

# AI ASSISTED CODING LAB

## ASSIGNMENT 4.4

NAME : Guangsinlung Phaomei

ENROLLMENT NO :2503A51L20

BATCH NO : 19

### TASK DESCRIPTION 1: Auto-Complete a Python Class for Bank

Account

- Write a class definition comment and start the constructor for a class called Bank Account 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.

**PROMPT 1:** generate a Python class called Bank Account. Start with a class definition comment and a constructor that takes `account_holder` and `balance` as attributes. Then use GitHub Copilot to auto-complete the rest of the class with methods for depositing money, withdrawing money (with balance check), and displaying the current balance.

The screenshot shows the VS Code editor with a file named `asn51.py` open. The code defines a `BankAccount` class with methods `__init__`, `deposit`, `withdraw`, and `display_balance`. The `__init__` method takes `account_holder` and `balance` as arguments. The `deposit` method checks if the amount is positive and updates the balance. The `withdraw` method checks if the account has sufficient funds and updates the balance. The `display_balance` method prints the account holder's name and the current balance. The `__main__` block prompts the user to enter the account holder's name.

The chat window on the right shows a prompt: "rewrite it so that it can take input". The chat response shows the `BankAccount` class definition with the `display_balance` method updated to take input from the user.

The screenshot shows the VS Code editor with the `asn51.py` file. The code is now complete, including a `choice` input and a loop that calls the `deposit`, `withdraw`, and `display_balance` methods based on the user's choice. The `__main__` block prompts the user to enter the account holder's name and the initial balance.

The terminal window at the bottom shows the program's output. It prompts the user to enter the account holder's name (king) and the initial balance (2000). It then prompts the user to choose an option (1-4). The user enters 1, and the program prompts for the deposit amount (500). The output shows "Deposited \$500.00. New balance: \$2500.00".

## TASK DESCRIPTION 2: Auto-Complete a For Loop to Sum Even

Numbers in a List

- 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.

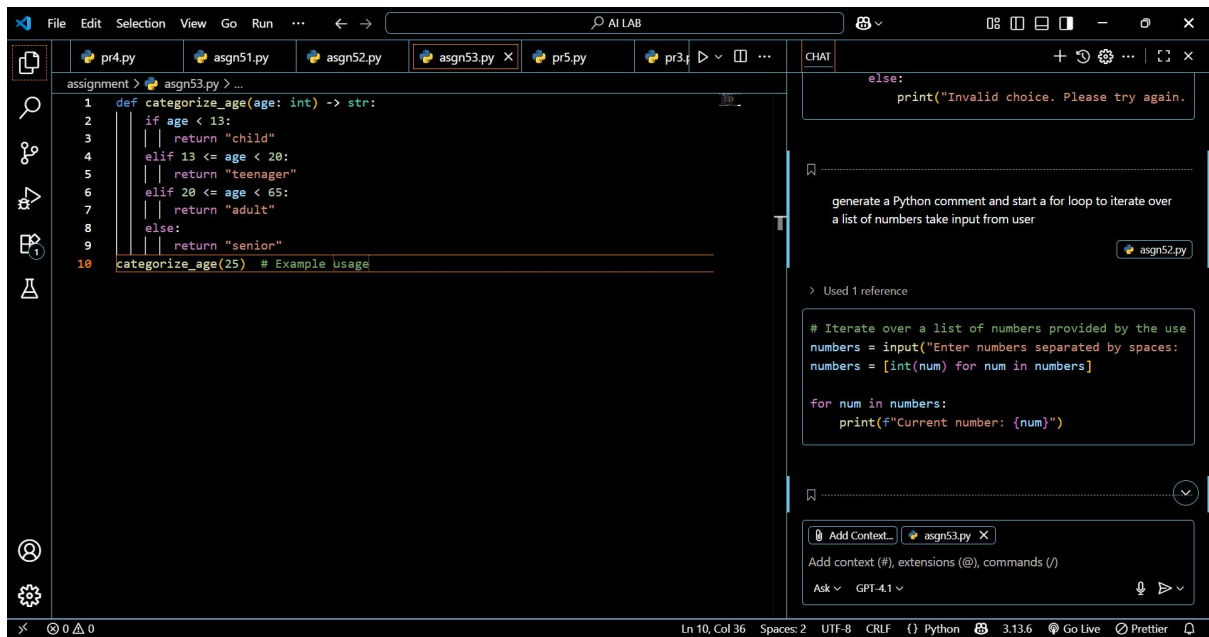
**PROMPT 1:** generate a Python comment and start a for loop to iterate over a list of numbers. Then use GitHub Copilot to auto-complete the loop so that it sums all the even numbers in the list.



