AI ASSISTED CODING

LAB:13.2

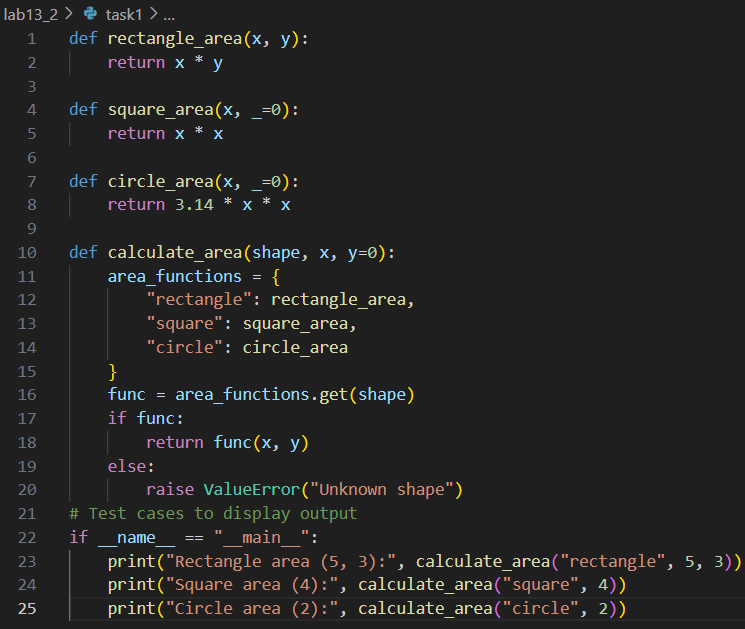
ROLLNO:2403A52096

BATCH:04

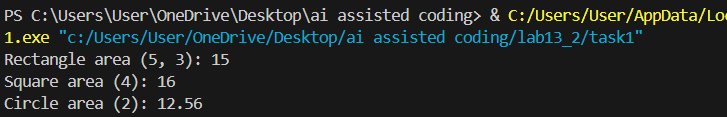
TASK1:

PROMPT: Refactor a Python program to calculate the area of a rectangle, square, and circle using a modular approach. Implement separate functions for each shape and use a dictionary to select the correct function based on user input. Print the area for a rectangle (5, 3), a square (4), and a circle (2).

CODE:



OUTPUT:



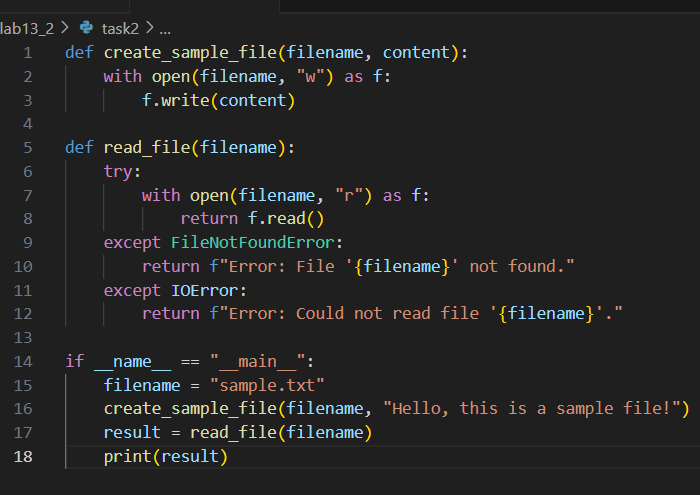
OBSERVATION:

The refactored program uses a dictionary to map shape names to their respective area functions, resulting in a modular and maintainable design. The output correctly displays the area for each shape, confirming the program works as intended.

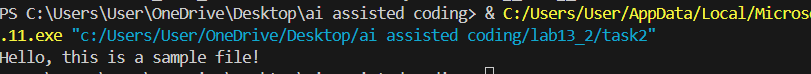
TASK2:

PROMPT: Refactor a Python program to safely create and read a text file. Implement error handling for file operations using try-except blocks. The program should write a sample message to a file, read the file content, and print it, handling any file-not-found or read errors gracefully.

CODE:



OUTPUT:



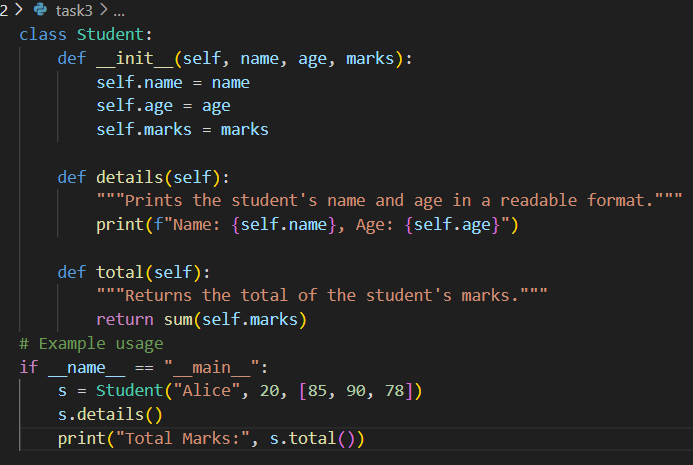
OBSERVATION:

The program creates a sample file, reads its content, and prints it to the console. Exception handling ensures that missing files or read errors are reported with clear messages, making the code robust and user-friendly. The output confirms successful file operations.

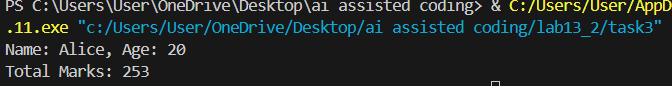
TASK3:

PROMPT: Refactor a legacy Python class representing a student to improve readability and modularity. Use clear variable names ([name](vscode-file://vscode-app/c:/Users/User/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), [age](vscode-file://vscode-app/c:/Users/User/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), [marks](vscode-file://vscode-app/c:/Users/User/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)), store marks as a list, add docstrings, and enhance the output formatting for displaying student details and total marks.

CODE:



OUTPUT:



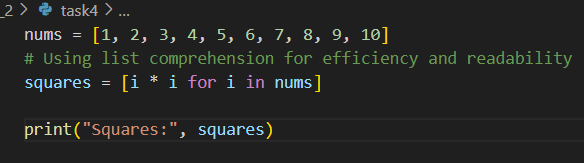
OBSERVATION:

The refactored class now uses descriptive names and stores marks in a list, making it more flexible and maintainable. Docstrings provide clarity on the class and its methods. The output is more readable, and the code structure is modular and easy to extend.

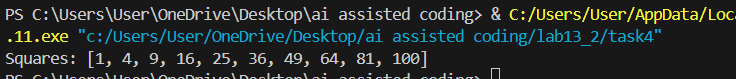
TASK4:

PROMPT: Refactor the following Python code to improve efficiency and readability. Replace the explicit for-loop that appends the square of each number in a list with a list comprehension.

CODE:



OUTPUT:



OBSERVATION:

The refactored code uses a list comprehension to generate the list of squares, making the code more concise and efficient. The output correctly displays the squares of the numbers from 1 to 10.