Software Engineering Exercise (Backend)

The purpose of this exercise is to create a simple "Redis-like" key-value store in Python. All data will be stored in memory for simplicity (so don't worry about disk persistence, or any such mechanisms)

The purpose is to have a program that we can run with "python script.py" which will give you a shell accepting commands. You should be able to interact with it via the console, so it would be advisable to think about separating core storage logic from console logic.

Step 1: It should implement the following commands

- GET <key> - retrieves a value for a given key
  - As a response, if the key is present in the store, the associated value will be returned. However, if the key is not present, the console should print back "NULL"
- SET <key> <value> - associates a key with a value and stores it in memory (all storage is in memory)
  - No response expected in the console (just a new-line)
- UNSET <key>
  - No response expected in the console (just a new-line)
- NUMEQUALTO <value>
  - Returns the number of keys that are associated with the given value
- END - exits the program (just exit, stored data is not expected to persist between runs)
- Any other command should return the string "error" in the console.

Before implementing these, please consider:

- What data structures would be more adequate?
- What is the Big O complexity of each operation?

Step 2: In the spirit of incremental development, as a follow-up, you need to implement transactions on top of what you built. To achieve this you should implement the following commands:

- BEGIN - begins a transaction (no parameters)
  - No response expected in the console (just a new-line)
- COMMIT - commits the transaction (i.e. any changes since the beginning of the transaction become permanent, and the transaction is closed)
  - No response expected in the console (just a new-line), unless no transaction currently exists, in which case it should return "NO TRANSACTION"
- ROLLBACK - rolls back the transaction (i.e. any changes since the beginning of the transaction to the data state are lost, and the transaction is closed)
  - No response expected in the console (just a new-line), unless no transaction currently exists, in which case it should return "NO TRANSACTION"

Before implementing these, please consider:

- What data structures would be more adequate
- Consider any potential tradeoffs out could make, complexity of the implementation, etc. In the scope of this exercise, it’s ok to go for a simpler solution as long as you’re aware of it’s shortcomings. Engineering is often about compromise :)

Other considerations:

- Please send the exercise back either via e-mail, or shared github repo.
- Please write down any assumptions. If anything is not clear, it’s ok to write “I’m assuming you mean X in this part of the exercise” and go with that.

- Also give some considerations to unit tests