AI ASSISTED CODING:
ASSIGNMENT-10.1

M.KEERTHANA
2403A51259
BATCH-11
CSE-GEN

TASK1: Refactor Nested Conditionals

Task: Provide AI with the following nested conditional code and ask it to simplify and refactor for readability.

GIVEN CODE:

```
def discount(price, category):
    if category == "student":
        if price > 1000:
            return price * 0.9
        else:
            return price * 0.95
    else:
        if price > 2000:
            return price * 0.85
        else:
            return price
```

PROMPT:

simplify and refactor for readability.

REFACTORED CODE:

```
def discount(price, category):
    """Calculate discounted price based on category and price."""
    if category == "student":
        return price * (0.9 if price > 1000 else 0.95)
    elif price > 2000:
        return price * 0.85
    return price
```

TASK2:

Task: Give AI this messy loop and ask it to refactor and optimize.

GIVEN CODE:

```
def discount(price, category):
    """Calculate discounted price based on category and price."""
    if category == "student":
        return price * (0.9 if price > 1000 else 0.95)
    elif price > 2000:
        return price * 0.85
    return price
```

PROMPT:

simplify and refactor for readability

REFACTORED CODE:

```
def find_common(a, b):
    """Return a list of common elements between two iterables."""
    return list(set(a) & set(b))
```

TASK3:

Task: Provide this class with poor readability and ask AI to improve:

- Naming conventions
- Encapsulation
- Readability & maintainability

GIVEN CODE:

```
class emp:
    def __init__(self,n,s):
        self.n=n
        self.s=s
    def inc(self,p):
        self.s=self.s+(self.s*p/100)
    def pr(self):
        print("emp:",self.n,"salary:",self.s)
```

PROMPT:

simplify and refactor for readability

REFACTORED CODE:

```
class Employee:
    """Represents an employee with salary management methods."""

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary

    def increase_salary(self, percent):
        """Increase salary by given percentage."""
        self.salary += self.salary * percent / 100

    def display_info(self):
        """Print employee details."""
        print(f"Employee: {self.name}, Salary: {self.salary:.2f}")
```

TASK4:Modularize Long Function

Task: Give AI this long unstructured function and let it modularize into
smaller helper functions.

GIVEN CODE:

```
def process_scores(scores):
    total = 0
    for s in scores:
        total += s
    avg = total / len(scores)
    highest = scores[0]
    for s in scores:
        if s > highest:
            highest = s
    lowest = scores[0]
    for s in scores:
        if s < lowest:
            lowest = s
    print("Average:", avg)
    print("Highest:", highest)
    print("Lowest:", lowest)
```

PROMPT:

simplify and refactor for readability

REFACTORED CODE:

```python
def calculate_average(scores):
    """Return the average of the scores."""
    return sum(scores) / len(scores)

def find_highest(scores):
    """Return the highest score."""
    return max(scores)

def find_lowest(scores):
    """Return the lowest score."""
    return min(scores)

def process_scores(scores):
    """Process scores and print statistics."""
    avg = calculate_average(scores)
    highest = find_highest(scores)
    lowest = find_lowest(scores)

    print(f"Average: {avg:.2f}")
    print(f"Highest: {highest}")
    print(f"Lowest: {lowest}")
```

TASK5:

Code Review on Error Handling

Task: Provide AI with this faulty code and ask it to improve error handling, naming, and readability.

GIVEN CODE:

```python
def div(a,b):
    return a/b
print(div(10,0))
```

PROMPT:

simplify and refactor for readability

REFACTORED CODE:

```
def divide_numbers(a, b):
    """
    Divide two numbers with error handling.

    Returns:
        float: result of division if valid
        str: error message if division by zero occurs
    """
    try:
        return a / b
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."

print(divide_numbers(10, 0))
```

TASK6:Complexity Reduction

Task: Use AI to simplify overly complex logic.

GIVEN CODE:

```
def is_even(num):
    if num % 2 == 0:
        return True
    else:
        return False
```

PROMPT:

simplify and refactor for readability

REFACTORED CODE:

```
def is_even(num):
    """Check if a number is even."""
    return num % 2 == 0
```