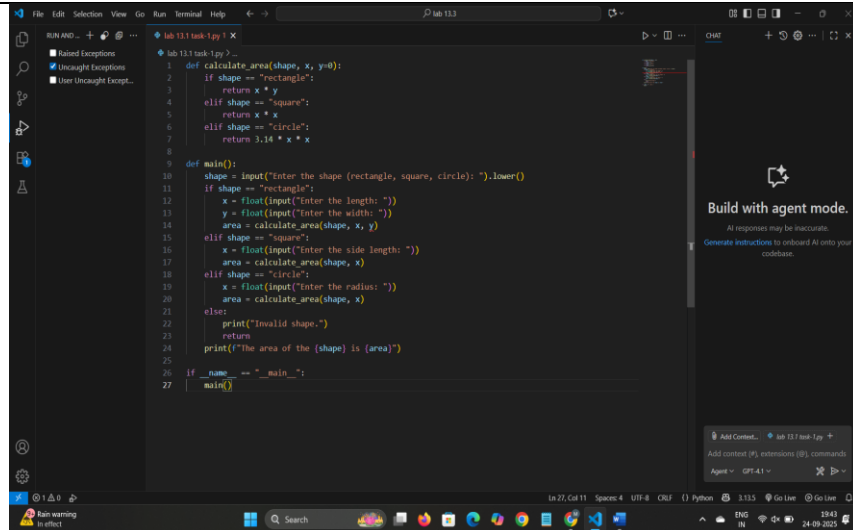


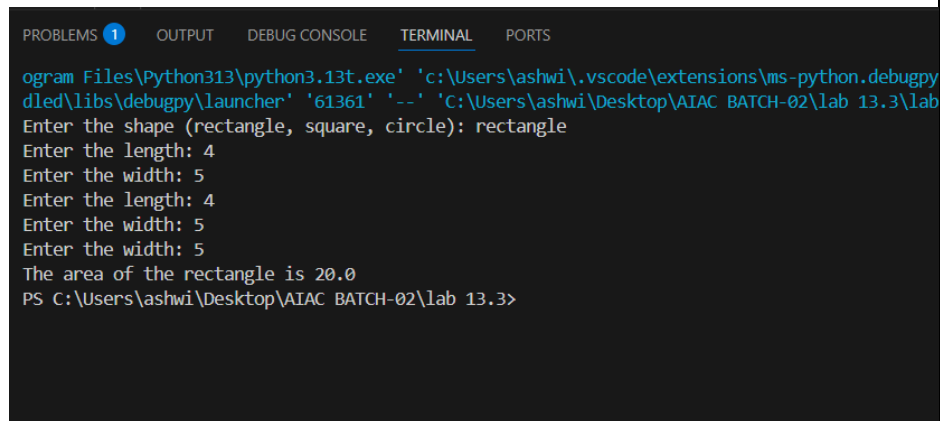
SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	Academic Year: 2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Instructor(s) Name		Dr. V. Venkataramana (Co-ordinator)	
		Dr. T. Sampath Kumar	
		Dr. Pramoda Patro	
		Dr. Brij Kishor Tiwari	
		Dr. J. Ravichander	
		Dr. Mohammand Ali Shaik	
		Dr. Anirodh Kumar	
		Mr. S. Naresh Kumar	
		Dr. RAJESH VELPULA	
		Mr. Kundhan Kumar	
		Ms. Ch. Rajitha	
		Mr. M Prakash	
		Mr. B. Raju	
		Intern 1 (Dharma teja)	
		Intern 2 (Sai Prasad)	
		Intern 3 (Sowmya)	
		NS_2 (Mounika)	
Course Code	24CS002PC215	Course Title	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	Week 7 - Wednesday	Time(s)	
Duration	2 Hours	Applicable to Batches	
Assignment Number: 13.3 (Present assignment number)/24 (Total number of assignments)			
Q.No.	Question		Expected Time to complete
1	Lab 13 – Code Refactoring: Improving Legacy Code with AI Suggestions Lab Objectives • To introduce the concept of code refactoring and why it matters (readability, maintainability, performance).		Week 5 - Monday

	<ul style="list-style-type: none"> • To practice using AI tools for identifying and suggesting improvements in legacy code. • To evaluate the before vs. after versions for clarity, performance, and correctness. • To reinforce responsible AI-assisted coding practices (avoiding over-reliance, validating outputs). <p>Learning Outcomes</p> <p>After completing this lab, students will be able to:</p> <ol style="list-style-type: none"> 1. Use AI to analyze and refactor poorly written Python code. 2. Improve code readability, efficiency, and error handling. 3. Document AI-suggested improvements through comments and explanations. 4. Apply refactoring strategies without changing functionality. 5. Critically reflect on AI's refactoring suggestions. <p>Task Description #1 – Remove Repetition</p> <p>Task: Provide AI with the following redundant code and ask it to refactor</p> <p>Python Code</p> <pre>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</pre> <p>Expected Output</p> <ul style="list-style-type: none"> • Refactored version with dictionary-based dispatch or separate functions. • Cleaner and modular design. <p>PROMPT: REFACTOR THE GIVEN CODE SEPARATE THE FUNCTIONS AND USER SHOULD ENTER THE INPUT PRINT THE OUTPUT WITHOUT ANY ERRORS</p>	
--	---	--



```
1 def calculate_area(shape, x, y=0):
2     if shape == "rectangle":
3         return x * y
4     elif shape == "square":
5         return x * x
6     elif shape == "circle":
7         return 3.14 * x * x
8
9 def main():
10     shape = input("Enter the shape (rectangle, square, circle): ").lower()
11     if shape == "rectangle":
12         x = float(input("Enter the length: "))
13         y = float(input("Enter the width: "))
14         area = calculate_area(shape, x, y)
15     elif shape == "square":
16         x = float(input("Enter the side length: "))
17         area = calculate_area(shape, x)
18     elif shape == "circle":
19         x = float(input("Enter the radius: "))
20         area = calculate_area(shape, x)
21     else:
22         print("Invalid shape.")
23     return
24     print("The area of the {shape} is {area}")
25
26 if __name__ == "__main__":
27     main()
```

OUTPUT;



```
Program Files\Python313\python3.13t.exe 'c:\Users\ashwi\.vscode\extensions\ms-python.debugpy\dlls\debugpy\launcher' '61361' '--' 'C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 13.3\lab 13.3 task 1.py'
Enter the shape (rectangle, square, circle): rectangle
Enter the length: 4
Enter the width: 5
Enter the length: 4
Enter the width: 5
Enter the width: 5
The area of the rectangle is 20.0
PS C:\Users\ashwi\Desktop\AIAC BATCH-02\lab 13.3>
```

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: REFACTOR THE CODE BY USING OPEN AND TRYEXCEPT PRINT THE OUTPUT AND THE USER ENTER THE INPUT

lab 13.3task-2.py X task3(1).py

C: > Users > MEGHANA > OneDrive > Documents > AIAC > ASSIGNMENT-13.3 > lab 13.3ta

```

1  def read_file(filename):
2      try:
3          with open(filename, "r") as f:
4              data = f.read()
5              return data
6      except FileNotFoundError:
7          print(f"Error: The file '{filename}' was not found.")
8          return None
9      except Exception as e:
10         print(f"An error occurred: {e}")
11         return None
12
13 if __name__ == "__main__":
14     filename = input("Enter the filename to read: ")
15     content = read_file(filename)
16     if content is not None:
17         print("File content:")
18         print(content)
19

```

Enter the filename to read: python
Error: The file 'python' was not found.
PS C:\Users\MEGHANA\OneDrive\Documents\AIAC\ASSIGNMENT-13.3> ^C

Problems Output Debug Console Terminal Ports

Enter the filename to read: lab 13.3task-2.py
File content:
def read_file(filename):
try:
with open(filename, "r") as f:
data = f.read()
return data
except FileNotFoundError:
print(f"Error: The file '{filename}' was not found.")
return None
except Exception as e:
File content:
def read_file(filename):
try:
with open(filename, "r") as f:
data = f.read()
return data
except FileNotFoundError:
print(f"Error: The file '{filename}' was not found.")
return None
except Exception as e:
try:
with open(filename, "r") as f:
data = f.read()
return data
except FileNotFoundError:
print(f"Error: The file '{filename}' was not found.")
return None
except Exception as e:
data = f.read()
return data
except FileNotFoundError:
print(f"Error: The file '{filename}' was not found.")
return None

