# SR-UNIVERSITY

## ASSIGNMENT:6.3

## B . NO:10

## **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.

Expected Output #1

- A Python class with a constructor (__init__) and a display_details() method.
- Sample object creation and output displayed on the console.
- Brief analysis of AI-generated code.

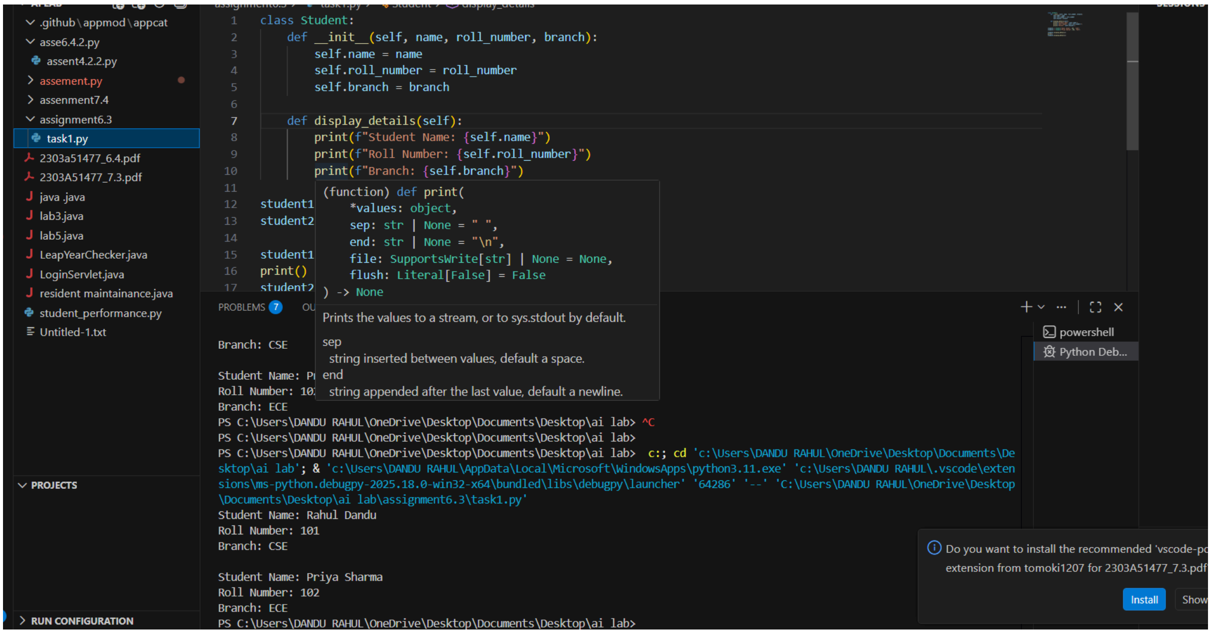
## Prompt:

Create a Python class named Student with the following requirements:
- Attributes: name, roll_number, branch
- A constructor (__init__) to initialize the attributes
- A method display_details() that prints the student's details
- Create at least two Student objects and display their details

## Code:

```python
class Student:
    def __init__(self, name, roll_number, branch):
        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}")

student1 = Student("Rahul Dandu", 101, "CSE")
```

```python
student2 = Student("Priya Sharma", 102, "ECE")

student1.display_details()
print()
student2.display_details()
```



```python
class Student:
    def __init__(self, name, roll_number, branch):
        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}")

student1
student2

student1
print()
student2
```

```
(function) def print(
    *values: object,
    sep: str | None = " ",
    end: str | None = "\n",
    file: SupportsWrite[str] | None = None,
    flush: Literal[False] = False
) -> None
```

Prints the values to a stream, or to sys.stdout by default.

sep
 string inserted between values, default a space.
end
 string appended after the last value, default a newline.

```
Branch: CSE

Student Name: Pr
Roll Number: 10
Branch: ECE
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab> ^C
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>  c:; cd 'c:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\De
sktop\ai lab'; & 'c:\Users\DANDU RAHUL\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\DANDU RAHUL\.vscode\exten
sions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64286' '--' 'C:\Users\DANDU RAHUL\OneDrive\Desktop
\Documents\Desktop\ai lab\assignment6.3\task1.py'
Student Name: Rahul Dandu
Roll Number: 101
Branch: CSE

Student Name: Priya Sharma
Roll Number: 102
Branch: ECE
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
```

Do you want to install the recommended 'vscode-p
extension from tomoki1207 for 2303A51477_7.3.pdf

## Analysis:

## The `Student` class correctly initializes attributes and displays student details using `display_details()`. The code is simple, readable, and works as intended.

## Task #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).

Expected Output #2

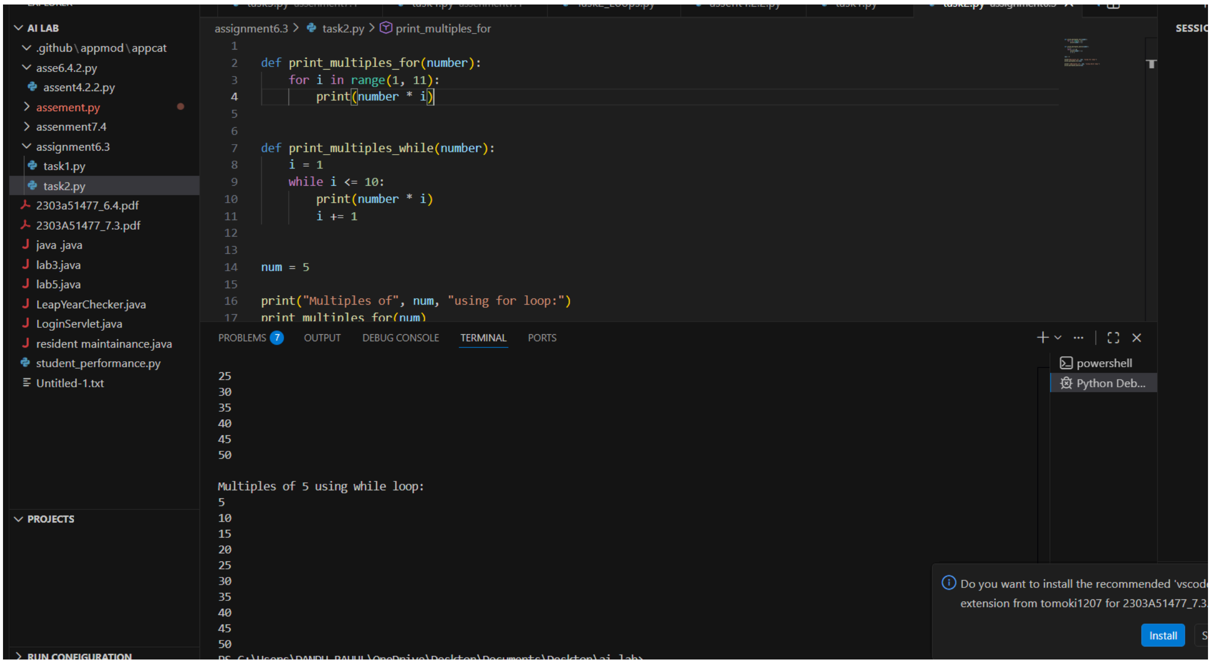
- Correct loop-based Python implementation.
- Output showing the first 10 multiples of a number.
- Comparison and analysis of different looping approaches.

# Prompt:

Write a Python program that prints the first 10 multiples of a given number.
1. First, use a for loop to print the multiples.
2. Then, use a while loop to print the same multiples.
3. Include example usage with a number like 5.

# Code:

```python
def print_multiples_for(number):
    for i in range(1, 11):
        print(number * i)


def print_multiples_while(number):
    i = 1
    while i <= 10:
        print(number * i)
        i += 1


num = 5

print("Multiples of", num, "using for loop:")
print_multiples_for(num)

print("\nMultiples of", num, "using while loop:")
print_multiples_while(num)
```



```python

def print_multiples_for(number):
    for i in range(1, 11):
        print(number * i)


def print_multiples_while(number):
    i = 1
    while i <= 10:
        print(number * i)
        i += 1


num = 5

print("Multiples of", num, "using for loop:")
print_multiples_for(num)
```

```
25
30
35
40
45
50

Multiples of 5 using while loop:
5
10
15
20
25
30
35
40
45
50
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
```

Do you want to install the recommended 'vscod
extension from tomoki1207 for 2303A51477_7.3

analysis:

The `for` loop is easy and clear for a fixed number of multiples.

The `while` loop works the same but needs a counter and is better for flexible conditions

**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).

Expected Output #3

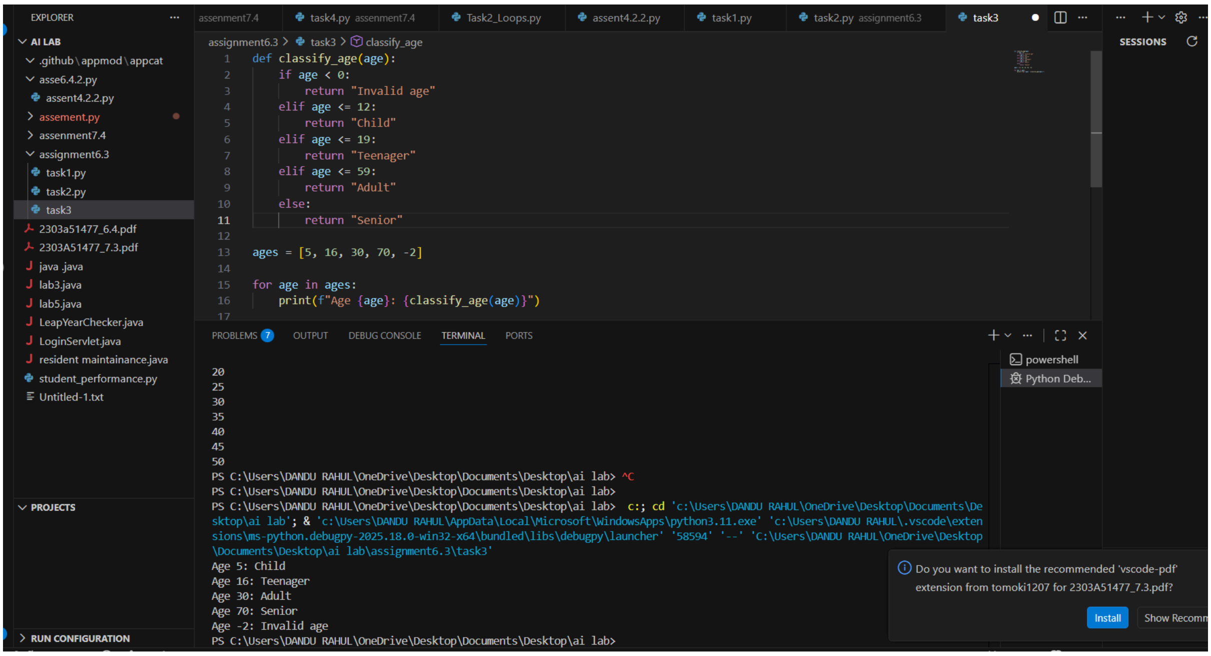
- A Python function that classifies age into appropriate groups.
- Clear and correct conditional logic.
- Explanation of how the conditions work

# Prompt:

Write a Python function that classifies a person's age into groups:
- Child (0–12), Teenager (13–19), Adult (20–59), Senior (60 and above).
Use nested if-elif-else statements to return the age group.
Also, create an alternative version using a dictionary with ranges to classify the same way.
Include example usage for ages like 5, 16, 30, 70, and -2.

## Code:

```python
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"

ages = [5, 16, 30, 70, -2]

for age in ages:
    print(f"Age {age}: {classify_age(age)}")
```



```python
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"

ages = [5, 16, 30, 70, -2]

for age in ages:
    print(f"Age {age}: {classify_age(age)}")
```

```
20
25
30
35
40
45
50
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab> ^C
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>  c:; cd 'c:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\De
sktop\ai lab'; & 'c:\Users\DANDU RAHUL\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\DANDU RAHUL\.vscode\exten
sions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58594' '--' 'C:\Users\DANDU RAHUL\OneDrive\Desktop
\Documents\Desktop\ai lab\assignment6.3\task3'
Age 5: Child
Age 16: Teenager
Age 30: Adult
Age 70: Senior
Age -2: Invalid age
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
```

Do you want to install the recommended 'vscode-pdf' extension from tomoki1207 for 2303A51477_7.3.pdf?

## Analysis:

**The `if-elif-else` version classifies age step by step and is easy to understand.**

**The dictionary approach does the same using ranges, making it more flexible for changes.**

## 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.

Expected Output #4

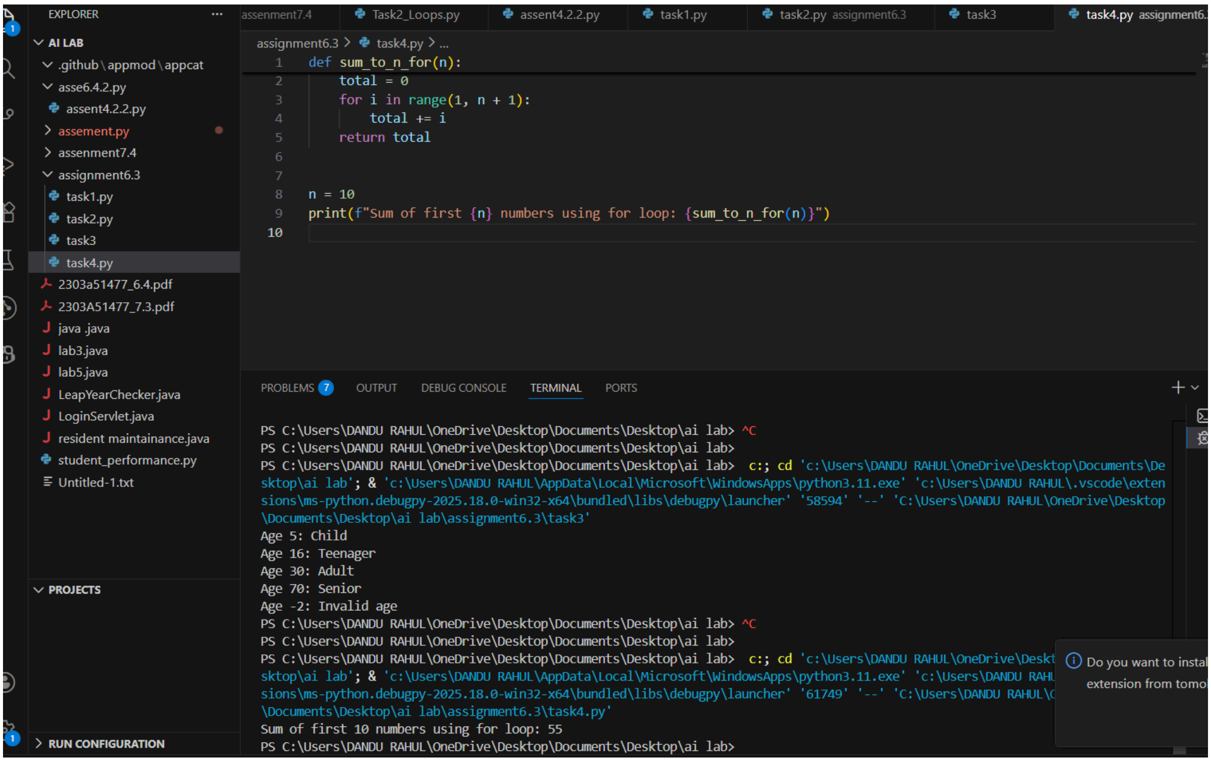
- Python function to compute the sum of first n numbers.
- Correct output for sample inputs.
- Explanation and comparison of different approaches

# Prompt:

**Write a Python program to calculate the sum of the first n natural numbers.**
**1. First, implement a function using a for loop.**
**2. Then, implement the same using a while loop.**
**3. Finally, implement it using the mathematical formula n*(n+1)//2.**
**Include example usage for n = 10 and display outputs for all three methods.**

## Code:

```python
def sum_to_n_for(n):
    total = 0
    for i in range(1, n + 1):
        total += i
    return total


n = 10
print(f"Sum of first {n} numbers using for loop: {sum_to_n_for(n)}")
```



```python
def sum_to_n_for(n):
    total = 0
    for i in range(1, n + 1):
        total += i
    return total


n = 10
print(f"Sum of first {n} numbers using for loop: {sum_to_n_for(n)}")

```

```
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab> ^C
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>  c:; cd 'c:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\De
sktop\ai lab'; & 'c:\Users\DANDU RAHUL\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\DANDU RAHUL\.vscode\exten
sions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58594' '--' 'C:\Users\DANDU RAHUL\OneDrive\Desktop
\Documents\Desktop\ai lab\assignment6.3\task3'
Age 5: Child
Age 16: Teenager
Age 30: Adult
Age 70: Senior
Age -2: Invalid age
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab> ^C
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>  c:; cd 'c:\Users\DANDU RAHUL\OneDrive\Deskt
sktop\ai lab'; & 'c:\Users\DANDU RAHUL\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\DANDU RAHU
sions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61749' '--' 'C:\Users\DANDU RAHUL\C
\Documents\Desktop\ai lab\assignment6.3\task4.py'
Sum of first 10 numbers using for loop: 55
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
```

**Analysis:**

The `for` loop correctly adds numbers from 1 to `n` and returns the total. It is simple, readable, and works well for calculating the sum of the first n natural numbers.

**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.

Expected Output #5

- Complete Python Bank Account class.
- Demonstration of deposit and withdrawal operations with updated balance.
- Well-commented code with a clear explanation.

**prompt:**

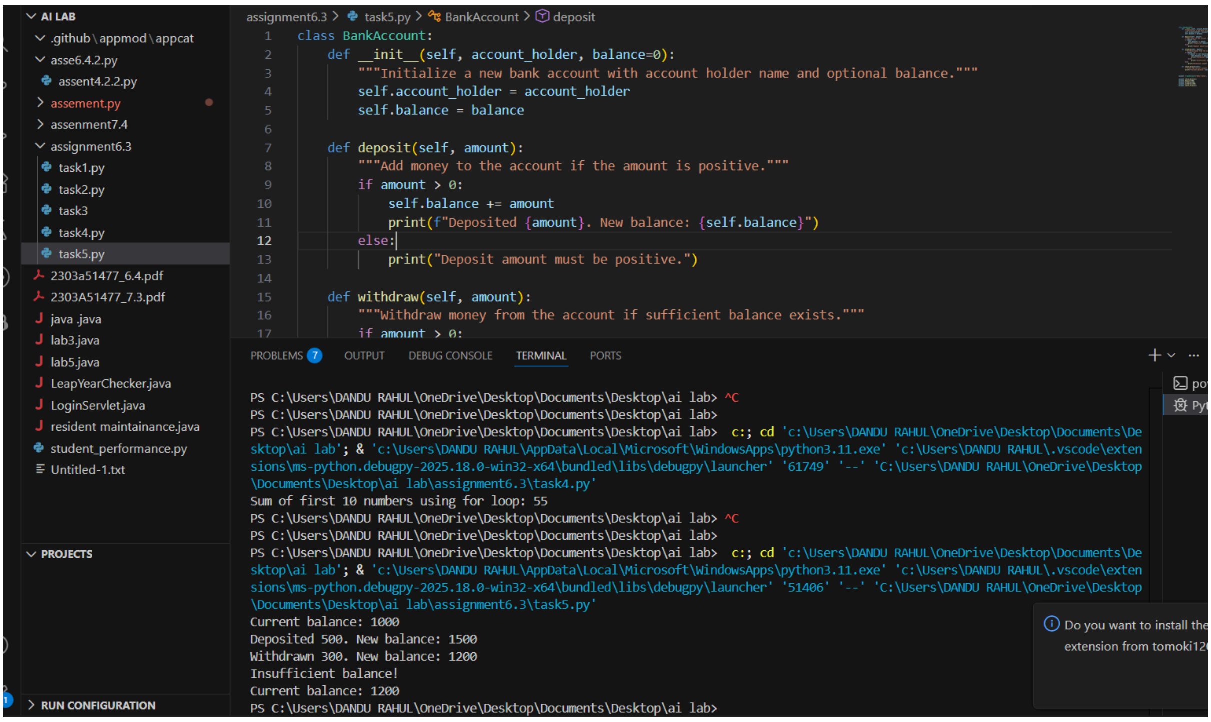
**Create a Python class called BankAccount with methods deposit(), withdraw(), and check_balance().**

**The class should track the account holder's name and balance.**

**Include example usage showing deposits, withdrawals, and balance checks.**

Code:

```python
class BankAccount:
    def __init__(self, account_holder, balance=0):
        """Initialize a new bank account with account holder name and optional balance."""
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        """Add money to the account if the amount is positive."""
        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 if sufficient balance exists."""
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"Withdrawn {amount}. New balance: {self.balance}")
            else:
                print("Insufficient balance!")
        else:
            print("Withdrawal amount must be positive.")

    def check_balance(self):
        """Print the current account balance."""
        print(f"Current balance: {self.balance}")



account = BankAccount("Rahul Dandu", 1000)

account.check_balance()
account.deposit(500)
account.withdraw(300)
account.withdraw(1500)
account.check_balance()
```



```python
class BankAccount:
    def __init__(self, account_holder, balance=0):
        """Initialize a new bank account with account holder name and optional balance."""
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        """Add money to the account if the amount is positive."""
        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 if sufficient balance exists."""
        if amount > 0:
```

```
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab> ^C
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>  c:; cd 'c:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\De
sktop\ai lab'; & 'c:\Users\DANDU RAHUL\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\DANDU RAHUL\.vscode\exten
sions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61749' '--' 'C:\Users\DANDU RAHUL\OneDrive\Desktop
\Documents\Desktop\ai lab\assignment6.3\task4.py'
Sum of first 10 numbers using for loop: 55
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab> ^C
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>  c:; cd 'c:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\De
sktop\ai lab'; & 'c:\Users\DANDU RAHUL\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\DANDU RAHUL\.vscode\exten
sions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51406' '--' 'C:\Users\DANDU RAHUL\OneDrive\Desktop
\Documents\Desktop\ai lab\assignment6.3\task5.py'
Current balance: 1000
Deposited 500. New balance: 1500
Withdrawn 300. New balance: 1200
Insufficient balance!
Current balance: 1200
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
```

analysis:

The `BankAccount` class correctly manages deposits, withdrawals, and balance checks with proper validation.
It is clear, readable, and demonstrates basic banking operations safely.