

## Performance Comparison and Documentation

### Efficiency Analysis:

Criteria	Copilot Version	Manual Version
Readability	✓ Very clean and Pythonic	✗ Less readable
Performance	✓ Uses optimized built-in sort	✗ $O(n^2)$ complexity
Edge-case safe	✓ Handles missing keys gracefully	✓ Also uses <code>.get()</code> with default
Developer effort	✓ Requires only 1 comment prompt	✗ Manual coding, debugging needed

---

GitHub Copilot's suggestion for sorting a list of dictionaries by a given key was concise, efficient, and Pythonic. It used Python's built-in `sorted()` function with a lambda function for key access, resulting in clean and performant code. Copilot's ability to understand intent from a natural-language comment (`# Function to sort...`) demonstrates its strength in rapid prototyping and eliminating boilerplate logic.

In contrast, the manual implementation used a bubble sort approach, which, while logically correct, is inefficient ( $O(n^2)$  time complexity) and more verbose. This version was included for learning purposes but would not scale well with large datasets.

The AI-suggested code not only required less effort but also adhered to Python best practices, showcasing how such tools can significantly reduce development time and increase reliability — especially for common utility functions.

However, it's important to validate AI-suggested code to ensure correctness in more complex scenarios. Developers should use these

tools as assistants, not as blind replacements for understanding algorithms.

```
sort_dicts.py X
> Users > ochie > Downloads > sort_dicts.py > ...
1  def sort_dicts_by_key(list_of_dicts, sort_key, reverse=False):
2      """
3      Sorts a list of dictionaries by a specified key.
4
5      Args:
6          list_of_dicts (list): List of dictionaries to sort.
7          sort_key (str): The key to sort the dictionaries by.
8          reverse (bool): Whether to sort in descending order. Default is False (ascending).
9
10     Returns:
11         list: A new list of dictionaries sorted by the specified key.
12     """
13     return sorted(list_of_dicts, key=lambda x: x.get(sort_key, None), reverse=reverse)
14
15 # Example usage:
16 if __name__ == "__main__":
17     data = [
18         {"name": "Alice", "age": 30},
19         {"name": "Bob", "age": 25},
20         {"name": "Charlie", "age": 35}
21     ]
22     sorted_data = sort_dicts_by_key(data, "age")
23     print(sorted_data)
24
25 def manual_sort_dicts_by_key(dict_list, key):
26     for i in range(len(dict_list)):
27         for j in range(i + 1, len(dict_list)):
28             if dict_list[i].get(key, '') > dict_list[j].get(key, ''):
29                 dict_list[i], dict_list[j] = dict_list[j], dict_list[i]
30     return dict_list
31
32 sample = [
```

Ln 43, Col 1 Spaces: 4 UTF-8 CRI