2403a51272

MARIPUDI ANJALI

CODE WRITTEN:

class BankAccount:

"""

A class to represent a bank account.

Attributes:

account\_holder (str): The name of the person who owns the account.

balance (float): The current balance in the account.

"""

def \_\_init\_\_(self, account\_holder, balance):

"""

Initialize a new BankAccount instance.

Parameters:

account\_holder (str): The name of the account holder.

balance (float): The initial balance of the account.

"""

self.account\_holder = account\_holder

self.balance = balance

CODE AUTO-COMPLETED BY GITHUB COPILOT:

class BankAccount:

"""

A class to represent a bank account.

Attributes:

account\_holder (str): The name of the person who owns the account.

balance (float): The current balance in the account.

"""

def \_\_init\_\_(self, account\_holder, balance):

"""

Initialize a new BankAccount instance.

Parameters:

account\_holder (str): The name of the account holder.

balance (float): The initial balance of the account.

"""

self.account\_holder = account\_holder

self.balance = balance

def deposit(self, amount):

"""

Deposit money into the account.

Parameters:

amount (float): The amount of money to deposit.

Returns:

None

"""

if amount > 0:

self.balance += amount

print(f"Deposited {amount}. New balance: {self.balance}")

else:

print("Deposit amount must be positive.")

def withdraw(self, amount):

"""

Withdraw money from the account.

Parameters:

amount (float): The amount of money to withdraw.

Returns:

None

"""

if amount > 0:

if amount <= self.balance:

self.balance -= amount

print(f"Withdrew {amount}. New balance: {self.balance}")

else:

print("Insufficient funds.")

else:

print("Withdrawal amount must be positive.")

def display\_balance(self):

"""

Display the current balance of the account.

Returns:

None

"""

print(f"Account Holder: {self.account\_holder}")

print(f"Current Balance: {self.balance}")

# ...existing code...

if \_name\_ == "\_main\_":

account = BankAccount("Alice", 1000)

account.deposit(500)

account.withdraw(200)

account.display\_balance()

EXPLANATION:

class BankAccount: defines a blueprint for creating bank account objects.

The constructor \_init\_ sets up account holder name and starting balance.

self.account\_holder keeps track of who owns the account.

self.balance stores how much money is inside the account.

The deposit method increases the balance by the given amount.

