SR UNIVERSITY

AI ASSIST CODING

Lab-4.4

ROLL NO:2503A51L13

NAME: B.Hasini

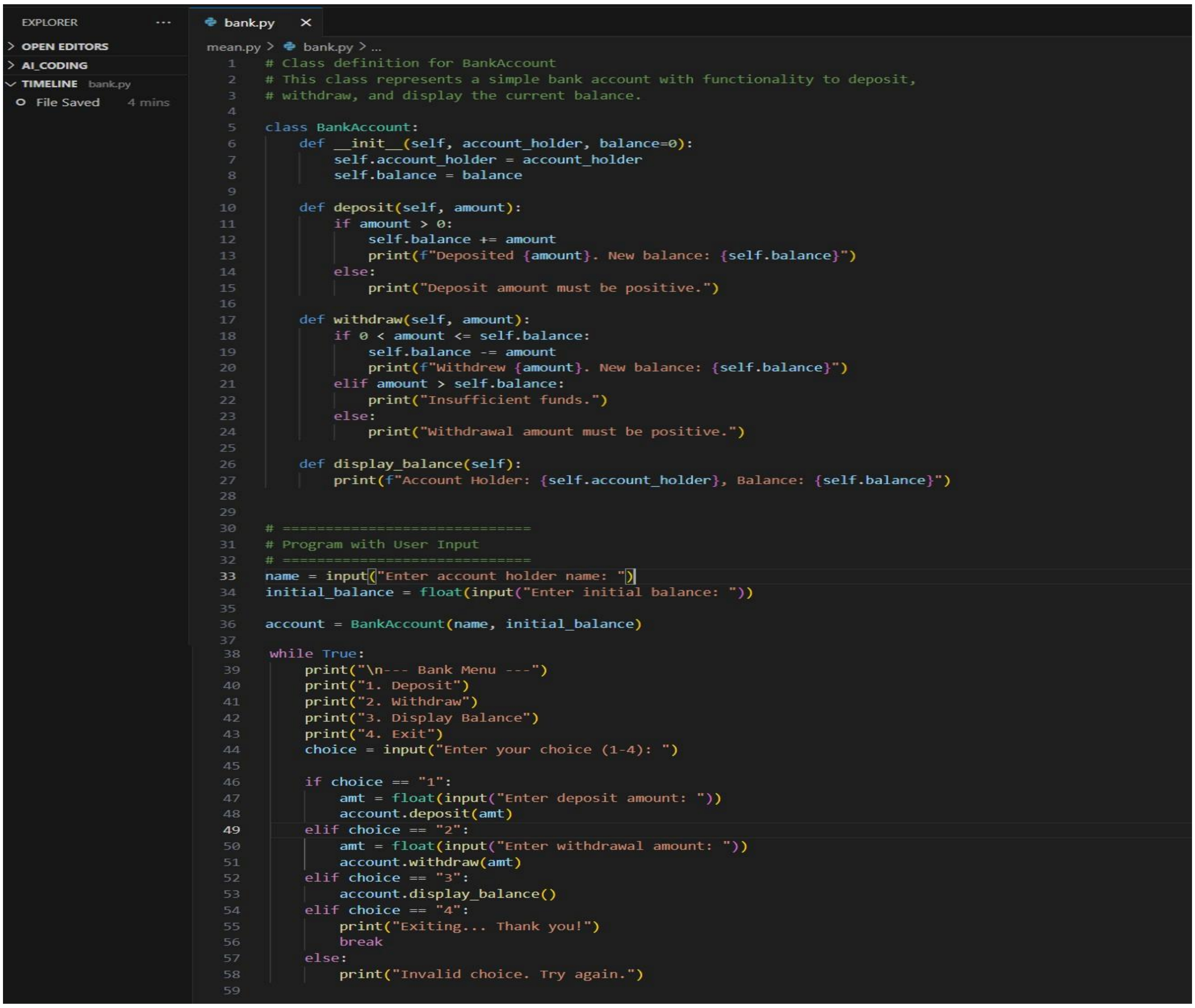
BATCH:19

TASK #1:

Prompt Used:

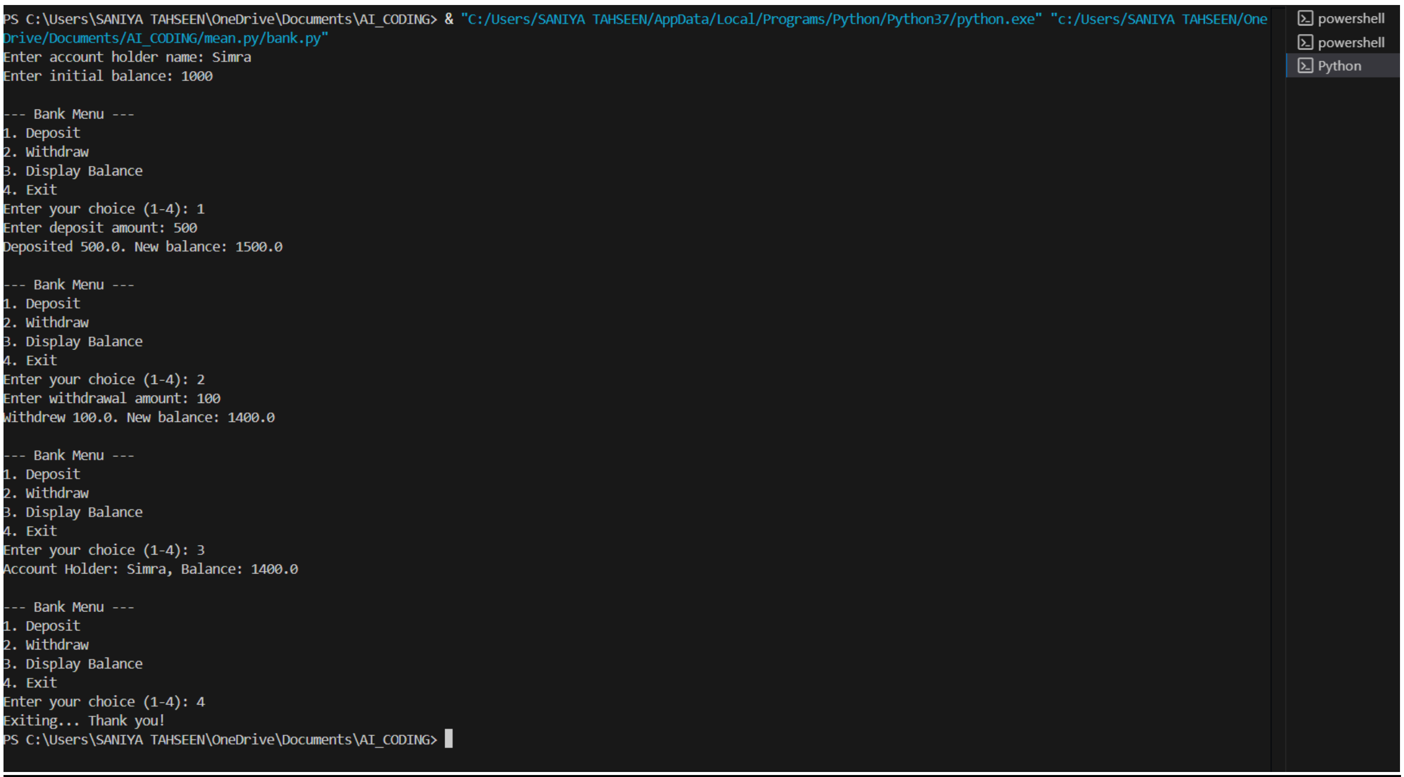
Write a class definition comment and start the constructor for a class called BankAccount with account_holder and balance attributes. Use GitHub Copilot to auto-complete the rest of the class, including methods to deposit, withdraw, and display balance.

