NAME : B.SRISHANTH ROLL NO : 2403A510G3

SUBJECT : AI ASSISTED CODING ASSIGNMENT : 12.1

Task Description #1 (Sorting – Merge Sort Implementation)

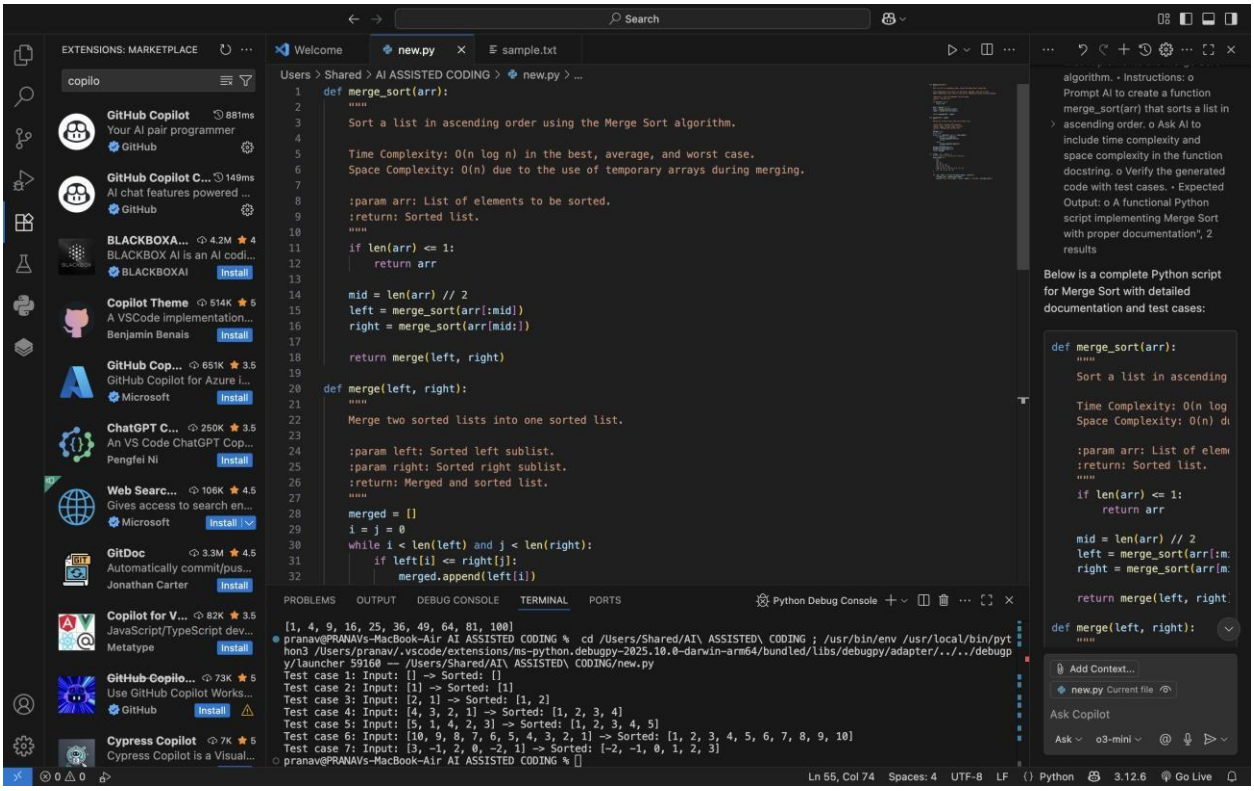
- Task: Use AI to generate a Python program that implements the Merge Sort algorithm.
- Instructions:
  - Prompt AI to create a function merge_sort(arr) that sorts a list in ascending order.
  - Ask AI to include time complexity and space complexity in the function docstring.
  - Verify the generated code with test cases.
- Expected Output:
  - A functional Python script implementing Merge Sort with proper documentation

PROMPT USED :