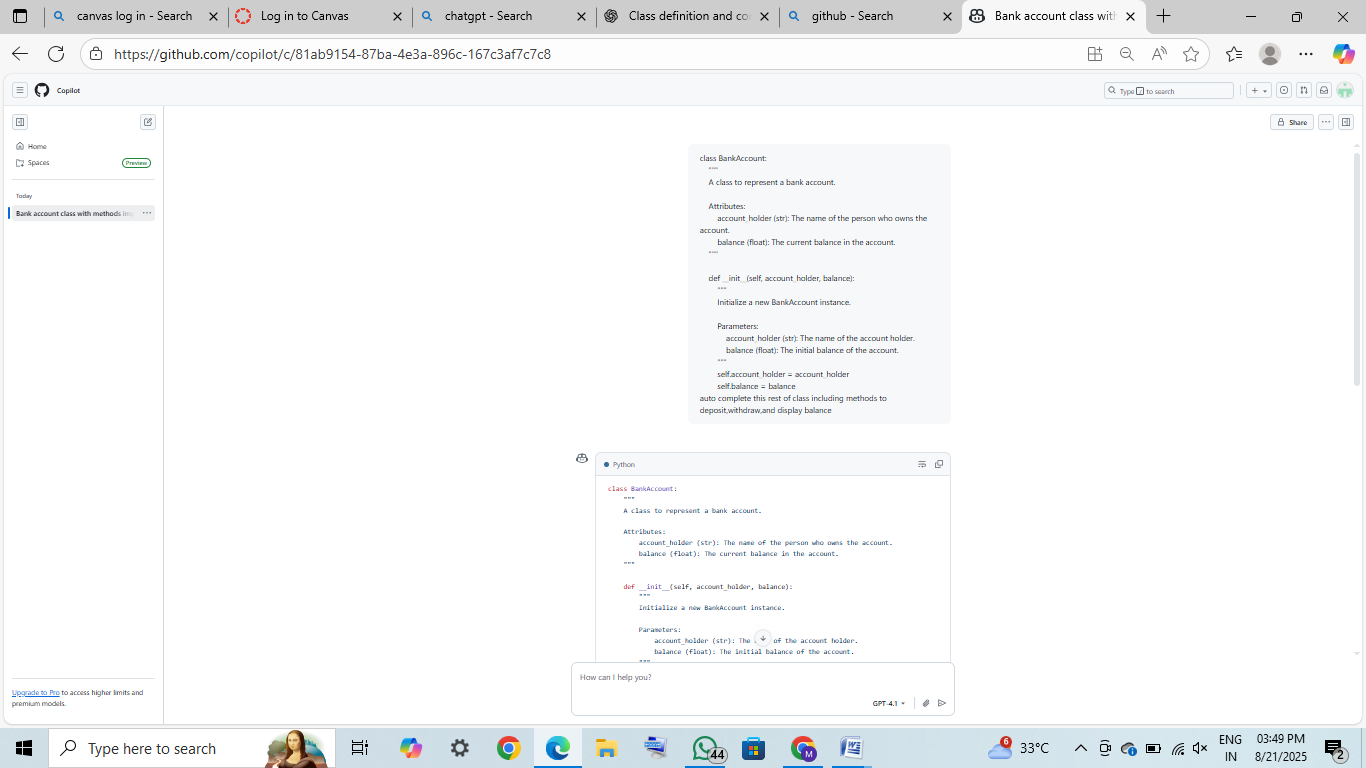
The withdraw method decreases the balance if there is enough money.