Code Generated:



```python
# Class definition for BankAccount
# This class represents a simple bank account with functionality to deposit,
# withdraw, and display the current balance.

class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        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):
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew {amount}. New balance: {self.balance}")
        elif amount > self.balance:
            print("Insufficient funds.")
        else:
            print("Withdrawal amount must be positive.")

    def display_balance(self):
        print(f"Account Holder: {self.account_holder}, Balance: {self.balance}")


# ==============================
# Program with User Input
# ==============================
name = input("Enter account holder name: ")
initial_balance = float(input("Enter initial balance: "))

account = BankAccount(name, initial_balance)

while True:
    print("\n--- Bank Menu ---")
    print("1. Deposit")
    print("2. Withdraw")
    print("3. Display Balance")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")

    if choice == "1":
        amt = float(input("Enter deposit amount: "))
        account.deposit(amt)
    elif choice == "2":
        amt = float(input("Enter withdrawal amount: "))
        account.withdraw(amt)
    elif choice == "3":
        account.display_balance()
    elif choice == "4":
        print("Exiting... Thank you!")
        break
    else:
        print("Invalid choice. Try again.")
```

Output After executing Code:



```
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> & "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/mean.py/bank.py"
Enter account holder name: Simra
Enter initial balance: 1000

--- Bank Menu ---
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Enter your choice (1-4): 1
Enter deposit amount: 500
Deposited 500.0. New balance: 1500.0

--- Bank Menu ---
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Enter your choice (1-4): 2
Enter withdrawal amount: 100
Withdrew 100.0. New balance: 1400.0

--- Bank Menu ---
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Enter your choice (1-4): 3
Account Holder: Simra, Balance: 1400.0

--- Bank Menu ---
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Enter your choice (1-4): 4
Exiting... Thank you!
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING>
```

Observations:

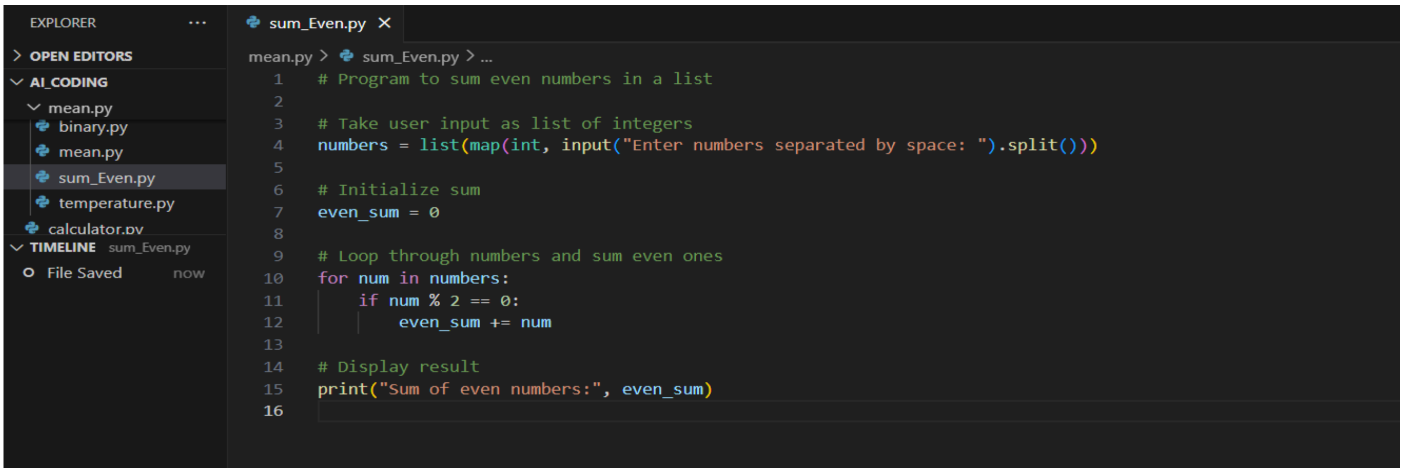
- The BankAccount class is created with account_holder and balance attributes, initialized via the constructor.
- It includes methods to deposit, withdraw, and display balance, each with proper validation.
- A menu-driven loop allows users to perform transactions interactively using input options.

