**SR UNIVERSITY  
  
AI ASSISTED CODING LAB**

**ASSIGNMENT 13.2**

**NAME:** MOHAMMED MUTAKABIR HUSSAIN

**H.NO: 2503A51L35  
BATCH: 20**

**TASK 1  
  
Task Description #1 – Remove Repetition**

Task: Provide AI with the following redundant code and ask it to refactor

**Python Code**

def calculate\_area(shape, x, y=0):

if shape == "rectangle":

return x \* y

elif shape == "square":

return x \* x

elif shape == "circle":

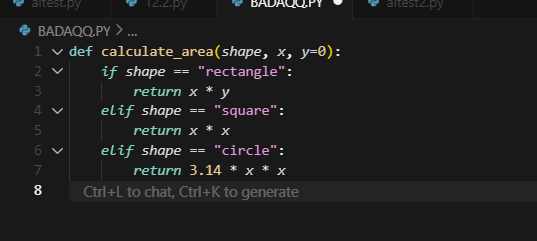
return 3.14 \* x \* x

**Expected Output**

* Refactored version with dictionary-based dispatch or separate functions.
* Cleaner and modular design.

**PROMPT:**  
A black screen with white text

AI-generated content may be incorrect.

**ERROR CODE GIVEN :**  


**OUTPUT:**  
A screenshot of a computer

AI-generated content may be incorrect.

**OBSERVATION:**   
we have gave the code to the cursor AI, and it refactored it into a simpler and more readable version. The new code removed repetition and still fulfilled output expectations with proper logic. This made the program cleaner and easier to maintain

**TASK 2:**  
**Task Description #2 – Error Handling in Legacy Code**

Task: Legacy function without proper error handling

**Python Code**

def read\_file(filename):

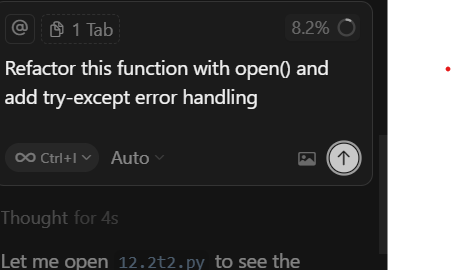
f = open(filename, "r")

data = f.read()

f.close()

return data

**Expected Output:**

AI refactors with with open() and try-except:  
  
**PROMPT:**  


**Given CODE :**   
A screen shot of a computer

AI-generated content may be incorrect.

**OUTPUT:**  
A screen shot of a computer program

AI-generated content may be incorrect.

**OBSERVATION:**

We have gave the code to the cursor AI, and it refactored it using with open() for safer file handling also added try-except blocks to catch errors, making the function more efficient .

**TASK 3**  
**Task Description #3 – Complex Refactoring**

Task: Provide this legacy class to AI for readability and modularity improvements:

**Python Code**

class Student:

def \_\_init\_\_(self, n, a, m1, m2, m3):

self.n = n

self.a = a

self.m1 = m1

self.m2 = m2

self.m3 = m3

def details(self):

print("Name:", self.n, "Age:", self.a)

def total(self):

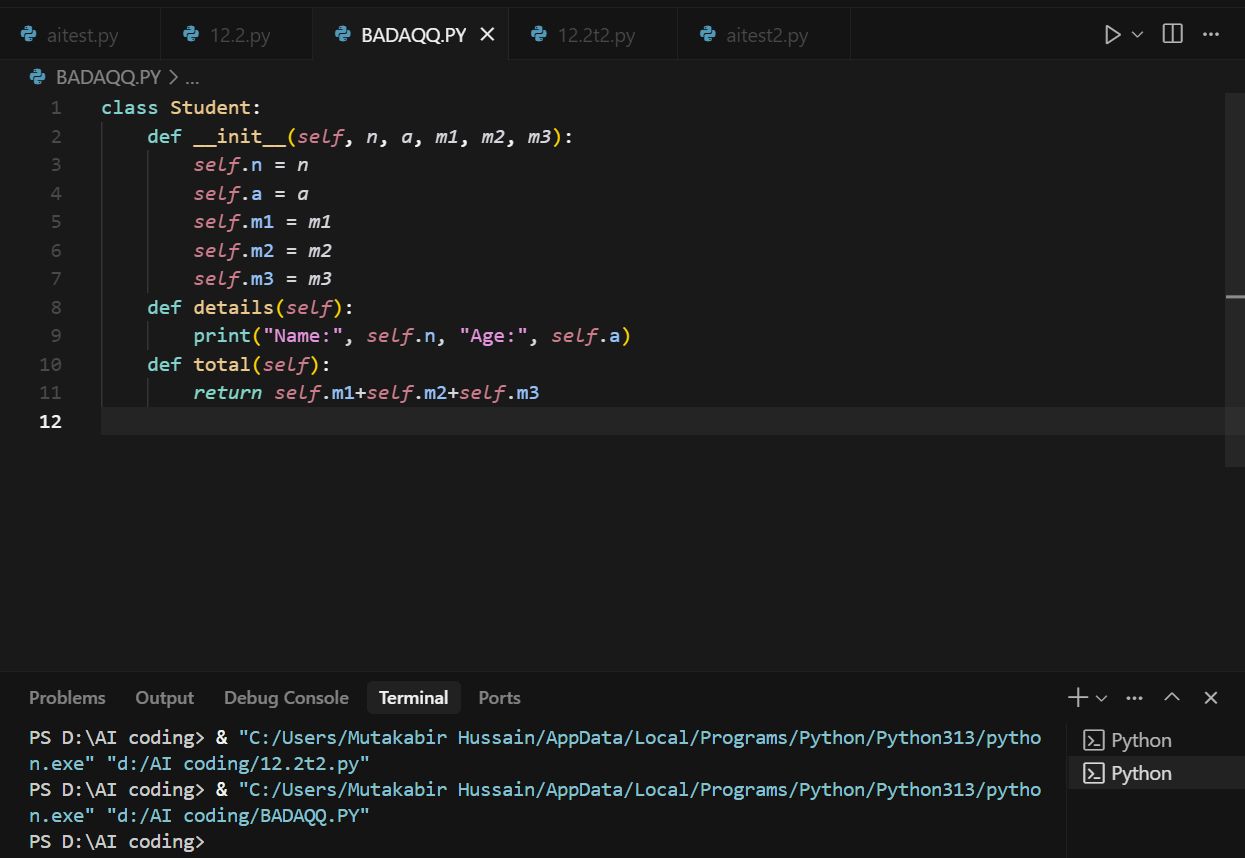
return self.m1+self.m2+self.m3

**Expected Output:**

* AI improves naming (name, age, marks).
* Adds docstrings.
* Improves print readability.
* Possibly uses sum(self.marks) if marks stored in a list.

**PROMPT:**  
A screenshot of a black screen

AI-generated content may be incorrect.

**CODE GIVEN:**  


**OUTPUT:**  
A screenshot of a computer program

AI-generated content may be incorrect.

**OBSERVATION:**

I have gave Student class code to the cursor AI and told what I require , and it made the code easier to read and use.   
Variable names were improved,  
Docstrings were added,   
The details print is cleaner,  
And total marks are now calculated using a list and sum().

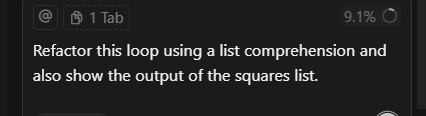
**TASK 4:**

**Task Description #4 – Inefficient Loop Refactoring**

Task: Refactor this inefficient loop with AI help

**Python Code**

nums = [1,2,3,4,5,6,7,8,9,10]  
squares = []  
for i in nums:  
squares.append(i \* i)

**Expected Output:** AI suggested a **list comprehension  
  
PROMPT:  
**

**CODE GIVEN:  
A screenshot of a computer

AI-generated content may be incorrect.**

**OUTPUT:  
A screenshot of a computer

AI-generated content may be incorrect.  
A black screen with white text

AI-generated content may be incorrect.**

**OBSERVATION:**I gave the loop code to the AI and asked it to refactor using a list comprehension and show the output. The AI returned a shorter, cleaner code and displayed the correct squares of the numbers, making it more efficient and readable.