**ASSIGNMENT-13.2**

**NAME:-J.RAKESH**

**ENROLL:2403a51321**

**BATCH:13**

**------------------------------------------------------------------------------------------------------------------------TASK1:-** **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:- 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 rewrite the given code .**

**Code:-**

**A computer screen with colorful text

AI-generated content may be incorrect.**

**Output:-**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Observation:-**

* **Used x\*\*2 instead of x\*x**
* **Proper indentation for readability**
* **Used if else instead of if elif**

**------------------------------------------------------------------------------------------------------------------------TASK2:-**

**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:-**

**def read\_file(filename):  
f = open(filename, "r")  
data = f.read()**

**f.close()  
return data refactor the code using open() and try and except technique.**

**Code:-**

A screen shot of a computer program

AI-generated content may be incorrect.

**Output:-**

A computer screen with white text

AI-generated content may be incorrect.

**Observation:-**

* **Create a class student**
* **Used try and except error handling technique.**
* **Used try and except to read and write details**

**------------------------------------------------------------------------------------------------------------------------**

**TASK3:-**

**\*\*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:-**

**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 rewrite the code improved naming,readability and add docstrings.**

**CODE:-**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**OUTPUT:-**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Observation:-**

* **Added docstrings for readability**
* **Used descriptive variables for clear understanding of the code**

**TASK4:-** **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:-**

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

**squares.append(i \* i) fix the errors and rewrite the code with examples.**

**CODE:-**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Output:-**

**A number on a black background

AI-generated content may be incorrect.**

**Observation:-**

* **List comprehension is concise and fast for simple transformations**
* **the loop version is equivalent**
* **it makes the list of squares from 1 to 10.**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**