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)
```

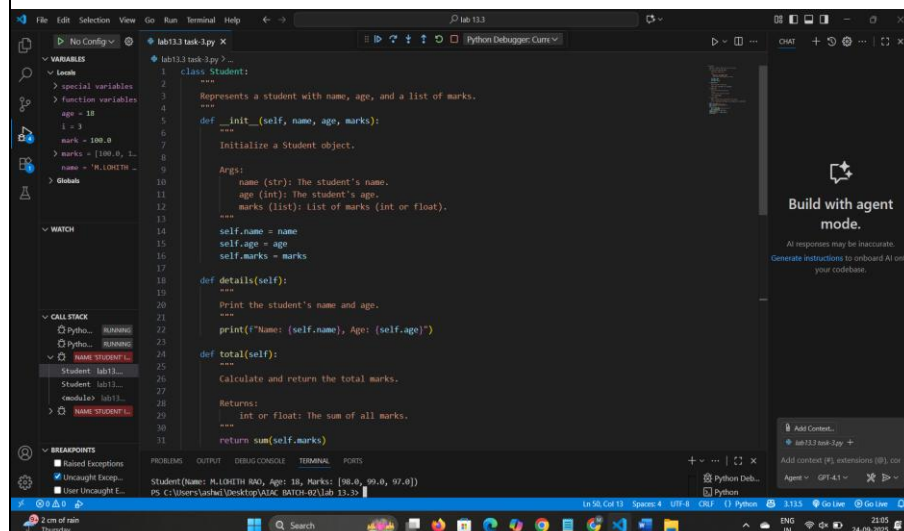
```
def total(self):
```

```
    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: REFACTOR THE CODE BY USING THE DOCSTRINGS
PRINT THE OUTPUT WHEN USER ENTERS THE INPUT



The screenshot shows a VS Code editor with a Python file named 'lab13.3 task 3.py'. The code defines a 'Student' class with an '.__init__' method that takes 'name', 'age', and 'marks' as arguments. The 'details' method prints the student's name and age. The 'total' method calculates and returns the sum of all marks. The code is refactored to use docstrings and a list for marks. The terminal window shows the output of the 'details' method: 'Student(Name: M.LONITH RAO, Age: 18, Marks: [88.0, 89.0, 97.0])'.

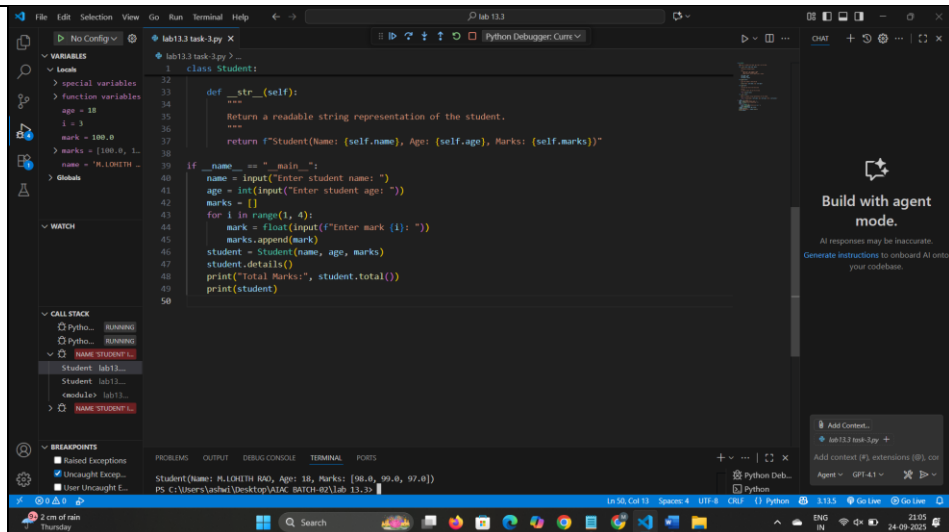
```
class Student:
    """
    Represents a student with name, age, and a list of marks.
    """
    def __init__(self, name, age, marks):
        """
        Initialize a Student object.
        """
        Args:
            name (str): The student's name.
            age (int): The student's age.
            marks (list): List of marks (int or float).
        """
        self.name = name
        self.age = age
        self.marks = marks

    def details(self):
        """
        Print the student's name and age.
        """
        print(f"Name: {self.name}, Age: {self.age}")

    def total(self):
        """
        Calculate and return the total marks.
        """
        Returns:
            int or float: The sum of all marks.
        """
        return sum(self.marks)
```

Terminal Output:

```
Student(Name: M.LONITH RAO, Age: 18, Marks: [88.0, 89.0, 97.0])
```



```
Enter student name: megha
Enter student age: 19
Enter mark 1: 98
Enter mark 2: 100
Enter mark 3: 97
Name: megha, Age: 19
Total Marks: 295.0
Student(Name: megha, Age: 19, Marks: [98.0, 100.0, 97.0])
PS C:\Users\MEGHANA\OneDrive\Documents\AIAC\ASSIGNMENT-13.3> ^C
```

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: REFACTOR THE CODE BY COMPLETING THE LOOP AND PRINT OUTPUT

