**AI ASSISTED CODING LAB**

**ASSIGNMENT 13.2**

**ENROLLMENT NO :**2503A51L47

**BATCH NO:** 20

**NAME:**  N.AKASH

TASK1

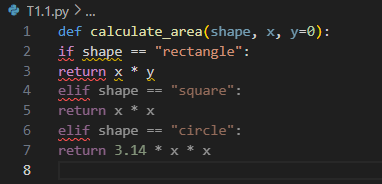
**TASK1 DESCRIPTION:-** Provide AI with the following redundant code and ask it to  
refactor

**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

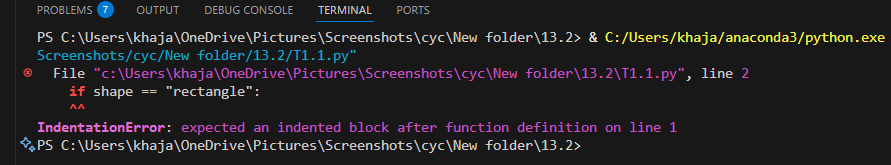
**PROMPT:-** Refactor the redundant code for calculating areas of shapes.

**CODE:-**

ERROR CODE:-



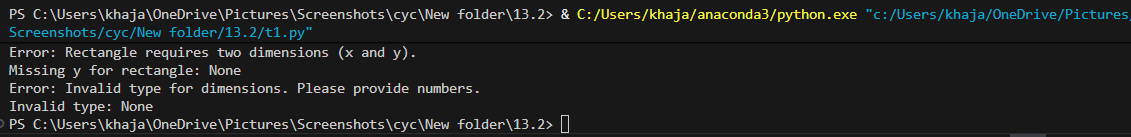
OUTPUT:-



AI GENERATED CODE:-



OUTPUT:-



**OBSERVATION:-**

The AI refactored the function, improved readability, and added error handling, making the

code more reliable.

TASK2

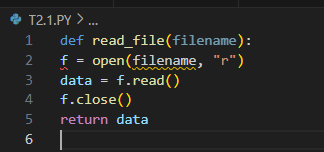
**TASK2 DESCRIPTION**:- Error Handling in Legacy Code  
Task: Legacy function without proper error handling  
Python Code  
def read\_file(filename):  
f = open(filename, "r")  
data = f.read()

f.close()  
return data

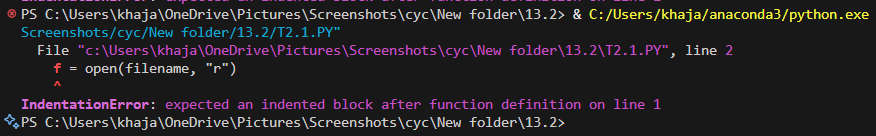
**PROMPT:-** Improve the legacy file reading function with error handling.

**CODE:-**

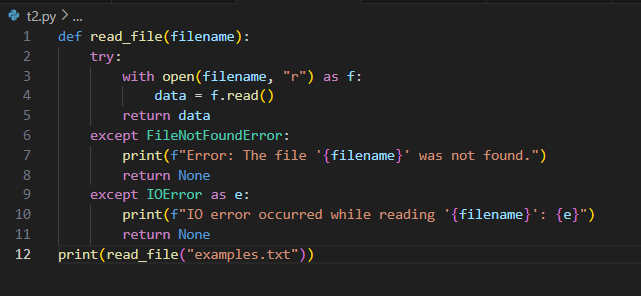
ERROR CODE:-



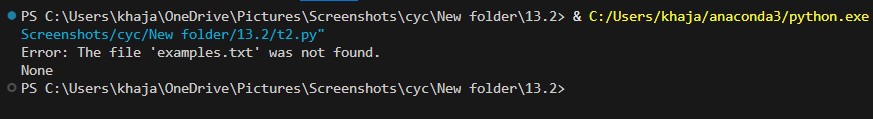
OUTPUT:-



AI GENERATED CODE:-



OUTPUT:-



**OBSERVATION:-** The AI added exception handling for missing files and ensured safe file operations, preventing runtime errors.

