Getting Started with Neo4py

Introduction

neo4py is a better alternative to py2neo. Though still in beta phase, you can use it in your projects.

Setting up Neo4py

I recommend creating a venv first using uv to start any of your Python project.

Follow these steps:

1. In your terminal run:

```
pip install uv
```

2. To create a venv run:

```
uv venv
```

This command will create a venv.

3. Now to activate the venv:

In Windows run:

```
.venv\Scripts\activate
```

In case of Linux run:

```
source .venv/Scripts/activate
```

4. Install the neo4j using this command:

```
uv pip install neo4j
```

5. Install the neo4py using this command:

```
uv pip install neo4py
```

Example Code

Let me show you how to use neo4py with a simple example.

Import the neo4py using this command:

```
from neo4py.neo4py import Graph
```

Now create a Graph object, and pass the connection details to the Graph class:

```
graph = Graph("connection_uri", ("user", "db_password"))
```

Or:

```
graph = Graph("bolt://localhost:7687", ("neo4j", "12345678"))
```

You can find your connection uri and user details when you start your database and open the neo4j browser. Password is the password that you have set while creating the database.

Give your data to the run method in this way:

```
data = {'name': 'Athar Naveed', 'age': 21}
```

```
graph = Graph("bolt://localhost:7687", ("neo4j", "12345678"))
```

Now to run the cypher query, go with:

```
graph.run(query, **data)
```

Utilizing Sloth Class

The Sloth class is for the Sloths (like me 😀). Let me show you how to use Sloth with a simple example.

Import the sloth using this command:

```
from neo4py.sloth import Sloth
```

Now create a Sloth object, and pass the connection details to the Sloth class:

```
sloth = Sloth("connection_uri", ("user", "db_password"))
```

Or:

```
sloth = Sloth("bolt://localhost:7687", ("neo4j", "12345678"))
```

You can find your connection uri and user details when you start your database and open the neo4j browser. Password is the password that you have set while creating the database.

Currently Sloth has only 2 methods:

1. create_node
2. read_node

Creating Nodes

You just have to pass your data in a dict format of Python, and it'll create the nodes and labels with respect to it.

To create nodes you have to use this command:

```
sloth.create_node([
    {
        'name': 'Athar',
        'age': 20,
        'gender': 'male',
        'labels': 'Single Label',
    },
    {
        'name': 'Naveed',
        'age': 50,
        'gender': 'male',
        'labels': ['Single Label', 'List of Labels']
    },
])
```

The create_node method takes a list of dictionaries and creates nodes based on the provided data. The labels attribute/key can take one to a list of labels.

Reading Nodes

Currently you can only retrieve all records. Each record will include its id.

To read all nodes you have to use this command:

```
sloth.read_node('*')
```

This method will return all of the records as a list of dictionaries you can iterate over the records to retrieve them. Like this:

```
# code
records = sloth.read_node('*')
for record in records:
    print(f'user id: {record["id"]}')
    print(f'user name: {record["name"]}')
    print(f'user age: {record["age"]}')
    print(f'user labels: {record["label"]}')
    print('====================================')

# output
user id: 0
user name: Athar
user age: 22
```

```
user labels: Human
====================================
user id: 1
user name: Naveed
user age: 52
user labels: ['Person', 'Human']
====================================
```

Content List