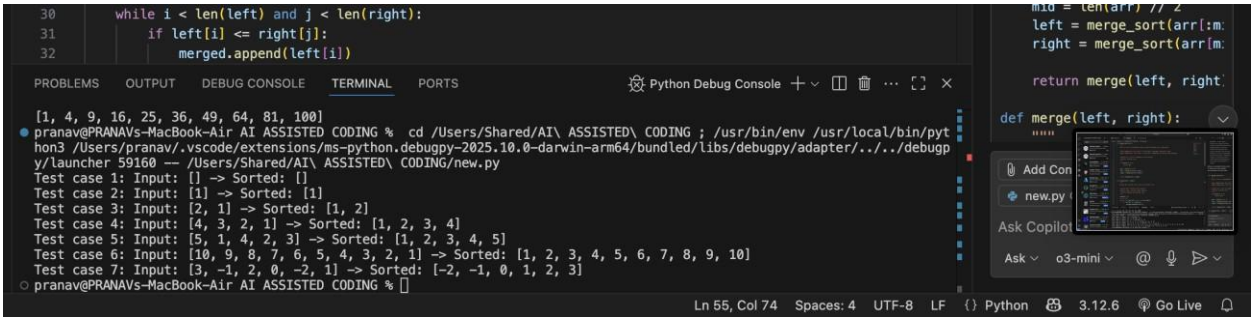
To generate a python program and to implement the merge sort algorithm

To sort in ascending order

CODE GENERATED :



OUTPUT :



Task Description #2 (Searching – Binary Search with AI Optimization)

• Task: Use AI to create a binary search function that finds a target element in a sorted list.

• Instructions:

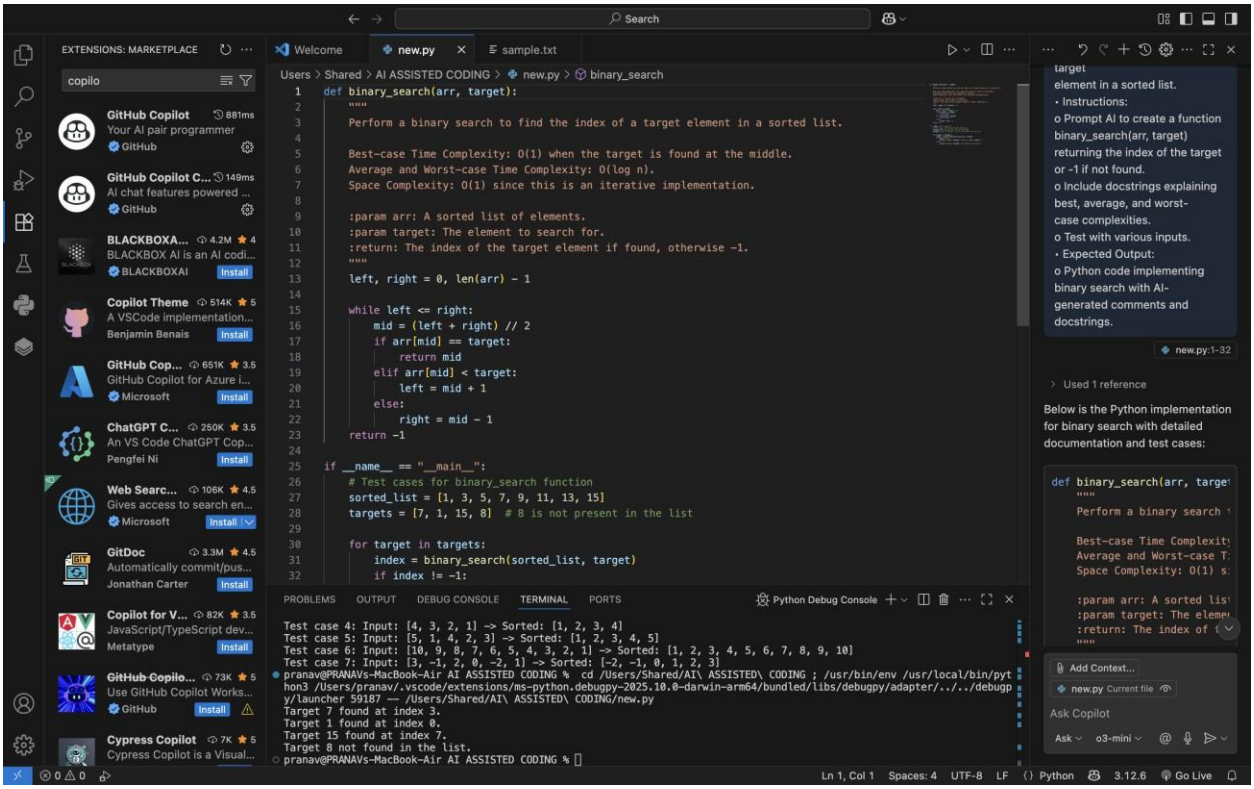
o Prompt AI to create a function binary_search(arr, target) returning the index of the target or -1 if not found.

o Include docstrings explaining best, average, and worst-case complexities.
o Test with various inputs.
• Expected Output:
o Python code implementing binary search with AI-generated comments and docstrings.