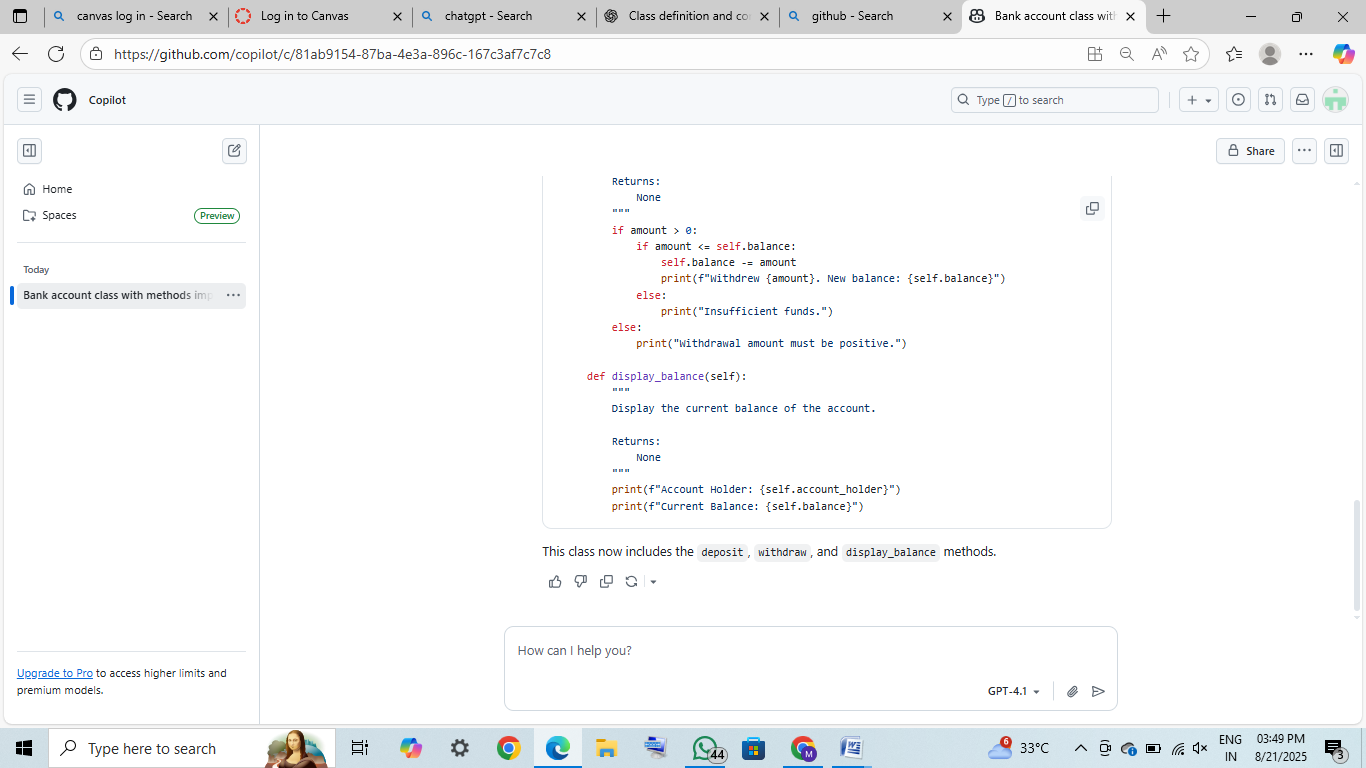
If not enough money, it prints "Insufficient funds".

