Augustus Mendy

Southern New Hampshire University

CS-340-10011-M01

Professor Gebre-Amlark

April 6, 2025

## About the Project/Project Title

Through the Pymongo driver, this class makes it easier to perform Create-Read-Update-Delete (CRUD) operations in MongoDB. All of the abstracted getter/setter methods required for MongoDB initialization and CRUD operations will be included in the class.

## Motivation

The purpose of this application was to assess my ability to deal with databases and manipulate their contents. The language selected to power MongoDB was Python. Python works nicely with MongoDB and is simple to write and manage. With Jupyter Notebook's built-in compiler, Python may also be compiled rapidly.

## Getting Started

In order to begin this program, you would first.

* Open Mongo and import the aac\_shelter\_outvome.csv file.
* The next step would be to develop both a basic and a sophisticated index to interpret the information contained in the paper.
* A user would now want to create both an admin account and an aacuser account in order to authenticate and access the database.
* The user would then need to run the program from a notebook and have access to or install Python.

## Installation

An up-to-date Python version that can execute both.py and.ipynb files.

* Python 3.6
* Pymongo 4.2

MongoDB 4.2

Data available in a Mongo database collection,

MongoDB - “to access the database”

* cd /usr/local/datasets/aac\_shelter\_outcome.csv
* /usr/local/datasets/
* from pymongo import MongoClient

## 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

A user can use the code to test, add, edit, read, and remove animals from a shelter. One would utilize to test this. After launching Mongo and loading the required Python files, a user might add animals by entering print(animals.create (STRING\_TYPE). The program would then throw an error if the addition was unsuccessful or a boolean if it was successful.

animals.createRecord({

age\_upon\_outcome': '2 years',

'animal\_id': 'A716330',

'animal\_type': 'Dog',

'breed': 'Chihuahua Shorthair Mix',

'color': 'Brown/White',

'date\_of\_birth': '2013-11-18',

'datetime': '2015-12-28 18:43:00',

'monthyear': '2015-12-28T18:43:00',

'name': 'Frank',

'outcome\_subtype': '',

'outcome\_type': 'Adoption',

'sex\_upon\_outcome': 'Neutered Male',

'location\_lat': 30.7595748121648,

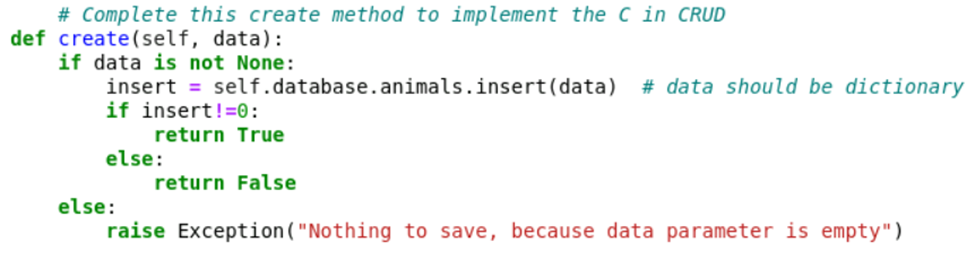
'location\_long': -97.5523753807133,

'age\_upon\_outcome\_in\_weeks': 110.111408730159

})

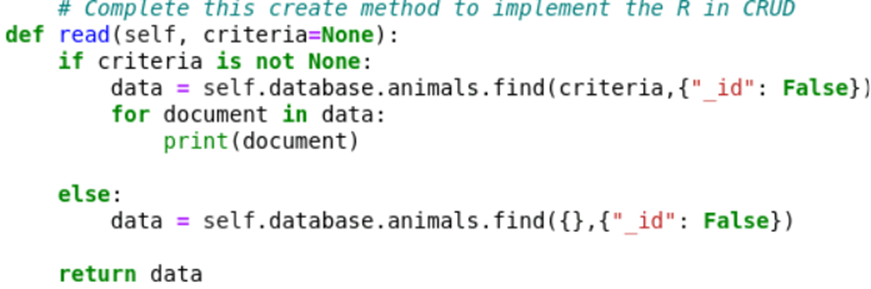
)

* **A Create method that inserts a document into a specified MongoDB database and collection**
  + Input -> argument to function will be a set of key/value pairs in the data type acceptable to the MongoDB driver insert API call.
  + Return -> “True” if successful insert, else “False”.

