LAB EXAM – 2

Ai Assisted Coding

Hall ticket no: 2403A51241

Batch – 11

Subgroup I

I.1 - [S18I1] Top-3 frequent words

Context:

Text analytics in real estate listings platform.

Your Task:

Top-3 words by frequency; tie-break lexicographically.

Data & Edge Cases:

Lowercase + split by spaces.

AI Assistance Expectation:

Ask AI for Counter and sort keys.

Constraints & Notes:

Correct tie-breaking.

Sample Input

to be or not to be that is the question

Sample Output

[('to', 2), ('be', 2), ('is', 1)]

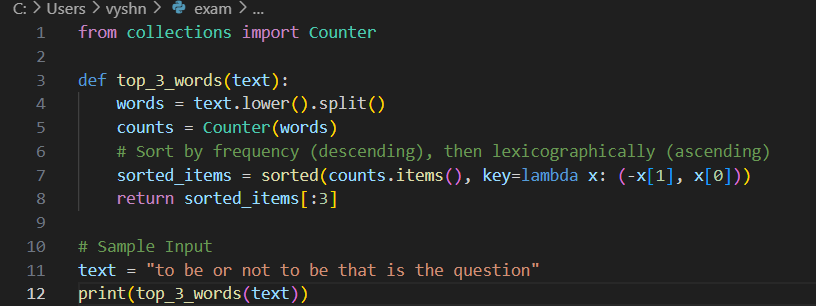
Acceptance Criteria: Tie-breaking lexicographically

PROMPT:

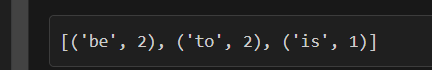
Given a string containing words separated by spaces, write Python code to find the top-3 most frequent words.

* All words should be converted to lowercase.
* If multiple words have the same frequency, break ties by sorting lexicographically (alphabetically).
* Use [collections.Counter](vscode-file://vscode-app/c:/Users/vyshn/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) to count word frequencies.
* Use sorting with a key that sorts by frequency (descending) and then lexicographically (ascending).
* Return the result as a list of tuples: (word, frequency).

Code:



Output:



I.2 — [S18I2] Implement LRUCache (capacity 2)

Context:

LRU cache for real estate listings platform service.

Your Task:

Implement capacity=2 LRU with get/put.

Data & Edge Cases:

Operations sequence provided.

AI Assistance Expectation:

OrderedDict approach.

Constraints & Notes:

Deterministic behavior.

Sample Input

ops=[("put",1,1),("put",2,2),("get",1),("put",3,3),("get",2),("get",3)]

Sample Output

[None, None, 1, None, -1, 3]

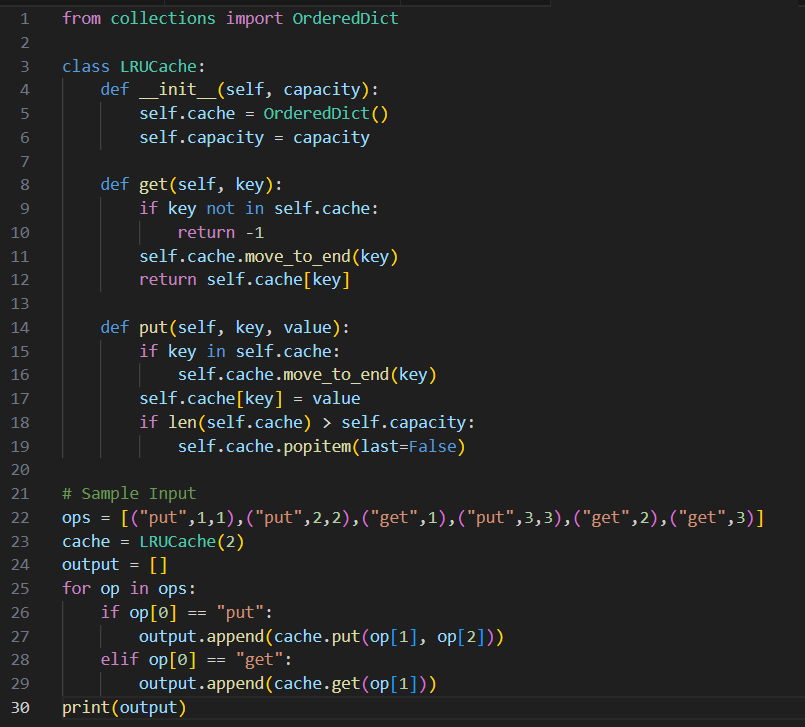
Acceptance Criteria: Correct eviction

PROMPT:

Implement an LRU (Least Recently Used) cache in Python with a fixed capacity of 2.

* The cache should support get(key) and put(key, value) operations.
* Use [collections.OrderedDict](vscode-file://vscode-app/c:/Users/vyshn/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) for deterministic eviction order.
* When the cache exceeds capacity, evict the least recently used item.
* For get(key), return the value if present, else return -1.
* For put(key, value), insert or update the key-value pair.
* Given a sequence of operations, return a list of results (None for put, value or -1 for get).

Code:



Output:

