# CS 340 README

## About the Project/Project Title

The CRUD Python Module now fully implements Create, Read, Update, and Delete operations for MongoDB. In addition to inserting and retrieving documents, the module now supports updating documents (returning the number of modified documents) and deleting documents (returning the number of deleted documents). This design enables detailed tracking of database modifications and is built using industry best practices.

## Motivation

I developed this module to deepen my understanding of MongoDB and to practice implementing real-world CRUD operations using Python. With prior experience in setting up user authentication and database connections, I was excited to transfer that knowledge into a practical application. Overcoming challenges like ensuring a robust database connection, input validation, and error handling has been invaluable for my growth as a developer.

**Python Driver:**  
This module uses the pymongo library—the official MongoDB driver for Python. It was chosen for its comprehensive support for MongoDB features, active maintenance, and widespread community adoption.

**CRUD Operation Details:**

• Create: Accepts a non-empty dictionary to insert a new document into the specified collection.

• Read: Uses a provided query (or returns all documents if none is given) and returns a list of documents.

• Update: Accepts a query and an update dictionary; performs an update operation that modifies all matching documents and returns the count of modified documents.

• Delete: Accepts a query and removes all matching documents from the collection, returning the count of deleted documents.

## Getting Started

To help get started using the module, follow these steps:

1. Download the file provided with this documentation.

2. MongoDB Connection: The module connects to a MongoDB instance using the pymongo library. It uses default connection details (host, port, database, and collection) that you can override by providing your own credentials.

3. The connection uses environment variables (MONGO\_USER and MONGO\_PASS) or will prompt for input if these are not set.

4. Install dependencies from requirements.txt if available or simply run pip install missing packages.

5. Import the module into your project via import the file name.

## Installation

Required tools and libraries:

• Python 3.9 - available at python.org

• MongoDB

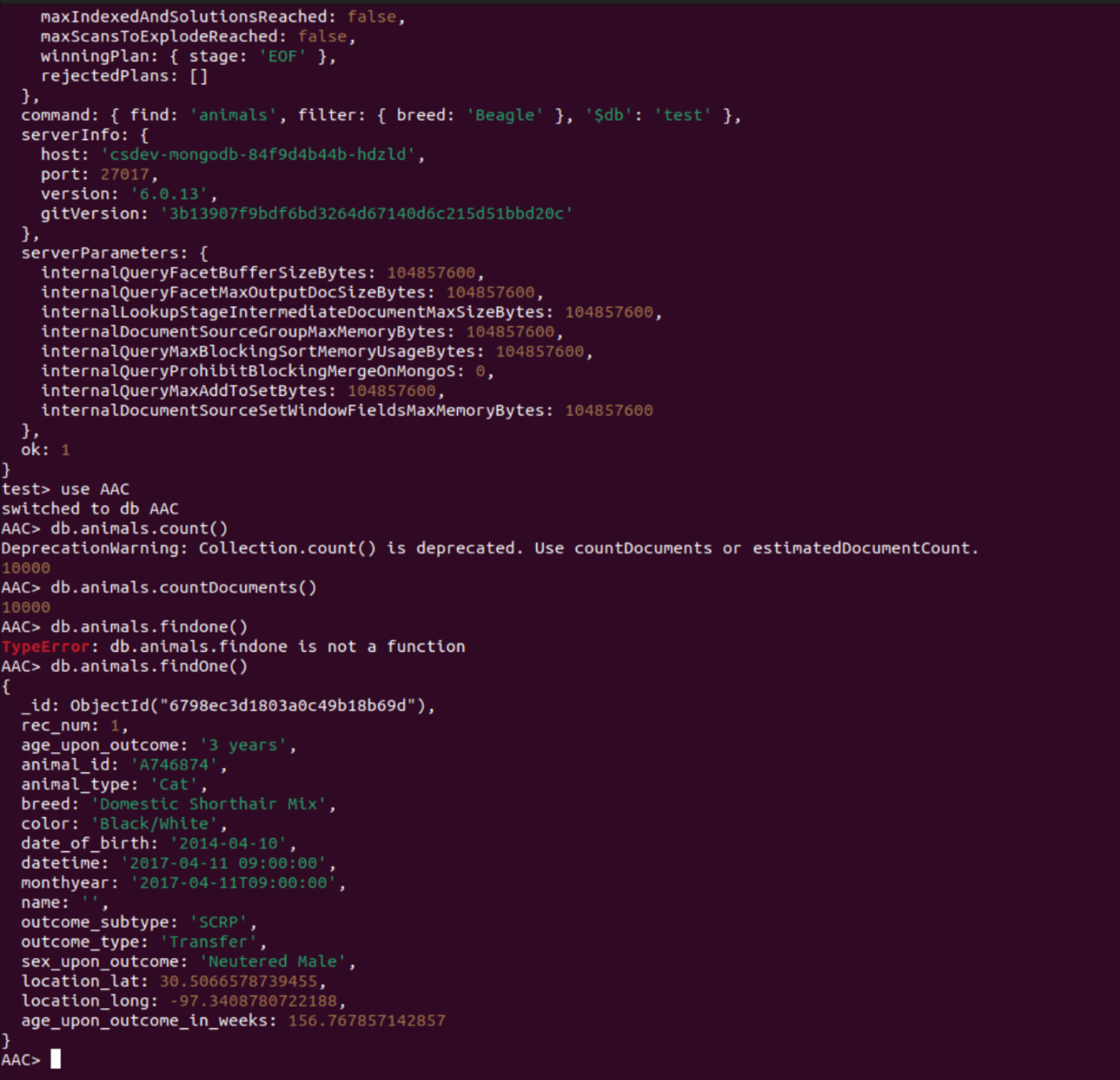
• pymongo

• bson

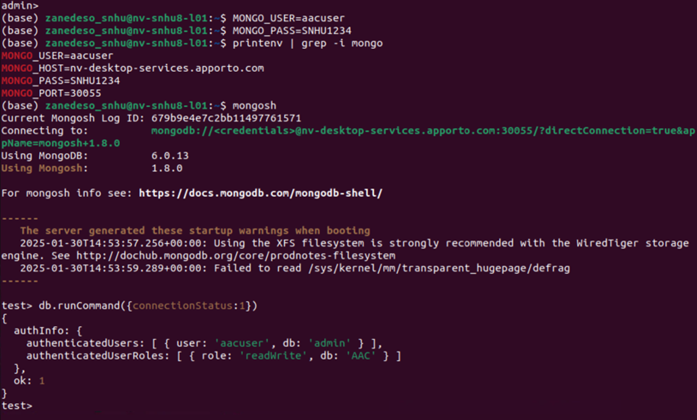
These can be installed via the command: pip install -r requirements.txt

Mongoimport Tool Command

mongoimport --host nv-desktop-services.apporto.com --port 31580 -u aacuser -p SNHU1234 --authenticationDatabase AAC --db AAC --collection animals --type csv --file /usr/local/datasets/aac\_shelter\_outcomes.csv --headerline

**Mongoimport Verification**  


**Log In process**



## Usage

This module demonstrates how to perform full CRUD operations on a MongoDB collection. When you run the module, it will:

• Prompt for or use provided credentials to establish a connection with MongoDB.

• Insert a new document (for example, with fields such as name and species) into a specified collection (Create).

• Retrieve and display the inserted document in the console (Read).

• Modify existing documents by applying update operations based on a query. The module returns the number of documents that were updated (Update).

• Remove documents from the collection using a specified query, returning the count of documents deleted (Delete).

## Tools Used and Rationale

1. MongoDB as the Model - Chosen for its flexible schema, easy integration with Python, and ability to store complex JSON-like documents.

2. Dash as the View/Controller - Dash is a web framework for Python that seamlessly integrates interactive charts, tables, and components.