TASK #2:

Prompt Used:

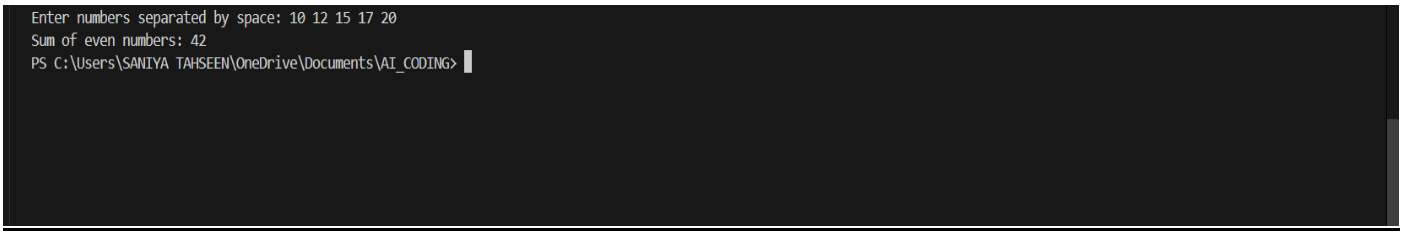
Write a comment and the initial line of a loop to iterate over a list. Allow GitHub Copilot to complete the logic to sum all even numbers in the list.

Code Generated:



```python
# Program to sum even numbers in a list

# Take user input as list of integers
numbers = list(map(int, input("Enter numbers separated by space: ").split()))

# Initialize sum
even_sum = 0

# Loop through numbers and sum even ones
for num in numbers:
    if num % 2 == 0:
        even_sum += num

# Display result
print("Sum of even numbers:", even_sum)
```

Output After executing Code:



```
Enter numbers separated by space: 10 12 15 17 20
Sum of even numbers: 42
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING>
```

Observations:

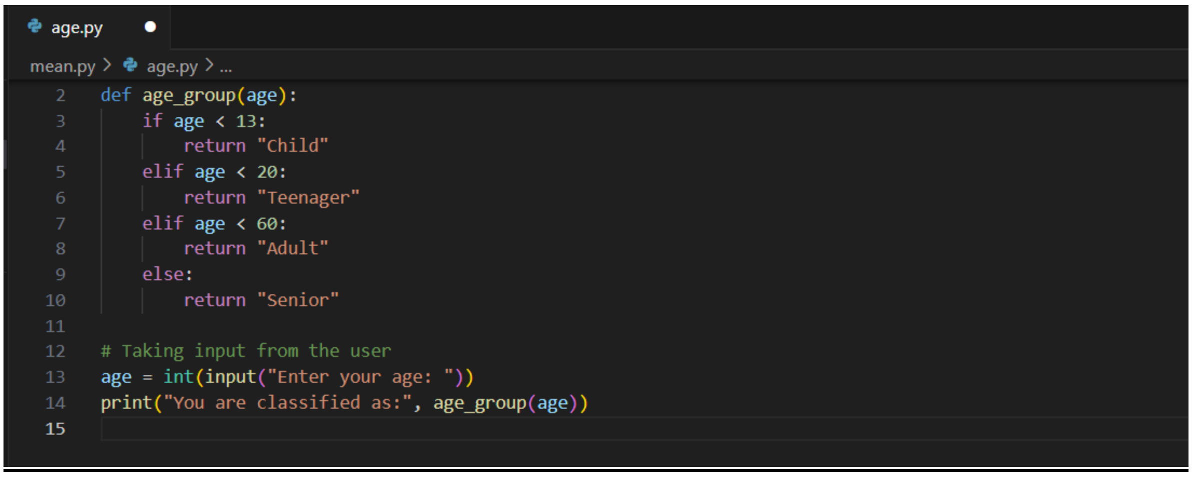
- The program accepts a list of integers from the user using input().split() and converts them with map(int, ...).
- An accumulator variable (even_sum) is initialized to 0 to store the running total of even numbers.
- Each number is checked with the condition num % 2 == 0 to determine if it is even.

