

AI ASSISTED CODING

ASSIGNMENT-6.3

M.AKHIL REDDY

2303A52315

BATCH-45

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.

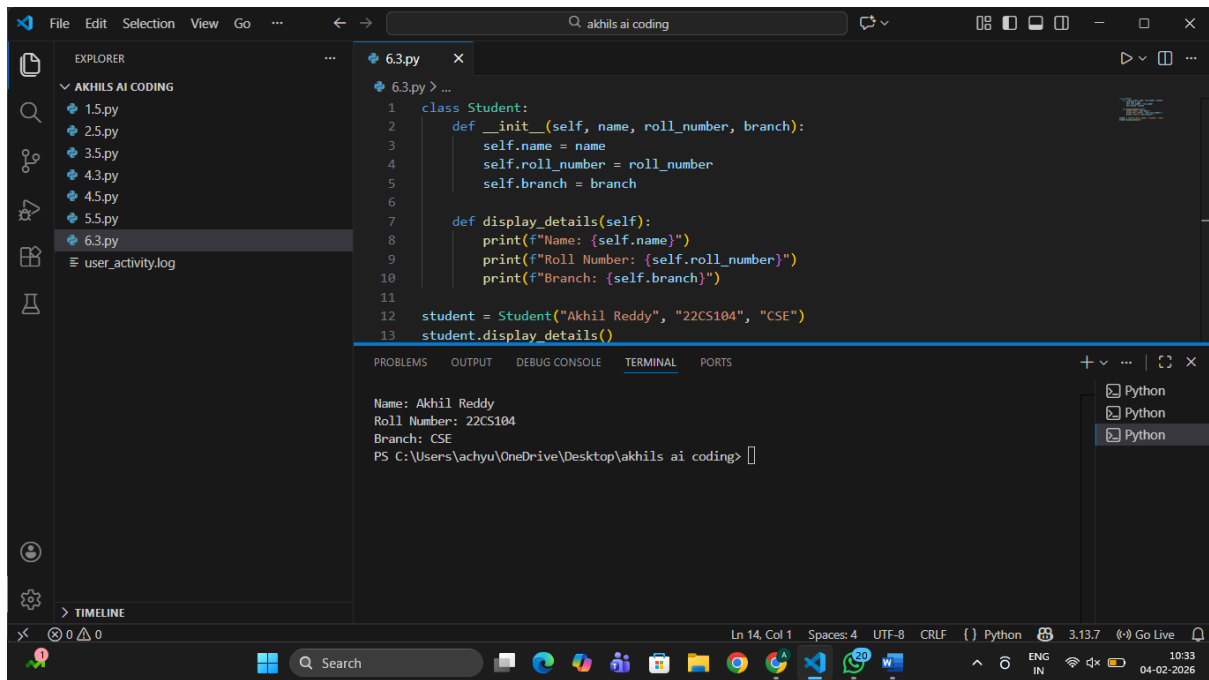
Expected Output #1

- 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.

PROMPT:

Write a Python class named Student with attributes for name, roll number, and branch. Include a constructor (`init`) to initialize these attributes and a method `display_details()` to print the student's information in a formatted way. Also, provide sample code to create an object of the class and call the `display_details()` method. Do not include any additional explanations or comments outside the code.

EXPECTED OUTPUT:



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).

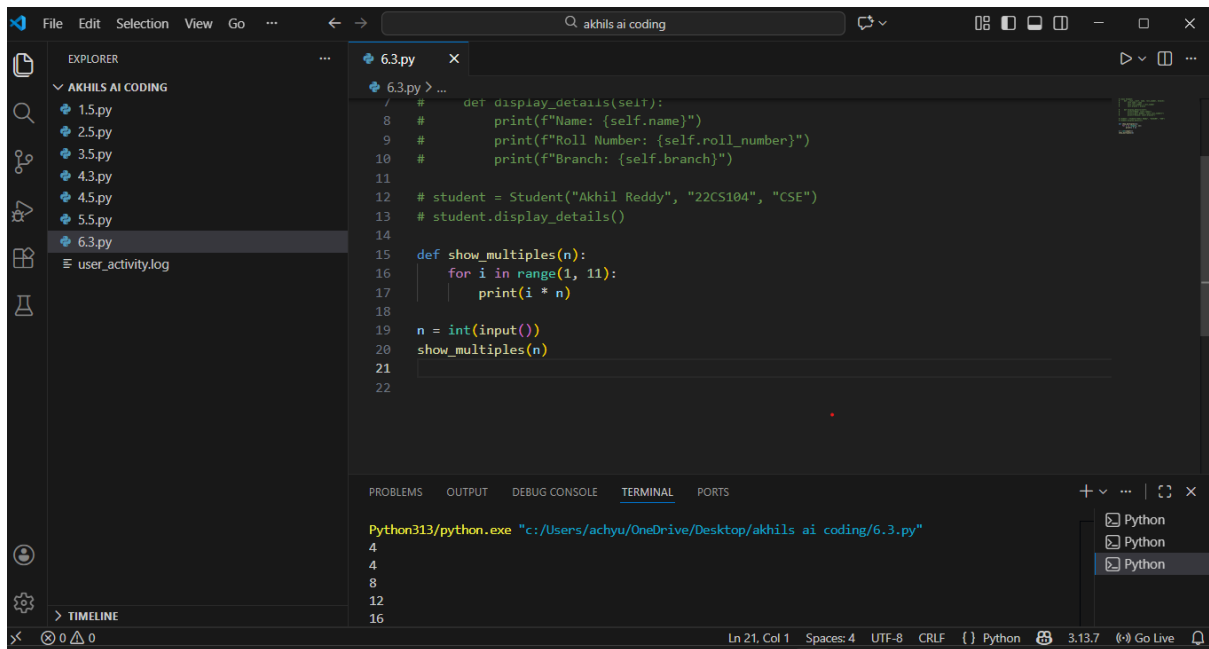
Expected Output #2

- Correct loop-based Python implementation.
- Output showing the first 10 multiples of a number.
- Comparison and analysis of different looping approaches.

PROMPT:

Write a Python function called `show_multiples` that takes an integer `n` as an argument. Use a `for` loop to calculate and print the first 10 multiples of that number (starting from $1 * n$). After the code, provide a brief explanation of how the loop range is defined.

EXPECTED OUTPUT:



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).

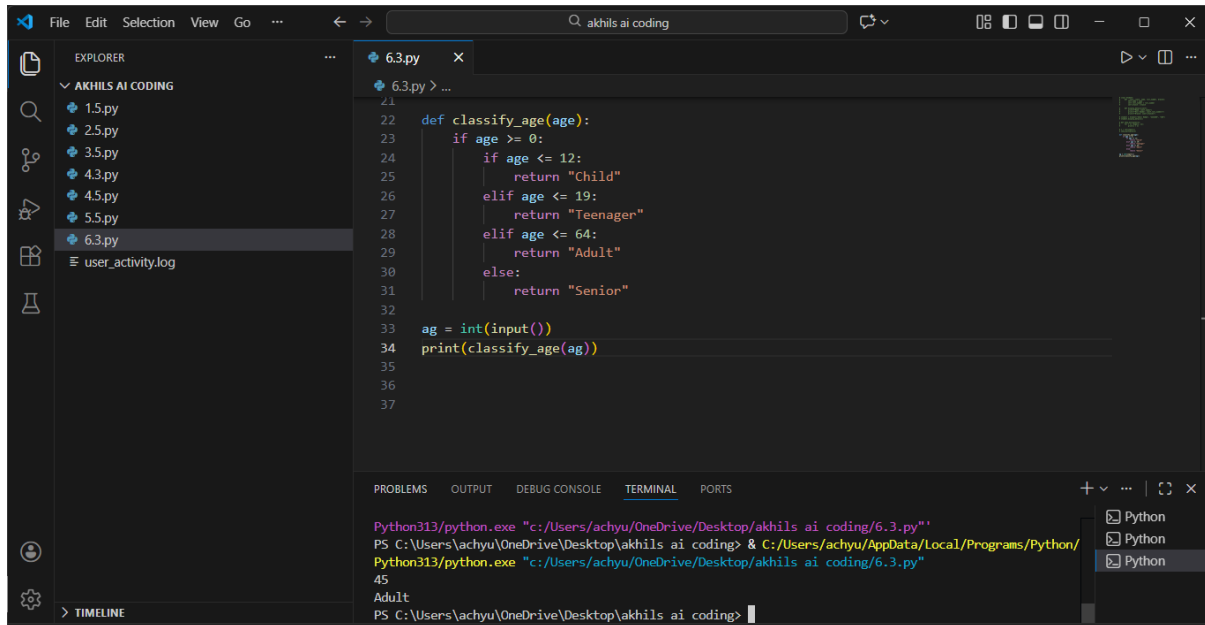
Expected Output #3

- A Python function that classifies age into appropriate groups.
- Clear and correct conditional logic.
- Explanation of how the conditions work.

