Assignment-6.3

H.NO:2303A54060

Batch:47_B

1. Scenario

You are developing a simple student information management module.

Task

• Use an AI tool (GitHub Copilot / Cursor AI / Gemini) to complete a Student class.

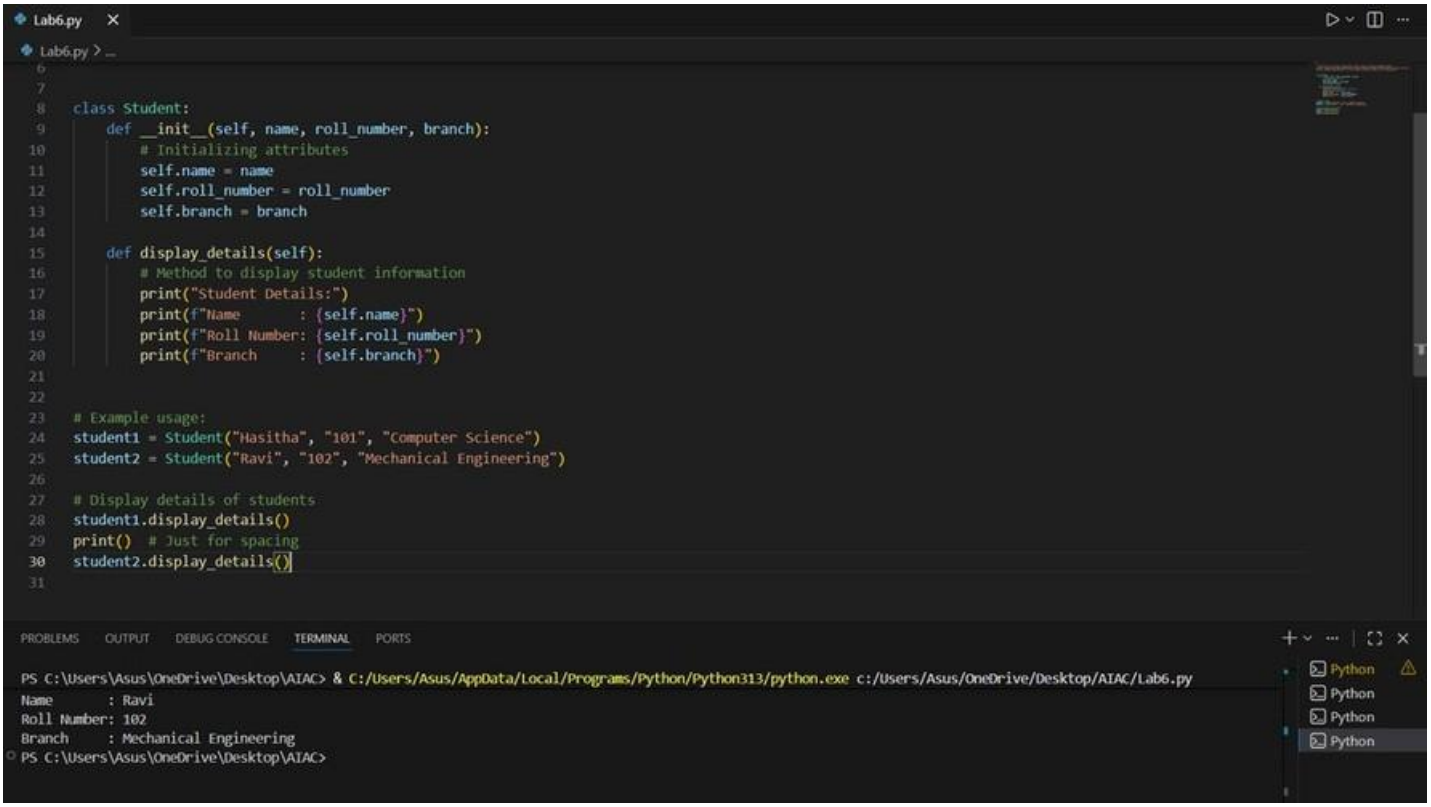
• The class should include attributes such as name, roll number, and branch.

• Add a method display_details() to print student information.

• Execute the code and verify the output.

• Analyze the code generated by the AI tool for correctness and clarity.

