

ASSIGNMENT – 4

NAME: A. MANIDEEPIKA

HT.NO: 2403A52052

BATCH NO: AIB03

TASK 1:

Complete class with methods like:

deposit(self, amount)

withdraw(self, amount)

display_balance(self)

PROMPT:

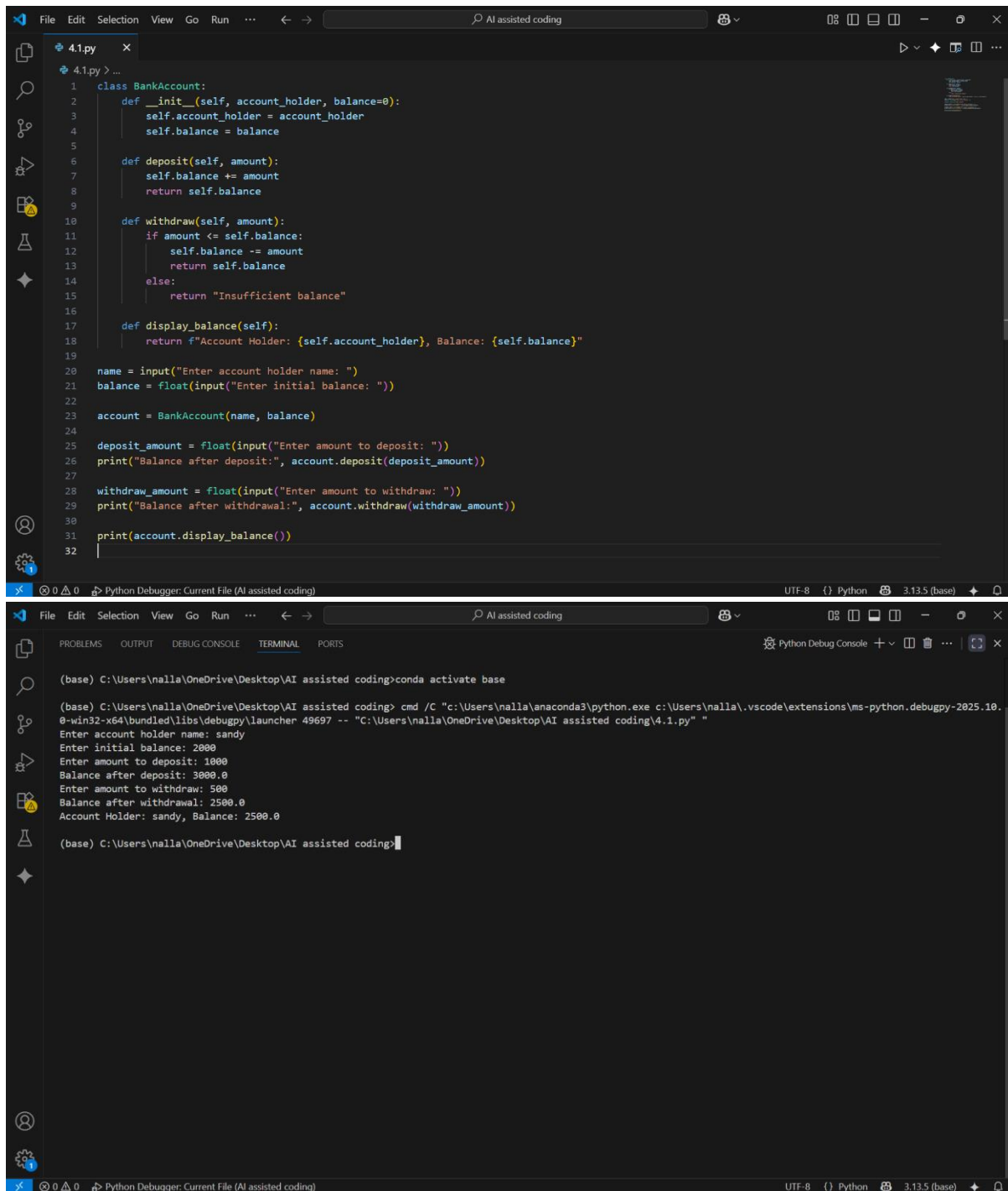
Complete class with methods like:

deposit(self, amount)

withdraw(self, amount)

display_balance(self)