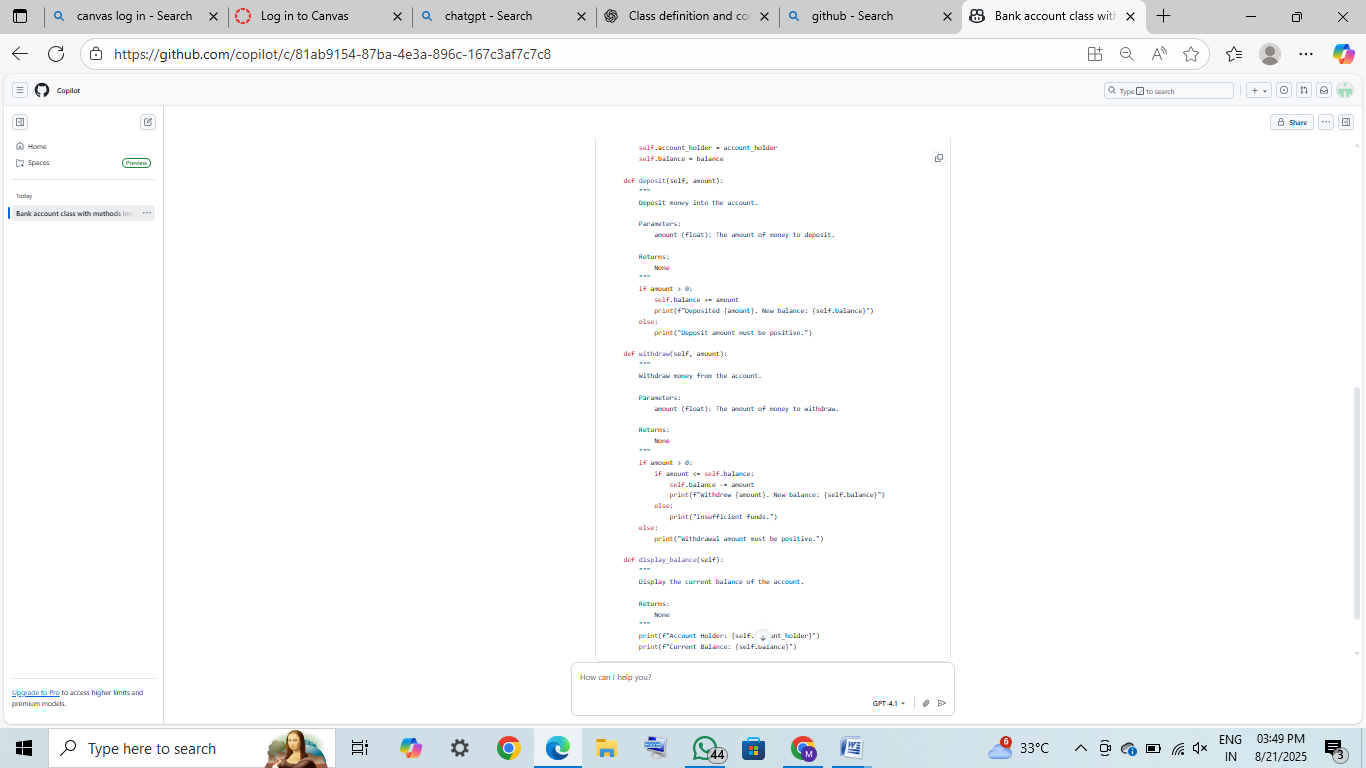
The display\_balance method prints the current balance.