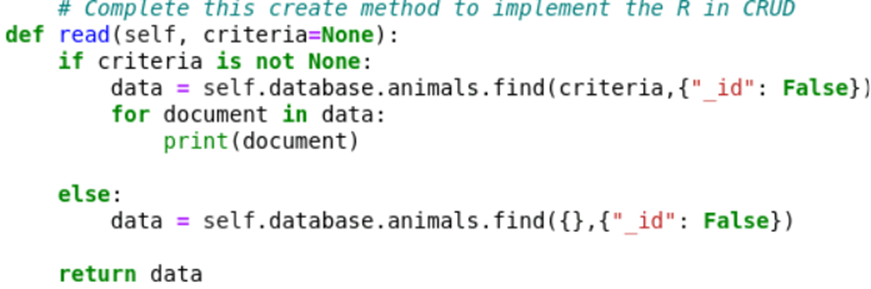


• **A Read method that queries for document(s) from a specified MongoDB database and specified collection.**

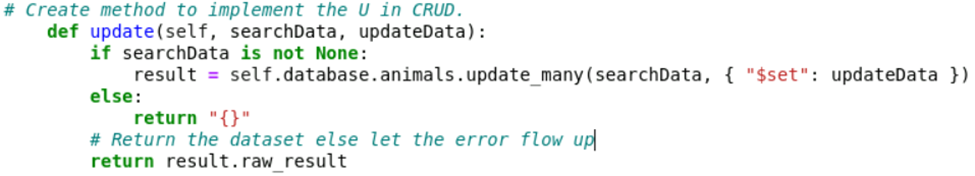
* Input -> arguments to function should be the key/value lookup pair to use with the MongoDB driver find API call.
* Return -> result in cursor if successful, else MongoDB returned error message.



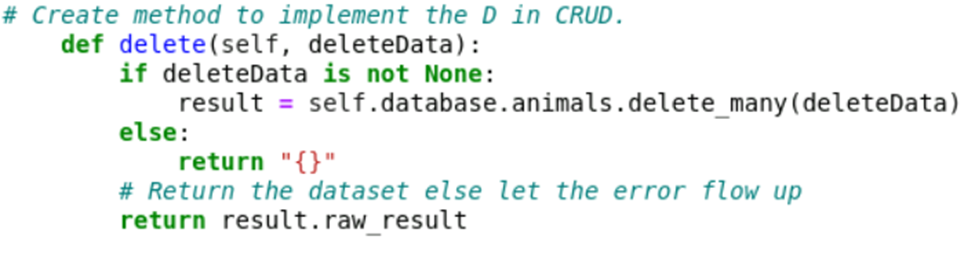
* **A Read method that queries for document(s) from a specified MongoDB database and specified collection.**
* Input -> arguments to function should be the key/value lookup pair to use with the MongoDB driver find API call.
* Return -> result in cursor if successful, else MongoDB returned error message.



* **An Update method that queries for and changes document(s) from a specified MongoDB database and specified collection**
* Input -> arguments to function should be the key/value lookup pair to use with the MongoDB driver find API call. Last argument to function will be a set of key/value pairs in the data type acceptable to the MongoDB driver insert API call.
* Return -> result in JSON format if successful, else MongoDB returned error message.



* **A Delete method that queries for and removes document(s) from a specified MongoDB database and specified collection.**
* Input -> arguments to function should be the key/value lookup pair to use with the MongoDB driver find API call.
* Return -> result in JSON format if successful, else MongoDB returned error message.