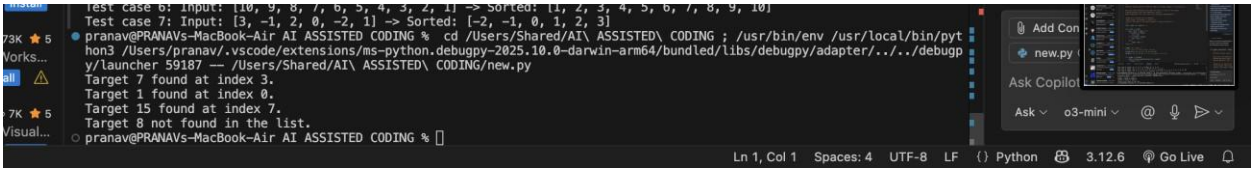
PROMPT USED :

To create a binary search function that finds a target element in a sorted list

CODE GENERATED :



OUTPUT :



Task Description #3 (Real-Time Application – Inventory Management System)

• Scenario: A

retail store's inventory system contains thousands of products, each with attributes like product ID, name, price, and stock quantity. Store staff need to:

1. Quickly search for a product by ID or name.
2. Sort products by price or quantity for stock analysis.

• Task:

o Use AI to suggest the most efficient search and sort

algorithms for this use case.

o Implement the recommended algorithms in Python.

o Justify the choice based on dataset size, update frequency, and performance requirements.

• Expected Output:

o A table mapping operation → recommended algorithm → justification.

o Working Python functions for searching and sorting the inventory.

Deliverables (For All Tasks)

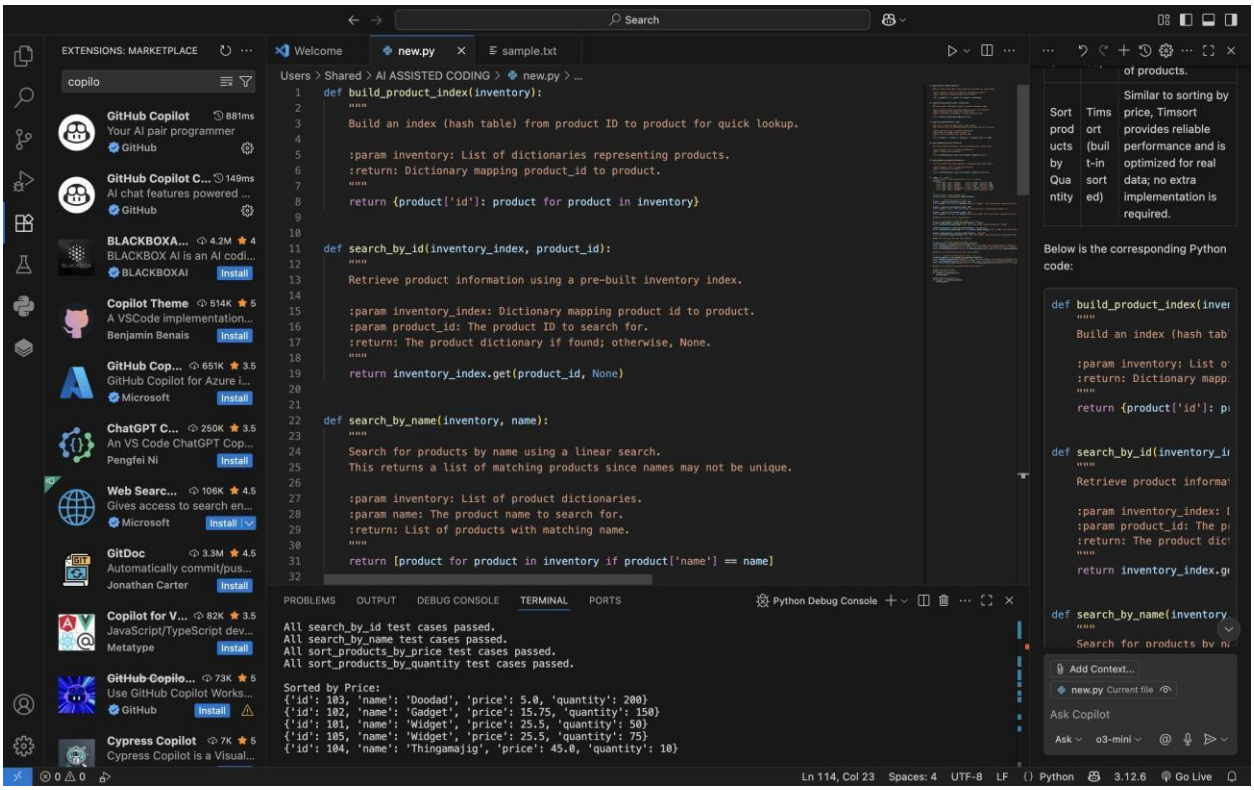
1. AI-generated prompts for code and test case generation.
2. At least 3 assert test cases for each task.
3. AI-generated initial code and execution screenshots.
4. Analysis of whether code passes all tests.
5. Improved final version with inline comments and explanation.
6. Compiled report (Word/PDF) with prompts, test cases, assertions, code, and output.