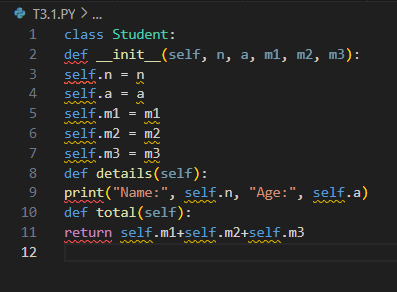
TASK3

**TASK3 DESCRIPTION:-** 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

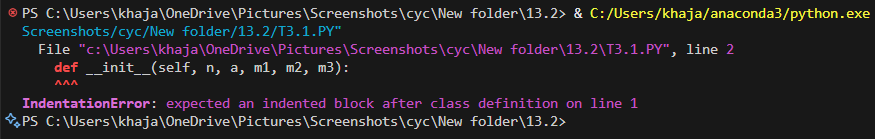
**PROMPT**:- Refactor the legacy Student class for readability and modularity.

**CODE:-**

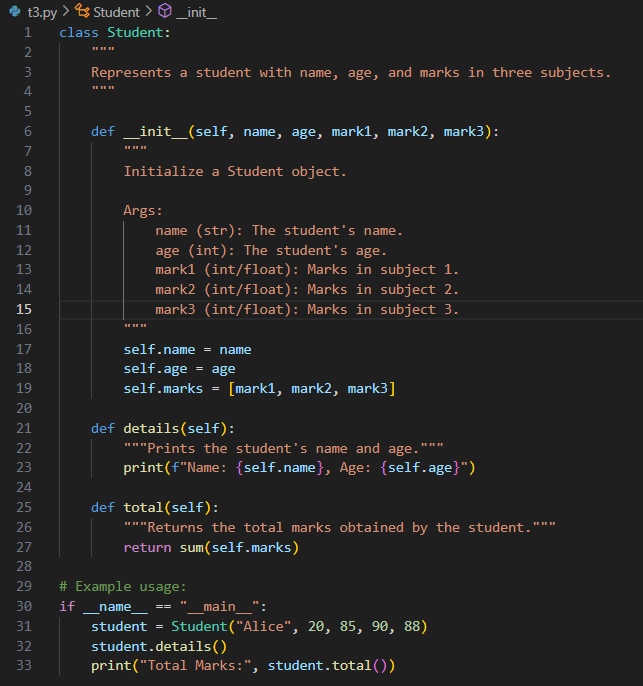
ERROR CODE:-



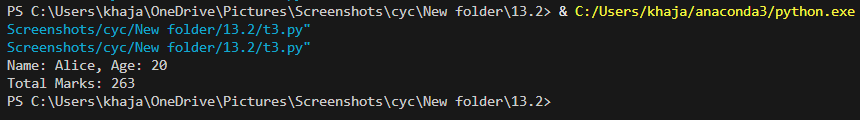
OUTPUT:-



AI GENERATED CODE:-



OUTPUT:-



**OBSERVATION**:- The AI improved the class structure, added clear methods, and made the code more organized and maintainable.

TASK4

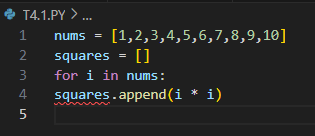
**TASK4 DESCRIPTION:-** Inefficient Loop Refactoring  
Task: Refactor this inefficient loop with AI help

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

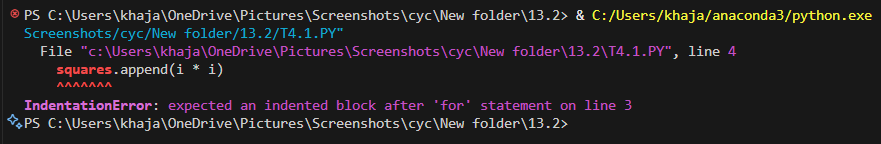
**PROMPT:-** Refactor the inefficient loop for better performance.

CODE:-

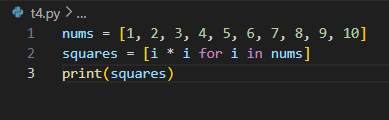
ERROR CODE:-



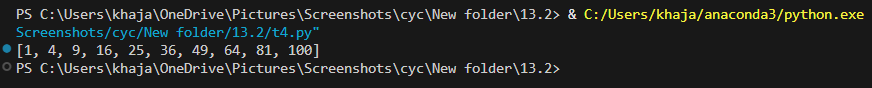
OUTPUT:-



AI GENERATED CODE:-



OUTPUT:-



**OBSERVATION:-** The AI replaced the loop with a list comprehension, simplifying the code and improving efficiency.