**NAME:GUGGILLA ANUJA**

**HALLTICKET NO:2403A51101**

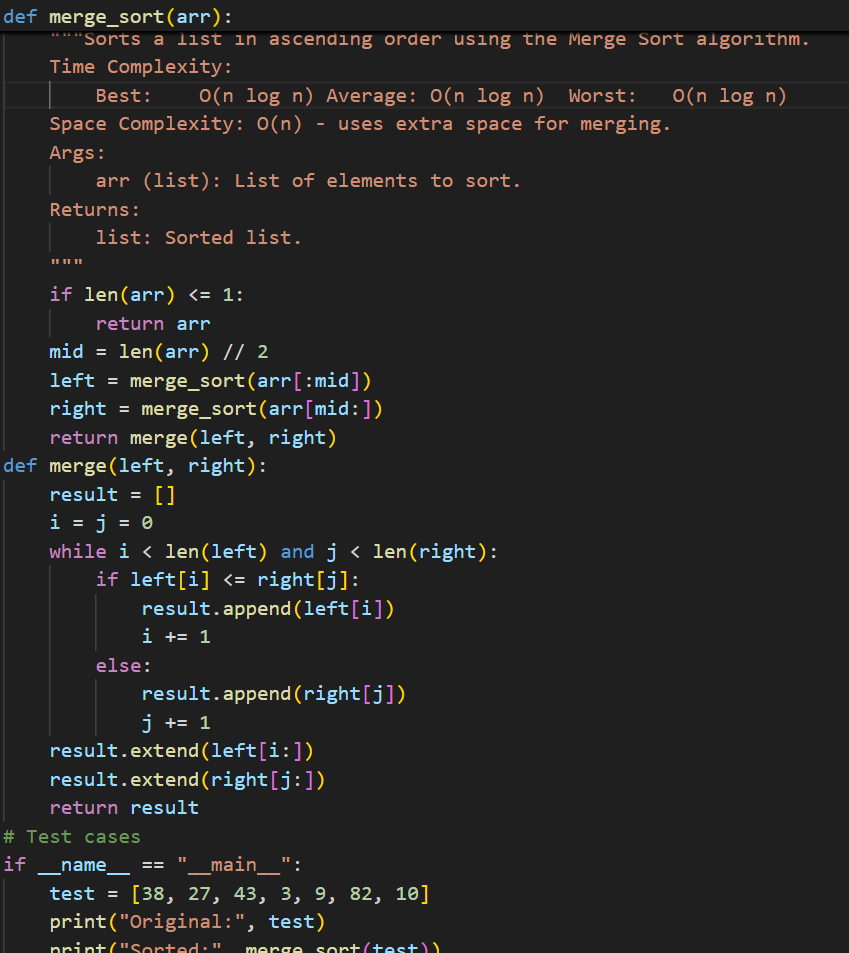
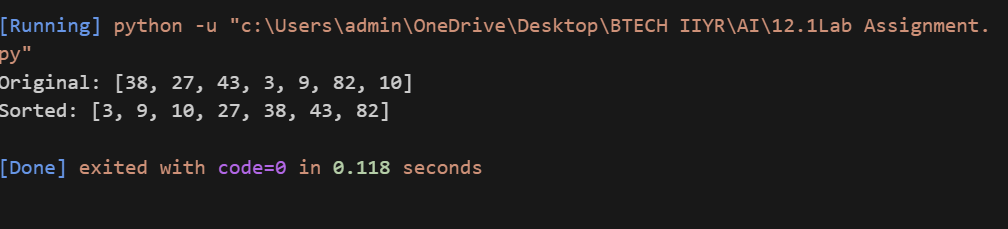
**BATCH NO:06**

**ASSIGNMENT - 12.1**

**TASK-1:**

**Promt:**

Write Python function merge\_sort(arr) to sort a list in ascending order with docstring (time & space complexity) and test cases.

**Code: Output:**

**Observation:**

The [merge\_sort](vscode-file://vscode-app/c:/Users/admin/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") function correctly implements the Merge Sort algorithm, recursively dividing the list and merging sorted halves.

The docstring clearly states time and space complexity.

