Task Description #1 – Remove Repetition

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 refactor the above code

```
import math

def calculate_area(shape, x, y=None):
    if shape == "rectangle":
        if y is not None:
            return x * y
        else:
            return "For a rectangle, please provide both length (x) and width (y)."
    elif shape == "square":
        return x * x
    elif shape == "circle":
        return math.pi * x * x
    else:
        return "Unsupported shape. Please use 'rectangle', 'square', or 'circle'."
```

OBSERVATION : here after refactoring the redundant code the refactored code contains more detailed structural format and uses dictionary based pre built functions

Task Description #2 – Error Handling in Legacy Code

PROMPT : def read_file(filename): f = open(filename, "r") data = f.read() f.close() return data rectify the code refactor the error

```
def read_file(filename):
    try:
        with open(filename, "r") as f:
            data = f.read()
        return data
    except FileNotFoundError:
        return f"Error: File '{filename}' not found."
    except Exception as e:
        return f"An error occurred: {e}"
```

OBSERVATION : Instead of open() fuction after refactoring the code try-except was replaced. previously with the refactor code we may get error if the file is not available then we may get error. After refactoring with try-except function we will not get any errors

Task Description #3 – Complex Refactoring

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 give the code after complex refactoring

```
class Student:
    def __init__(self, name, age, marks):
        # Basic data validation
        if not isinstance(name, str) or not name:
            raise ValueError("Name must be a non-empty string.")
        if not isinstance(age, (int, float)) or age <= 0:
            raise ValueError("Age must be a positive number.")
        if not isinstance(marks, list) or len(marks) != 3 or not all(isinstance(m, (int, float)) and 0 <= m <= 1
            raise ValueError("Marks must be a list of 3 numbers between 0 and 100.")

        self._name = name  # Use underscore for internal attribute
        self._age = age
        self._marks = marks
```

```
        @property
        def name(self):
            """Get the student's name."""
            return self._name

        @property
        def age(self):
            """Get the student's age."""
            return self._age

        @property
        def marks(self):
            """Get the student's marks as a list."""
            return self._marks

        @property
        def total_marks(self):
            """Calculate and return the total marks."""
            return sum(self._marks)

        def display_details(self):
            """Display student details."""
            print(f"Name: {self.name}, Age: {self.age}")

    # Example usage:
    try:
        student1 = Student("Alice", 17, [85, 90, 78])
        student1.display_details()
        print(f"Total Marks: {student1.total_marks}")

        # Example of invalid data
        # student2 = Student("Bob", -18, [95, 88]) # This will raise a ValueError
    except ValueError as e:
        print(f"Error creating student: {e}")
```

```
Name: Alice, Age: 17
Total Marks: 253
```

OBSERVATION : A student class method to display student details and calculate their total scores from the 3 subjects the output shows the student name and age followed by their calculated total score

Task Description #4 – Inefficient Loop Refactoring

PROMPT : nums = [1,2,3,4,5,6,7,8,9,10] squares = [] for i in nums: squares.append(i * i) refactor the above inefficient loop

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

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

OBSERVATION : here the for loop is running inside the list while getting stored in the list squares[] both the tasks are being done at a time

```
Start coding or generate with AI.
```