Code & Output:



Prompt: You are in charge of developing a simple student information management

module.create a
student class while making sure the class includes attributes such as name,roll number and branch.Include a method display_details() to print student information.create all this using python

Code-Explanation: the student class consists of attributes such as Name,Roll

number,Branch.Multiple objects
such as student1 and student2 are created and their details are accessed outside the class using display_details() method.

2. Scenario

You are writing a utility function to display multiples of a given number.

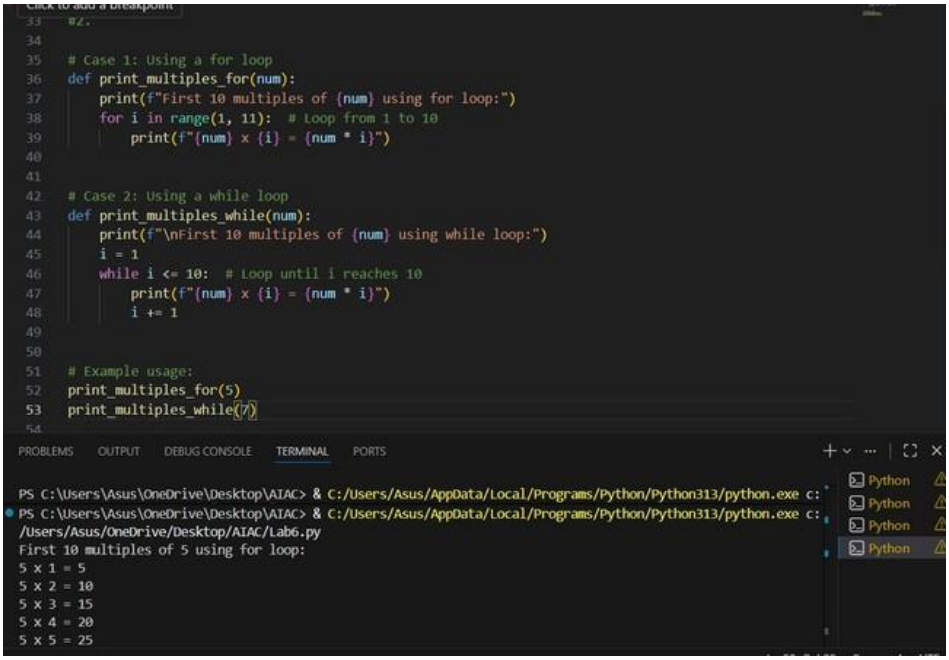
Task

• Prompt the AI tool to generate a function that prints the first 10 multiples of a given number
using a loop.

• Analyze the generated loop logic.

• Ask the AI to generate the same functionality using another controlled looping structure (e.g.,
while instead of for).

code & Output:



Prompt: you are writing a utility function to display multiples of a given number.make sure to print the
first 10 multiples of a given number using a loop.print 2 cases where in one case you use a for loop and in one case you use a while loop to execute it **Code Explanation:** Here**,**For loop version: Iterates directly over numbers 1–10 using range(1, 11).Where as While
loop version Starts with i = 1 and increments until i <= 10..Both prints the same values.

3. Scenario

You are building a basic classification system based on age.

