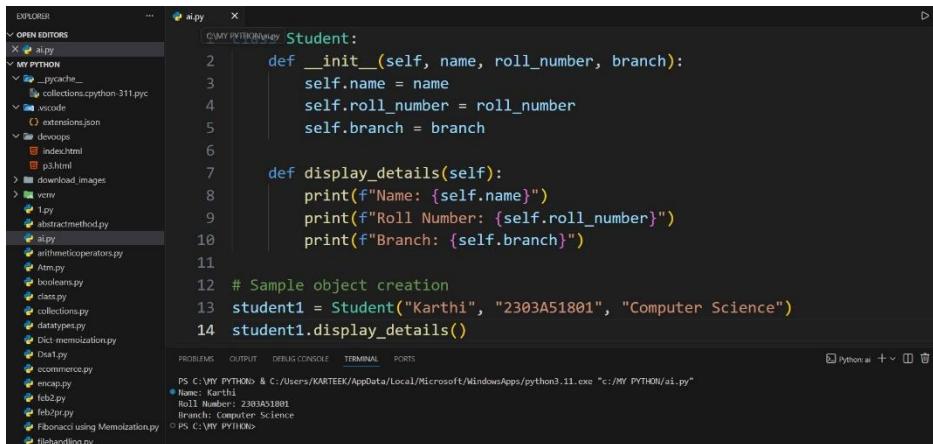


SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	
Course Coordinator Name		Dr. Rishabh Mittal	
Instructor(s) Name		Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan 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. 		Week3 - Wednesday

	<ul style="list-style-type: none"> To evaluate the correctness, readability, and completeness of AI-generated Python code. <p>Lab Outcomes (LOs) 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. <hr/> <p>Task Description #1: Classes (Student Class)</p> <p>Scenario You are developing a simple student information management module.</p> <p>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.</p> <p>Expected Output #1</p> <ul style="list-style-type: none">  <pre> 1 #!/usr/bin/python 2 class Student: 3 def __init__(self, name, roll_number, branch): 4 self.name = name 5 self.roll_number = roll_number 6 self.branch = branch 7 8 def display_details(self): 9 print(f"Name: {self.name}") 10 print(f"Roll Number: {self.roll_number}") 11 print(f"Branch: {self.branch}") 12 13 # Sample object creation 14 student1 = Student("Karthi", "2303A51801", "Computer Science") 15 student1.display_details() </pre> <hr/> <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>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).</p> <p>Expected Output #2</p>	
--	---	--

```
self.branch = branch

def display_details(self):
    print(f"Name: {self.name}")
    print(f"Roll Number: {self.roll_number}")
    print(f"Branch: {self.branch}")

# Example usage
student1 = Student("Arun", "1735", "Computer Science")
student1.display_details()

#Write a Python function print_multiples(n) that prints first 10 multiples of n using for loop
def print_multiples(n):
    for i in range(1, 11):
        print(n * i)

# Example usage
print_multiples(5)
```

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

# Example usage
print_multiples(5)"""

#Rewrite print_multiples(n) function to print first 10 multiples using while loop instead of for
def print_multiples(n):
    i = 1
    while i <= 10:
        print(n * i)
        i += 1

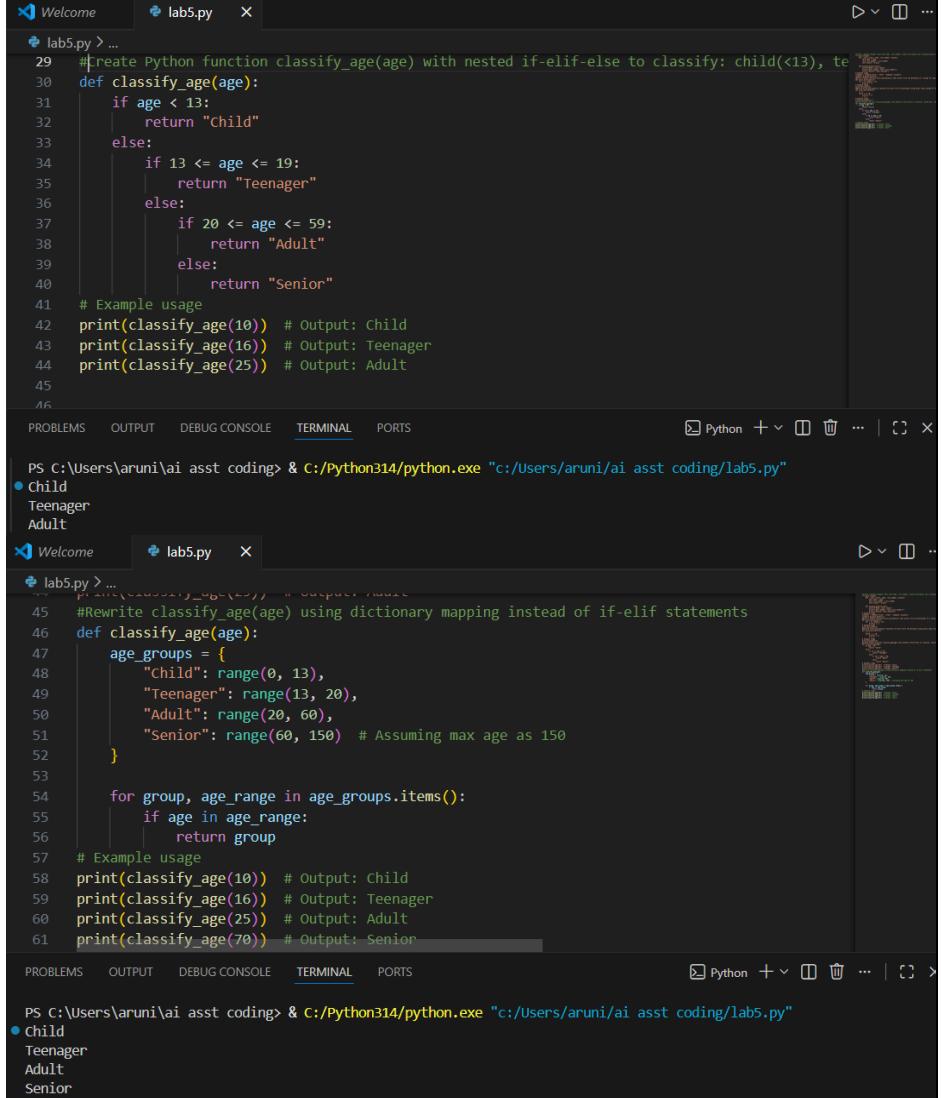
# Example usage
print_multiples(5)
```

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.,

	<p>simplified conditions or dictionary-based logic).</p> <p>Expected Output #3</p> <ul style="list-style-type: none"> •  <pre> 29 #Create Python function classify_age(age) with nested if-elif-else to classify: child(<13), te 30 def classify_age(age): 31 if age < 13: 32 return "Child" 33 else: 34 if 13 <= age <= 19: 35 return "Teenager" 36 else: 37 if 20 <= age <= 59: 38 return "Adult" 39 else: 40 return "Senior" 41 # Example usage 42 print(classify_age(10)) # Output: Child 43 print(classify_age(16)) # Output: Teenager 44 print(classify_age(25)) # Output: Adult 45 46 47 PS C:\Users\aruni\ai asst coding> & C:/Python314/python.exe "c:/Users/aruni/ai asst coding/lab5.py" ● Child Teenager Adult </pre> <pre> 45 #Rewrite classify_age(age) using dictionary mapping instead of if-elif statements 46 def classify_age(age): 47 age_groups = { 48 "Child": range(0, 13), 49 "Teenager": range(13, 20), 50 "Adult": range(20, 60), 51 "Senior": range(60, 150) # Assuming max age as 150 52 } 53 54 for group, age_range in age_groups.items(): 55 if age in age_range: 56 return group 57 # Example usage 58 print(classify_age(10)) # Output: Child 59 print(classify_age(16)) # Output: Teenager 60 print(classify_age(25)) # Output: Adult 61 print(classify_age(70)) # Output: Senior </pre> <pre> PS C:\Users\aruni\ai asst coding> & C:/Python314/python.exe "c:/Users/aruni/ai asst coding/lab5.py" ● Child Teenager Adult Senior </pre>
	<p>Task Description #4: For and While Loops (Sum of First n Numbers)</p> <p>Scenario You need to calculate the sum of the first n natural numbers.</p> <p>Task</p> <ul style="list-style-type: none"> • 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. <p>Expected Output #4</p> <ul style="list-style-type: none"> •

The image shows a terminal window with three vertically stacked code snippets for a Python file named `lab5.py`. Each snippet demonstrates different ways to calculate the sum of the first `n` natural numbers.

Top Snippet:

```
56     return group
57 # Example usage
58 print(classify_age(10)) # Output: Child
59 print(classify_age(16)) # Output: Teenager
60 print(classify_age(25)) # Output: Adult
61 print(classify_age(70)) # Output: Senior"""
62 #Write Python function sum_to_n(n) using for loop to calculate sum of first n natural numbers
63 def sum_to_n(n):
64     total = 0
65     for i in range(1, n + 1):
66         total += i
67     return total
68
69 # Example usage
70 print(sum_to_n(5)) # Output: 15
71
72
```

Middle Snippet:

```
68
69 # Example usage
70 print(sum_to_n(5)) # Output: 15"""
71 #Convert sum_to_n(n) to use while loop instead of for loop
72 def sum_to_n(n):
73     total = 0
74     i = 1
75     while i <= n:
76         total += i
77         i += 1
78     return total
79
80 # Example usage
81 print(sum_to_n(5)) # Output: 15
82
```

Bottom Snippet:

```
73     total = 0
74     i = 1
75     while i <= n:
76         total += i
77         i += 1
78     return total
79
80 # Example usage
81 print(sum_to_n(5)) # Output: 15"""
82 #Add mathematical formula version to sum_to_n(n) function as alternative to loops
83 def sum_to_n(n):
84     # Using the formula n(n + 1) / 2
85     return n * (n + 1) // 2
86
87 # Example usage
88 print(sum_to_n(5)) # Output: 15
89
```

Terminal Output:

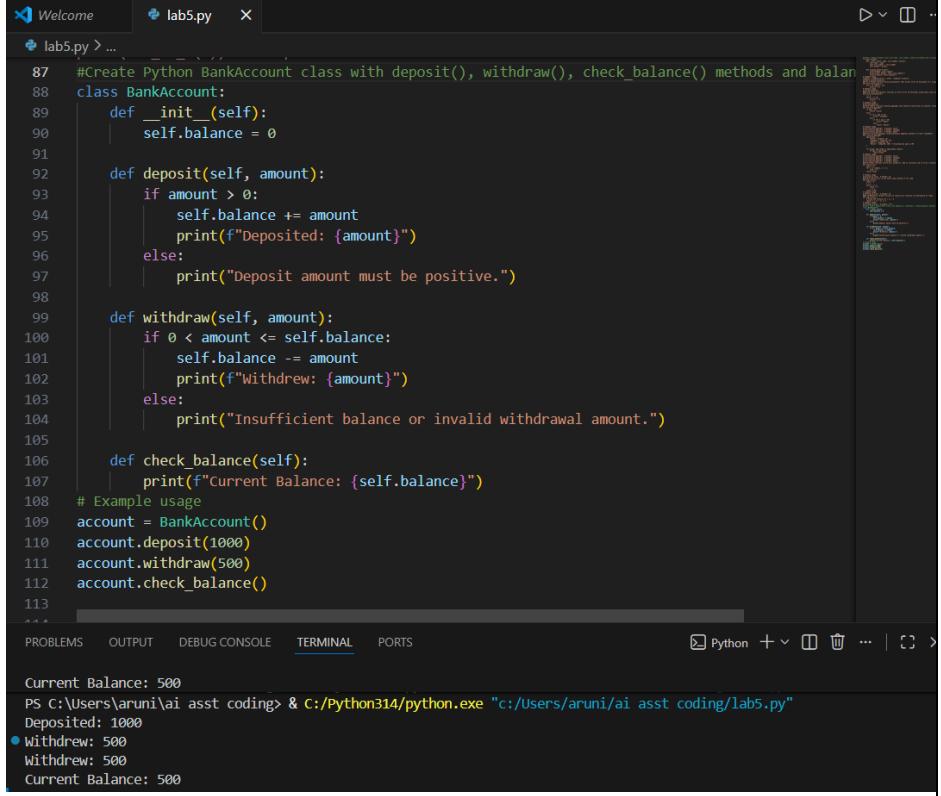
```
PS C:\Users\aruni\ai asst coding> & C:/Python314/python.exe "c:/Users/aruni/ai asst coding/lab5.py"
● PS C:\Users\aruni\ai asst coding> & C:/Python314/python.exe "c:/Users/aruni/ai asst coding/lab5.py"
15

PS C:\Users\aruni\ai asst coding> & C:/Python314/python.exe "c:/Users/aruni/ai asst coding/lab5.py"
● PS C:\Users\aruni\ai asst coding> & C:/Python314/python.exe "c:/Users/aruni/ai asst coding/lab5.py"
15

PS C:\Users\aruni\ai asst coding> & C:/Python314/python.exe "c:/Users/aruni/ai asst coding/lab5.py"
● PS C:\Users\aruni\ai asst coding> & C:/Python314/python.exe "c:/Users/aruni/ai asst coding/lab5.py"
15
```

Task Description #5: Classes (Bank Account Class)

Scenario

	<p>You are designing a basic banking application.</p> <p>Task</p> <ul style="list-style-type: none">• 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. <p>Expected Output #5</p> <ul style="list-style-type: none">•  <pre>87 #Create Python BankAccount class with deposit(), withdraw(), check_balance() methods and balan 88 class BankAccount: 89 def __init__(self): 90 self.balance = 0 91 92 def deposit(self, amount): 93 if amount > 0: 94 self.balance += amount 95 print(f"Deposited: {amount}") 96 else: 97 print("Deposit amount must be positive.") 98 99 def withdraw(self, amount): 100 if 0 < amount <= self.balance: 101 self.balance -= amount 102 print(f"Withdrew: {amount}") 103 else: 104 print("Insufficient balance or invalid withdrawal amount.") 105 106 def check_balance(self): 107 print(f"Current Balance: {self.balance}") 108 109 # Example usage 110 account = BankAccount() 111 account.deposit(1000) 112 account.withdraw(500) 113 account.check_balance()</pre> <p>PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS</p> <p>Current Balance: 500 PS C:\Users\aruni\ai asst coding> & C:/Python314/python.exe "c:/Users/aruni/ai asst coding/lab5.py" Deposited: 1000 Withdrew: 500 Withdrew: 500 Current Balance: 500</p> <p>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.</p>	
--	---	--