CRUD Operation in Python with Neo4j database and Cypher queries

Neo4j Explanation:

Neo4j is a popular and powerful graph database management system designed for efficient storage, retrieval, and management of graph-structured data. Unlike traditional relational databases, which use tables to organize data, Neo4j utilizes a graph-based model to represent and handle data. In a Neo4j graph, data is organized as nodes, relationships, and properties. Neo4j employs Cypher queries, a specialized query language, to interact with and perform operations on the graph data.

- **Nodes**: Represent entities in the graph and hold properties.
- **Relationships**: Define connections between nodes and can also hold properties.
- **Properties**: Key-value pairs associated with nodes and relationships, providing additional information about them.

Neo4j is highly efficient in handling relationships between data points, making it ideal for applications involving social networks, recommendation systems, fraud detection, and more.

Code Explanation:

The code utilizes a virtual environment to isolate and manage the required libraries, ensuring a clean and controlled environment for the project's dependencies. Below are the steps for making this project in python after installing and importing required libraries.

1. Connecting to Neo4j:

The code establishes a connection to Neo4j using the `GraphDatabase` class from the `neo4j` module.

2. CRUD Operations (Create, Read, Update, Delete):

The code defines functions for creating people and pets, establishing an ownership relationship, getting all pets and persons, updating names, and deleting nodes.

3. Menu-Based Interface:

A menu-based interface is implemented to facilitate interaction with the Neo4j database, allowing users to choose CRUD operations through numbered options.

4. Creating Nodes and Relationships:

Functions like `create_person`, `create_pet`, and `create_owns_pet_relationship` create nodes (persons, pets) and establish relationship for ownership. This relationship also have pet amount as its property.

5. Retrieving Nodes:

Functions like `get_all_pets` and `get_all_persons` retrieve all pets and all persons from the Neo4j database, respectively.

6. Updating Node Names:

Functions like `update_person_node_name` and `update_pet_node_name` update the names of persons and pets.

7. Deleting Nodes:

Functions like `delete_person_node` and `delete_pet_node` delete nodes (persons, pets) from the Neo4j database.