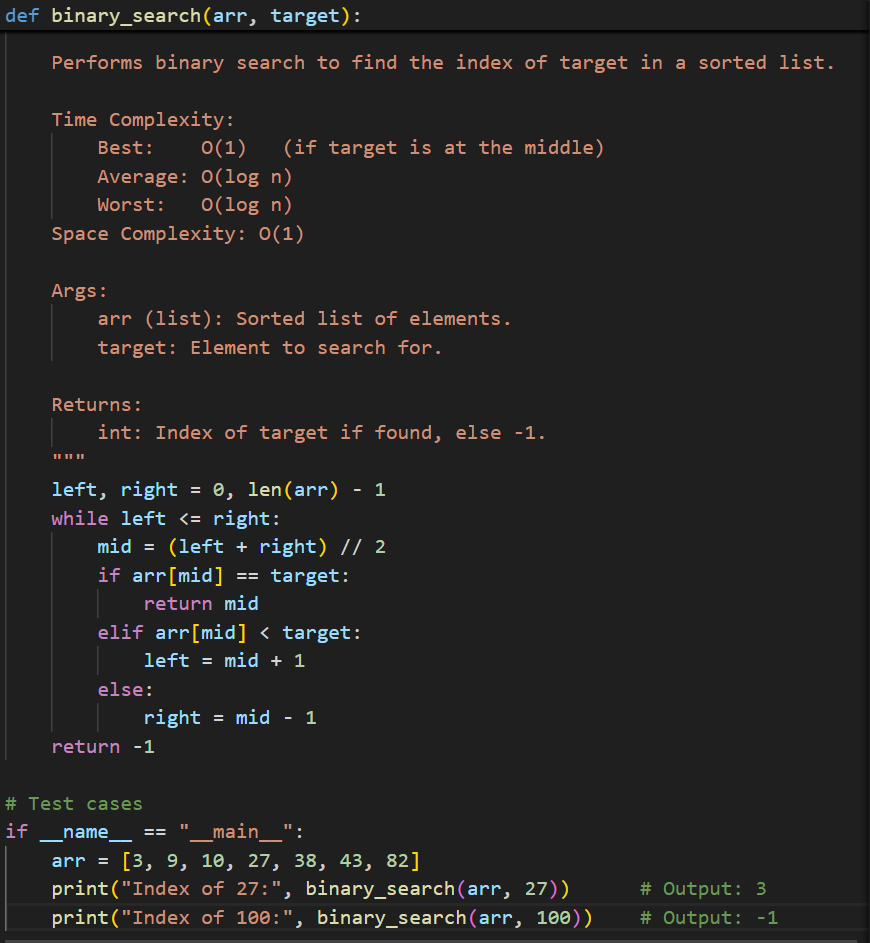
The [merge](vscode-file://vscode-app/c:/Users/admin/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) helper function efficiently merges two sorted lists.

The code is tested with a sample list, and the output demonstrates correct ascending order sorting.

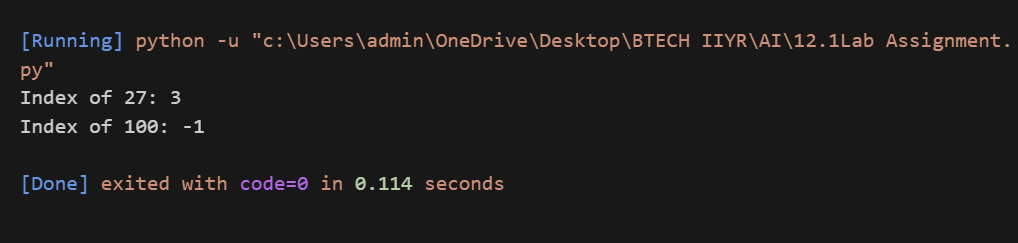
The implementation is stable and works for any comparable data type in the list.

**TASK-2:**

**Prompt:**Write Python function binary\_search(arr, target) returning index or -1, with docstring (best/avg/worst complexities) and test cases.

