AI Assisted Coding

Assignment - 6

K.VAMSHIDHAR|| 2303A510H7|| Batch:- 8

Task Description #1: Classes (Student Class)

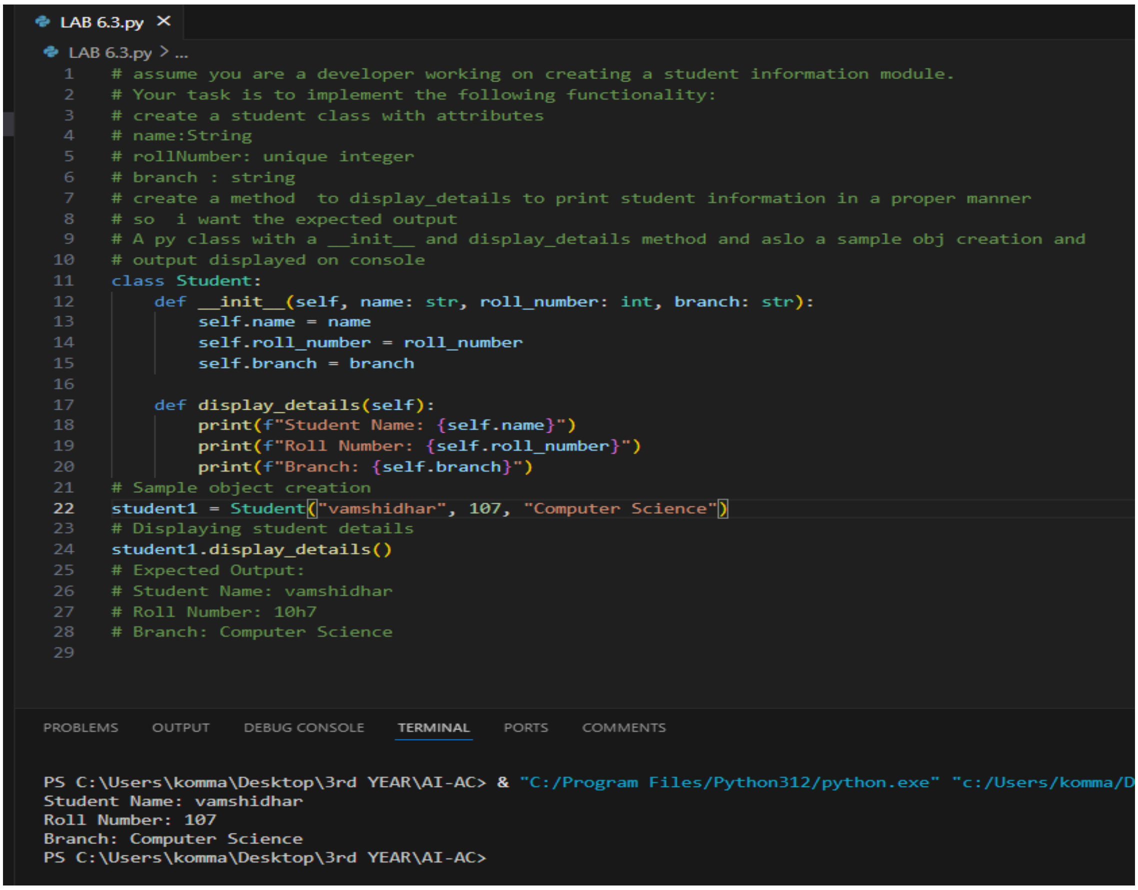
Scenario

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:



```python
# assume you are a developer working on creating a student information module.
# Your task is to implement the following functionality:
# create a student class with attributes
# name:String
# rollNumber: unique integer
# branch : string
# create a method  to display_details to print student information in a proper manner
# so  i want the expected output
# A py class with a __init__ and display_details method and aslo a sample obj creation and
# output displayed on console
class Student:
    def __init__(self, name: str, roll_number: int, branch: str):
        self.name = name
        self.roll_number = roll_number
        self.branch = branch

    def display_details(self):
        print(f"Student Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Branch: {self.branch}")
# Sample object creation
student1 = Student("vamshidhar", 107, "Computer Science")
# Displaying student details
student1.display_details()
# Expected Output:
# Student Name: vamshidhar
# Roll Number: 10h7
# Branch: Computer Science