At the bottom, we create an account for Anjali with ₹1000.

We deposit ₹500, withdraw ₹200, then display the final balance (₹1300).





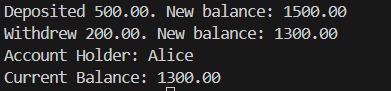


OUTPUT:

Deposited 500. New balance: 1500

Withdrew 200. New balance: 1300

Account Holder: Alice



TASK 2:

CODE WRITTEN:

# Iterate over each item in the list

my\_list = [1, 2, 3, 4, 5, 6]

for item in my\_list:

AUTO –COMPLETED CODE BY GITHUB COPILOT:

# Iterate over each item in the list and sum all even numbers

my\_list = [1, 2, 3, 4, 5, 6]

total = 0

for item in my\_list:

if item % 2 == 0:

total += item

print("Sum of even numbers:", total)

EXPLANATION:

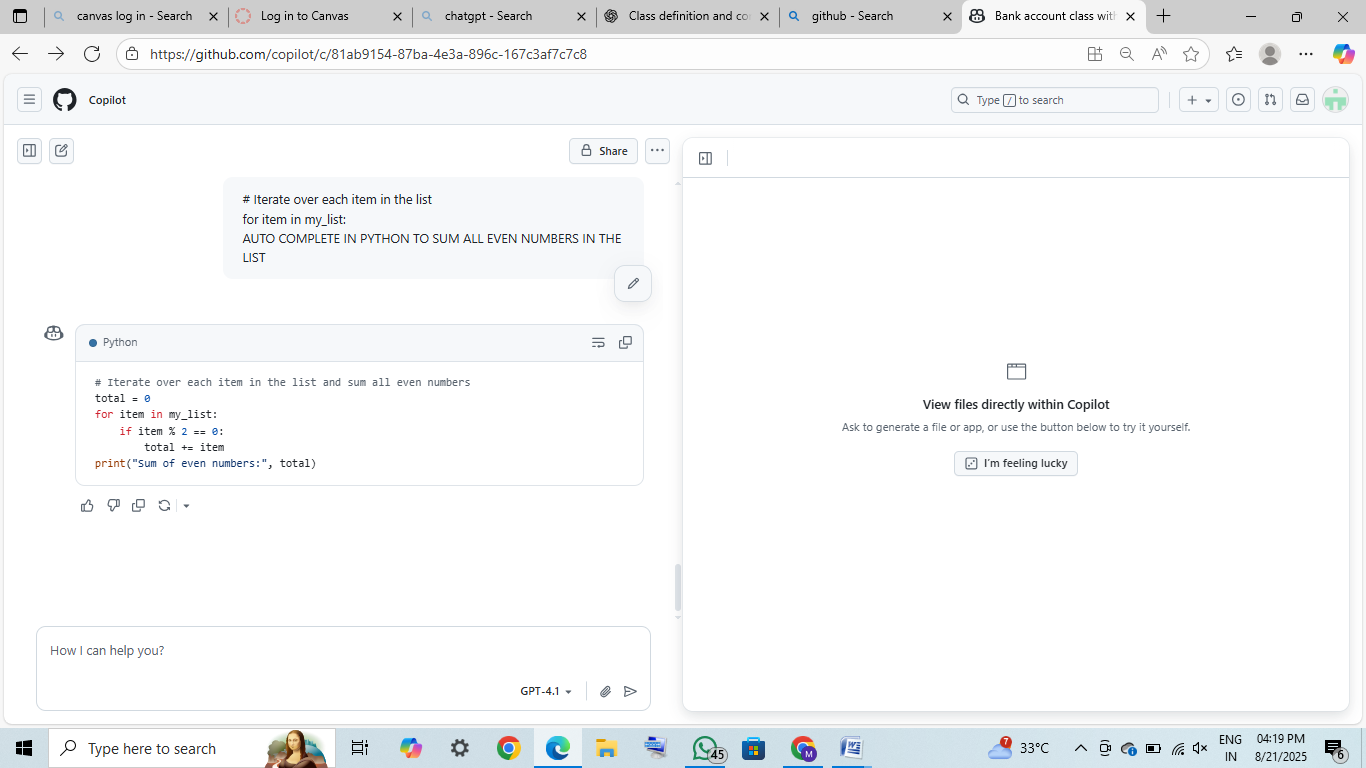
A variable even\_sum is set to 0 to store the total.