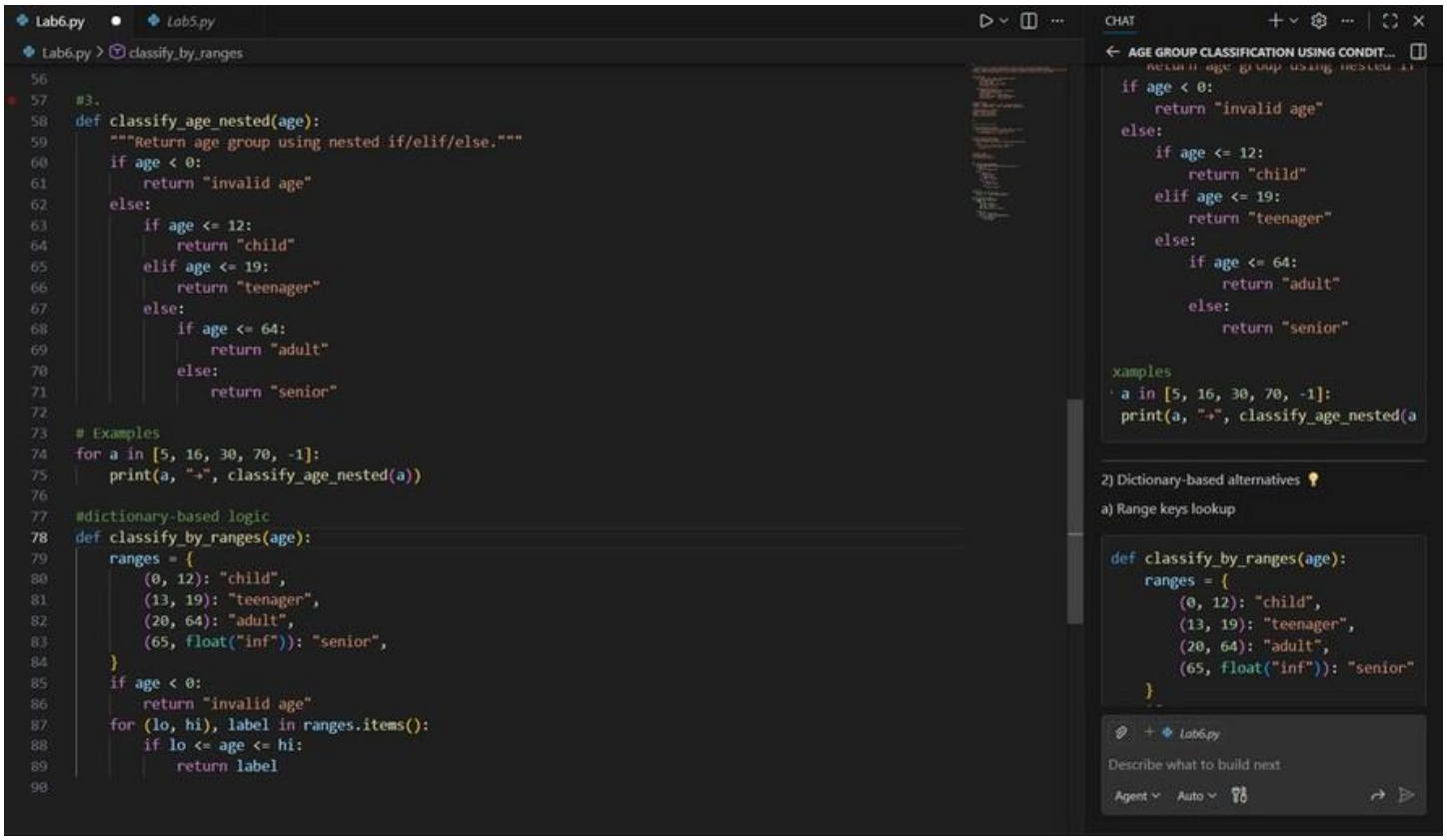
Task

• Ask the AI tool to generate nested if-elif-else conditional statements to classify age groups (e.g., child, teenager, adult, senior).

• Analyze the generated conditions and logic.

• Ask the AI to generate the same classification using alternative conditional structures (e.g., simplified conditions or dictionary-based logic).

code & output:



```
PS C:\Users\Asus\OneDrive\Desktop\AIAC> & C:/Users/Asus/AppData/Local/Programs/Python/Python313/python.exe c:/Users
/Asus/OneDrive/Desktop/AIAC/Lab6.py
5 → child
16 → teenager
30 → adult
70 → senior
-1 → invalid age
PS C:\Users\Asus\OneDrive\Desktop\AIAC>
```

Prompt:
imagine you are building a classification system based on age.use nested if-elif-else control statements to classify age groups(child,teenager,adult,senior).Also provide the same classification using alternative conditional structures such as dictionary-based logic.

code-Explanation

Nested if-else are simple and fast.you can use them when a problem demands fixed rules whereas dictionary based approach is more flexible and easier to chnage but it is slightly slower because it iterates and evaluates functions

4. Scenario

You need to calculate the sum of the first n natural numbers.

