LAB ASSIGNMENT- 4.4

Name: E. Harini

Enrollment No.: 2503A51L40

Course Code: 24CS002PC215

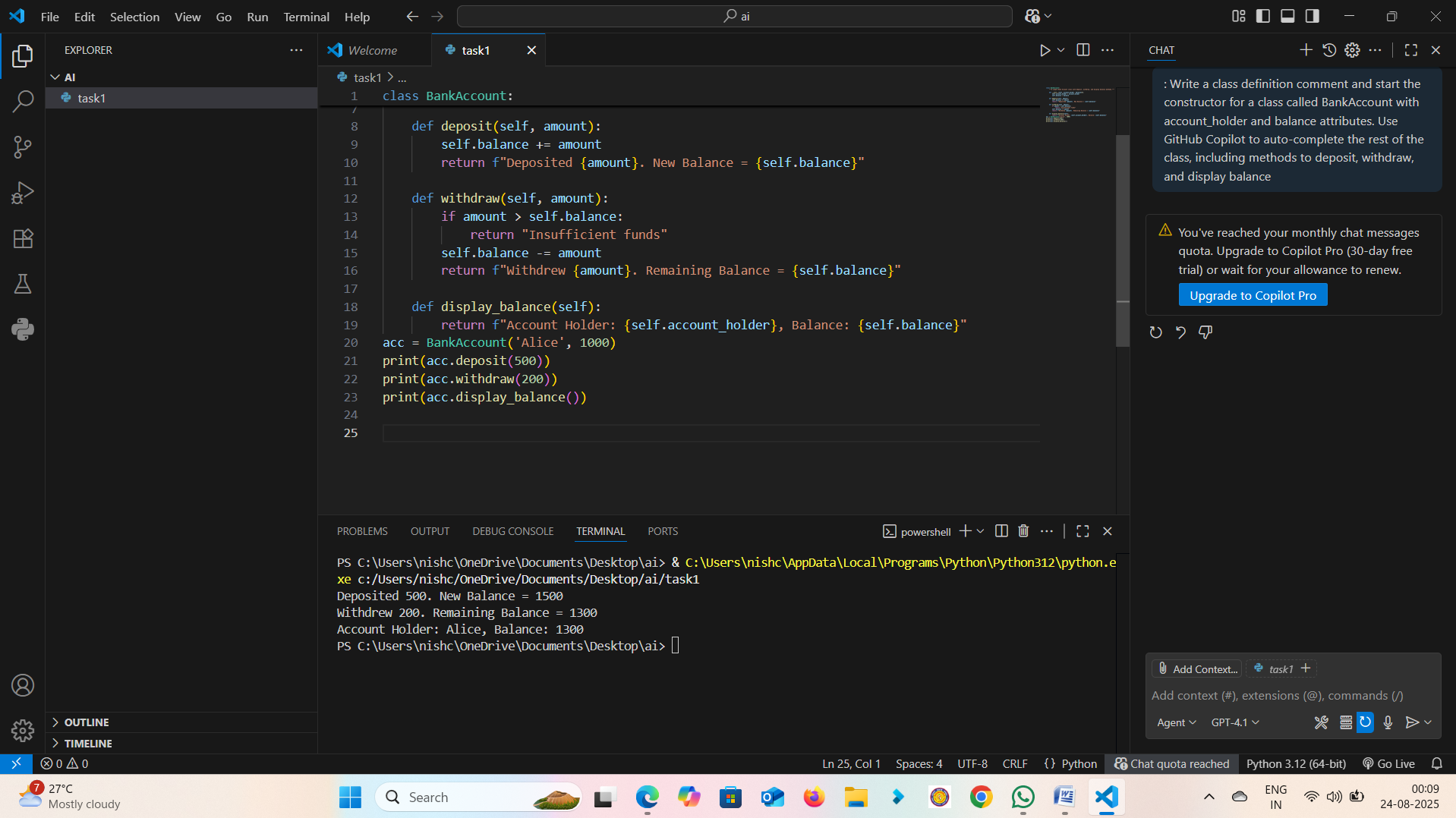
Course Title: AI Assisted Coding

Lab Number: 4.4

BRANCH: cse  
  
Task 1: Auto-Complete a Python Class for Bank Account

Prompt: Write a class definition comment and start the constructor for a class called BankAccount with account\_holder and balance attributes. Use GitHub Copilot to auto-complete the rest of the class, including methods to deposit, withdraw, and display balance.

Python Code:   
class BankAccount:  
 """A simple Bank Account class with deposit, withdraw, and display balance methods."""  
 def \_\_init\_\_(self, account\_holder, balance=0):  
 self.account\_holder = account\_holder  
 self.balance = balance  
  
 def deposit(self, amount):  
 self.balance += amount  
 return f"Deposited {amount}. New Balance = {self.balance}"  
  
 def withdraw(self, amount):  
 if amount > self.balance:  
 return "Insufficient funds"  
 self.balance -= amount  
 return f"Withdrew {amount}. Remaining Balance = {self.balance}"  
  
 def display\_balance(self):  
 return f"Account Holder: {self.account\_holder}, Balance: {self.balance}"

Explanation: The class has attributes for account holder and balance. Methods allow deposit, withdrawal with balance check, and displaying account details. 

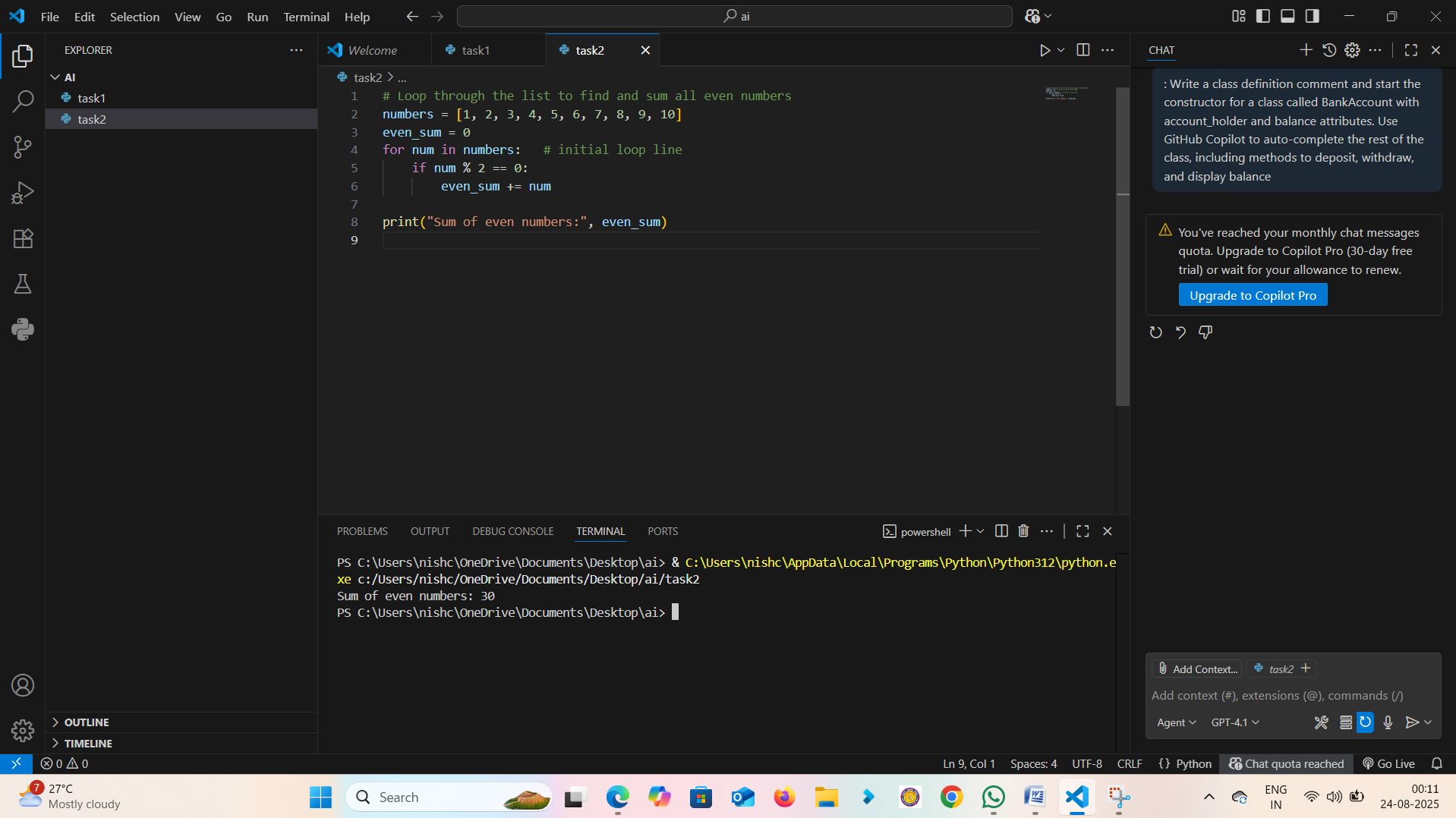
Sample Output:  
>>> acc = BankAccount('Alice', 1000)  
>>> print(acc.deposit(500))  
Deposited 500. New Balance = 1500  
>>> print(acc.withdraw(200))  
Withdrew 200. Remaining Balance = 1300  
>>> print(acc.display\_balance())  
Account Holder: Alice, Balance: 1300

Observation: The BankAccount class successfully handled deposits, withdrawals, and displayed balance accurately.

Task 2: Auto-Complete a For Loop to Sum Even Numbers in a List

Prompt: Write a comment and the initial line of a loop to iterate over a list. Allow GitHub Copilot to complete the logic to sum all even numbers in the list.

Python Code:  
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
even\_sum = 0  
for num in numbers:  
 if num % 2 == 0:  
 even\_sum += num  
print("Sum of even numbers:", even\_sum)

Explanation: The loop iterates through the list, checks if each number is even, and adds it to the running sum. 

Sample Output:  
Sum of even numbers: 30

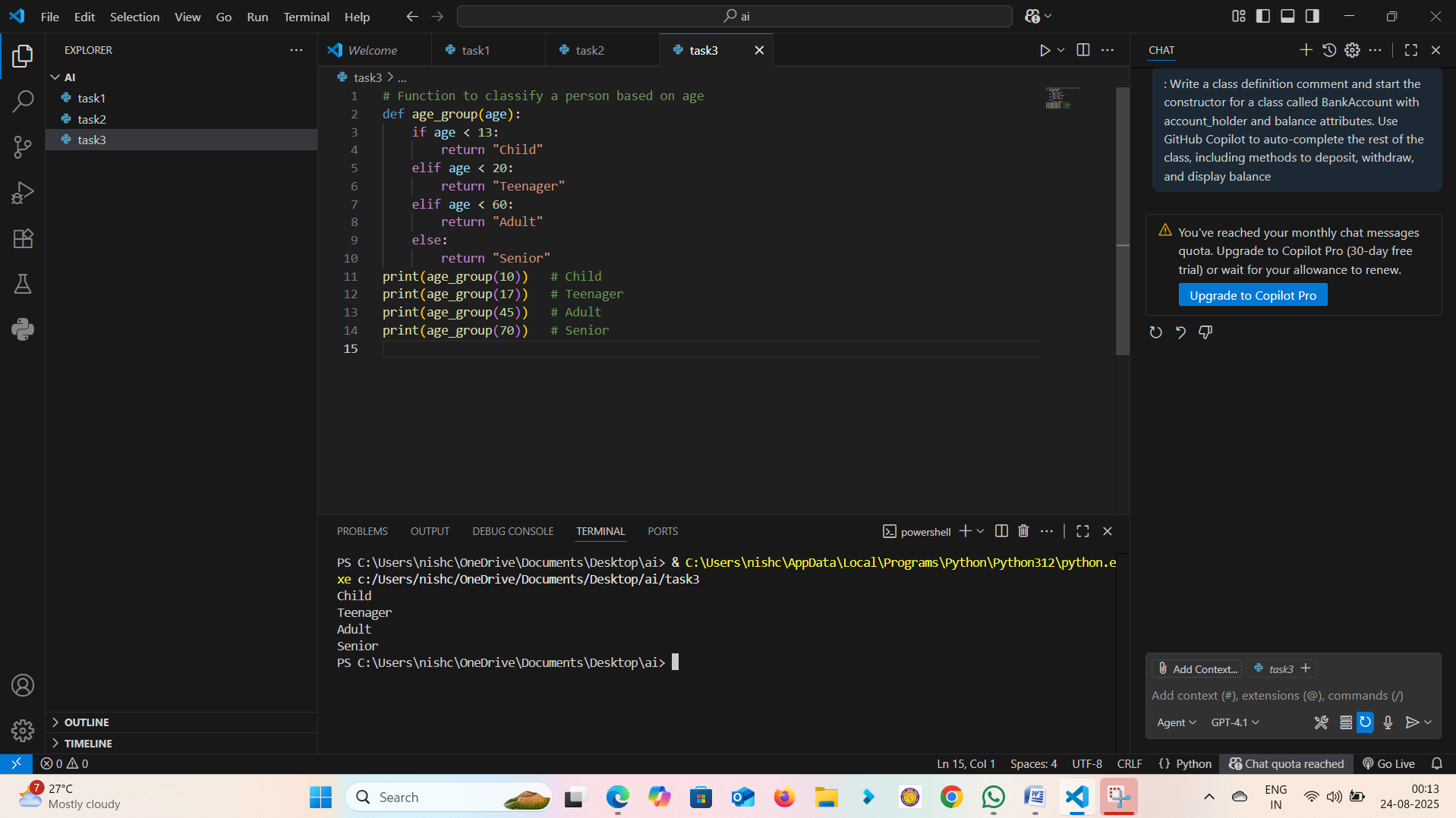
Observation: The loop correctly iterated and summed even numbers from the list.