3. Python 3.9 & pymongo - Python is widely used for data manipulation and rapid prototyping. pymongo ensures compatibility and reliability.

4. Additional Tools - Jupyter Notebooks, Plotly Express, and dash\_leaflet for maps and data visualization.

## Steps Taken to Complete the Project

1. Set Up MongoDB and imported data.

2. Developed CRUD Python Module with robust error handling.

3. Integrated Dash for interactive filtering and visualization.

4. Tested functionality using Jupyter Notebook and manual execution.

5. Refined layout and branding for a professional look.

## Challenges Encountered and How They Were Overcome

• Base64 Image Encoding: Initially failed to display; resolved by decoding properly.

• Filtering Logic: Adjusted DataFrame operations to ensure correct data selection.

• Map Integration: Ensured lat/lon values were valid and correctly formatted.

• Callback ID Mismatches: Fixed mismatches to enable Dash interactivity.

## Deployment Screenshots

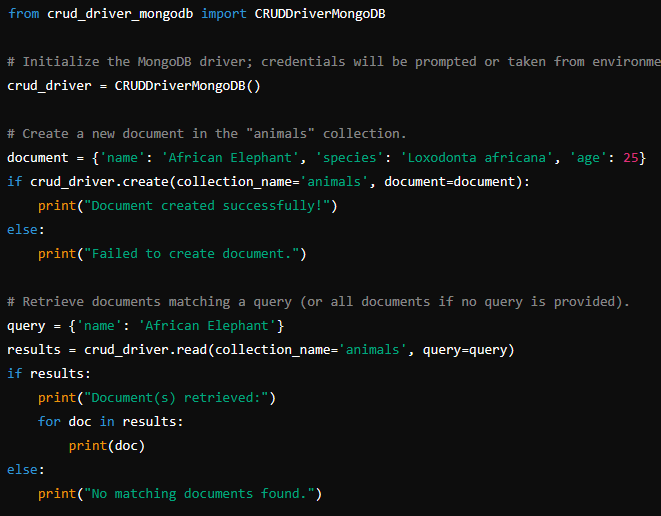
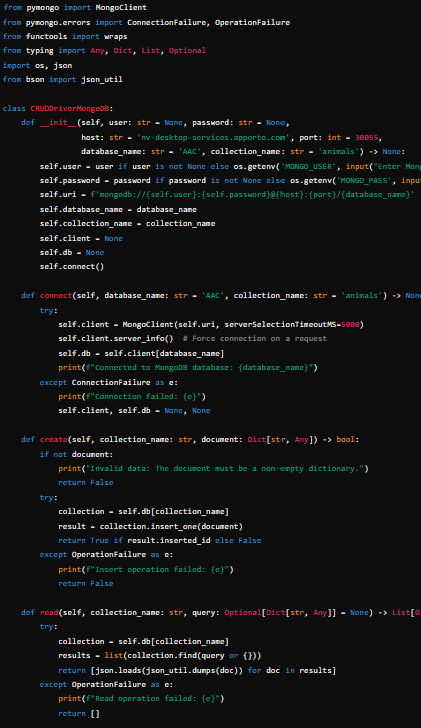
A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