```

```
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/D
Student Name: vamshidhar
Roll Number: 107
Branch: Computer Science
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>
```

Task Description #2: Loops (Multiples of a Number)

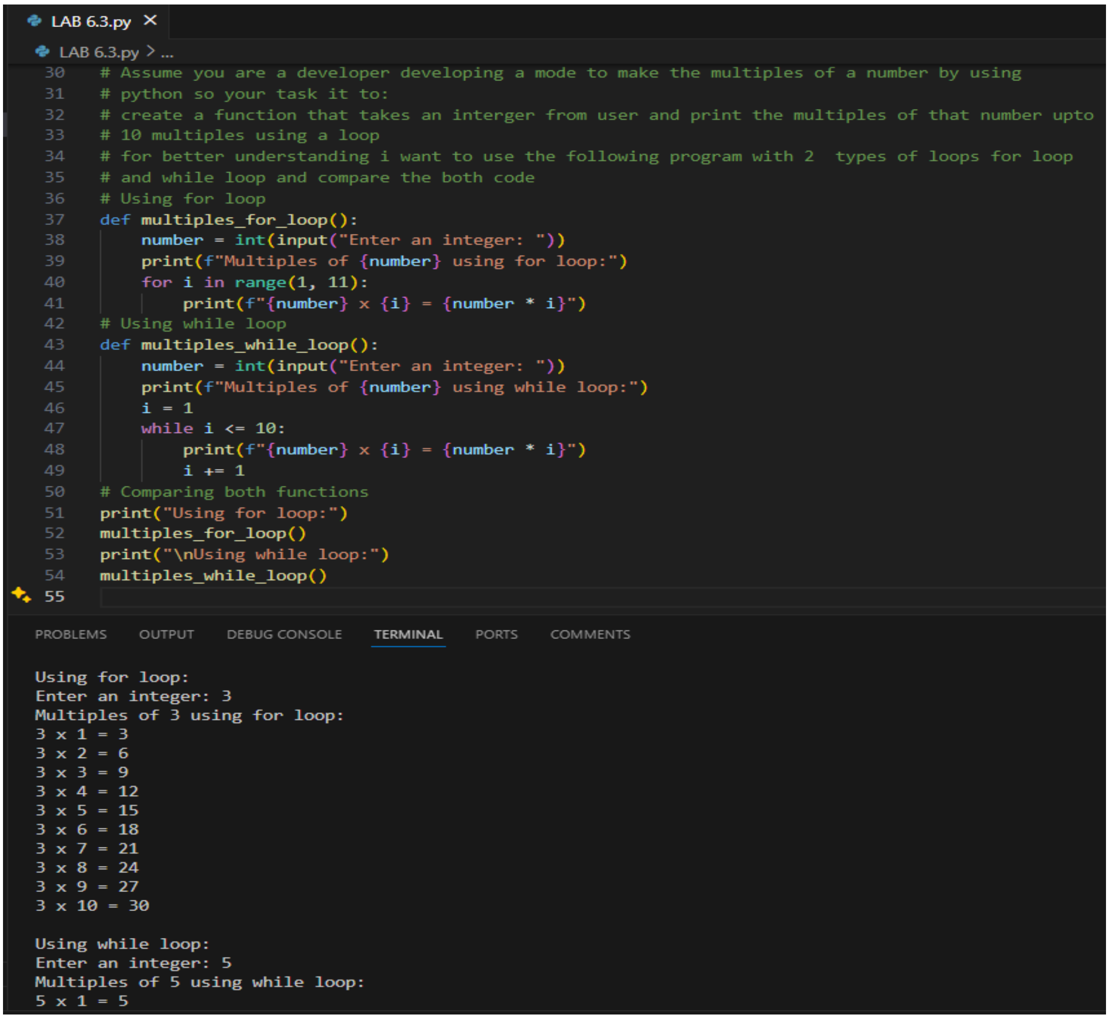
Scenario

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:



```python
# Assume you are a developer developing a mode to make the multiples of a number by using
# python so your task it to:
# create a function that takes an interger from user and print the multiples of that number upto
# 10 multiples using a loop
# for better understanding i want to use the following program with 2  types of loops for loop
# and while loop and compare the both code
# Using for loop
def multiples_for_loop():
    number = int(input("Enter an integer: "))
    print(f"Multiples of {number} using for loop:")
    for i in range(1, 11):
        print(f"{number} x {i} = {number * i}")
# Using while loop
def multiples_while_loop():
    number = int(input("Enter an integer: "))
    print(f"Multiples of {number} using while loop:")
    i = 1
    while i <= 10:
        print(f"{number} x {i} = {number * i}")
        i += 1
# Comparing both functions
print("Using for loop:")
multiples_for_loop()
print("\nUsing while loop:")
multiples_while_loop()

```

```
Using for loop:
Enter an integer: 3
Multiples of 3 using for loop:
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

Using while loop:
Enter an integer: 5
Multiples of 5 using while loop:
5 x 1 = 5
```

Task Description #3: Conditional Statements (Age Classification)

Scenario

You are building a basic classification system based on age.

