Task 1: AI-Powered Code Completion

Tool Used: GitHub Copilot (AI code completion)

Objective: Write a Python function to sort a list of dictionaries by a specific key, compare the AI-suggested code with a manually written version, and analyze which is more efficient and why.

Manual Implementation

```
# Manual Python function to sort a list of dictionaries by a key
def sort_list_of_dicts(data, key):
    """
    Sort a list of dictionaries by a specified key.
    :param data: list of dictionaries
    :param key: key to sort by
    :return: sorted list of dictionaries
    """
    return sorted(data, key=lambda x: x[key])

# Example usage
data = [{'name': 'Alice', 'age': 25}, {'name': 'Bob', 'age': 19}, {'name': 'Charlie',
sorted_data = sort_list_of_dicts(data, 'age')
print(sorted_data)
```

AI-Suggested Implementation (GitHub Copilot)

```
# GitHub Copilot Suggested Function
def sort_dicts(data, key):
    try:
        return sorted(data, key=lambda item: item.get(key, 0))
    except Exception as e:
        print("Error:", e)
        return data
```

200-Word Analysis

The manual function provides a simple, efficient, and readable approach to sorting dictionaries by a given key. It leverages Python’s built-in sorted() function with a lambda expression, resulting in O(n log n) complexity. This approach works perfectly for well-structured data where all dictionaries contain the target key. The AI-suggested version by GitHub Copilot demonstrates greater fault tolerance. It includes error handling with a try-except block and uses the .get() method to avoid key errors, returning a default value when the key is missing. This makes it more robust when handling inconsistent or incomplete data. In terms of performance, both versions are computationally similar. However, the Copilot version sacrifices a bit of readability for resilience. While the manual code is ideal for controlled data, Copilot’s code is preferable in real-world, noisy datasets where exceptions might occur. Overall, the AI-generated code shows how AI tools can enhance developer productivity, introduce safer coding practices, and accelerate software creation — though human review remains essential for logic validation and optimization.