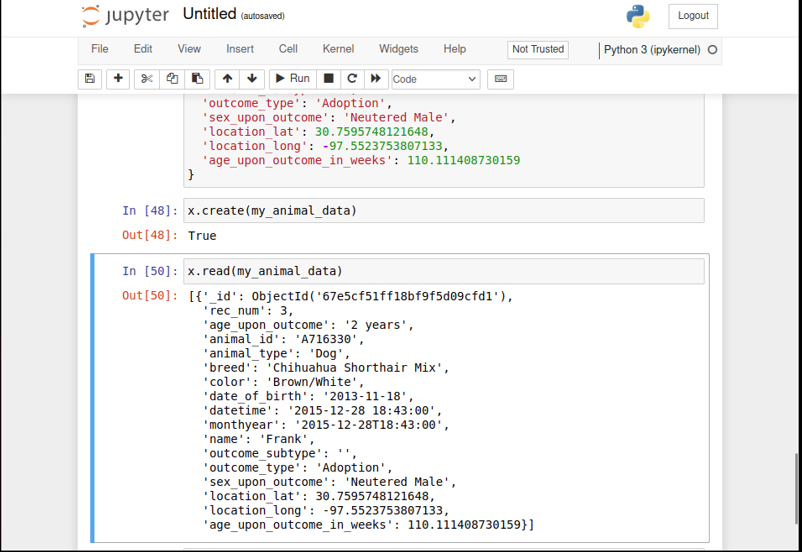


**Test**

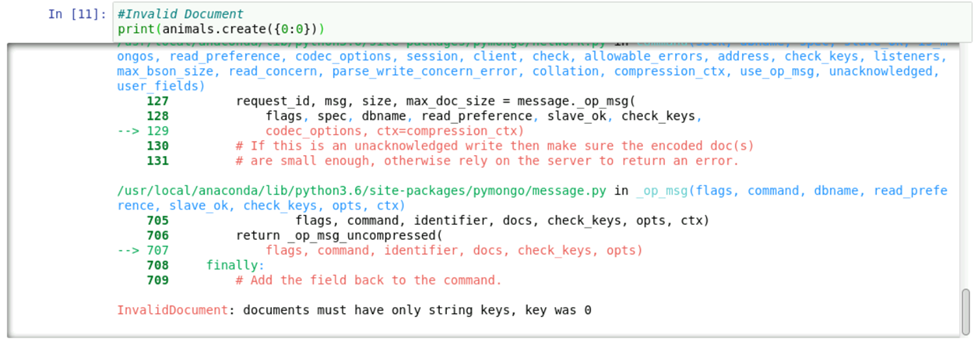
An invalid print(animals.create(0:0)) statement was used to test this code, attempting to construct an invalid data type while producing an invalid argument.

You might use query = animals to find your added animal and make sure it was added.(\"name": "NAME"})

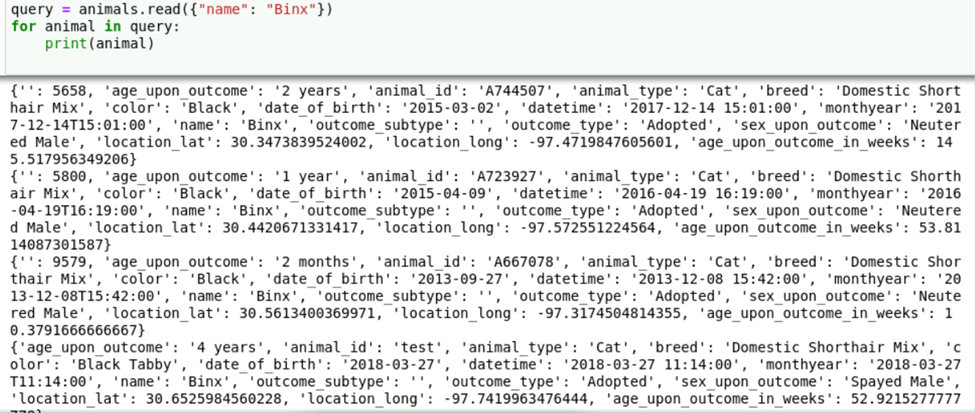
### Screenshots

**

**Invalid creation script with outcome:**

**

**Query to locate my created animal:**

****

**In valid query to search for animal:**

*A screenshot of a computer program

AI-generated content may be incorrect.*

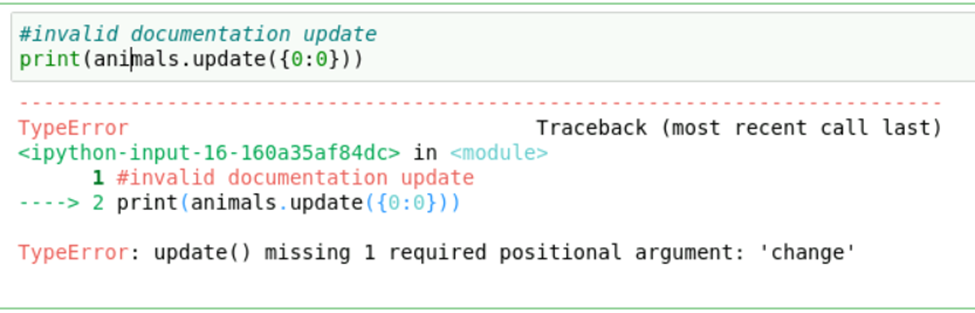
**Animal Documentation update:**

****

**Invalid documentation update:**

Text

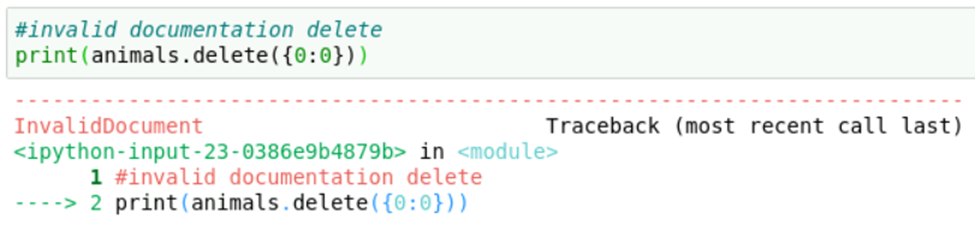
Description automatically generated

**

## Valid documentation deletion:



## Invalid deletion:



## Roadmap/Features (Optional)

MongoDB startup will be the main focus of future work. Currently, the database, username, and password need to be hardcoded. This can be provided to the constructor in later editions.   
  
Additionally, for every new object, the constructor will attempt to initialize the database. Future releases will include a singleton.

## Contact

Your name: Augustus Mendy