CODE:



The image consists of two screenshots of a Visual Studio Code editor. The top screenshot shows a Python file named '4.1.py' with a class definition for 'BankAccount' and a main program that interacts with the user. The class has methods for deposit, withdrawal, and displaying the balance. The main program prompts the user for their name, initial balance, deposit amount, and withdrawal amount, then updates the account and displays the final balance. The bottom screenshot shows the terminal output of the program, demonstrating the execution flow and the resulting balance after a deposit and a withdrawal.

```
1 class BankAccount:
2     def __init__(self, account_holder, balance=0):
3         self.account_holder = account_holder
4         self.balance = balance
5
6     def deposit(self, amount):
7         self.balance += amount
8         return self.balance
9
10    def withdraw(self, amount):
11        if amount <= self.balance:
12            self.balance -= amount
13            return self.balance
14        else:
15            return "Insufficient balance"
16
17    def display_balance(self):
18        return f"Account Holder: {self.account_holder}, Balance: {self.balance}"
19
20 name = input("Enter account holder name: ")
21 balance = float(input("Enter initial balance: "))
22
23 account = BankAccount(name, balance)
24
25 deposit_amount = float(input("Enter amount to deposit: "))
26 print("Balance after deposit:", account.deposit(deposit_amount))
27
28 withdraw_amount = float(input("Enter amount to withdraw: "))
29 print("Balance after withdrawal:", account.withdraw(withdraw_amount))
30
31 print(account.display_balance())
32
```

```
(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>conda activate base
(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>cmd /C "c:\Users\nalla\anaconda3\python.exe c:\Users\nalla\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher 49697 -- "c:\Users\nalla\OneDrive\Desktop\AI assisted coding\4.1.py" "
Enter account holder name: sandy
Enter initial balance: 2000
Enter amount to deposit: 1000
Balance after deposit: 3000.0
Enter amount to withdraw: 500
Balance after withdrawal: 2500.0
Account Holder: sandy, Balance: 2500.0
(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>
```

OBSERVATION:

This code defines a `BankAccount` class to manage a simple bank account with a holder's name and balance. It has methods to deposit money, withdraw money if enough balance is available, and display account details. The program asks the user for their name, initial balance, deposit, and withdrawal amounts, updates the account accordingly, and shows the final balance.

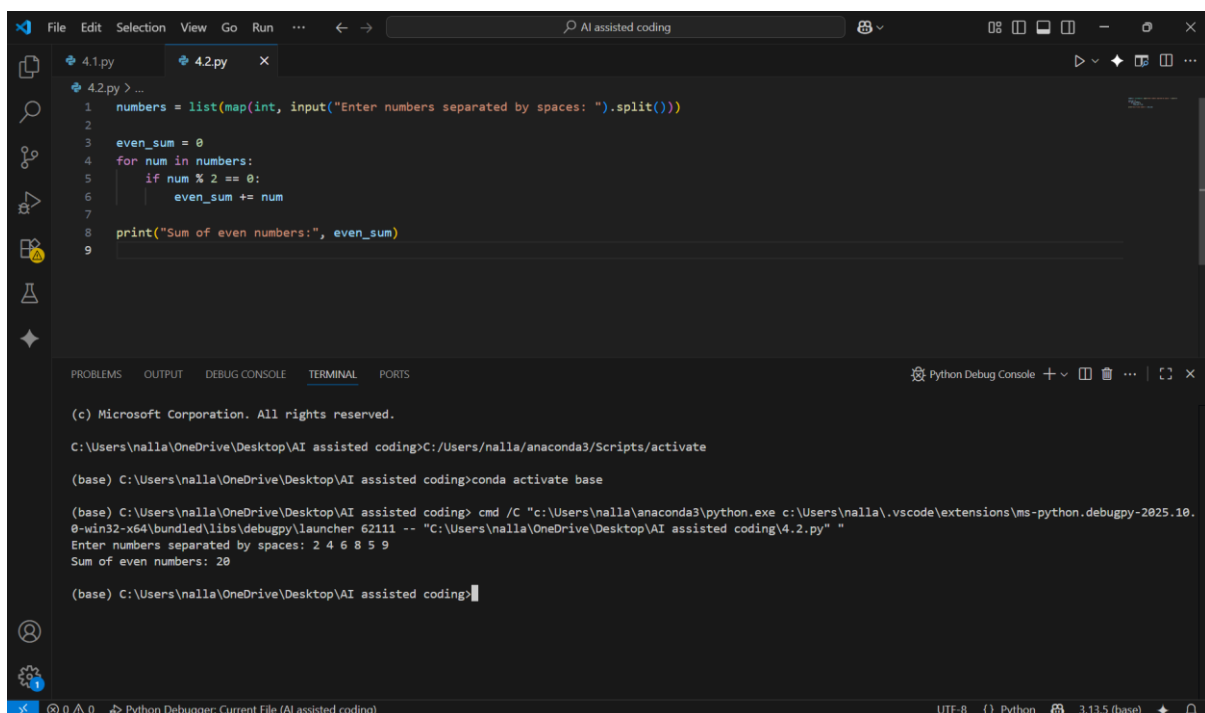
TASK 2:

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.

PROMPT:

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

CODE:



The screenshot shows a Visual Studio Code editor with a Python file named 4.2.py. The code in the file is as follows:

```
1 numbers = list(map(int, input("Enter numbers separated by spaces: ").split()))
2
3 even_sum = 0
4 for num in numbers:
5     if num % 2 == 0:
6         even_sum += num
7
8 print("Sum of even numbers:", even_sum)
9
```

Below the editor, the TERMINAL panel shows the execution of the script. The user has activated the base environment and run the script using the command: `cmd /C "c:\Users\nalla\anaconda3\python.exe c:\Users\nalla\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher 62111 -- "C:\Users\nalla\OneDrive\Desktop\AI assisted coding\4.2.py"`. The terminal output shows the prompt "Enter numbers separated by spaces: 2 4 6 8 5 9" and the result "Sum of even numbers: 20".

OBSERVATION:

This Python code calculates the sum of even numbers from a list entered by the user. It first prompts the user to input numbers separated by spaces, then converts that input into a list of integers using `map` and `split`. It initializes a variable `even_sum` to zero, then iterates through each number in the list. If a number is divisible by 2 (i.e., it's even), it adds that number to `even_sum`. Finally, it prints the total sum of all even numbers entered by the user.

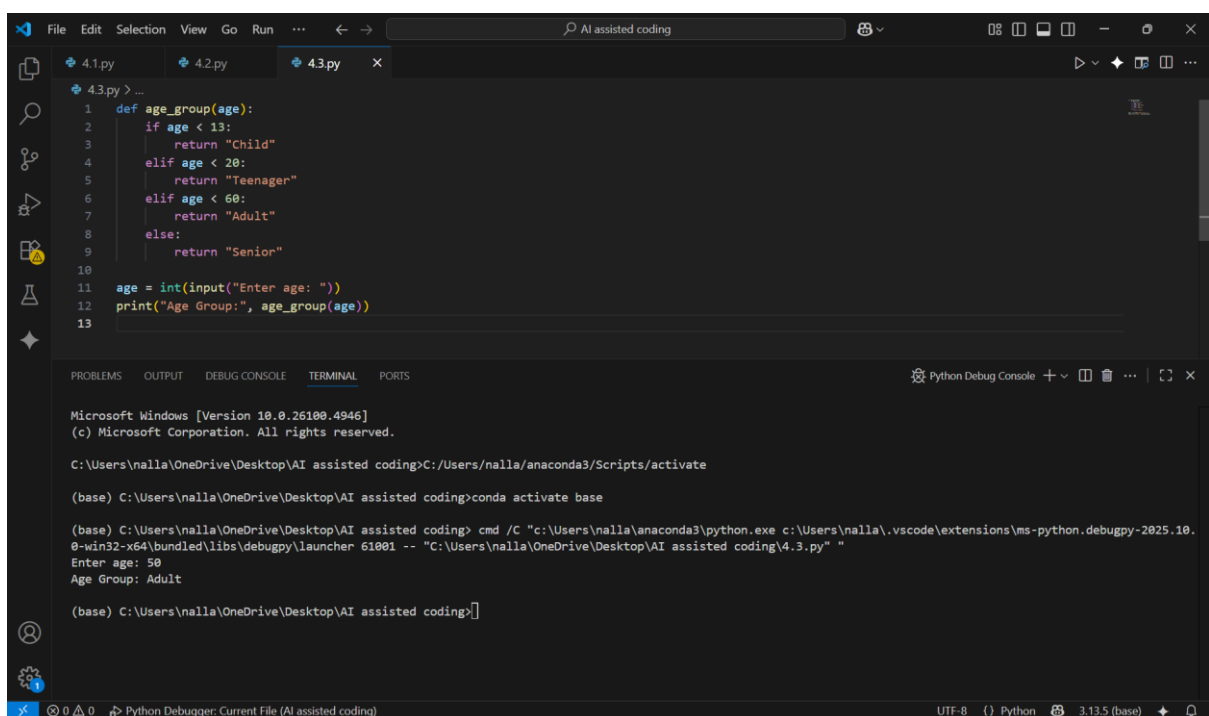
TASK 3:

Start a function that takes age as input and returns whether the person is a child, teenager, adult, or senior using if-elif-else. Use Copilot to complete the conditionals.

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.

CODE:



The screenshot shows a Visual Studio Code editor with a Python file named 4.3.py. The code defines a function `age_group` that categorizes a person based on their age. The function uses if-elif-else statements to return "Child", "Teenager", "Adult", or "Senior". Below the function, the code prompts the user to enter their age, converts it to an integer, and prints the resulting age group. The terminal window at the bottom shows the execution of the script, where the user entered 50 and the output was "Age Group: Adult".

```
1 def age_group(age):
2     if age < 13:
3         return "Child"
4     elif age < 20:
5         return "Teenager"
6     elif age < 60:
7         return "Adult"
8     else:
9         return "Senior"
10
11 age = int(input("Enter age: "))
12 print("Age Group:", age_group(age))
13
```

```
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nalla\OneDrive\Desktop\AI assisted coding>C:\Users\nalla\anaconda3\Scripts\activate

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>conda activate base

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>cmd /C "c:\Users\nalla\anaconda3\python.exe c:\Users\nalla\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher 61001 -- "C:\Users\nalla\OneDrive\Desktop\AI assisted coding\4.3.py" "
Enter age: 50
Age Group: Adult

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>
```

OBSERVATION:

This Python code defines a function `age_group` that categorizes a person based on their age. It takes an integer age as input and checks several conditions: if the age is less than 13, it returns "Child"; if it is between 13 and 19, it returns "Teenager"; if it is between 20 and 59, it returns "Adult"; and if it is 60 or above, it returns "Senior". The program prompts the user to enter their age, converts it to an integer, passes it to the `age_group` function, and then prints the resulting age category.

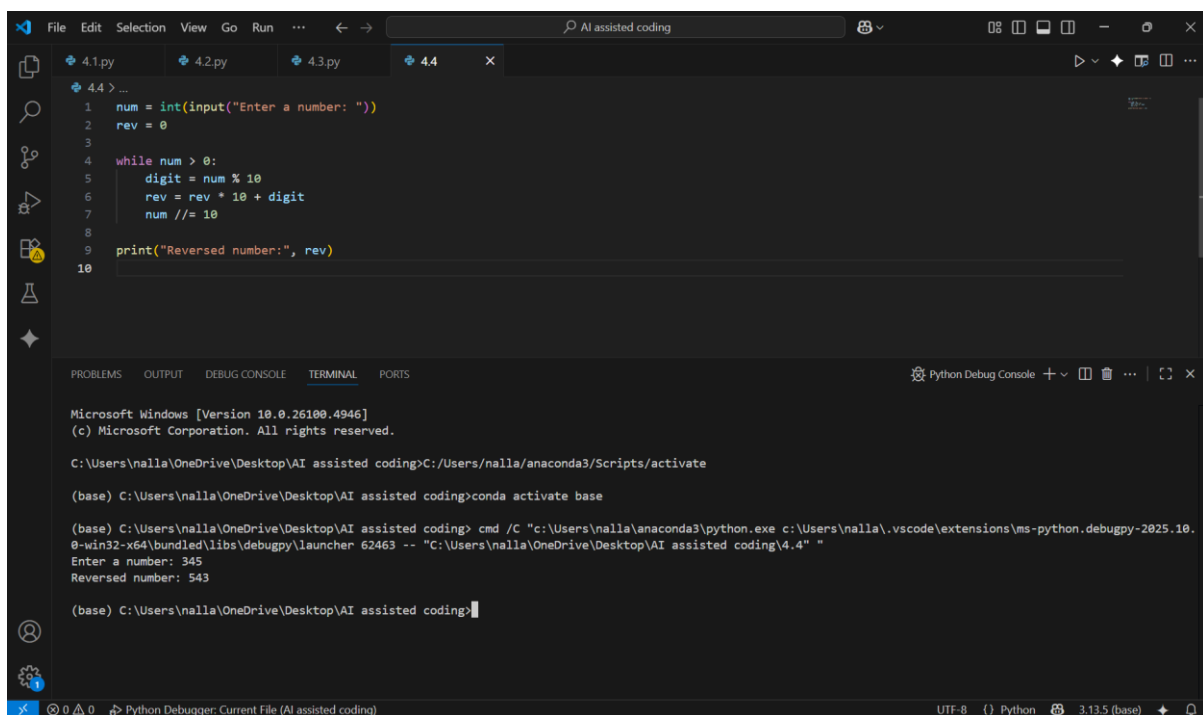
TASK 4:

Write a comment and start a while loop to reverse the digits of a number. Let Copilot complete the loop logic

PROMPT:

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

CODE:



```
4.4 > ...
1 num = int(input("Enter a number: "))
2 rev = 0
3
4 while num > 0:
5     digit = num % 10
6     rev = rev * 10 + digit
7     num //= 10
8
9 print("Reversed number:", rev)
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console

Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nalla\OneDrive\Desktop\AI assisted coding>C:\Users\nalla\anaconda3\Scripts\activate

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>conda activate base

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>cmd /C "c:\Users\nalla\anaconda3\python.exe c:\Users\nalla\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher 62463 -- "C:\Users\nalla\OneDrive\Desktop\AI assisted coding\4.4" "

Enter a number: 345
Reversed number: 543

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>

Python Debugger: Current File (AI assisted coding) UTF-8 Python 3.13.5 (base)

OBSERVATION:

This Python code reverses the digits of a number entered by the user. It first prompts the user to input a number and stores it in `num`, while initializing `rev` to 0 to hold the reversed number. Inside a while loop that runs as long as `num` is greater than 0, it extracts the last digit using `num % 10`, adds it to `rev` after shifting the existing digits to the left (`rev * 10 + digit`), and removes the last digit from `num` using integer division `num //= 10`. After the loop finishes, it prints the reversed number stored in `rev`.

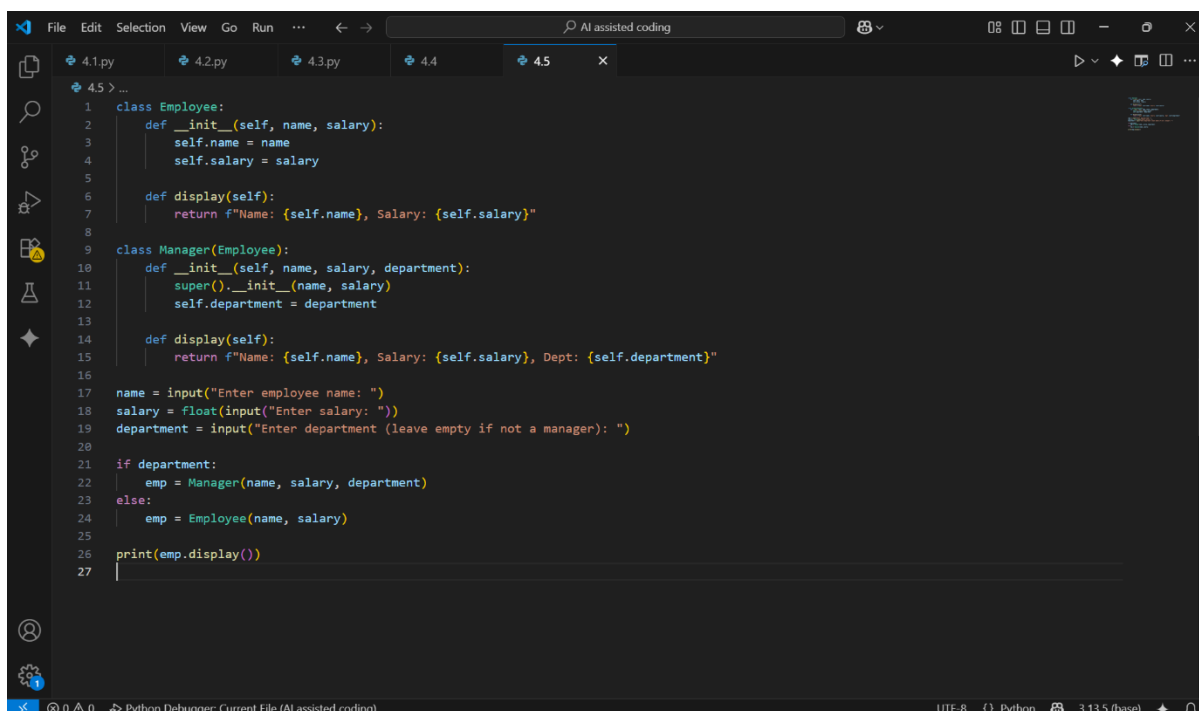
TASK 5:

Begin a class `Employee` with attributes `name` and `salary`. Then, start a derived class `Manager` that inherits from `Employee` and adds a `department`. Let GitHub Copilot complete the methods and constructor chaining

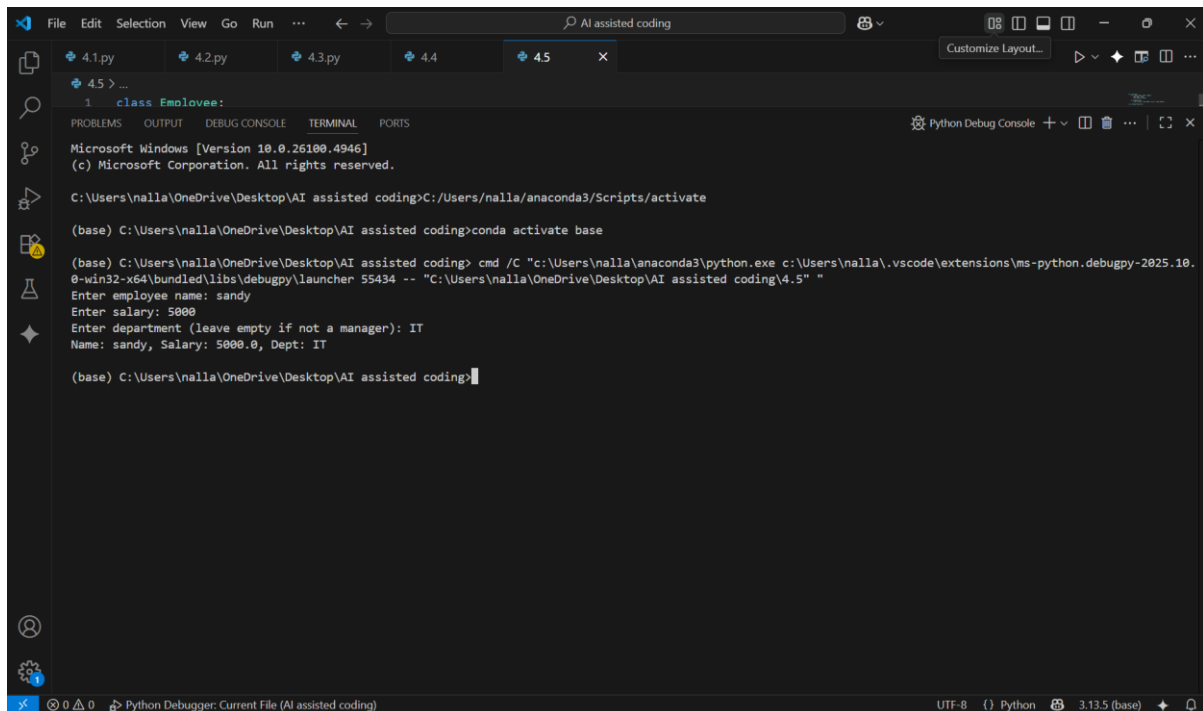
PROMPT:

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

CODE:

A screenshot of a code editor interface, likely Visual Studio Code, showing a Python file named 4.5.py. The code defines two classes: Employee and Manager. The Employee class has attributes name and salary, and a display method. The Manager class inherits from Employee and adds a department attribute. The code also includes a main block that takes user input for name, salary, and department, creates an instance of either Manager or Employee, and prints the display output. The editor shows line numbers from 1 to 27. The status bar at the bottom indicates the file is UTF-8, Python 3.13.5 (base), and is being debugged with AI assisted coding.

