Southern New Hampshire University

Back-End Python CRUD Module README

About the Project/Project Title
Python CRUD Module for MongoDB
This module is a Python based module that handles the Create, Read, Update, and Delete functions found within a backend/server-side application, ultimately handling all CRUD based operations to a connected MongoDB database.

Motivation
The overall purpose of this project is to design and develop a Python Module that handles Create, Read, Update, and Delete methods to a connected MongoDB database. By designing this module, it helps create a reusable module for CRUD based operations that can be used and configured to a Mongo database.

Getting Started
To get started with a local setup of this module, follow these steps outlined below:

1. Fork this Repository
2. Clone Your Forked Repository
    a. Use this command to clone the repository
    git clone {Forked GitHub website URL goes here}

3. Create A Branch
    a. It is often advised to never work off of the main branch of a repository.
        i. To create a branch without switching to it, run 'git branch <new-branch-name>'
        ii. To create and switch to the new branch immediately, run 'git checkout -b <new-branch-name>'

4. Import the dataset
    a. You may use any dataset you would like, however, for the example dataset provided, simply run the following command: mongoimport --type=csv --headerline --db aac --collection animals --drop ./aac_shelter_outcomes.csv
5. Run command 'mongosh' to enter the mongo shell
6. Run command 'use {database name here}' to change databases

For user authentication and adding users, follow these steps:

1. Switch to admin by running command: use admin
2. Run command: db.createUser({ user: "{username goes here}", pwd: passwordPrompt(), roles: [{ role: "{Role Access goes here}", db: "{database name goes here}" }] })
3. Verify user creation and authentication
    a. Run command: db.auth("{username here}", passwordPrompt())
    b. Run command: db.runCommand({ connectionStatus: 1 })

Finally for the Python Module setup, to ensure that the correct connections to the database are made, ensure to create a .env file with the username and password of the authenticated user you created, ensure to change the PORT and HOST as needed based on the mongodb credentials, and of course alter

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

the DB and COL variables to match your database and collection names that you would like to use this module for as needed.

Installation
Python (Latest Version)
Jupyter Lab (Latest Version)
IDE of choice (PyCharm is recommended)
PyMongo (run 'python -m pip install pymongo' in cmd or terminal)
bson.objectid (Included in the pymongo package install above)

Usage
Use this space to show useful examples of how your project works and how it can be used. Be sure to include examples of your code, tests, and screenshots.

Code Example
The code below represents the CRUD Functions found in the code base. All functions also include input validation, as well as exception handling.

```
# Create method to implement the C in CRUD.
    def create(self, data):
        # Check that some data was provided
        if data is not None:
            try:
                # Insert the provided dictionary as a new document into the animals collection
                self.database.animals.insert_one(data)  # data should be dictionary
                return True # Indicate that the insert succeeded

            except errors.PyMongoError as e:
                # Catch and display any database related errors during insert
                print(f"An error occurred while inserting the data: {e}")
                return False # Indicate that the insert failed

        else:
            # If no data is given, raise an exception rather than inserting nothing
            raise Exception("Nothing to save, because data parameter is empty")

    # Create method to implement the R in CRUD.
    def read(self, query):
        # Ensure a query filter was provided
        if query is not None:
            try:
                # Find all documents that match the query filter
                # Convert the cursor to a list so it can be returned directly
                documents = list(self.collection.find(query))
                return documents # Return matching documents as a list
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```
            except errors.PyMongoError as e:
                # Catch and display any database related errors during the query
                print(f"An error occurred while reading the data: {e}")
                return [] # Return an empty list if the read fails
        else:
            # If no query is given, raise an exception
            raise Exception("Query parameter is empty")

    # Method to implement the U in CRUD.
    def update(self, lookup_pair, update_data):
        # Ensure both lookup filter and update data are provided
        if lookup_pair is not None and update_data is not None:
            try:
                # Check if update_data already contains an operator
                # If not, assume the user wants to set the fields
                if not any(key.startswith('$') for key in update_data.keys()):
                    update_operation = {'$set': update_data}
                else:
                    update_operation = update_data

                # Use update_many to allow for modification of multiple documents
                result = self.collection.update_many(lookup_pair, update_operation)

                # Return the count of documents modified
                return result.modified_count

            except errors.PyMongoError as e:
                # Catch and display any database related errors during the update
                print(f"An error occurred while updating the data: {e}")
                return 0 # Return 0 objects modified if an error occurs

        else:
            # Raise an exception if required parameters are missing
            raise Exception("Required parameters for update are missing: lookup_pair and/or
update_data")

    # Method to implement the D in CRUD.
    def delete(self, lookup_pair):
        # Ensure a lookup filter was provided
        if lookup_pair is not None:
            try:
                # Use delete_many to allow for removal of multiple documents
                result = self.collection.delete_many(lookup_pair)

                # Return the count of documents removed
                return result.deleted_count
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```
            except errors.PyMongoError as e:
                # Catch and display any database related errors during the delete
                print(f"An error occurred while deleting the data: {e}")
                return 0 # Return 0 objects removed if an error occurs

        else:
            # Raise an exception if the required parameter is missing
            raise Exception("Required parameter for delete is missing: lookup_pair")
```