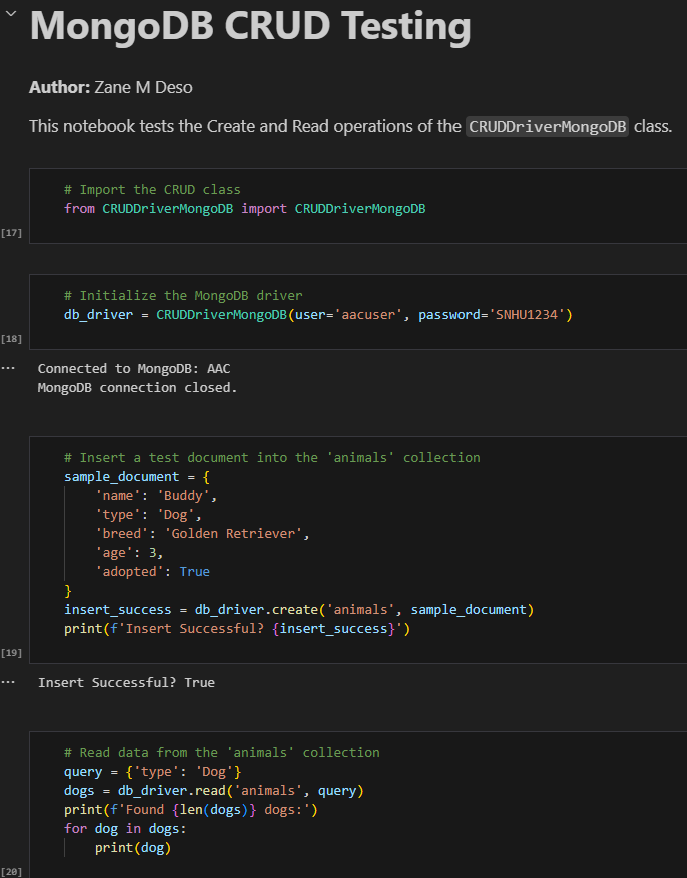
AI-generated content may be incorrect.

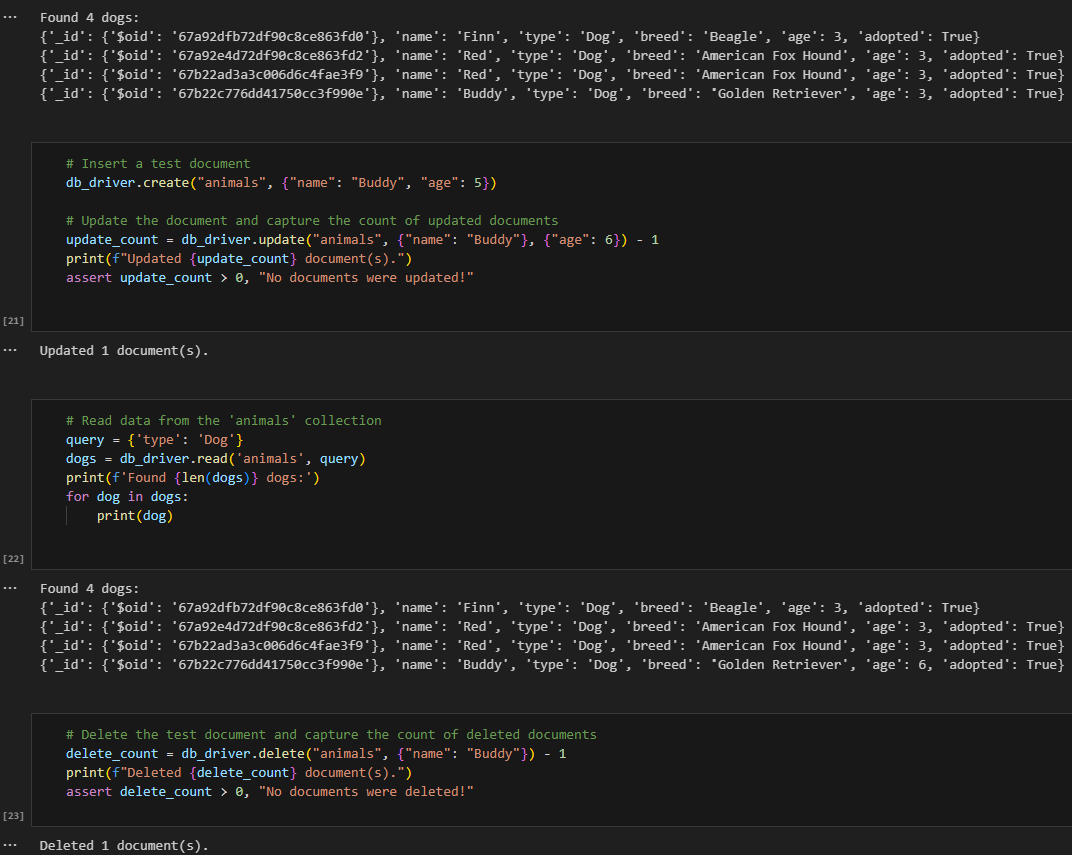
### Code Example