A loop goes through each number in the list.

If a number is divisible by 2 (num % 2 == 0), it is even.

Even numbers are added to even\_sum.

After the loop ends, the program prints the sum of all even numbers.



OUTPUT:

WhatsApp Image 2025-08-21 at 16.15.48_68fbdfef.jpg

TASK 3:

CODE WRITTEN:

def categorize\_age(age):

"""

Categorize a person based on their age.

Parameters:

age (int): The age of the person.

Returns:

str: The category ('child', 'teenager', 'adult', or 'senior').

"""

AUTO-COMPLETED CODE BY GITHUB COPILOT:

def categorize\_age(age):

"""

Categorize a person based on their age.

Parameters:

age (int): The age of the person.

Returns:

str: The category ('child', 'teenager', 'adult', or 'senior').

"""

if age < 13:

return "child"

elif age < 20:

return "teenager"

elif age < 65:

return "adult"

else:

return "senior"

# ...existing code...

print(categorize\_age(25)) # This will print: adult

EPLANATION:

The function check\_age(age) takes an age as input.

if age < 13: → returns "Child" if the age is less than 13.

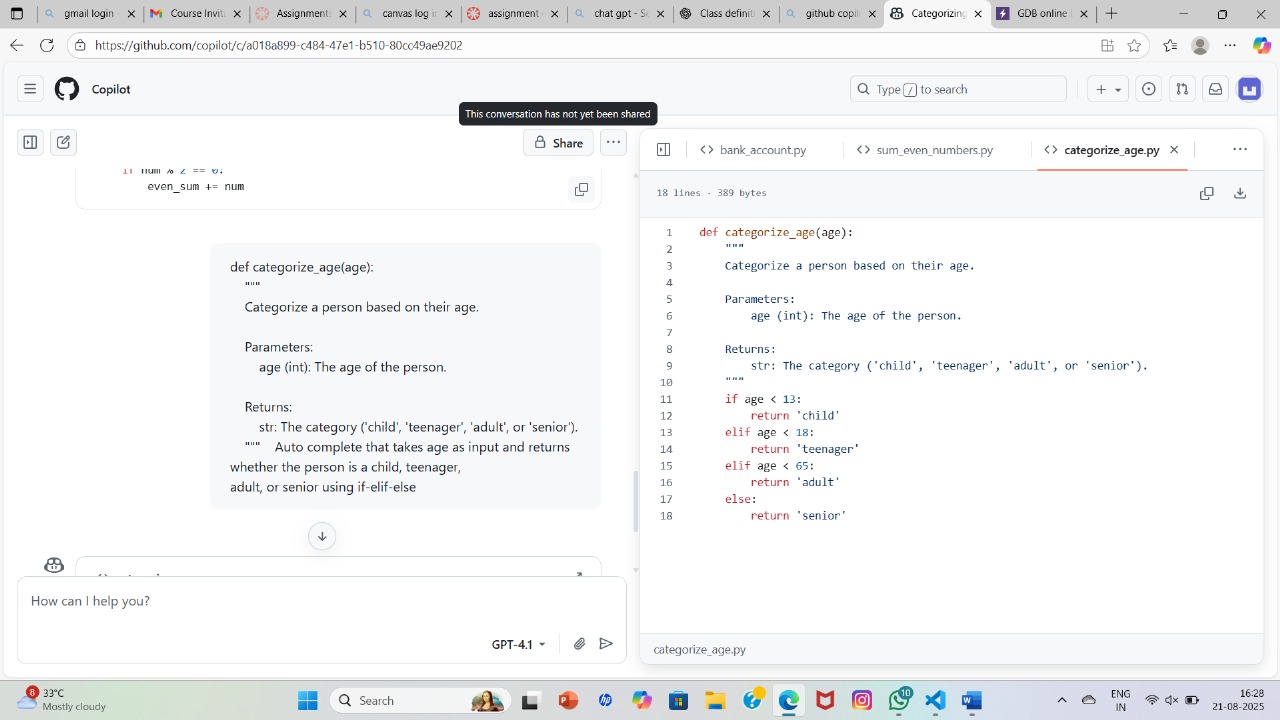
elif age < 20: → returns "Teenager" if the age is between 13 and 19.

elif age < 60: → returns "Adult" if the age is between 20 and 59.

else: → returns "Senior" for ages 60 and above.

The function always returns one of the four categories.

Example: check\_age(16) will return "Teenager".



OUTPUT:

WhatsApp Image 2025-08-21 at 16.30.32_9fb1aea4.jpg

TASK 4:

CODE WRITTEN:

# Program to reverse the digits of a number entered by the user

# take input from the user

num = int(input("Enter a number: "))

reverse = 0 # to store reversed number

# while loop to reverse digits

while num > 0:

AUTO –COMPLETED CODE BY GITHUB COPILOT:

# Program to reverse the digits of a number entered by the user

# take input from the user

num = int(input("Enter a number: "))

reverse = 0 # to store reversed number

# while loop to reverse digits

while num > 0:

digit = num % 10 # get the last digit

reverse = reverse \* 10 + digit # append digit to reversed number

num = num // 10 # remove the last digit

print("Reversed number:", reverse)

EXPLNATION:

A variable num stores the original number (e.g., 12345).