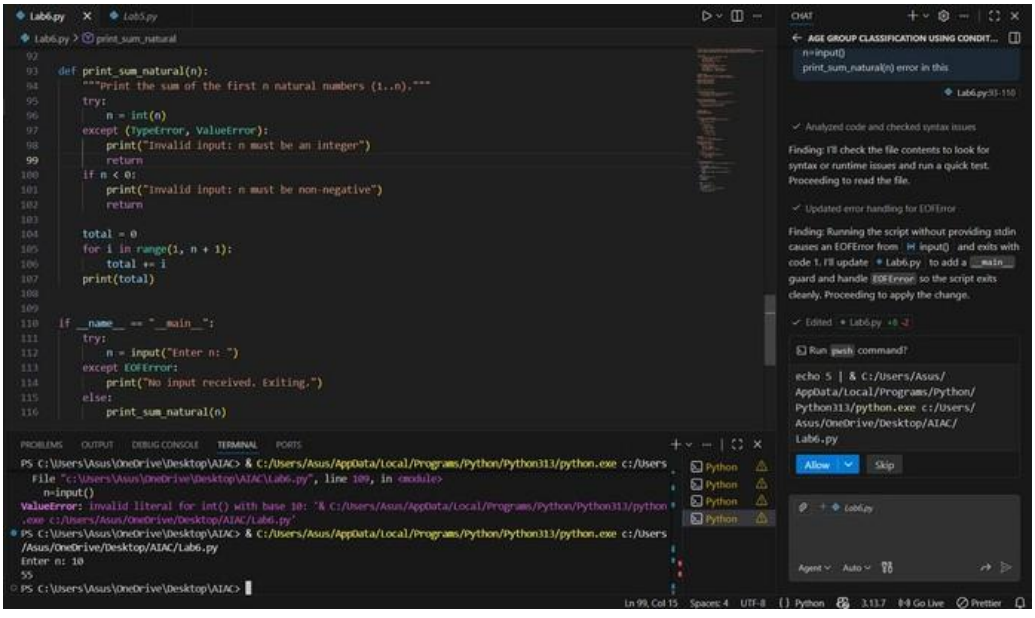
Task

• Use AI assistance to generate a sum_to_n() function using a for loop.

• Analyze the generated code.

• Ask the AI to suggest an alternative implementation using a while loop or a mathematical formula.

code & output:



Prompt:

you are building a function to print the sum of n natural numbers.use a for loop to get the output

5. Scenario

You are designing a basic banking application.