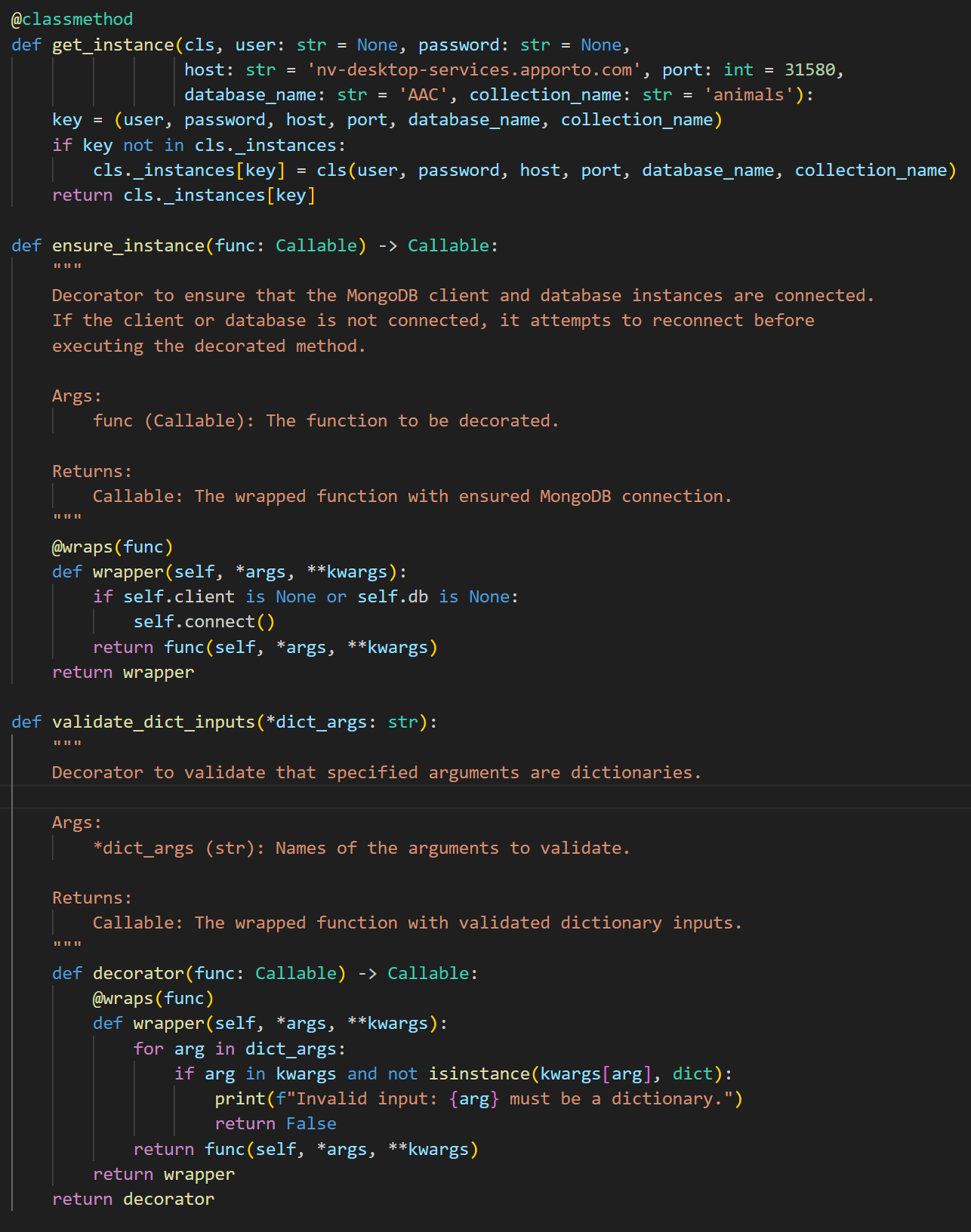
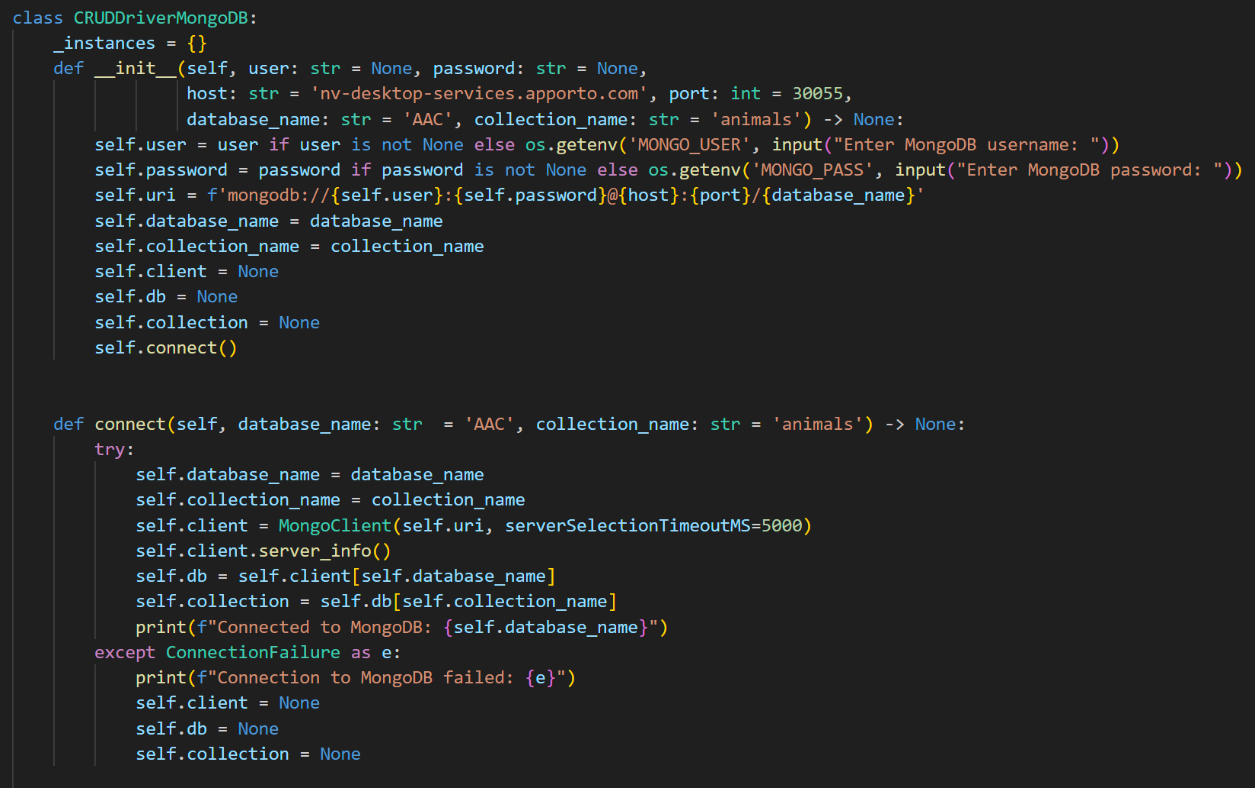
### Tests