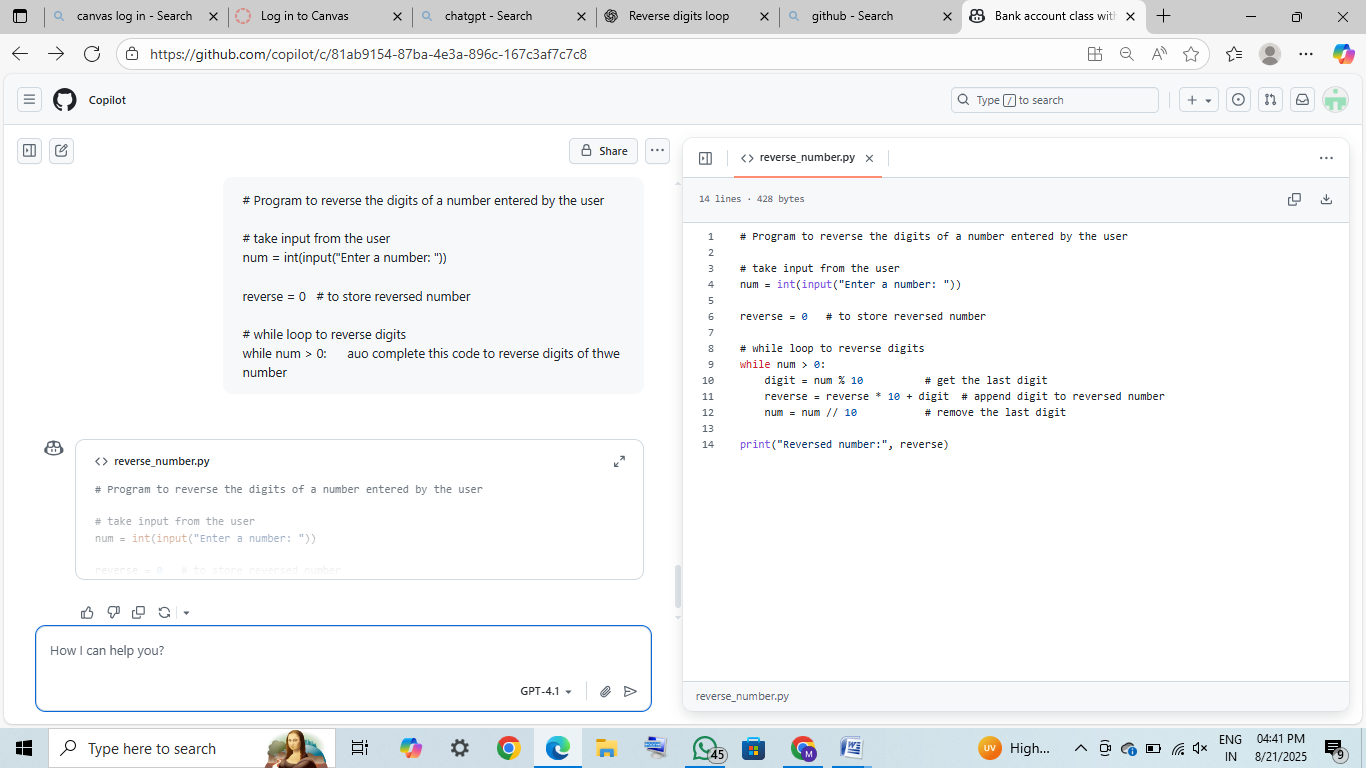
Another variable reverse is set to 0 to build the reversed number.

The while num > 0: loop runs until all digits are processed.

digit = num % 10 → extracts the last digit of num.

reverse = reverse \* 10 + digit → adds the digit to the reversed number.

num //= 10 → removes the last digit from num.



OUTPUT:

WhatsApp Image 2025-08-21 at 16.48.04_1f5a3bd3.jpg

TASK 5:

CODE WRITTEN:

class Employee:

"""

A class to represent an employee.

Attributes:

name (str): The name of the employee.

salary (float): The salary of the employee.

"""

def \_init\_(self, name, salary):

self.name = name

self.salary = salary

class Manager(Employee):

"""

A class to represent a manager, which is a type of employee.

Attributes:

name (str): Inherited from Employee.

salary (float): Inherited from Employee.

department (str): The department the manager oversees.

"""

def \_init\_(self, name, salary, department):

super().\_init\_(name, salary)

self.department = department

AUTO-COMPLETED CODE BY GITHUB COPILOT:

class Employee:

"""

A class to represent an employee.

Attributes:

name (str): The name of the employee.

salary (float): The salary of the employee.

"""

def \_init\_(self, name, salary):

self.name = name

self.salary = salary

def display\_info(self):

"""

Display information about the employee.

Returns:

None

"""

print(f"Name: {self.name}")

print(f"Salary: {self.salary:.2f}")

class Manager(Employee):

"""

A class to represent a manager, which is a type of employee.

Attributes:

name (str): Inherited from Employee.

salary (float): Inherited from Employee.

department (str): The department the manager oversees.

"""

def \_init\_(self, name, salary, department):

super().\_init\_(name, salary)

self.department = department

def display\_info(self):

"""

Display information about the manager, including department.