TASK #3:

Prompt Used:

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.

Code Generated:

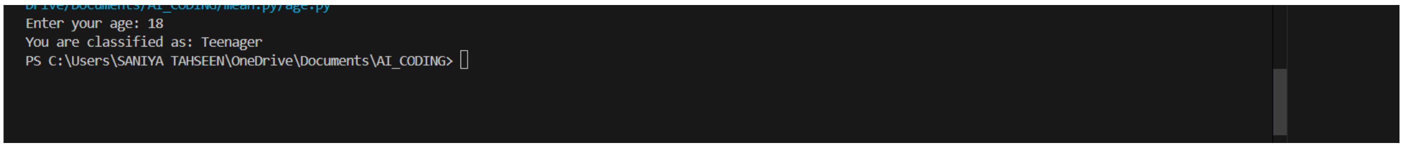


age.py

mean.py > age.py > ...

```python
def age_group(age):
    if age < 13:
        return "Child"
    elif age < 20:
        return "Teenager"
    elif age < 60:
        return "Adult"
    else:
        return "Senior"

# Taking input from the user
age = int(input("Enter your age: "))
print("You are classified as:", age_group(age))
```

Output After executing Code:



```
Enter your age: 18
You are classified as: Teenager
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING>
```

Observations:

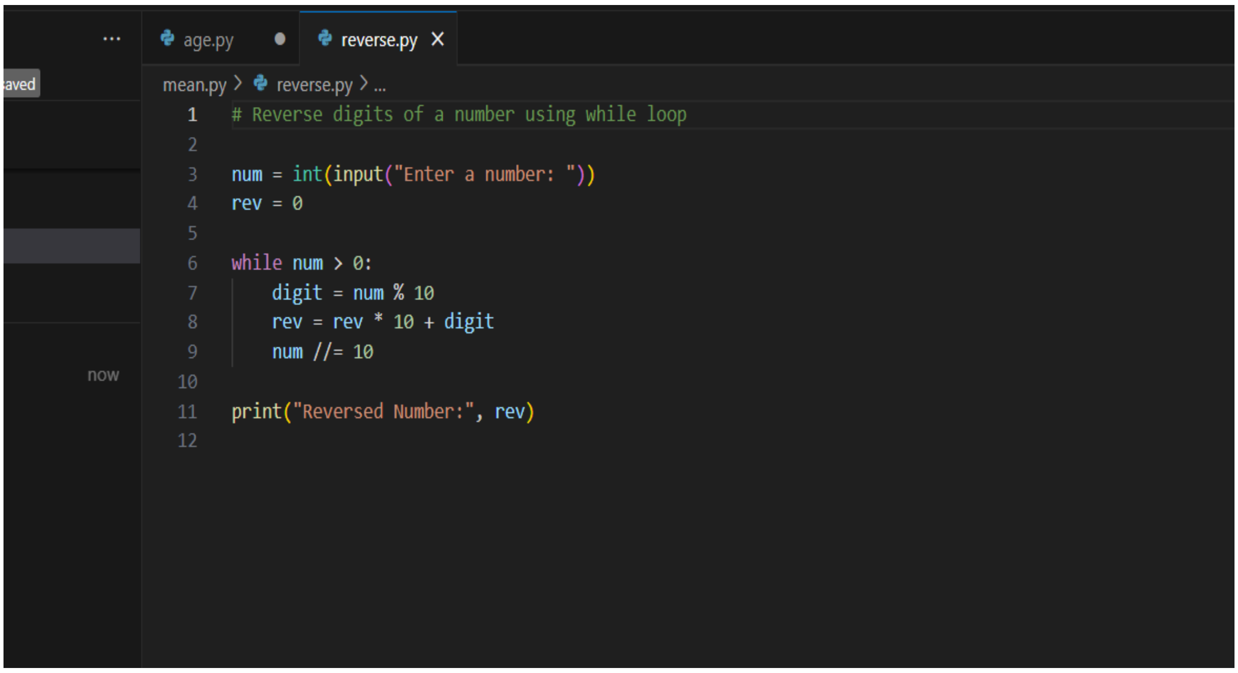
- The function age_group(age) uses if–elif–else to categorize the given age.
- Age ranges are checked in order: <13 → Child, <20 → Teenager, <60 → Adult, else → Senior.
- The result is returned as a string.