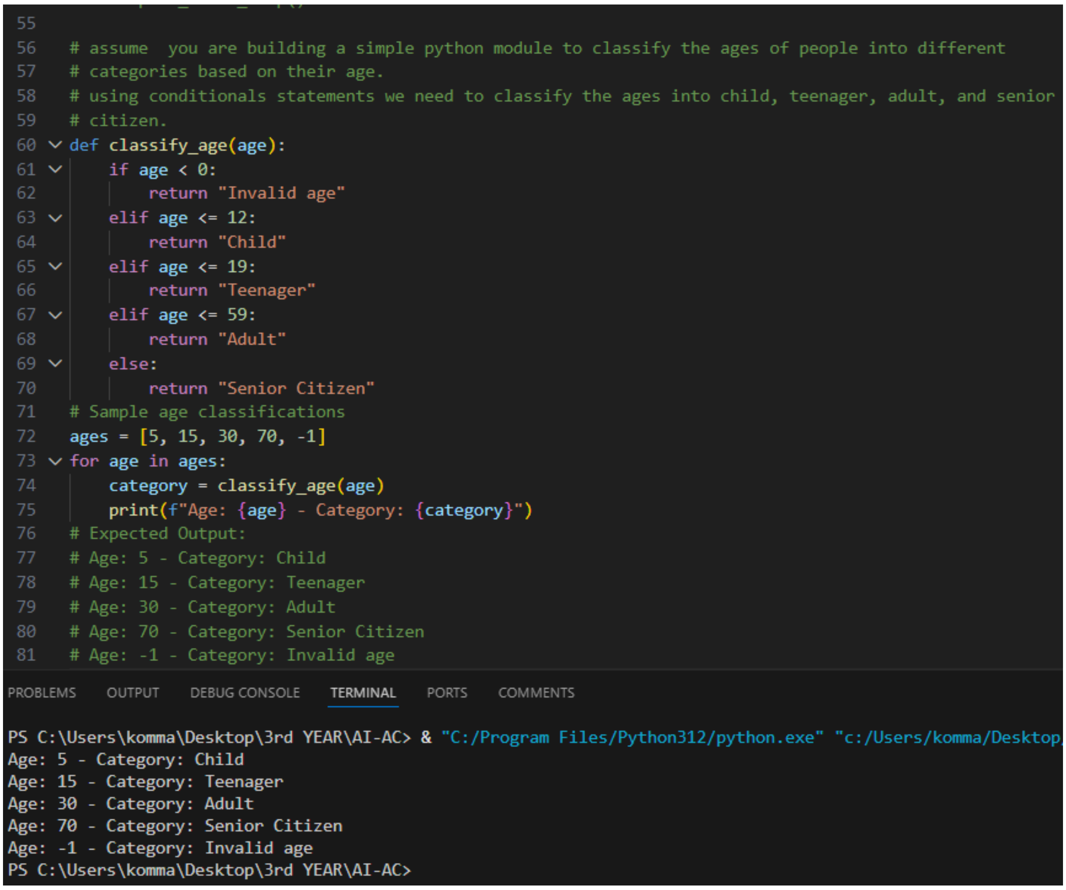
Task

- 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:



```python

# assume  you are building a simple python module to classify the ages of people into different
# categories based on their age.
# using conditionals statements we need to classify the ages into child, teenager, adult, and senior
# citizen.
def classify_age(age):
    if age < 0:
        return "Invalid age"
    elif age <= 12:
        return "Child"
    elif age <= 19:
        return "Teenager"
    elif age <= 59:
        return "Adult"
    else:
        return "Senior Citizen"
# Sample age classifications
ages = [5, 15, 30, 70, -1]
for age in ages:
    category = classify_age(age)
    print(f"Age: {age} - Category: {category}")
# Expected Output:
# Age: 5 - Category: Child
# Age: 15 - Category: Teenager
# Age: 30 - Category: Adult
# Age: 70 - Category: Senior Citizen
# Age: -1 - Category: Invalid age
```

```
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop
Age: 5 - Category: Child
Age: 15 - Category: Teenager
Age: 30 - Category: Adult
Age: 70 - Category: Senior Citizen
Age: -1 - Category: Invalid age
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>
```

Task Description #4: For and While Loops (Sum of First n Numbers)

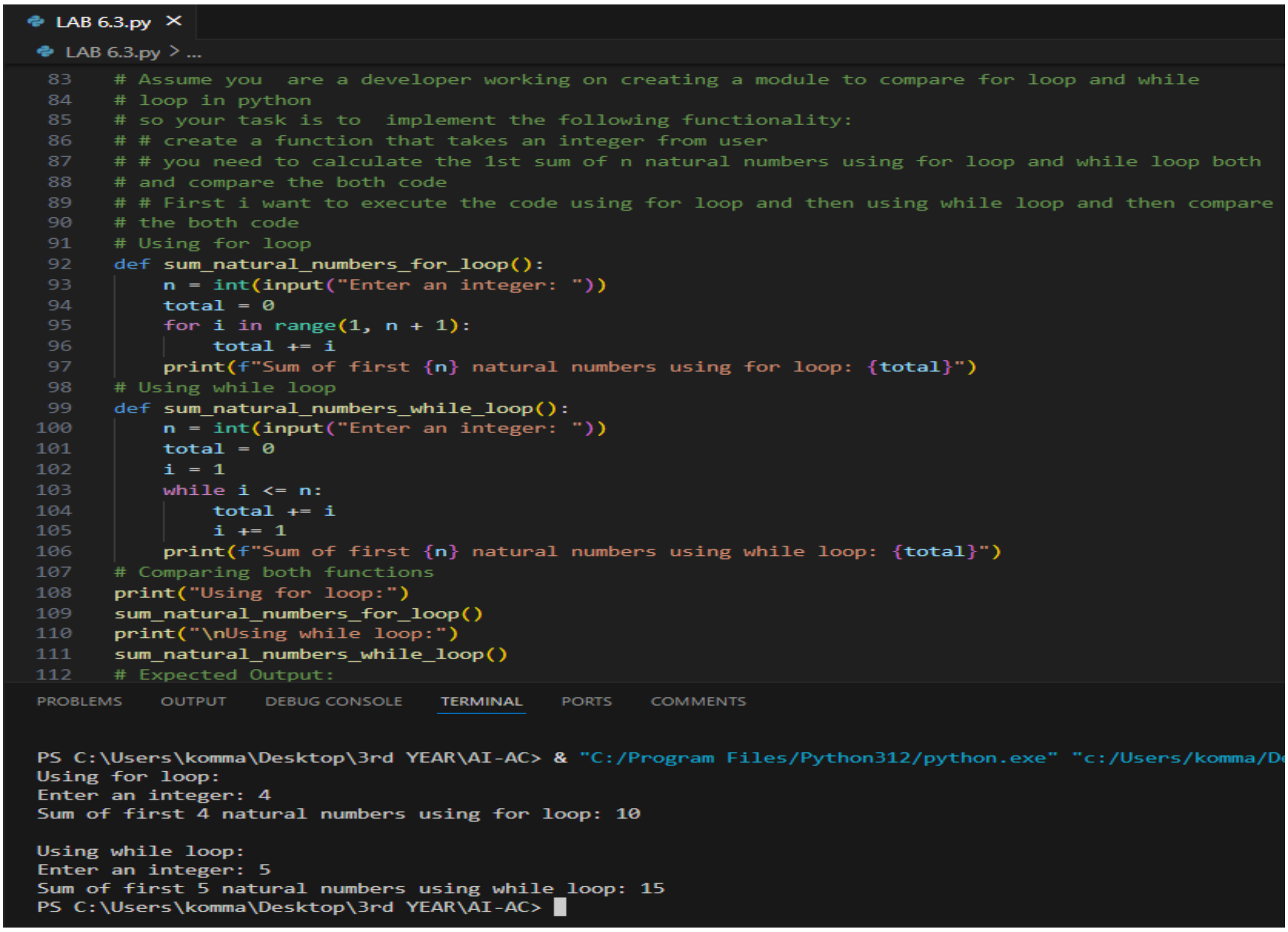
Scenario

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:



```python
# Assume you  are a developer working on creating a module to compare for loop and while
# loop in python
# so your task is to  implement the following functionality:
# # create a function that takes an integer from user
# # you need to calculate the 1st sum of n natural numbers using for loop and while loop both
# and compare the both code
# # First i want to execute the code using for loop and then using while loop and then compare
# the both code
# Using for loop
def sum_natural_numbers_for_loop():
    n = int(input("Enter an integer: "))
    total = 0
    for i in range(1, n + 1):
        total += i
    print(f"Sum of first {n} natural numbers using for loop: {total}")
# Using while loop
def sum_natural_numbers_while_loop():
    n = int(input("Enter an integer: "))
    total = 0
    i = 1
    while i <= n:
        total += i
        i += 1
    print(f"Sum of first {n} natural numbers using while loop: {total}")
# Comparing both functions
print("Using for loop:")
sum_natural_numbers_for_loop()
print("\nUsing while loop:")
sum_natural_numbers_while_loop()
# Expected Output:
```

```
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/De
Using for loop:
Enter an integer: 4
Sum of first 4 natural numbers using for loop: 10

Using while loop:
Enter an integer: 5
Sum of first 5 natural numbers using while loop: 15
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> 
```

Task Description #5: Classes (Bank Account Class)

Scenario

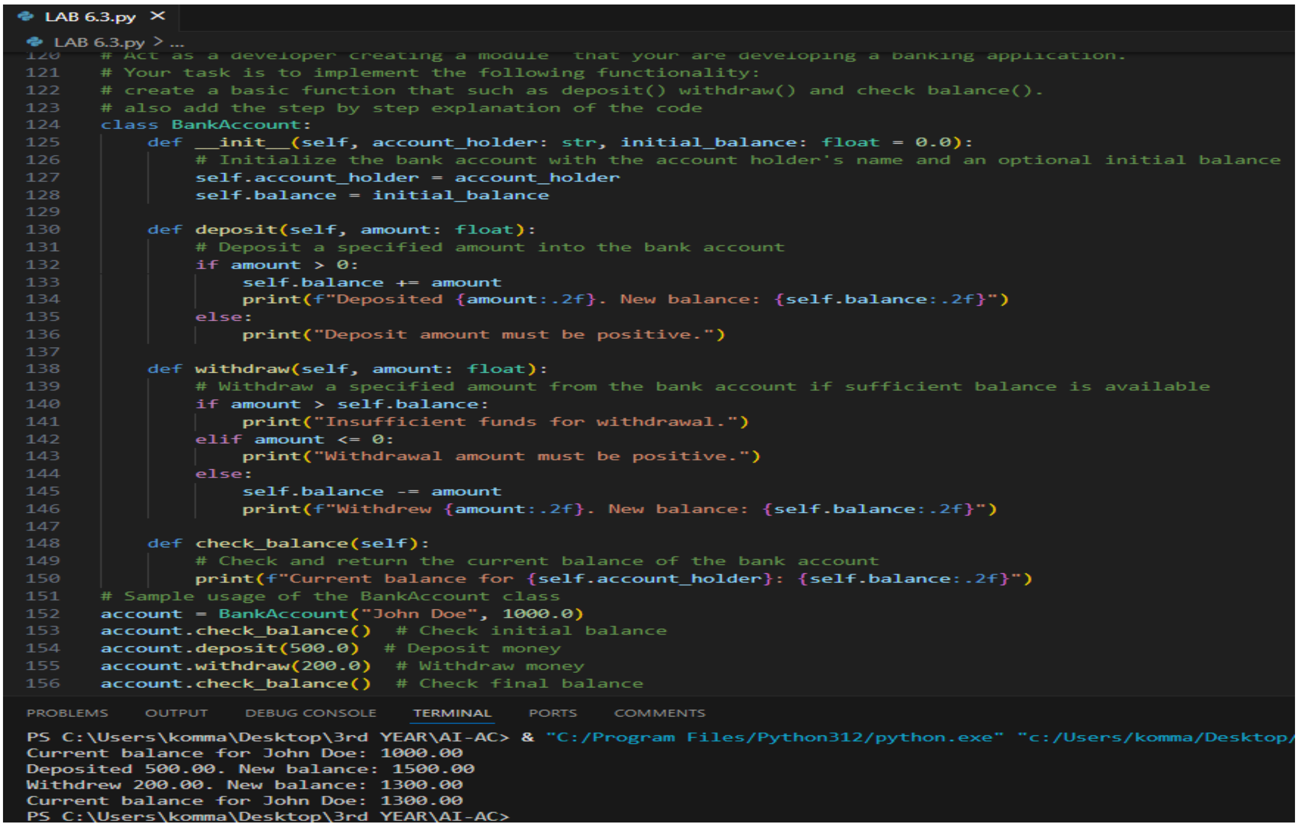
You are designing a basic banking application.

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:



```python
# Act as a developer creating a module  that your are developing a banking application.
# Your task is to implement the following functionality:
# create a basic function that such as deposit() withdraw() and check balance().
# also add the step by step explanation of the code
class BankAccount:
    def __init__(self, account_holder: str, initial_balance: float = 0.0):
        # Initialize the bank account with the account holder's name and an optional initial balance
        self.account_holder = account_holder
        self.balance = initial_balance

    def deposit(self, amount: float):
        # Deposit a specified amount into the bank account
        if amount > 0:
            self.balance += amount
            print(f"Deposited {amount:.2f}. New balance: {self.balance:.2f}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount: float):
        # Withdraw a specified amount from the bank account if sufficient balance is available
        if amount > self.balance:
            print("Insufficient funds for withdrawal.")
        elif amount <= 0:
            print("Withdrawal amount must be positive.")
        else:
            self.balance -= amount
            print(f"Withdrew {amount:.2f}. New balance: {self.balance:.2f}")

    def check_balance(self):
        # Check and return the current balance of the bank account
        print(f"Current balance for {self.account_holder}: {self.balance:.2f}")
# Sample usage of the BankAccount class
account = BankAccount("John Doe", 1000.0)
account.check_balance()  # Check initial balance
account.deposit(500.0)  # Deposit money
account.withdraw(200.0)  # Withdraw money
account.check_balance()  # Check final balance
```

```
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/
Current balance for John Doe: 1000.00
Deposited 500.00. New balance: 1500.00
Withdrew 200.00. New balance: 1300.00
Current balance for John Doe: 1300.00
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>
```