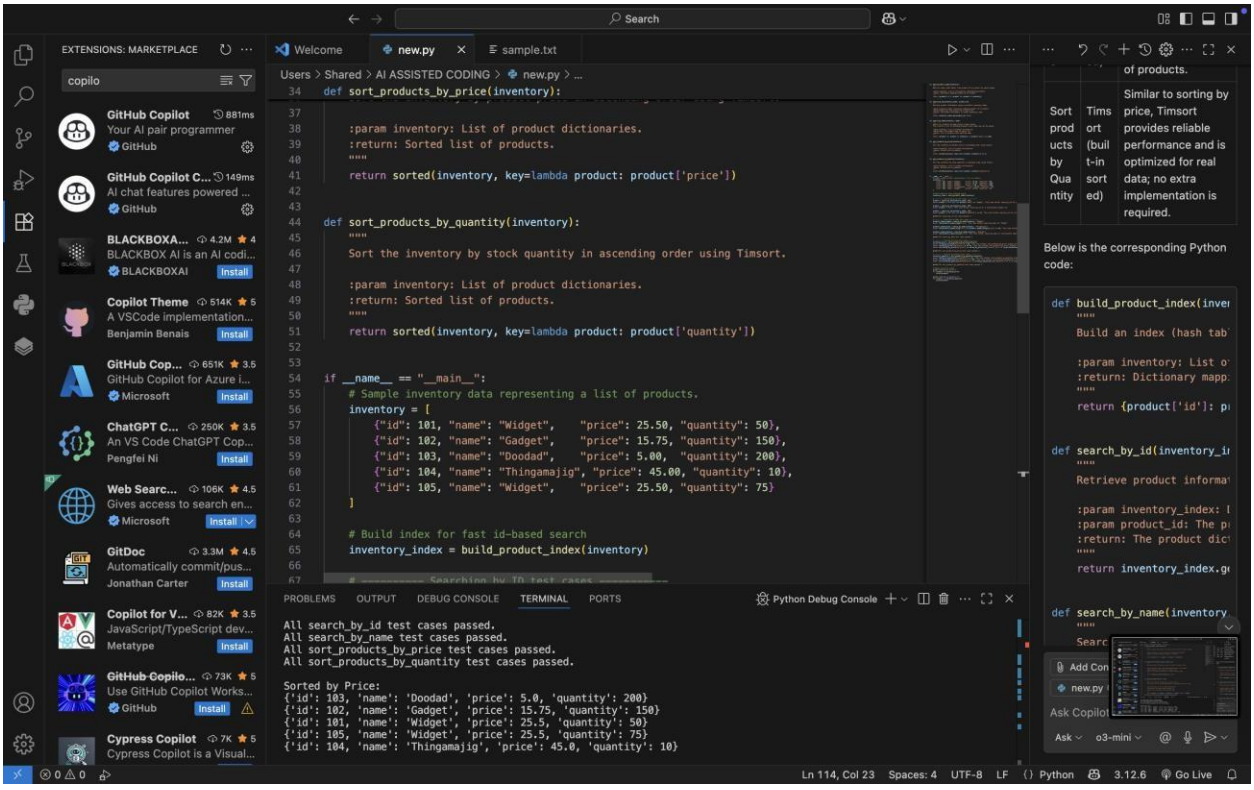
PROMPT USED :

