      sample_ages = [3, 15, 30, 70, -1]
      print("Nested if-elif-else:")
      for a in sample_ages:
          print(a, classify_age_nested(a))
      print("Simplified chained conditions:")
      for a in sample_ages:
          print(a, classify_age_simplified(a))
      print("Dictionary-threshold approach:")
      for a in sample_ages:
          print(a, classify_age_dict(a))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\shash\Downloads\counter-app>  c:; cd 'c:\Users\shash\Downloads\counter-app'; & 'c:\Us
shidhar\python.exe' 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bund
49215' '--' 'c:\Users\shash\Downloads\counter-app\AAC A 6.3.py'
Nested if-elif-else:
-1 invalid
Dictionary-threshold approach:
3 child
15 teenager
30 adult
70 senior
-1 invalid
PS C:\Users\shash\Downloads\counter-app>
```

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.

NAME:B.NITHIN H.NO:2303A51803 B-26

C: > Users > shash > Downloads > counter-app > AAC A 6.3.py > ...

```python
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):
    if n < 0:
        return None
    return n * (n + 1) // 2

if __name__ == "__main__":
    samples = [0, 1, 10, 100]
    for n in samples:
        print(n, sum_to_n_for(n), sum_to_n_while(n), sum_to_n_formula(n))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\shash\Downloads\counter-app>  c:; cd 'c:\Users\shash\Downloads\coun
da3\envs\Shashidhar\python.exe' 'c:\Users\shash\.vscode\extensions\ms-python.de
libs\debugpy\launcher' '65013' '--' 'c:\Users\shash\Downloads\counter-app\AAC A
0 0 0 0
PS C:\Users\shash\Downloads\counter-app>  c:; cd 'c:\Users\shash\Downloads\coun
da3\envs\Shashidhar\python.exe' 'c:\Users\shash\.vscode\extensions\ms-python.de
libs\debugpy\launcher' '61058' '--' 'c:\Users\shash\Downloads\counter-app\AAC A
0 0 0 0
1 1 1 1
10 55 55 55
100 5050 5050 5050
PS C:\Users\shash\Downloads\counter-app>
```

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.

NAME:B.NITHIN H.NO:2303A51803 B-26

```python
class BankAccount:
    def __init__(self, owner, balance=0.0):
        self.owner = owner
        self.balance = float(balance)
    def deposit(self, amount):
        if amount <= 0:
            raise ValueError("Deposit amount must be positive")
        self.balance += amount
        return self.balance
    def withdraw(self, amount):
        if amount <= 0:
            raise ValueError("Withdrawal amount must be positive")
        if amount > self.balance:
            return False
        self.balance -= amount
        return True
    def check_balance(self):
        return self.balance

    def __repr__(self):
        return f"BankAccount(owner={self.owner!r}, balance={self.balance:.2f})"


if __name__ == "__main__":
    owner = input("Enter account owner name: ").strip()
    bal = input("Enter starting balance (leave empty for 0): ").strip()
    try:
        start_balance = float(bal) if bal else 0.0
    except ValueError:
        start_balance = 0.0
    acct = BankAccount(owner or "Unknown", start_balance)
    print("Account created:", acct)
```

```
PS C:\Users\shash\Downloads\counter-app>  c:; cd 'c:\Users\shash\Downloads\counter-app'
da3\envs\Shashidhar\python.exe' 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-26
libs\debugpy\launcher' '63018' '--' 'c:\Users\shash\Downloads\counter-app\AAC A 6.3.py'
Enter account owner name: Shashidhar Ashadapu
Enter starting balance (leave empty for 0): 1000
Account created: BankAccount(owner='Shashidhar Ashadapu', balance=1000.00)

Options: [d]eposit, [w]ithdraw, [c]heck balance, [q]uit
Choose option: w
Amount to withdraw: 500
Success: True Balance: 500.0
```

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.