

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
<b>Program Name:</b> B. Tech		<b>Assignment Type:</b> Lab	
<b>Course Coordinator Name</b>		Dr. Rishabh Mittal	
<b>Instructor(s) Name</b>		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	
<b>Course Code</b>	23CS002PC304	<b>Course Title</b>	AI Assisted Coding
<b>Year/Sem</b>	III/II	<b>Regulation</b>	R23
<b>Date and Day of Assignment</b>	Week3 – Wednesday	<b>Time(s)</b>	23CSBTB01 To 23CSBTB52
<b>Duration</b>	2 Hours	<b>Applicable to Batches</b>	All batches
<b>AssignmentNumber:</b> 6.3(Present assignment number)/24(Total number of assignments)			
<b>Q.No.</b>	<b>Question</b>		<b>Expected Time to complete</b>
1	<b>Lab 6: AI-Based Code Completion – Classes, Loops, and Conditionals</b> <b>Lab Objectives</b> <ul style="list-style-type: none"> <li>• To explore AI-powered auto-completion features for core Python constructs such as classes, loops, and conditional statements.</li> <li>• To analyze how AI tools suggest logic for object-oriented programming and control structures.</li> </ul>		Week3 - Wednesday

	<ul style="list-style-type: none"><li>• To evaluate the correctness, readability, and completeness of AI-generated Python code.</li></ul> <p><b>Lab Outcomes (LOs)</b> After completing this lab, students will be able to:</p> <ul style="list-style-type: none"><li>• Use AI tools to generate and complete Python class definitions and methods.</li><li>• Understand and assess AI-suggested loop constructs for iterative tasks.</li><li>• Generate and evaluate conditional statements using AI-driven prompts.</li><li>• Critically analyze AI-assisted code for correctness, clarity, and efficiency.</li></ul>	
	<p><b>Task Description #1: Classes (Student Class)</b></p> <p><b>Scenario</b> You are developing a simple student information management module.</p> <p><b>Task</b></p> <ul style="list-style-type: none"><li>• Use an AI tool (GitHub Copilot / Cursor AI / Gemini) to complete a Student class.</li><li>• The class should include attributes such as name, roll number, and branch.</li><li>• Add a method <code>display_details()</code> to print student information.</li><li>• Execute the code and verify the output.</li><li>• Analyze the code generated by the AI tool for correctness and clarity.</li></ul> <p><b>Expected Output #1</b></p> <ul style="list-style-type: none"><li>• A Python class with a constructor (<code>__init__</code>) and a <code>display_details()</code> method.</li><li>• Sample object creation and output displayed on the console.</li><li>• Brief analysis of AI-generated code.</li></ul>	

The screenshot shows a terminal window with the following content:

```
JS resume devops.js  ↗ Tomato .html  ↗ lab 3 devops.html  ↗ DOM.html
C: > Users > shash > Downloads > counter-app > AAC A 6.3.py > ...
1  class Student:
2      def __init__(self, name, roll_number, branch):
3          self.name = name
4          self.roll_number = roll_number
5          self.branch = branch
6
7      def display_details(self):
8          print(f"Name: {self.name}")
9          print(f"Roll Number: {self.roll_number}")
10         print(f"Branch: {self.branch}")
11
12 if __name__ == "__main__":
13     s = Student("Alice Smith", "CS2026", "Computer Science")
14     s.display_details()

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
conda : The term 'conda' is not recognized as the name of a cmdlet, func
+ conda activate Shashidhar
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (conda:String) [], CommandNot
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\shash\Downloads\counter-app> & 'c:\Users\shash\anaconda3\envs
\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\de
hash\Downloads\counter-app\AAC A 6.3.py'
● Name: Alice Smith
● Roll Number: CS2026
● Branch: Computer Science
○ PS C:\Users\shash\Downloads\counter-app>
```

### 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.

The screenshot shows a terminal window with the following content:

```
JS resume devops.js      Tomato .html      lab 3 devops.h
C: > Users > shash > Downloads > counter-app > AAC A 6.3.py >
1  def print_multiples_for(n):
2      for i in range(1, 11):
3          print(n * i)
4
5  def print_multiples_while(n):
6      i = 1
7      while i <= 10:
8          print(n * i)
9          i += 1
10
11 if __name__ == "__main__":
12     number = 7
13     print("For loop multiples:")
14     print_multiples_for(number)
15     print("While loop multiples:")
16     print_multiples_while(number)

PROBLEMS      OUTPUT      DEBUG CONSOLE      TERMINAL      PORTS

PS C:\Users\shash\Downloads\counter-app> c;; cd 'c:\da3\envs\Shashidhar\python.exe' 'c:\Users\shash\.vscode\libs\debugpy\launcher' '56132' '--' 'c:\Users\shash\For loop multiples:
28
35
42
49
56
63
70
PS C:\Users\shash\Downloads\counter-app>
```