PROMPT:

Write a Python function called `classify_age` that takes an integer `age` as an input. Use a nested `if-elif-else` structure to return a category: 'Child' (0–12), 'Teenager' (13–19), 'Adult' (20–64), and 'Senior' (65+). After the code, analyze why the order of conditions matters and then provide an alternative version using a dictionary-based approach or simplified logical boundaries.

EXPECTED OUTPUT:



The screenshot shows a VS Code editor window with a file explorer on the left and a code editor in the center. The file explorer shows a folder named 'AKHILS AI CODING' containing several Python files (1.5.py, 2.5.py, 3.5.py, 4.3.py, 4.5.py, 5.5.py, 6.3.py) and a file named 'user_activity.log'. The code editor shows the content of '6.3.py'.

```
21
22 def classify_age(age):
23     if age >= 0:
24         if age <= 12:
25             return "Child"
26         elif age <= 19:
27             return "Teenager"
28         elif age <= 64:
29             return "Adult"
30         else:
31             return "Senior"
32
33 ag = int(input())
34 print(classify_age(ag))
35
36
37
```

The terminal at the bottom shows the command prompt output:

```
Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/6.3.py"
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/6.3.py"
45
Adult
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>
```

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.

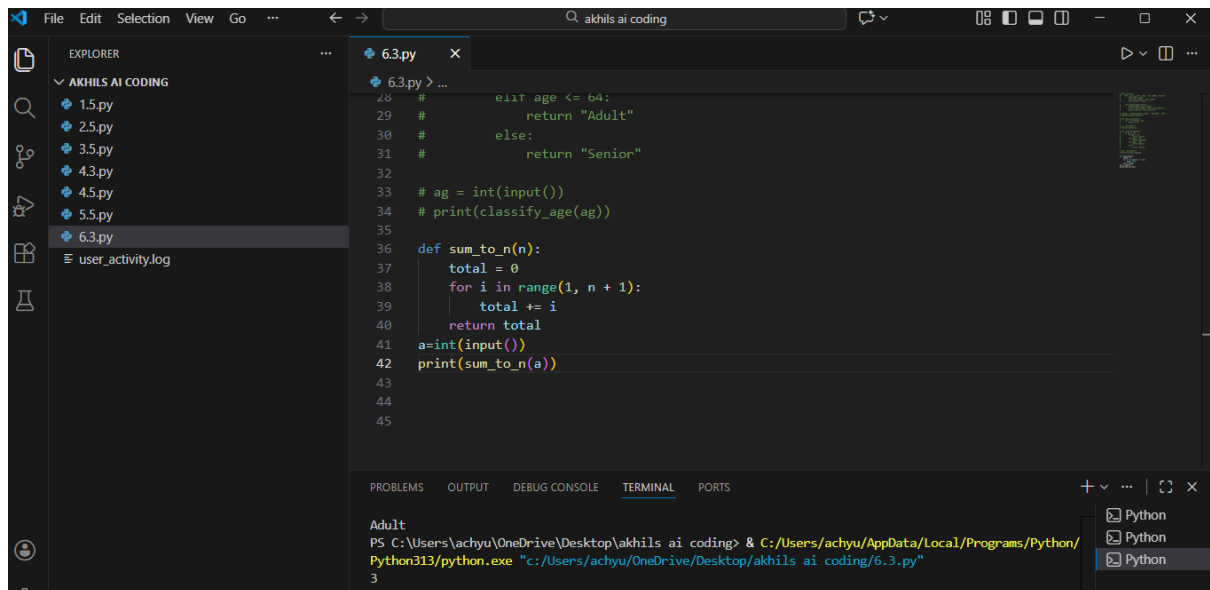
Expected Output #4

- Python function to compute the sum of first n numbers.
- Correct output for sample inputs.
- Explanation and comparison of different approaches.

PROMPT:

Write a Python function `sum_to_n(n)` that calculates the sum of the first `n` natural numbers using a `for` loop. After providing the code, analyze the logic behind the accumulator variable. Finally, provide two alternative implementations: one using a `while` loop and one using the mathematical formula (Arithmetic Progression), and compare the performance and readability of all three methods.