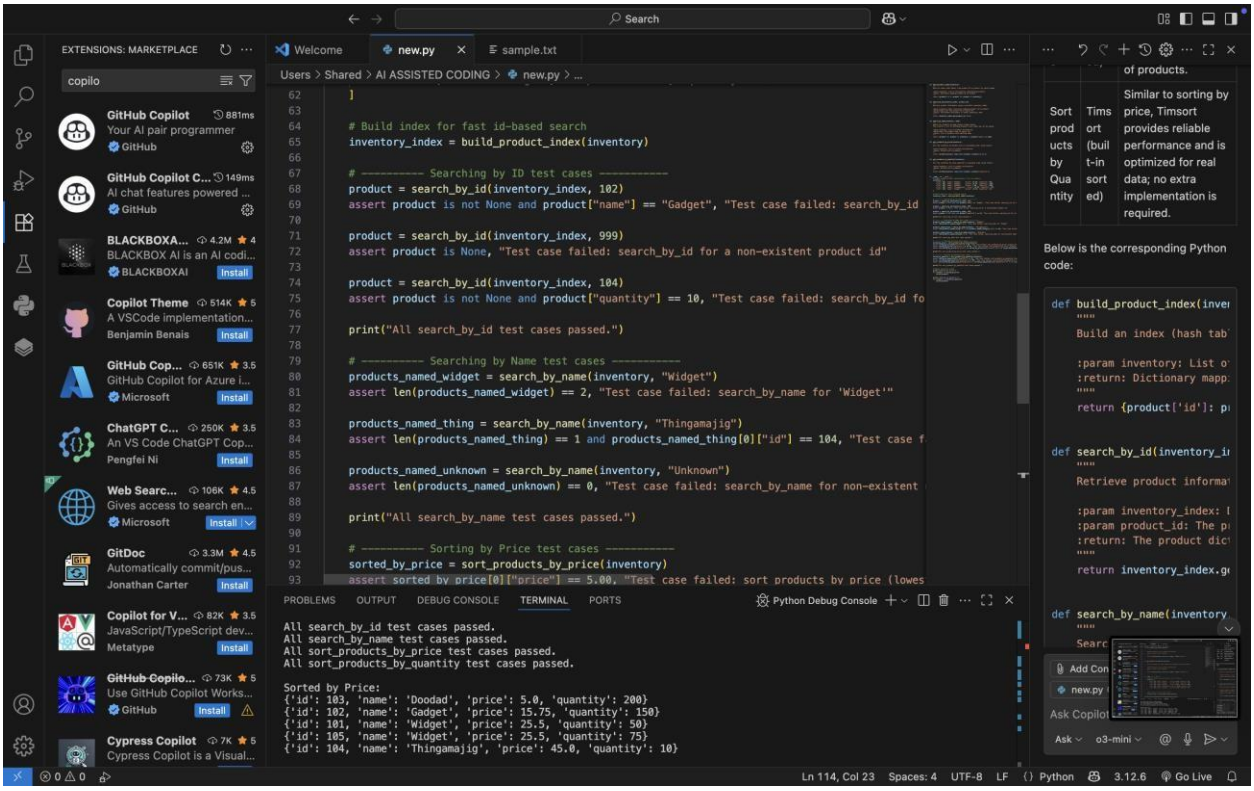
To write the code for a retail store's inventory system contains thousands of products, each with attributes like product ID, name, price, and stock quantity. Store staff need to:

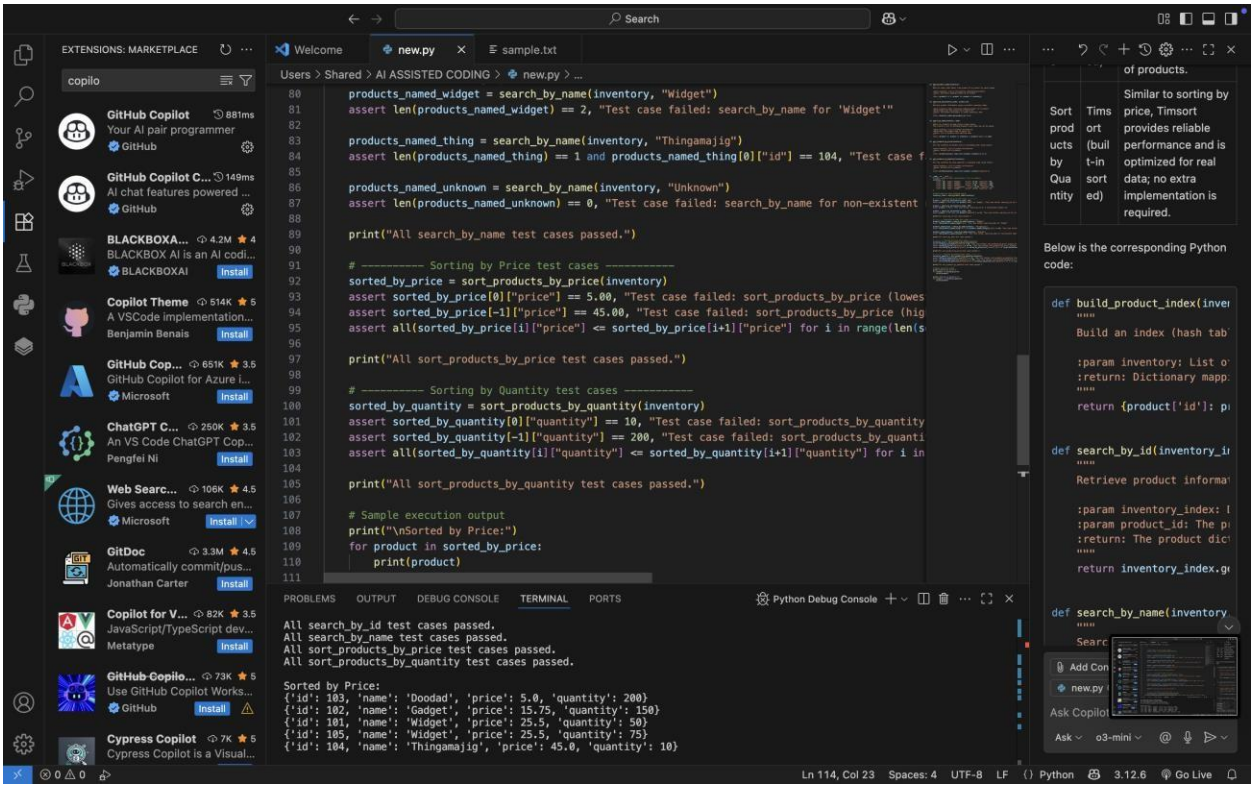
1. Quickly search for a product by ID or name.
2. Sort products by price or quantity for stock analysis.

CODE GENERATED :

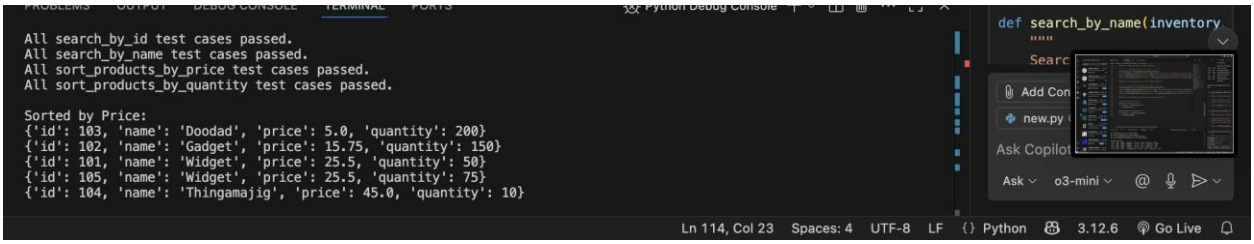


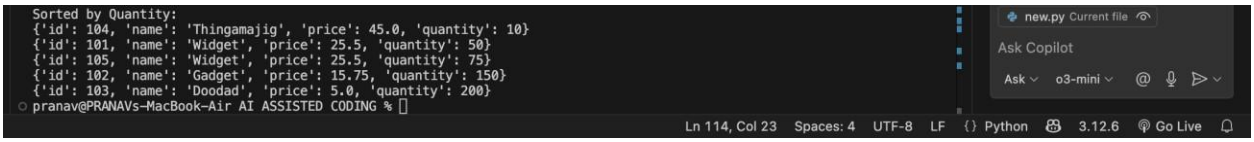



Search
EXTENSIONS: MARKETPLACE
copilo
GitHub Copilot 881ms
Your AI pair programmer
GitHub
GitHub Copilot C... 149ms
AI chat features powered ...
GitHub
BLACKBOXA... 4.2M 4
BLACKBOX AI is an AI codi...
BLACKBOXAI Install
Copilot Theme 514K 5
A VSCode implementation...
Benjamin Benais Install
GitHub Cop... 651K 3.5
GitHub Copilot for Azure i...
Microsoft Install
ChatGPT C... 250K 3.5
An VS Code ChatGPT Cop...
Pengfei Ni Install
Web Searc... 106K 4.5
Gives access to search en...
Microsoft Install
GitDoc 3.3M 4.5
Automatically commit/pus...
Jonathan Carter Install
Copilot for V... 82K 3.5
JavaScript/TypeScript dev...
Metatype Install
GitHub Copilo... 73K 5
Use GitHub Copilot Works...
GitHub Install
Cypress Copilot 7K 5
Cypress Copilot is a Visual...
Welcome
new.py
sample.txt
Users > Shared > AI ASSISTED CODING > new.py > ...
# Build index for fast id-based search
inventory_index = build_product_index(inventory)
# ---------- Searching by ID test cases ----------
product = search_by_id(inventory_index, 102)
assert product is not None and product["name"] == "Gadget", "Test case failed: search_by_id
product = search_by_id(inventory_index, 999)
assert product is None, "Test case failed: search_by_id for a non-existent product id"
product = search_by_id(inventory_index, 104)
assert product is not None and product["quantity"] == 10, "Test case failed: search_by_id fo
print("All search_by_id test cases passed.")
# ---------- Searching by Name test cases ----------
products_named_widget = search_by_name(inventory, "Widget")
assert len(products_named_widget) == 2, "Test case failed: search_by_name for 'Widget'"
products_named_thing = search_by_name(inventory, "Thingamajig")
assert len(products_named_thing) == 1 and products_named_thing[0]["id"] == 104, "Test case f
products_named_unknown = search_by_name(inventory, "Unknown")
assert len(products_named_unknown) == 0, "Test case failed: search_by_name for non-existent
print("All search_by_name test cases passed.")
# ---------- Sorting by Price test cases ----------
sorted_by_price = sort_products_by_price(inventory)
assert sorted_by_price[0]["price"] == 5.00, "Test case failed: sort_products_by_price (lowes
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python Debug Console
All search_by_id test cases passed.
All search_by_name test cases passed.
All sort_products_by_price test cases passed.
All sort_products_by_quantity test cases passed.
Sorted by Price:
{'id': 103, 'name': 'Doodad', 'price': 5.0, 'quantity': 200}
{'id': 102, 'name': 'Gadget', 'price': 15.75, 'quantity': 150}
{'id': 101, 'name': 'Widget', 'price': 25.5, 'quantity': 50}
{'id': 105, 'name': 'Widget', 'price': 25.5, 'quantity': 75}
{'id': 104, 'name': 'Thingamajig', 'price': 45.0, 'quantity': 10}
of products.
Sort products by Quantity
Timsort (built-in sorted)
Similar to sorting by price, Timsort provides reliable performance and is optimized for real data; no extra implementation is required.
Below is the corresponding Python code:
def build_product_index(inve
"""
Build an index (hash tab
:param inventory: List o
:return: Dictionary mapp
"""
return {product['id']: p
def search_by_id(inventory_i
"""
Retrieve product informa
:param inventory_index: I
:param product_id: The p
:return: The product dic
"""
return inventory_index.g
def search_by_name(inventory
"""
Searc
Add Con
new.py
Ask Copilot
Ask
o3-mini
Ln 114, Col 23
Spaces: 4
UTF-8
LF
Python
3.12.6
Go Live


Search
EXTENSIONS: MARKETPLACE
copilo
GitHub Copilot 881ms
Your AI pair programmer
GitHub
GitHub Copilot C... 149ms
AI chat features powered ...
GitHub
BLACKBOXA... 4.2M 4
BLACKBOX AI is an AI codi...
BLACKBOXAI Install
Copilot Theme 514K 5
A VSCode implementation...
Benjamin Benais Install
GitHub Cop... 651K 3.5
GitHub Copilot for Azure i...
Microsoft Install
ChatGPT C... 250K 3.5
An VS Code ChatGPT Cop...
Pengfei Ni Install
Web Searc... 106K 4.5
Gives access to search en...
Microsoft Install
GitDoc 3.3M 4.5
Automatically commit/pus...
Jonathan Carter Install
Copilot for V... 82K 3.5
JavaScript/TypeScript dev...
Metatype Install
GitHub Copilo... 73K 5
Use GitHub Copilot Works...
GitHub Install
Cypress Copilot 7K 5
Cypress Copilot is a Visual...
Welcome
new.py
sample.txt
Users > Shared > AI ASSISTED CODING > new.py > ...
products_named_widget = search_by_name(inventory, "Widget")
assert len(products_named_widget) == 2, "Test case failed: search_by_name for 'Widget'"
products_named_thing = search_by_name(inventory, "Thingamajig")
assert len(products_named_thing) == 1 and products_named_thing[0]["id"] == 104, "Test case f
products_named_unknown = search_by_name(inventory, "Unknown")
assert len(products_named_unknown) == 0, "Test case failed: search_by_name for non-existent
print("All search_by_name test cases passed.")
# ---------- Sorting by Price test cases ----------
sorted_by_price = sort_products_by_price(inventory)
assert sorted_by_price[0]["price"] == 5.00, "Test case failed: sort_products_by_price (lowes
assert sorted_by_price[-1]["price"] == 45.00, "Test case failed: sort_products_by_price (hig
assert all(sorted_by_price[i]["price"] <= sorted_by_price[i+1]["price"] for i in range(len(s
print("All sort_products_by_price test cases passed.")
# ---------- Sorting by Quantity test cases ----------
sorted_by_quantity = sort_products_by_quantity(inventory)
assert sorted_by_quantity[0]["quantity"] == 10, "Test case failed: sort_products_by_quantity
assert sorted_by_quantity[-1]["quantity"] == 200, "Test case failed: sort_products_by_quanti
assert all(sorted_by_quantity[i]["quantity"] <= sorted_by_quantity[i+1]["quantity"] for i in
print("All sort_products_by_quantity test cases passed.")
# Sample execution output
print("\nSorted by Price:")
for product in sorted_by_price:
print(product)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python Debug Console
All search_by_id test cases passed.
All search_by_name test cases passed.
All sort_products_by_price test cases passed.
All sort_products_by_quantity test cases passed.
Sorted by Price:
{'id': 103, 'name': 'Doodad', 'price': 5.0, 'quantity': 200}
{'id': 102, 'name': 'Gadget', 'price': 15.75, 'quantity': 150}
{'id': 101, 'name': 'Widget', 'price': 25.5, 'quantity': 50}
{'id': 105, 'name': 'Widget', 'price': 25.5, 'quantity': 75}
{'id': 104, 'name': 'Thingamajig', 'price': 45.0, 'quantity': 10}
of products.
Sort products by Quantity
Timsort (built-in sorted)
Similar to sorting by price, Timsort provides reliable performance and is optimized for real data; no extra implementation is required.
Below is the corresponding Python code:
def build_product_index(inve
"""
Build an index (hash tab
:param inventory: List o
:return: Dictionary mapp
"""
return {product['id']: p
def search_by_id(inventory_i
"""
Retrieve product informa
:param inventory_index: I
:param product_id: The p
:return: The product dic
"""
return inventory_index.g
def search_by_name(inventory
"""
Searc
Add Con
new.py
Ask Copilot
Ask
o3-mini
Ln 114, Col 23
Spaces: 4
UTF-8
LF
Python
3.12.6
Go Live

OUTPUT :


All search_by_id test cases passed.
All search_by_name test cases passed.
All sort_products_by_price test cases passed.
All sort_products_by_quantity test cases passed.
Sorted by Price:
{'id': 103, 'name': 'Doodad', 'price': 5.0, 'quantity': 200}
{'id': 102, 'name': 'Gadget', 'price': 15.75, 'quantity': 150}
{'id': 101, 'name': 'Widget', 'price': 25.5, 'quantity': 50}
{'id': 105, 'name': 'Widget', 'price': 25.5, 'quantity': 75}
{'id': 104, 'name': 'Thingamajig', 'price': 45.0, 'quantity': 10}
def search_by_name(inventory
Add Con
new.py
Ask Copilot
Ask
o3-mini
Ln 114, Col 23
Spaces: 4
UTF-8
LF
Python
3.12.6
Go Live


Sorted by Quantity:
{'id': 104, 'name': 'Thingamajig', 'price': 45.0, 'quantity': 10}
{'id': 101, 'name': 'Widget', 'price': 25.5, 'quantity': 50}
{'id': 105, 'name': 'Widget', 'price': 25.5, 'quantity': 75}
{'id': 102, 'name': 'Gadget', 'price': 15.75, 'quantity': 150}
{'id': 103, 'name': 'Doodad', 'price': 5.0, 'quantity': 200}
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
new.py Current file
Ask Copilot
Ask
o3-mini
Ln 114, Col 23
Spaces: 4
UTF-8
LF
Python
3.12.6
Go Live