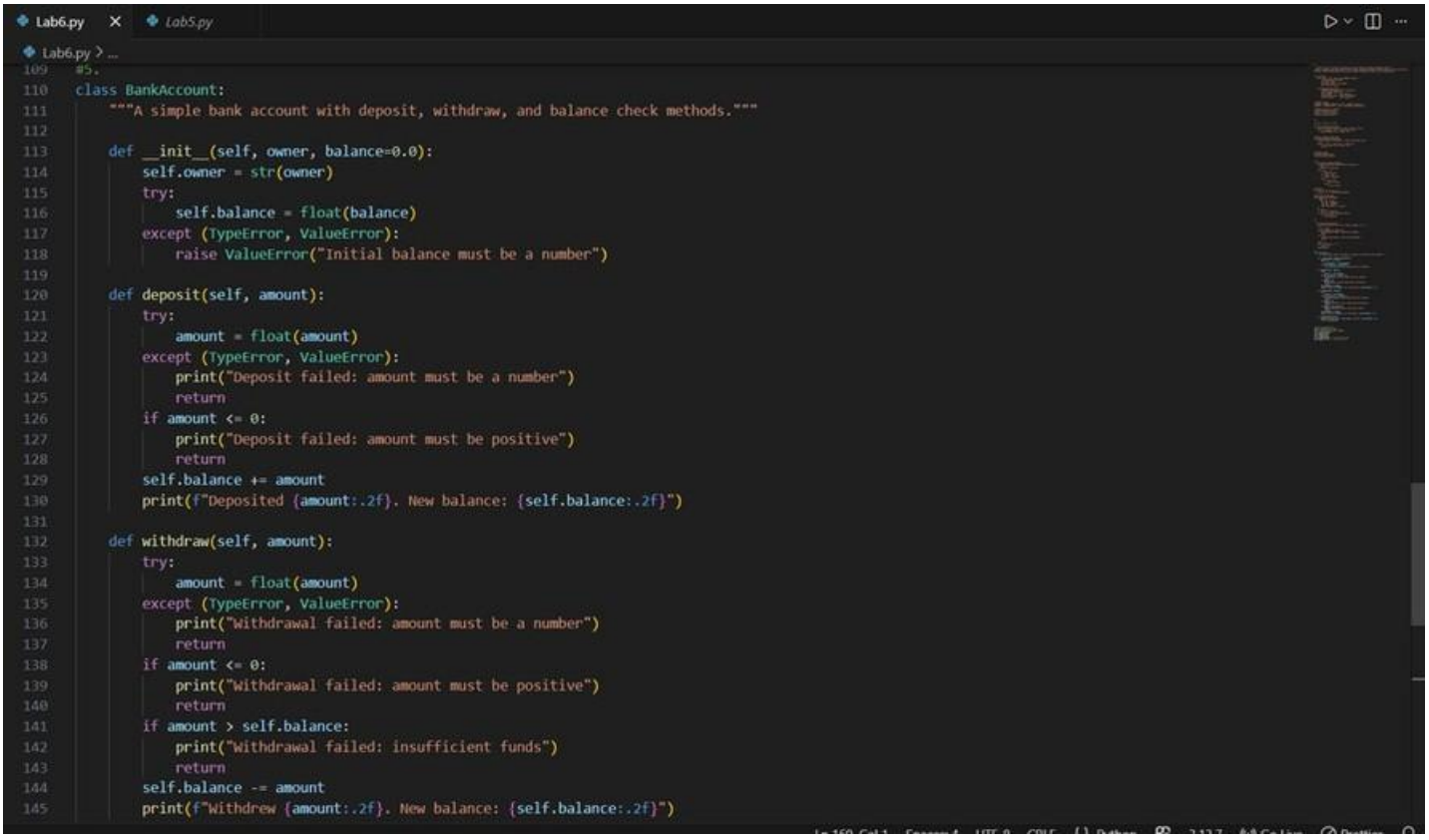
Task

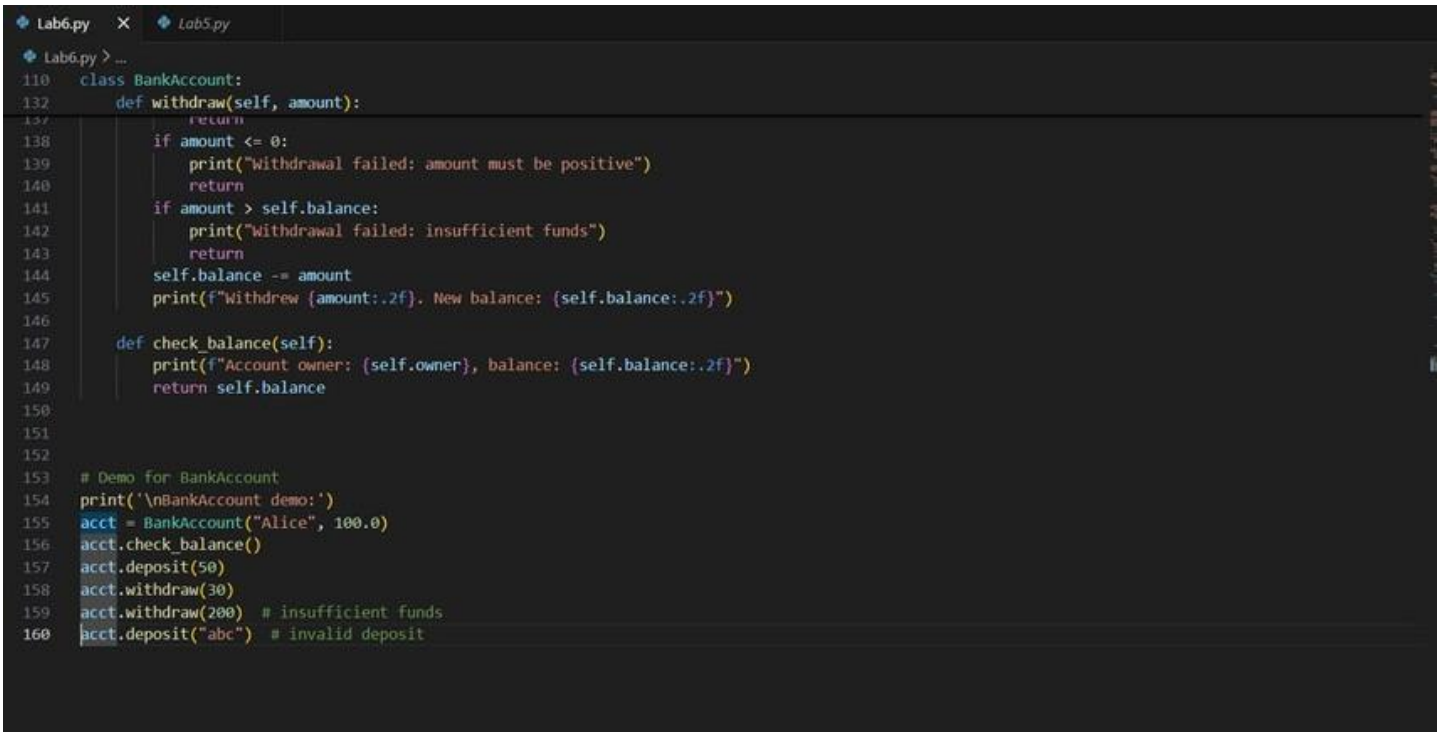
• Use AI tools to generate a Bank Account class with methods such as deposit(), withdraw(), and check_balance().