EXPECTED OUTPUT:



```
28 #     elif age <= 64:
29 #         return "Adult"
30 #     else:
31 #         return "Senior"
32
33 # ag = int(input())
34 # print(classify_age(ag))
35
36 def sum_to_n(n):
37     total = 0
38     for i in range(1, n + 1):
39         total += i
40     return total
41 a=int(input())
42 print(sum_to_n(a))
43
44
45
```

Adult

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/6.3.py"

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

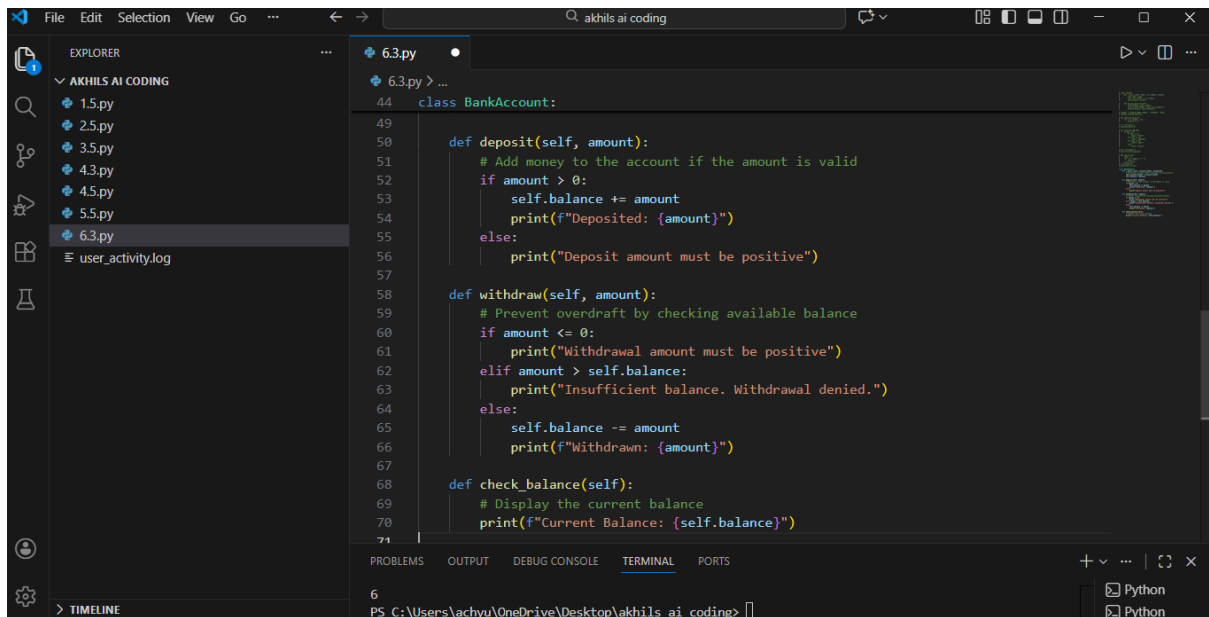
- Complete Python Bank Account class.
- Demonstration of deposit and withdrawal operations with updated balance.

- Well-commented code with a clear explanation.

PROMPT:

create a Python class named `BankAccount`. The class should initialize with an `account_holder` name and a `balance` (defaulting to 0). Implement three methods: `deposit(amount)`, `withdraw(amount)`, and `check_balance()`. Ensure the `withdraw` method includes logic to prevent overdrafts. Please provide the code with meaningful comments, a brief analysis of the class structure, and a demonstration of the class in action with multiple transactions.

EXPECTED OUTPUT:



```
44 class BankAccount:
45
46     def deposit(self, amount):
47         # Add money to the account if the amount is valid
48         if amount > 0:
49             self.balance += amount
50             print(f"Deposited: {amount}")
51         else:
52             print("Deposit amount must be positive")
53
54     def withdraw(self, amount):
55         # Prevent overdraft by checking available balance
56         if amount <= 0:
57             print("Withdrawal amount must be positive")
58         elif amount > self.balance:
59             print("Insufficient balance. Withdrawal denied.")
60         else:
61             self.balance -= amount
62             print(f"Withdrawn: {amount}")
63
64     def check_balance(self):
65         # Display the current balance
66         print(f"Current Balance: {self.balance}")
67
68
69
70
71
```

6
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>