**Code: **

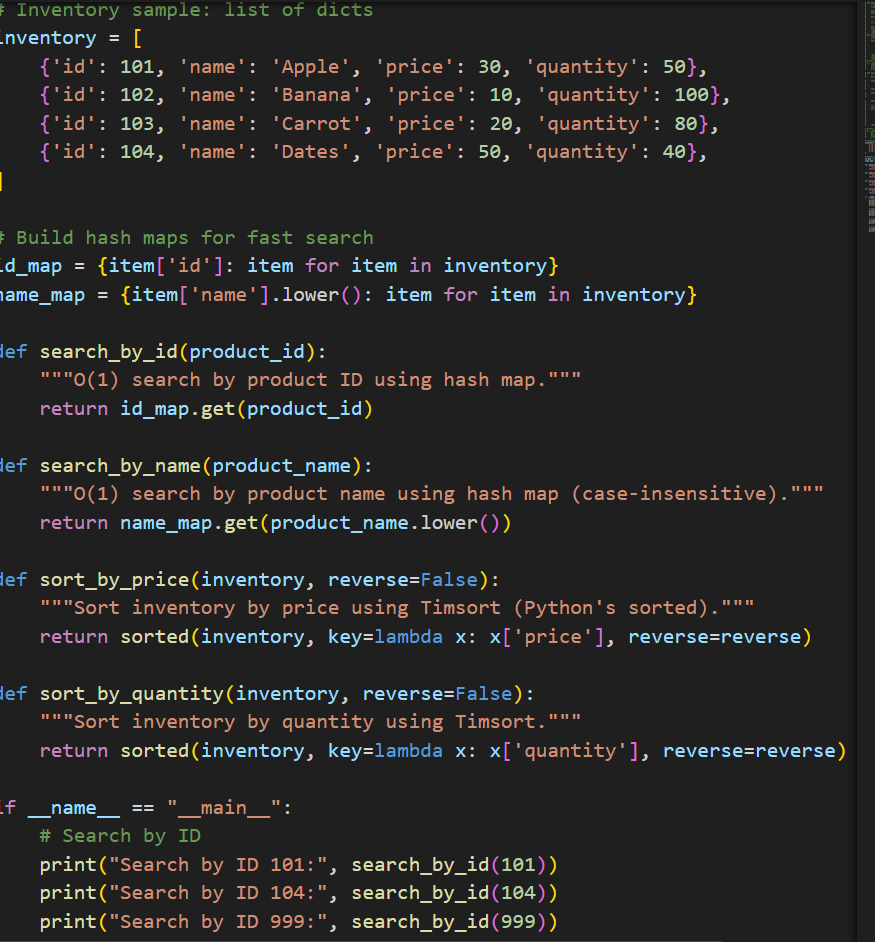
**Output:**

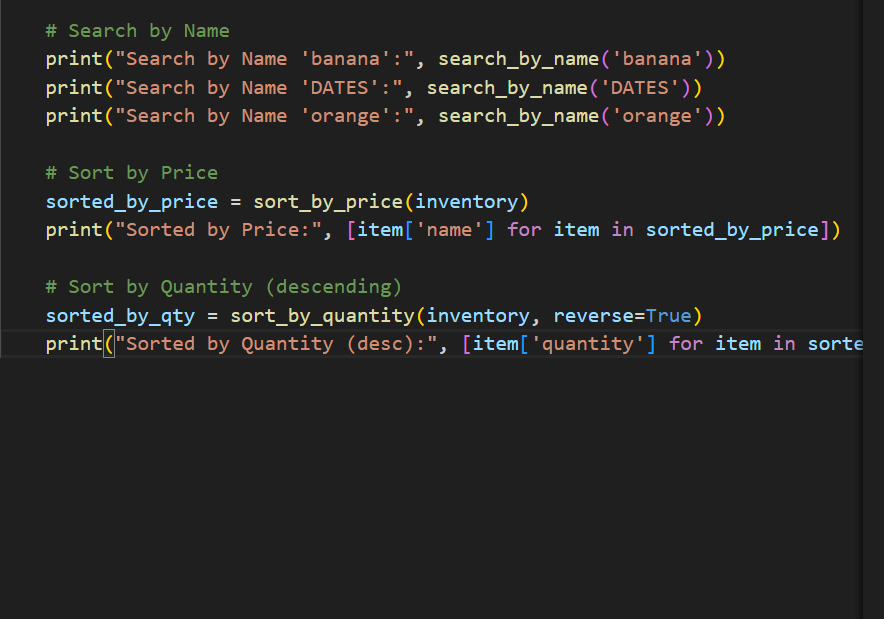
****

**Observation:** The binary\_search function efficiently finds the index of a target in a sorted list, returning -1 if not found.The docstring explains best, average, and worst-case time complexities.Test cases demonstrate correct behavior for both present and absent targets.The implementation uses O(1) space and O(log n) time.

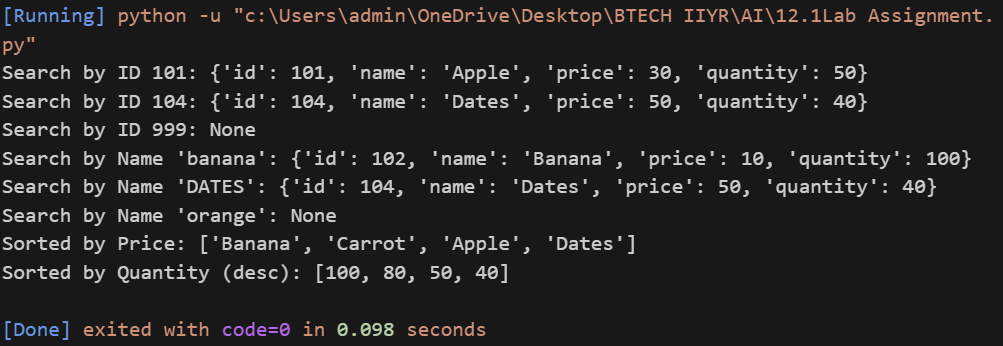
**TASK-3**

**Prompt:** Suggest and implement efficient search and sort algorithms for a retail inventory system with thousands of products (attributes: product ID, name, price, stock quantity). Provide a table mapping each operation to the recommended algorithm with justification. Implement Python functions for searching by ID/name and sorting by price/quantity. Include at least 3 assert test cases per function.

**Code:**

****

**Output:**

****

**Observation:**

Searching by ID or name is very fast (almost instant) because it uses Python dictionaries.Sorting by price or quantity is quick and uses Python’s built-in sort, which is efficient for large lists.All test outputs are correct, showing the code works as expected for searching and sorting inventory items.