**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.

```
C:\> Users > shash > Downloads > counter-app > * AAC A Counter > ...
 1 def classify_age_nested(age):
 2     if age < 0:
 3         return "invalid"
 4     if age <= 12:
 5         return "child"
 6     elif age <= 17:
 7         return "teenager"
 8     elif age <= 64:
 9         return "adult"
10     else:
11         return "senior"
12
13 def classify_age_simplified(age):
14     if age < 0:
15         return "invalid"
16     if 0 <= age <= 12:
17         return "child"
18     if 13 <= age <= 17:
19         return "teenager"
20     if 18 <= age <= 64:
21         return "adult"
22     return "senior"
23
24 def classify_age_dict(age):
25     if age < 0:
26         return "invalid"
27     thresholds = [(12, "child"), (17, "teenager"), (64, "adult"), (float('inf'), "senior")]
28     for limit, label in thresholds:
29         if age <= limit:
30             return label
31
32 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\shash\python.exe' 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\lib\site-packages\debugpy\launcher\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> []
```

```
C:\> Users > shash > Downloads > counter-app > AAC A 6.3.py > ...
13 def classify_age_simplified(age):
14     if 18 <= age <= 64:
15         return "adult"
16     return "senior"
17
18 def classify_age_dict(age):
19     if age < 0:
20         return "invalid"
21     thresholds = [(12, "child"), (17, "teenager"), (64, "adult"), (float('inf'), "senior")]
22     for limit, label in thresholds:
23         if age <= limit:
24             return label
25
26 if __name__ == "__main__":
27     sample_ages = [3, 15, 30, 70, -1]
28     print("Nested if-elif-else:")
29     for a in sample_ages:
30         print(a, classify_age_nested(a))
31     print("Simplified chained conditions:")
32     for a in sample_ages:
33         print(a, classify_age_simplified(a))
34     print("Dictionary-threshold approach:")
35     for a in sample_ages:
36         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:\Users\shash\python.exe' 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\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.

The screenshot shows a VS Code interface with a code editor and a terminal. The code editor contains a Python script named `devops.py` which defines three functions: `sum_to_n_for`, `sum_to_n_while`, and `sum_to_n_formula`. The terminal below shows the script being run and its output for sample inputs.

```
C:\> Users > shash > Downloads > counter-app > AAC A 6.3.py > ...
1 def sum_to_n_for(n):
2     total = 0
3     for i in range(1, n + 1):
4         total += i
5     return total
6
7 def sum_to_n_while(n):
8     total = 0
9     i = 1
10    while i <= n:
11        total += i
12        i += 1
13    return total
14
15 def sum_to_n_formula(n):
16     if n < 0:
17         return None
18     return n * (n + 1) // 2
19
20 if __name__ == "__main__":
21     samples = [0, 1, 10, 100]
22     for n in samples:
23         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\counter-app' & Shashidhar\python.exe 'c:\Users\shash\.vscode\extensions\ms-python.de...
PS C:\Users\shash\Downloads\counter-app> c;; cd 'c:\Users\shash\Downloads\counter-app' & Shashidhar\python.exe 'c:\Users\shash\.vscode\extensions\ms-python.de...
PS C:\Users\shash\Downloads\counter-app> c;; cd 'c:\Users\shash\Downloads\counter-app' & Shashidhar\python.exe 'c:\Users\shash\.vscode\extensions\ms-python.de...
PS C:\Users\shash\Downloads\counter-app> 0 0 0 0
PS C:\Users\shash\Downloads\counter-app> 0 0 0 0
PS C:\Users\shash\Downloads\counter-app> 1 1 1 1
PS C:\Users\shash\Downloads\counter-app> 10 55 55 55
PS C:\Users\shash\Downloads\counter-app> 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.

The screenshot shows a VS Code interface with a terminal window open. The terminal displays Python code for a `BankAccount` class and its execution.

```
C:\> Users > shash > Downloads > counter-app > AAC A 6.3.py > BankAccount > withdraw
1  class BankAccount:
2      def __init__(self, owner, balance=0.0):
3          self.owner = owner
4          self.balance = float(balance)
5      def deposit(self, amount):
6          if amount <= 0:
7              raise ValueError("Deposit amount must be positive")
8          self.balance += amount
9          return self.balance
10     def withdraw(self, amount):
11         if amount <= 0:
12             raise ValueError("Withdrawal amount must be positive")
13         if amount > self.balance:
14             return False
15         self.balance -= amount
16         return True
17     def check_balance(self):
18         return self.balance
19
20     def __repr__(self):
21         return f"BankAccount(owner={self.owner!r}, balance={self.balance:.2f})"
22
23
24 if __name__ == "__main__":
25     owner = input("Enter account owner name: ").strip()
26     bal = input("Enter starting balance (leave empty for 0): ").strip()
27     try:
28         start_balance = float(bal) if bal else 0.0
29     except ValueError:
30         start_balance = 0.0
31     acct = BankAccount(owner or "Unknown", start_balance)
32     print("Account created:", acct)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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-20
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.