TASK #4:

Prompt Used:

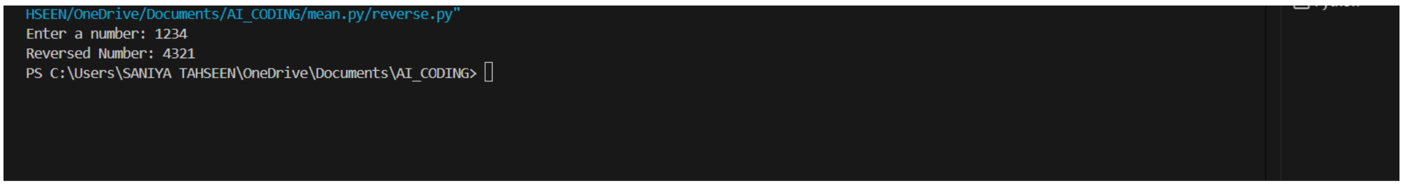
Write a comment and start a while loop to reverse the digits of a number. Let Copilot complete the loop logic

Code Generated:



```python
# Reverse digits of a number using while loop

num = int(input("Enter a number: "))
rev = 0

while num > 0:
    digit = num % 10
    rev = rev * 10 + digit
    num //= 10

print("Reversed Number:", rev)

```

Output After executing Code:



```
HSEEN/OneDrive/Documents/AI_CODING/mean.py/reverse.py"
Enter a number: 1234
Reversed Number: 4321
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING>
```

Observations:

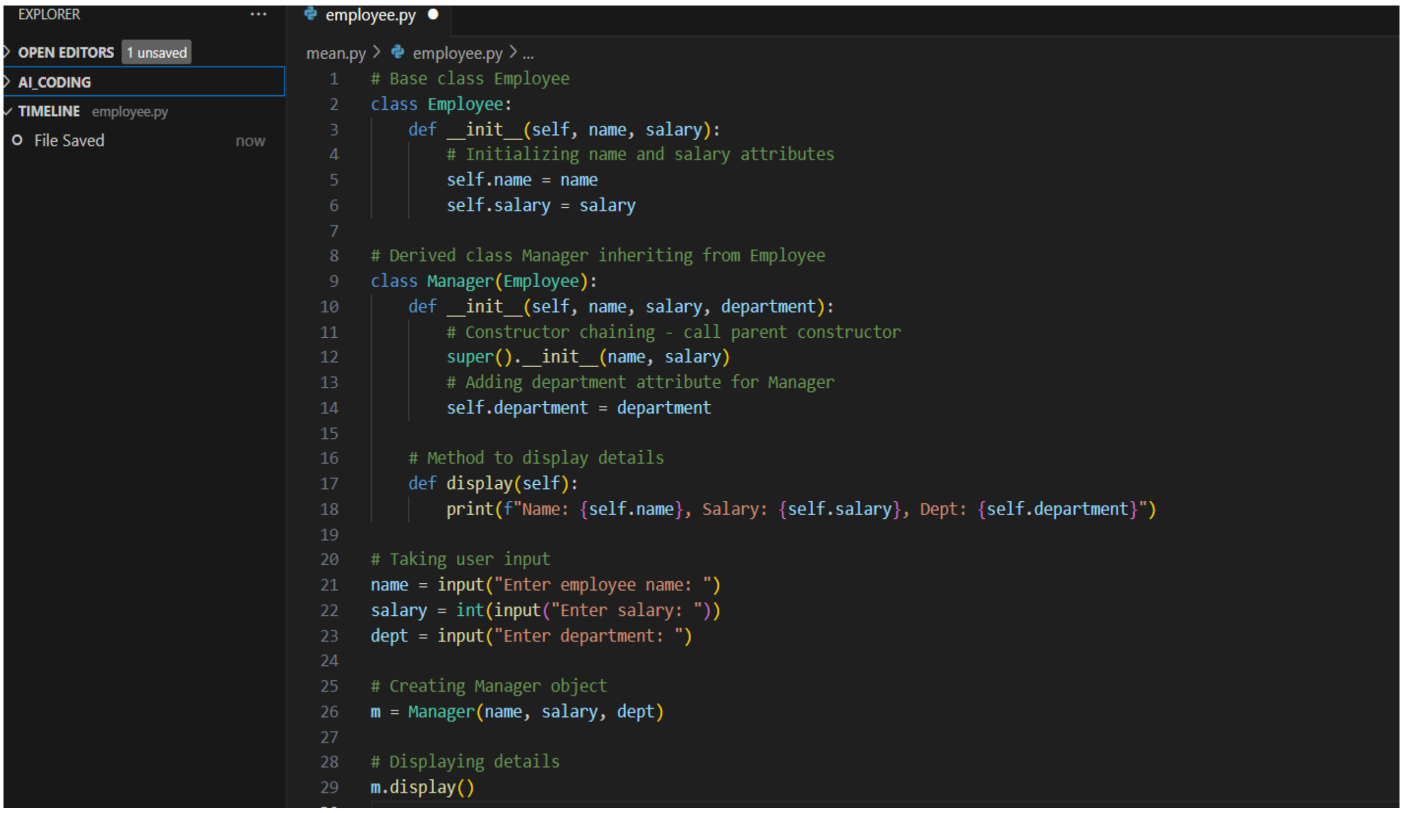
- The last digit is obtained using num % 10 and added to the reversed number.
- The reversed number is built step by step using rev = rev * 10 + digit.
- After each step, the last digit is removed from the original number using num //= 10.

