🧠 AI-Powered Code Completion

Task 1 – Week 4 Assignment: AI in Software Engineering

🎯 Task Overview

Write a Python function to sort a list of dictionaries by a specific key. Compare the AI-suggested code (e.g., using GitHub Copilot) with a manually written version. Analyze which version is more efficient and why.

✍️ Manual Implementation

In [1]:
```python
def sort_dicts_by_key(data, key):
    return sorted(data, key=lambda x: x.get(key, ''))

# Example usage
people = [
    {"name": "Alice", "age": 30},
    {"name": "Bob", "age": 25},
    {"name": "Charlie", "age": 35}
]

sorted_people = sort_dicts_by_key(people, "age")
print(sorted_people)
```

```
[{'name': 'Bob', 'age': 25}, {'name': 'Alice', 'age': 30}, {'name': 'Charlie', 'ag
e': 35}]
```

🤖 AI-Suggested Code (GitHub Copilot)

In [2]:
```python
def sort_dicts_by_key(dict_list, sort_key):
    return sorted(dict_list, key=lambda d: d[sort_key])

# Example usage
people = [
    {"name": "Alice", "age": 30},
    {"name": "Bob", "age": 25},
    {"name": "Charlie", "age": 35}
]

sorted_people = sort_dicts_by_key(people, "age")
print(sorted_people)
```

```
[{'name': 'Bob', 'age': 25}, {'name': 'Alice', 'age': 30}, {'name': 'Charlie', 'ag
e': 35}]
```

📊 Analysis & Comparison (200 Words)

Both the manual and AI-generated functions aim to sort a list of dictionaries by a given key, but they differ slightly in error handling and generality.

The AI-generated version:

```
def sort_dicts_by_key(dict_list, sort_key):
    return sorted(dict_list, key=lambda d: d[sort_key])
```

is more concise but assumes all dictionaries contain the specified key. If a key is missing, it raises a `KeyError`, which might crash the program.

The manual version:

```
def sort_dicts_by_key(data, key):
    return sorted(data, key=lambda x: x.get(key, ''))
```

is more robust. It uses `dict.get()` with a default fallback value, preventing crashes when some dictionaries lack the sorting key. This makes it more suitable for real-world data where missing fields are common.

In terms of efficiency, both use Python's built-in `sorted()` with O(n log n) complexity. However, the manual version trades a tiny bit of performance for better safety and flexibility.

Conclusion: The AI-generated code is efficient for clean, uniform data, while the manual version is safer for unpredictable input. Depending on your use case, one may be more appropriate than the other.