Returns:

None

"""

super().display\_info()

print(f"Department: {self.department}")

# ...existing code...

if \_name\_ == "\_main\_":

emp = Employee("John", 50000)

emp.display\_info()

mgr = Manager("Alice", 80000, "HR")

mgr.display\_info()

EXPLANTION:

The class BankAccount represents a simple bank account with two attributes: account\_holder and balance.

The constructor \_init\_ initializes these attributes when an object is created.

The deposit method increases the balance if the amount is positive, otherwise shows an error.

The withdraw method decreases the balance if there are enough funds, else prints “Insufficient funds.”

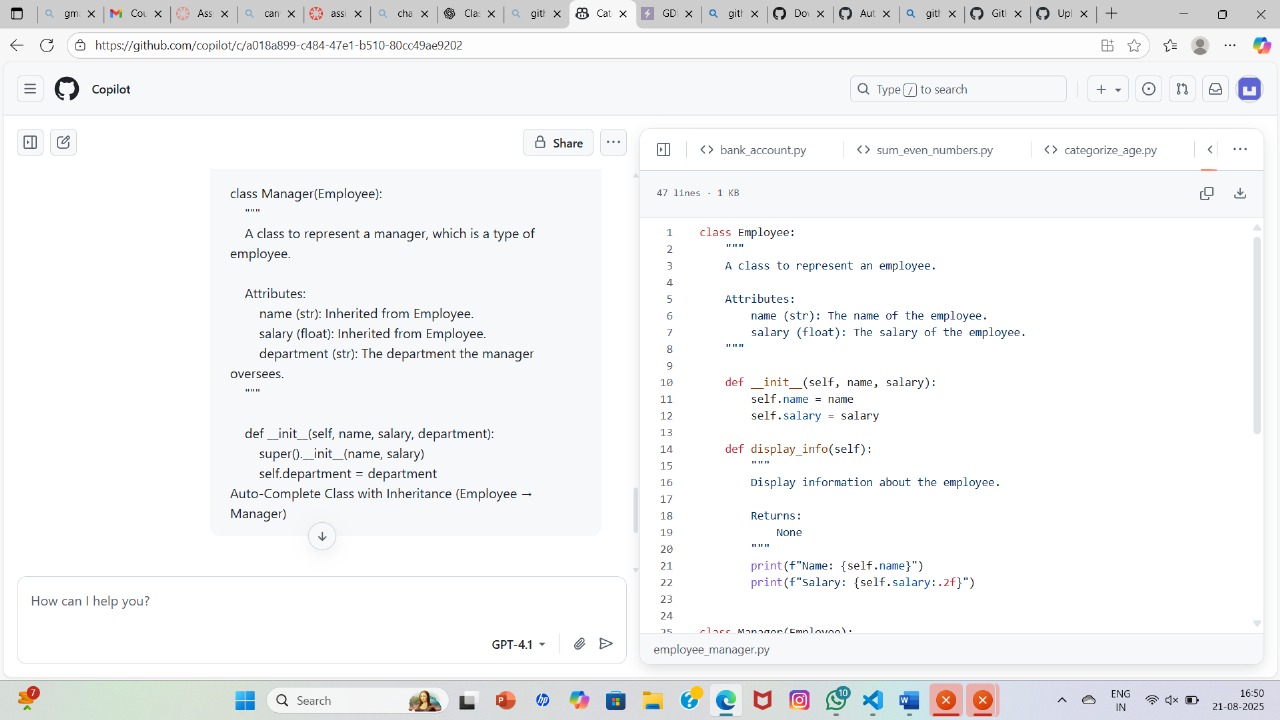
The display\_balance method prints the account holder’s name and the current balance.

In the main block, an account for Alice is created with ₹1000.

A deposit of ₹500 updates the balance to ₹1500.

A withdrawal of ₹200 updates the balance to ₹1300.

Finally, the balance is displayed showing Alice’s name and ₹1300.



OUTPUT:

