**AI ASSISTED CODING**

**LAB-13.2**

**NAME**:KANDUKURI SRIKAR

**ROLL NO**:2503A51L03(B19)

**Task Description #1 – Remove Repetition**  
Task: Provide AI with the following redundant code and ask it torefactor  
Python Code  
defcalculate\_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 separatefunctions.  
• Cleaner and modular design

**PROMPT:**

Python Code

defcalculate\_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

for this code give Refactored version with dictionary-based dispatch or separate

functions with Cleaner and modular design.



**OUTPUT:**

****

**OBSERVATION:**

The code was refactored from using multiple if-elif statements inside the calculate\_area function to a cleaner, modular design. Now, each shape's area calculation is handled by a separate function, and a dictionary-based dispatch is used to select the correct function based on the shape.

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

Task: Legacy function without proper error handling  
Python Code  
defread\_file(filename):  
f = open(filename, "r")  
data = f.read()

f.close()  
return data

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

**PROMPT:**

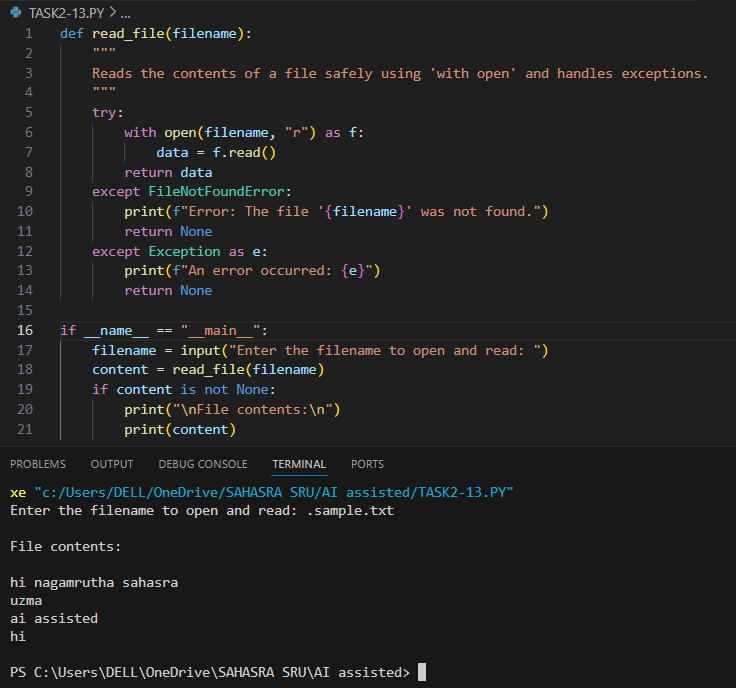
defread\_file(filename):

f = open(filename, "r")

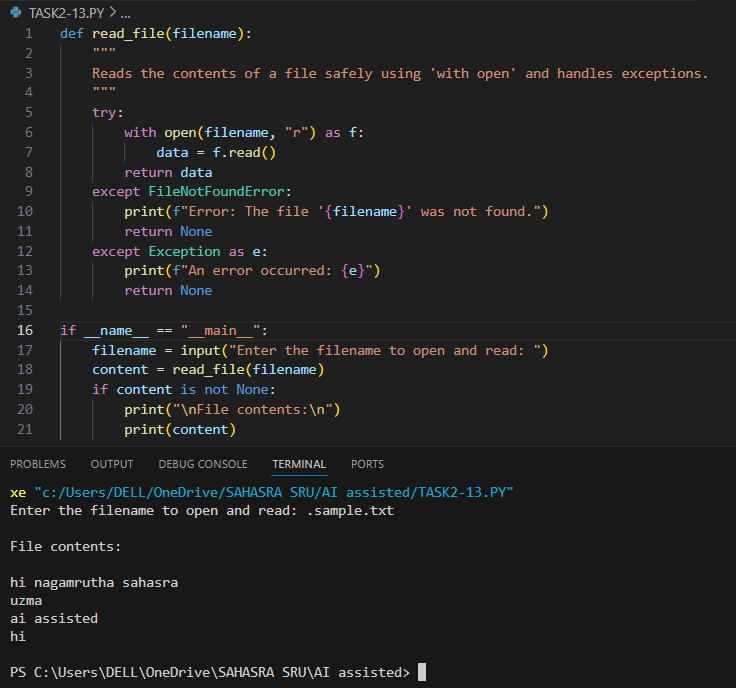
data = f.read()

f.close()

return data for the above code refactor with with open() and try-except let user give input to open and read the file.



**OUTPUT:**

****

**OBSERVATION:**

The code safely reads and displays the contents of a user-specified file using with open and exception handling. It prompts the user for a filename, reads the file if it exists, and prints its contents. Errors such as missing files are handled gracefully with clear messages.

**Task Description #3 – Complex Refactoring**  
Task: Provide this legacy class to AI for readability and modularityimprovements:  
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:**

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

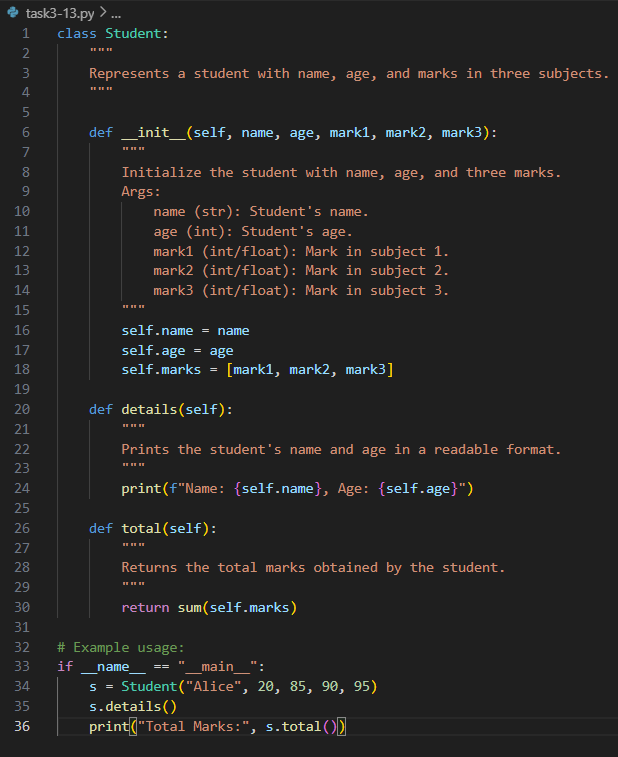
def details(self):

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

def total(self):

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

for the above code improve naming (name, age, marks). Add docstrings and use sum(self.marks) if marks stored in a list.



**OUTPUT:**

****

**OBSERVATION:**

The code now allows the user to input a student's name, age, and marks interactively. It stores the marks in a list for better structure and uses clear, descriptive variable names. The program prints the student's details and the total marks in a readable format, making it user-friendly and easy to maintain.

**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:**

Python Code

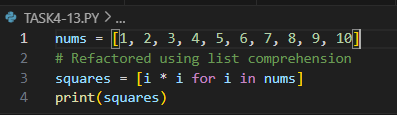
nums = [1,2,3,4,5,6,7,8,9,10]

squares = []

for i in nums:

squares.append(i \* i)

refactor the above code and suggest a list comprehension and print a example square

****

**OUTPUT:**

****

**OBSERVATION:**

The code efficiently generates a list of squares for numbers 1 to 10 using list comprehension, making it more concise and readable compared to a traditional loop. The output displays all the squared values in a single list.