```
1 class Employee:
2     def __init__(self, name, salary):
3         self.name = name
4         self.salary = salary
5
6     def display(self):
7         return f"Name: {self.name}, Salary: {self.salary}"
8
9 class Manager(Employee):
10     def __init__(self, name, salary, department):
11         super().__init__(name, salary)
12         self.department = department
13
14     def display(self):
15         return f"Name: {self.name}, Salary: {self.salary}, Dept: {self.department}"
16
17 name = input("Enter employee name: ")
18 salary = float(input("Enter salary: "))
19 department = input("Enter department (leave empty if not a manager): ")
20
21 if department:
22     emp = Manager(name, salary, department)
23 else:
24     emp = Employee(name, salary)
25
26 print(emp.display())
27
```



```
1 class Employee:
2     def __init__(self, name, salary):
3         self.name = name
4         self.salary = salary
5     def display(self):
6         return f"Name: {self.name}, Salary: {self.salary}"
7
8 class Manager(Employee):
9     def __init__(self, name, salary, department):
10        self.name = name
11        self.salary = salary
12        self.department = department
13    def display(self):
14        return f"Name: {self.name}, Salary: {self.salary}, Dept: {self.department}"
15
16 if __name__ == "__main__":
17     name = input("Enter employee name: ")
18     salary = input("Enter salary: ")
19     department = input("Enter department (leave empty if not a manager): ")
20     if department:
21         manager = Manager(name, salary, department)
22         print(manager.display())
23     else:
24         employee = Employee(name, salary)
25         print(employee.display())
```

Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nalla\OneDrive\Desktop\AI assisted coding>C:\Users\nalla\anaconda3\Scripts\activate

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>conda activate base

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>cmd /C "C:\Users\nalla\anaconda3\python.exe c:\Users\nalla\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher 55434 -- "C:\Users\nalla\OneDrive\Desktop\AI assisted coding\4.5" "

Enter employee name: sandy

Enter salary: 5000

Enter department (leave empty if not a manager): IT

Name: sandy, Salary: 5000.0, Dept: IT

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>

OBSERVATION:

This Python code demonstrates **inheritance** in classes. It defines an `Employee` class with a constructor that stores the employee's name and salary, and a `display` method that returns these details as a string. Then it defines a `Manager` class that inherits from `Employee` and adds a `department` attribute. The `Manager` class also overrides the `display` method to include the department in the output. The program asks the user to enter a name, salary, and optionally a department. If a department is provided, it creates a `Manager` object; otherwise, it creates a regular `Employee` object. Finally, it prints the details of the created object using the `display` method.