Task 3: Auto-Complete Conditional Logic to Check Age Group

Prompt: Start a function that takes age as input and returns whether the person is a child, teenager, adult, or senior using if-elif-else.

Python Code:  
def age\_group(age):  
 if age < 13:  
 return "Child"  
 elif age < 20:  
 return "Teenager"  
 elif age < 60:  
 return "Adult"  
 else:  
 return "Senior"

Explanation: The function uses if-elif-else conditionals to classify age groups.



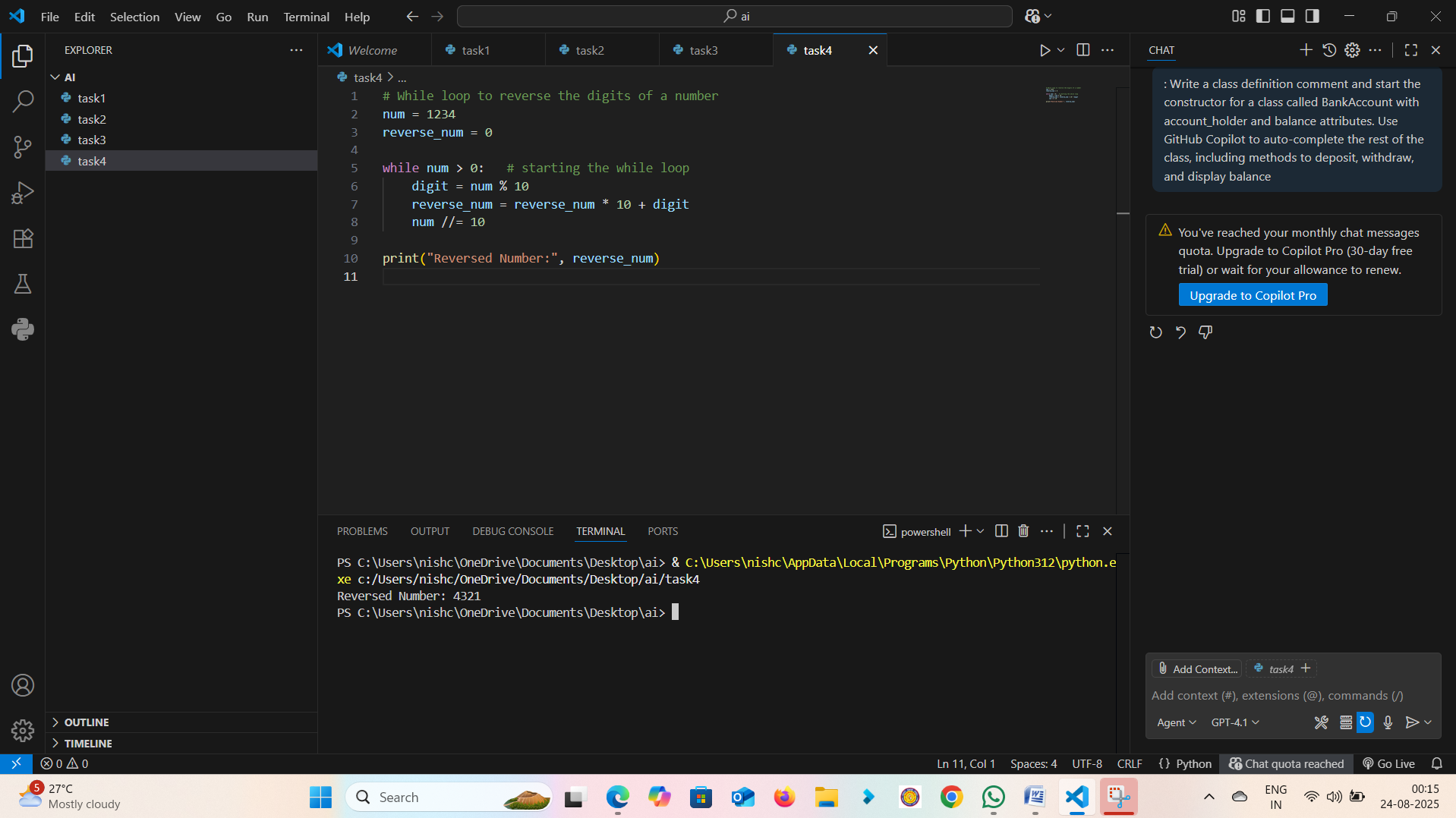
Sample Output:  
>>> age\_group(10) -> Child  
>>> age\_group(17) -> Teenager  
>>> age\_group(45) -> Adult  
>>> age\_group(70) -> Senior

Observation: The function correctly classified age groups based on input values.

Task 4: Auto-Complete a While Loop to Reverse Digits of a Number

Prompt: Write a comment and start a while loop to reverse the digits of a number.

Python Code:  
num = 1234  
reverse\_num = 0  
while num > 0:  
 digit = num % 10  
 reverse\_num = reverse\_num \* 10 + digit  
 num //= 10  
print("Reversed Number:", reverse\_num)

Explanation: The loop extracts the last digit using modulo, builds the reversed number, and reduces the original number using integer division. 

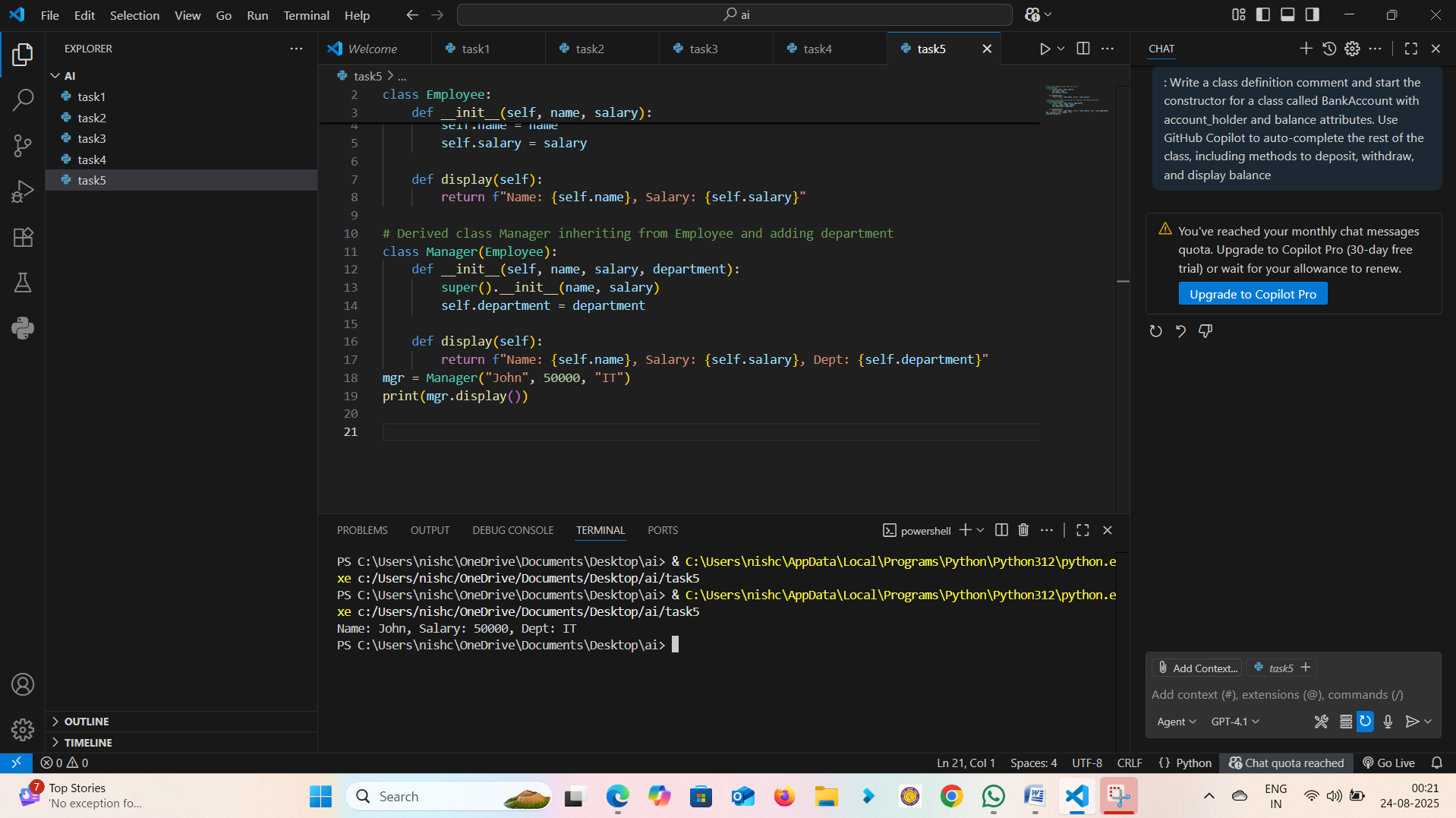
Sample Output:  
Reversed Number: 4321

Observation: The while loop reversed the digits of the number without errors.

Task 5: Auto-Complete Class with Inheritance (Employee → Manager)

Prompt: Begin a class Employee with attributes name and salary. Then, start a derived class Manager that inherits from Employee and adds a department.

Python Code:  
class Employee:  
 def \_\_init\_\_(self, name, salary):  
 self.name = name  
 self.salary = salary  
  
 def display(self):  
 return f"Name: {self.name}, Salary: {self.salary}"  
  
class Manager(Employee):  
 def \_\_init\_\_(self, name, salary, department):  
 super().\_\_init\_\_(name, salary)  
 self.department = department  
  
 def display(self):  
 return f"Name: {self.name}, Salary: {self.salary}, Dept: {self.department}"

Explanation: The Manager class inherits from Employee using super() for constructor chaining and overrides the display method.  


Sample Output:  
Name: John, Salary: 50000, Dept: IT

Observation: The Manager class inherited Employee attributes and methods correctly while extending functionality.

Observation

In this lab, we explored GitHub Copilot's ability to auto-complete Python code for classes, loops, and conditionals. We practiced building classes with inheritance, loops for summing and reversing, and conditional logic for classification. This enhanced understanding of AI-assisted coding and Python fundamentals.