TASK #5:

Prompt Used:

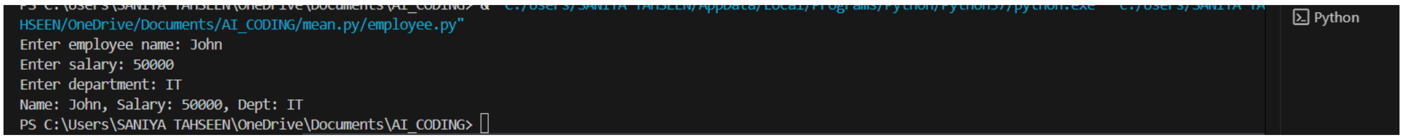
Begin a class Employee with attributes name and salary. Then, start a derived class Manager that inherits from Employee and adds a department. Let GitHub Copilot complete the methods and constructor chaining.

Code Generated:



```python
# Base class Employee
class Employee:
    def __init__(self, name, salary):
        # Initializing name and salary attributes
        self.name = name
        self.salary = salary

# Derived class Manager inheriting from Employee
class Manager(Employee):
    def __init__(self, name, salary, department):
        # Constructor chaining - call parent constructor
        super().__init__(name, salary)
        # Adding department attribute for Manager
        self.department = department

    # Method to display details
    def display(self):
        print(f"Name: {self.name}, Salary: {self.salary}, Dept: {self.department}")

# Taking user input
name = input("Enter employee name: ")
salary = int(input("Enter salary: "))
dept = input("Enter department: ")

# Creating Manager object
m = Manager(name, salary, dept)

# Displaying details
m.display()
```

Output After executing Code:



```
HSEEN/OneDrive/Documents/AI_CODING/mean.py/employee.py"
Enter employee name: John
Enter salary: 50000
Enter department: IT
Name: John, Salary: 50000, Dept: IT
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> 
```

Observations:

- The Manager class inherits attributes from the Employee class and adds a new attribute department.
- Constructor chaining is achieved using super(), which allows reuse of the parent class constructor.
- Input is taken from the user to dynamically create an object with given values.