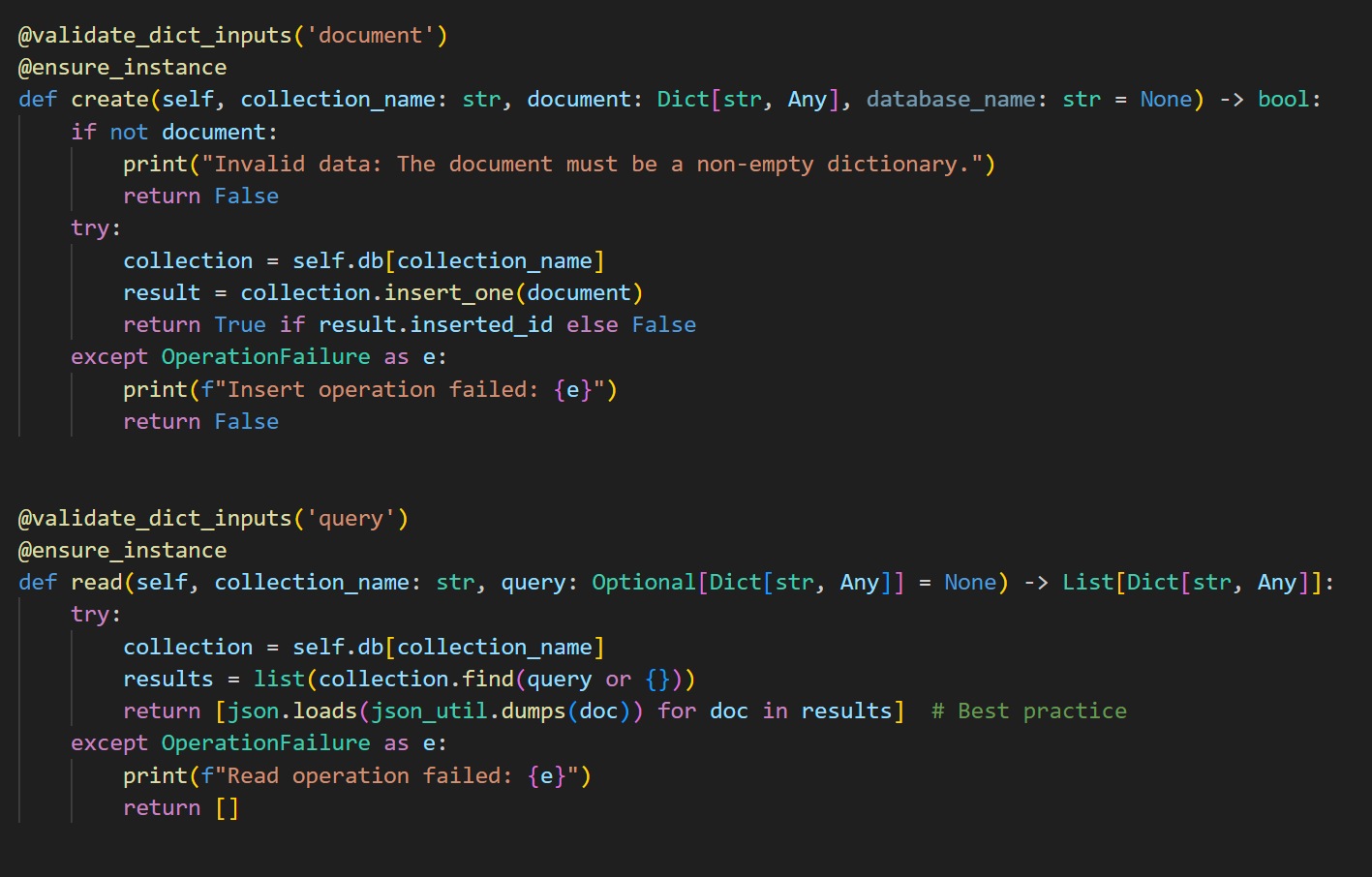
For testing I simply conducted manual testing of the creation and read operations via jupyter notebook script.





### Code Screenshots





## Contact

Zane M Deso @ Zane.Deso@snhu.edu