## TASK DESCRIPTION 3: Auto-Complete Conditional Logic to Check Age Group

- 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 1:** generate a Python function that takes age as input. Start the function definition and an if-elif-else structure, then use GitHub Copilot to auto-complete the logic so it returns whether the person is a child, teenager, adult, or senior."



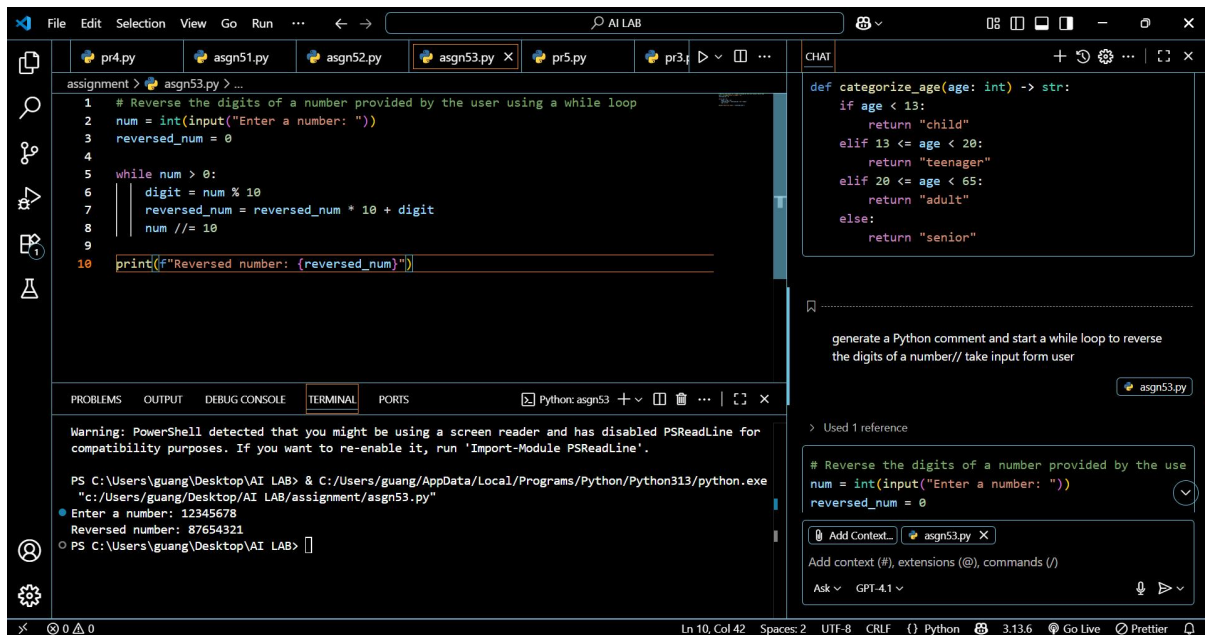
## TASK DESCRIPTION 4: Auto-Complete a While Loop to Reverse

Digits of a Number

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

PROMPT 1: generate a Python comment and start a while loop to reverse the digits of a number. Then use GitHub Copilot to auto-complete the loop logic.

PROMPT 1: generate a Python comment and start a while loop to reverse the digits of a number. Then use GitHub Copilot to auto-complete the loop logic.

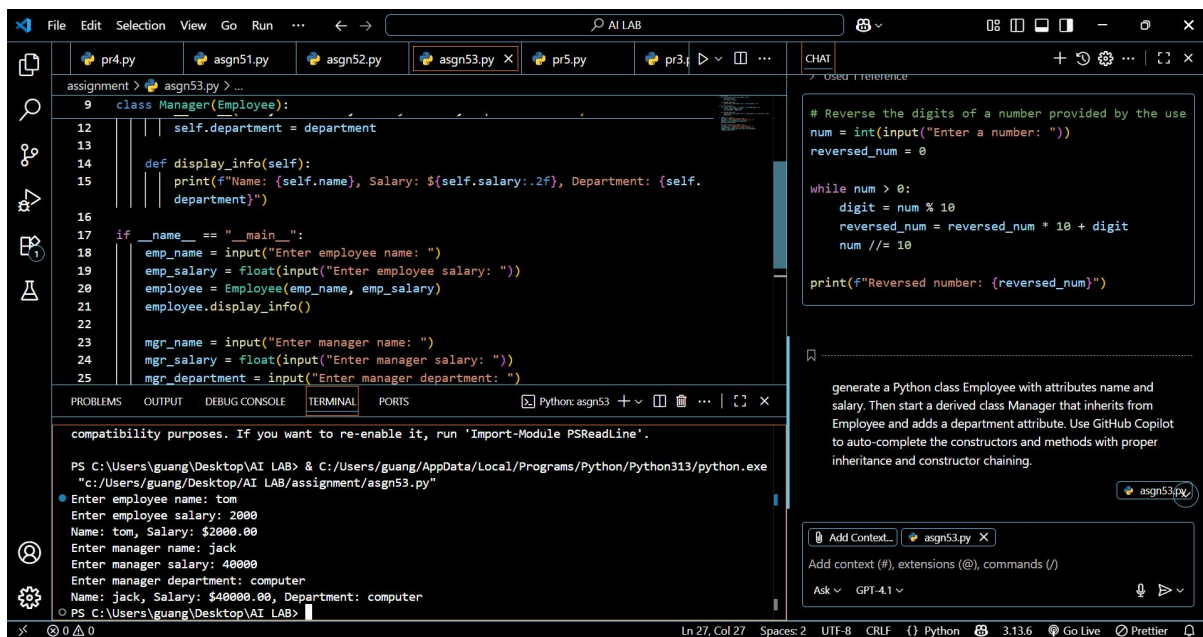


## TASK DESCRIPTION 5: Auto-Complete Class with Inheritance

(Employee → Manager)

- 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 1:** generate a Python class `Employee` with attributes `name` and `salary`. Then start a derived class `Manager` that inherits from `Employee` and adds a `department` attribute. Use GitHub Copilot to auto-complete the constructors and methods with proper inheritance and constructor chaining.



**OBSERVATION:** In this assignment, GitHub Copilot was effectively utilized to auto-completely different Python coding tasks. Each task was structured with a clear description and a starting prompt, allowing Copilot to generate the remaining logic. The tasks covered fundamental programming concepts such as class creation, loops, conditional statements, and inheritance. Specifically:

1. **Bank Account Class** – Demonstrated object-oriented programming with constructor, methods for deposit, withdraw, and balance display.
2. **For Loop for Even Numbers** – Showed how iteration and conditional checks can be automated for summing even numbers.
3. **Conditional Age Group Check** – Applied if-elif-else structure to classify age groups efficiently.
4. **While Loop for Reversing Digits** – Practiced loop control to reverse digits in a number.

5. **Class Inheritance (Employee → Manager)** – Illustrated constructor chaining and method overriding in object-oriented programming.