Tests

Describe and show how to run the tests with code examples.

Using Jupyter Notes, you can run simple tests by following these simple steps:

Python Testing Script

- Import for your CRUD module.

```
from Animal_Shelter_DB_CRUD_Python_Module import AnimalShelter
```

- Instantiate an instance of your class.

```
shelter = AnimalShelter()
```

- Use your create function create a new record in the aac database.

```
new_animal = {
   "age_upon_outcome": "6 years",
   "animal_id": "Z8675309",
   "animal_type": "Dog",
   "breed": "Boston Terror",
   "color": "Black/White",
   "date_of_birth": "2019-07-23",
   "name": "Zoey Glittersparkles",
   "outcome_subtype": "",
   "outcome_type": "Adoption",
   "sex_upon_outcome": "Spayed Female",
   "location_lat": 45.833506,
   "location_long": -108.469357
}

created = shelter.create(new_animal)
print("Create result:", created)
```

- Use your read funtion to return records from the aac database.

```
results = shelter.read({"breed": "Boston Terror"})
print("Read results:")
for r in results:
   print(r)
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

- Use your update function to update and return the number of documents updated in the aac database.

```
count = shelter.update({"breed": "Boston Terror"},{"breed": "Boston Terrier"})
print(f"Updated {count} documents.")

print("Verifying update...")
results = shelter.read({"breed": "Boston Terrier"})
for r in results:
    print(r)
```

- Use your delete function to delete selected documents and return the number of deleted documents in the aac database.

```
count = shelter.delete({"name":"Zoey Glittersparkles"})
print(f"Deleted {count} documents.")

print("Verifying purge...")
results = shelter.read({"name":"Zoey Glittersparkles"})
if results:
    for r in results:
        print(r)
else:
    print("No records found.")
```

Screenshots

The image below shows the source code of the CRUD methods of the module:

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```python
    # Create a method to return the next available record number for use in the create method
    def getNextRecordNum(self):
        # Query the animals collection for the document with the highest rec_num value
        out = self.database.animals.find().sort([('rec_num', -1)]).limit(1)

        # Loop through the result and return the next record number
        for dict in out:
            return (dict['rec_num'] + 1)

    # Create method to implement the C in CRUD.
    def create(self, data):
        # Check that some data was provided
        if data is not None:
            try:
                # Insert the provided dictionary as a new document into the animals collection
                self.database.animals.insert_one(data)  # data should be dictionary
                return True # Indicate that the insert succeeded

            except errors.PyMongoError as e:
                # Catch and display any database related errors during insert
                print(f"An error occurred while inserting the data: {e}")
                return False # Indicate that the insert failed

        else:
            # If no data is given, raise an exception rather than inserting nothing
            raise Exception("Nothing to save, because data parameter is empty")

    # Create method to implement the R in CRUD.
    def read(self, query):
        # Ensure a query filter was provided
        if query is not None:
            try:
                # Find all documents that match the query filter
                # Convert the cursor to a list so it can be returned directly
                documents = list(self.collection.find(query))
                return documents # Return matching documents as a list

            except errors.PyMongoError as e:
                # Catch and display any database related errors during the query
                print(f"An error occurred while reading the data: {e}")
                return [] # Return an empty list if the read fails
        else:
            # If no query is given, raise an exception
            raise Exception("Query parameter is empty")
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```python
# Method to implement the U in CRUD.
    def update(self, lookup_pair, update_data):
        # Ensure both lookup filter and update data are provided
        if lookup_pair is not None and update_data is not None:
            try:
                # Check if update_data already contains an operator
                # If not, assume the user wants to set the fields
                if not any(key.startswith('$') for key in update_data.keys()):
                    update_operation = {'$set': update_data}
                else:
                    update_operation = update_data

                # Use update_many to allow for modification of multiple documents
                result = self.collection.update_many(lookup_pair, update_operation)

                # Return the count of documents modified
                return result.modified_count

            except errors.PyMongoError as e:
                # Catch and display any database related errors during the update
                print(f"An error occurred while updating the data: {e}")
                return 0 # Return 0 objects modified if an error occurs

        else:
            # Raise an exception if required parameters are missing
            raise Exception("Required parameters for update are missing: lookup_pair and/or update_data")

    # Method to implement the D in CRUD.
    def delete(self, lookup_pair):
        # Ensure a lookup filter was provided
        if lookup_pair is not None:
            try:
                # Use delete_many to allow for removal of multiple documents
                result = self.collection.delete_many(lookup_pair)

                # Return the count of documents removed
                return result.deleted_count

            except errors.PyMongoError as e:
                # Catch and display any database related errors during the delete
                print(f"An error occurred while deleting the data: {e}")
                return 0 # Return 0 objects removed if an error occurs

        else:
            # Raise an exception if the required parameter is missing
            raise Exception("Required parameter for delete is missing: lookup_pair")
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

The images below showcase an example test file using Jupyter Notes to test the CRUD Functions.

[1]:
```python
# Python Testing Script

# Importing CRUD module
from Animal_Shelter_DB_CRUD_Python_Module import AnimalShelter

# Instantiate an instance of the AnimalShelter class
shelter = AnimalShelter()

# Create a new record in the aac database
new_animal = {
    "age_upon_outcome": "6 years",
    "animal_id": "Z8675309",
    "animal_type": "Dog",
    "breed": "Boston Terror",
    "color": "Black/White",
    "date_of_birth": "2019-07-23",
    "name": "Zoey Glittersparkles",
    "outcome_subtype": "",
    "outcome_type": "Adoption",
    "sex_upon_outcome": "Spayed Female",
    "location_lat": 45.833506,
    "location_long": -108.469357
}
created = shelter.create(new_animal)
print("Create result:", created)

# Read records from the aac database
results = shelter.read({"breed": "Boston Terror"})
print("Read results:")
for r in results:
    print(r)

# Update records from the aac database
count = shelter.update({"breed": "Boston Terror"},{"breed": "Boston Terrier"})
print(f"Updated {count} documents.")

print("Verifying update...")
results = shelter.read({"breed": "Boston Terrier"})
for r in results:
    print(r)

# Delete records from the aac database
count = shelter.delete({"name":"Zoey Glittersparkles"})
print(f"Deleted {count} documents.")

print("Verifying purge...")
results = shelter.read({"name":"Zoey Glittersparkles"})
if results:
    for r in results:
        print(r)
else:
    print("No records found.")
```

Contact

Your name: Eric Foster

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.