AI ASSISTED CODING

**ASSIGNMENT -13.3**

**H.no:2403a51182**

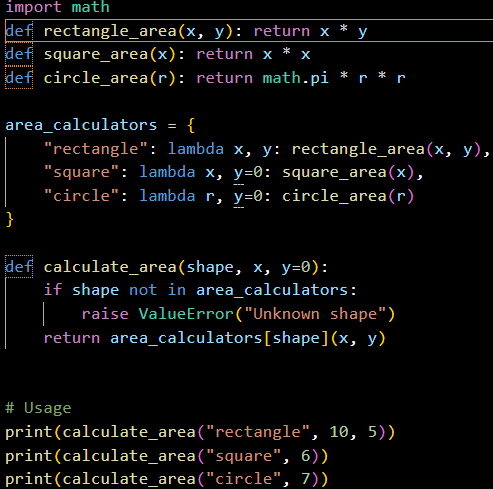
**Batch-06**

**TASK 1:**

**PROMPT:**

Refactor the given redundant calculate\_area function to improve readability, reduce duplication, and support scalability when adding new shapes.

**CODE:**

****

**OUTPUT:**

****

**OBSERVATION:**

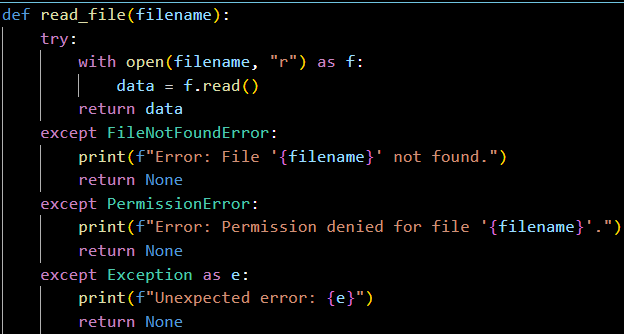
The original code contained repetitive if/elif checks, making it hard to maintain.  
After refactoring, each shape’s logic is modular and reusable, ensuring better **readability, maintainability, and extensibility**.

TASK-2:

**PROMPT:**

Refactor the legacy read\_file function to use with open() for automatic file handling and add proper try-except blocks for error handling.

**CODE:**

****

**OBSERVATION:**

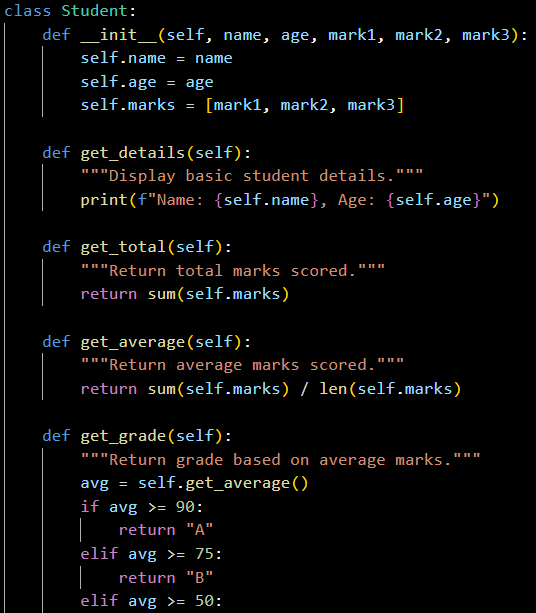
The original code lacked error handling and required manual file closing, which could lead to resource leaks.  
The refactored version ensures **safe file handling, automatic closure, and clear error reporting**, making the function more **robust and reliable**.

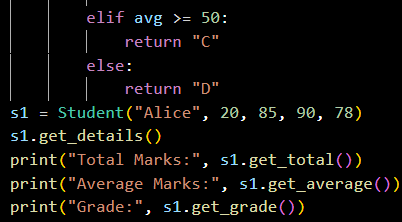
TASK-3:

PROMPT:

Refactor the legacy Student class to use meaningful variable names, modular methods, and improve readability while supporting extensibility (like average and grade calculation).

code:





OUTPUT:



OBSERVATION:

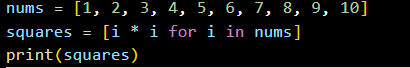
The original class had cryptic variable names and limited functionality.  
The refactored version improves **readability, modularity, and scalability**, making the class more reusable and easier to maintain.

TASK-4:

**PROMPT:**

Refactor the given loop to use a **Pythonic list comprehension** for better readability and efficiency.

**CODE:**

****

**OUTPUT:**

****

**OBSERVATION:**

The original code used an explicit loop with append(), which is less efficient and verbose.  
The refactored version with a **list comprehension** is concise, faster, and improves **readability and performance**.