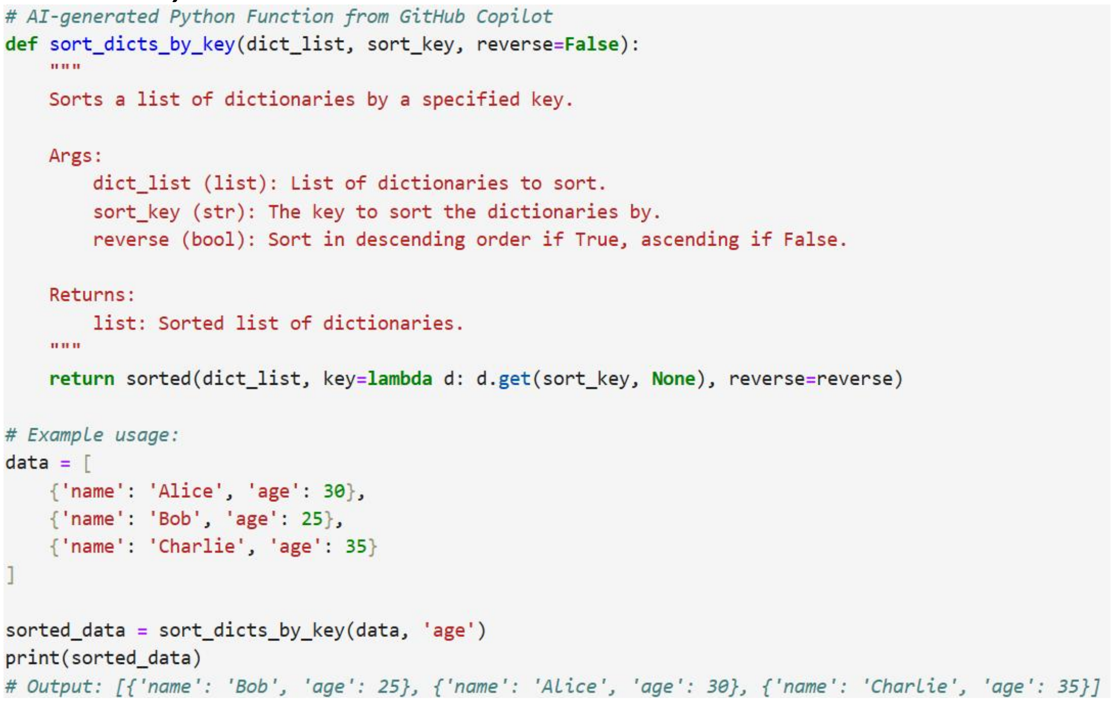
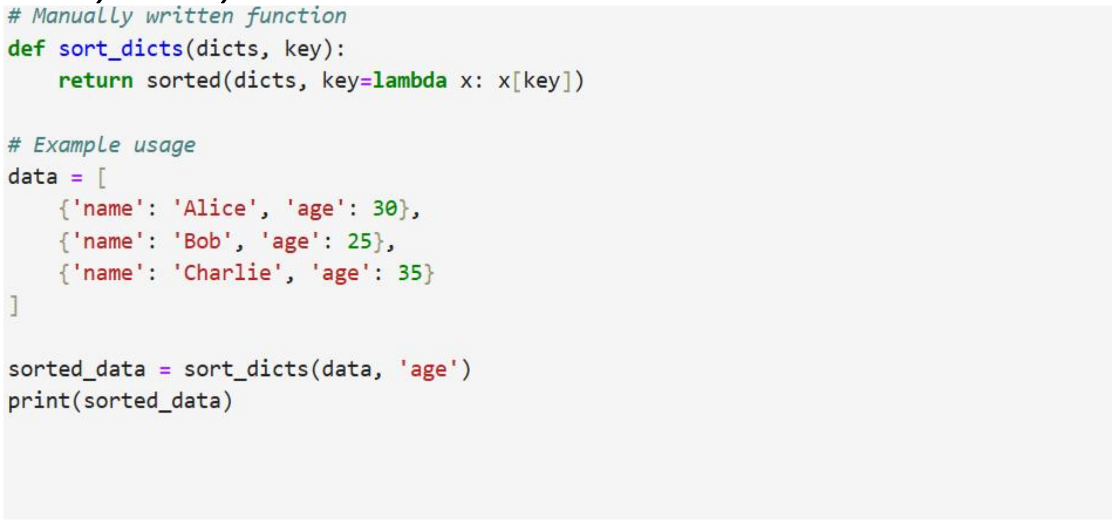
TASK 1: AI-Powered Code Completion

AI-Generated Python Function



```python
# AI-generated Python Function from GitHub Copilot
def sort_dicts_by_key(dict_list, sort_key, reverse=False):
    """
    Sorts a list of dictionaries by a specified key.

    Args:
        dict_list (list): List of dictionaries to sort.
        sort_key (str): The key to sort the dictionaries by.
        reverse (bool): Sort in descending order if True, ascending if False.

    Returns:
        list: Sorted list of dictionaries.
    """
    return sorted(dict_list, key=lambda d: d.get(sort_key, None), reverse=reverse)

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

sorted_data = sort_dicts_by_key(data, 'age')
print(sorted_data)
# Output: [{'name': 'Bob', 'age': 25}, {'name': 'Alice', 'age': 30}, {'name': 'Charlie', 'age': 35}]
```

Manually Written Python Function



```python
# Manually written function
def sort_dicts(dicts, key):
    return sorted(dicts, key=lambda x: x[key])

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

sorted_data = sort_dicts(data, 'age')
print(sorted_data)
```

Both the AI-generated and the manually written Python functions are designed to sort a list of dictionaries based on a given key, and while they produce the same result in ideal conditions, they differ significantly in their robustness and flexibility. The **AI-generated code** incorporates **.get()** to access dictionary values safely, which minimizes the risk of raising exceptions if the specified key is missing from any dictionary. This practice makes the function more fault-tolerant in real-world scenarios where data might be incomplete or inconsistent. Additionally, the AI-generated code includes a **reverse parameter** that gives users control over sorting order without modifying the core logic, and it's well-documented with a **descriptive docstring**—making the function easier to understand and reuse.

On the other hand, the **manual implementation** is more **concise and straightforward**, which is ideal for quick tasks or when working with clean and predictable datasets. It assumes the presence of the target key in all dictionaries and **lacks parameterization** or error handling. This simplicity can benefit beginners who are focusing on core logic without the complexity of additional safeguards. However, it comes with the drawback that it may raise a **KeyError** if the required key is missing.

In summary, while both approaches serve the same purpose, the **AI-generated code** is more robust, readable, and production-ready. It reflects best practices for writing reusable and defensive code, especially in dynamic or uncertain environments. The **manual version**, though elegant and efficient, is best suited for controlled situations where data integrity can be guaranteed.