• Analyze the AI-generated class structure and logic.

• Add meaningful comments and explain the working of the code.

code & output:

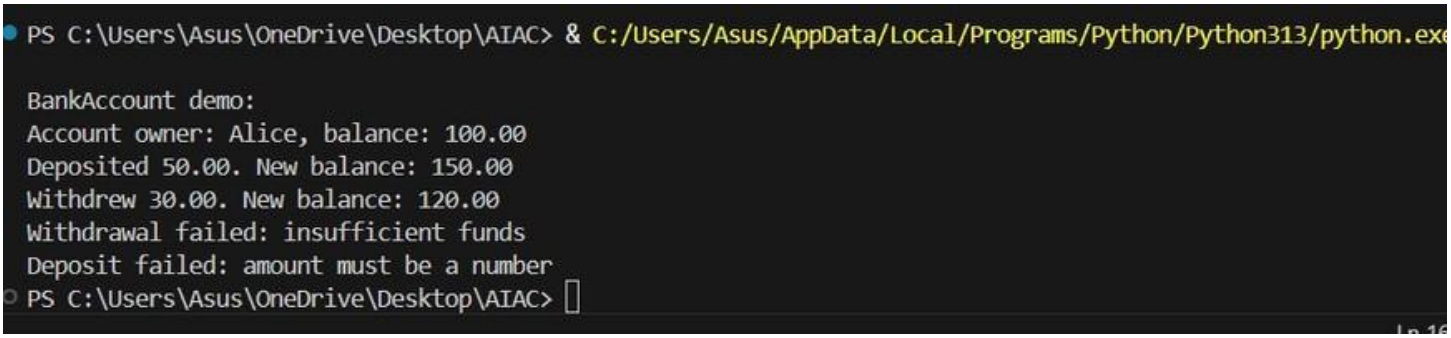




```python
class BankAccount:
    def withdraw(self, amount):
        if amount <= 0:
            print("Withdrawal failed: amount must be positive")
            return
        if amount > self.balance:
            print("Withdrawal failed: insufficient funds")
            return
        self.balance -= amount
        print(f"Withdrew {amount:.2f}. New balance: {self.balance:.2f}")

    def check_balance(self):
        print(f"Account owner: {self.owner}, balance: {self.balance:.2f}")
        return self.balance


# Demo for BankAccount
print('\nBankAccount demo:')
acct = BankAccount("Alice", 100.0)
acct.check_balance()
acct.deposit(50)
acct.withdraw(30)
acct.withdraw(200)  # insufficient funds
acct.deposit("abc")  # invalid deposit
```



```
PS C:\Users\Asus\OneDrive\Desktop\AIAC> & C:/Users/Asus/AppData/Local/Programs/Python/Python313/python.exe

BankAccount demo:
Account owner: Alice, balance: 100.00
Deposited 50.00. New balance: 150.00
Withdrew 30.00. New balance: 120.00
Withdrawal failed: insufficient funds
Deposit failed: amount must be a number
PS C:\Users\Asus\OneDrive\Desktop\AIAC>
```

Prompt:

you are designing a banking application.create a Bank Account class with methods such as deposit(),withdraw(),and check_balance().