# CS 340 Python Mongo CRUD - README

## About the Project/Motivation

This project is a web application that connects a client-side user interface (dashboard) to a database. The project was designed in a way that it can be easily implemented into various projects to give a program the functionality to connect to an existing mongo database. Along with connecting to a mongo db, you will also have the functionality to create, read, update, and delete documents within a given collection. Future iterations of this project will add a user dashboard, rather than typing all the commands within the terminal.

*Purpose of the CRUD Python module:*

The motivation behind this project is to make the development of projects quicker and easier when implementing CRUD operations with a MongoDB. The CRUD operations are helpful because they provide a simple and intuitive interface to interact with the database. They allow you to create, read, update, and delete data in a straightforward manner, making it easier to build applications that store and manipulate data. Furthermore, the flexibility of MongoDB's document model allows you to handle evolving data structures and adapt to changing application needs without requiring significant schema modifications.

*Why use a Python driver for Mongo?*

The Python driver for MongoDB is called PyMongo. PyMongo is the official MongoDB driver for Python and is widely used in the Python ecosystem for interacting with MongoDB databases. It provides a high-level API for performing CRUD operations and handling other database-related tasks. Due to its official status, feature-rich API, extensive documentation, and strong community support. It provides a reliable and efficient means of interacting with MongoDB, making it a suitable driver for building Python applications that utilize MongoDB as the underlying database.

*Explanation of the CRUD operation and working functionality:*

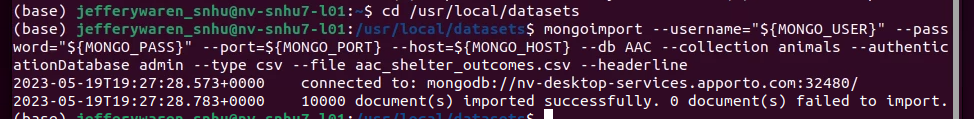
* Create (Insert): In the context of MongoDB, the Create operation involves inserting a new document or multiple documents into a collection. You provide the data as a document (in BSON format) and specify the collection where the document should be inserted. MongoDB assigns a unique identifier (ObjectId) to each inserted document if an identifier is not explicitly provided.
* Read (Query): The Read operation involves retrieving data from a MongoDB collection based on specified criteria. You can query the database for specific documents that match certain conditions. MongoDB's query language supports a wide range of criteria, such as field values, ranges, regular expressions, logical operators, and more.
* Update: The Update operation modifies existing documents in a MongoDB collection. You can change the values of specific fields within a document or update multiple documents that match a given condition. MongoDB provides various update operators that allow you to perform specific updates, such as setting a field to a new value, incrementing/decrementing a numeric field, adding/removing elements from arrays, renaming fields, and performing other atomic modifications.
* Delete: The Delete operation involves removing documents from a MongoDB collection. MongoDB provides delete operators like deleteOne() and deleteMany() to facilitate these deletions. You can specify the conditions to identify which documents should be deleted. Upon execution, the documents satisfying the specified criteria are permanently removed from the collection.

*Purpose for using the Dash Framework:*

Dash is a popular Python framework developed by Plotly for creating interactive web-based visualizations. It allows you to build web applications with a rich user interface and data visualization capabilities. The framework follows a reactive approach with a view and controller structure. The view is defined using layout components like HTML elements, graphs, and tables, organized hierarchically. The controller aspect is achieved through callbacks, which are Python functions executed in response to user interactions or changes in the application state. Dash's declarative syntax allows developers to specify the desired state of the application, with Dash handling automatic updates to the view based on changes in the state.

## Getting Started

To get a local copy up and running, follow these simple example steps:

* If you would like to start working with the python module to connect to a mongo database, you will firstly need to have an active mongo database running. An example of how to import a mongo database using the mongoimport tool is shown below:
* A picture containing text, screenshot, font

  Description automatically generatedCreate a user account within the mongo shell. An example of how to create a user with the proper database access and roles is shown below. Make sure that you adjust the user, role, and db to fill the needs of your specific database. It is also recommended to switch the user role to “readWrite” instead of “read” in the example.
* Open your IDE of choice and project that you would like to implement the module
* Simply add the attached python file to your project directory
* You should now be able to create an AnimalShelter object that will connect to a mongo database
  + A picture containing text, screenshot, font

    Description automatically generatedWhen creating the AnimalShelter object, make sure to adjust the mongo user, pass, host, port, db, and collection information as needed to connect to your unique db. How you can get this information is shown below:
* After creating the AnimalShelter object, you will now be able to perform the 4 built-in methods (create, read, update, and delete) to manipulate the documents within the db collection

## Installation

Listed below is all the necessary software to be able to run the project:

* Access to an integrated development environment (IDE) of your choice. A link has been provided for Pycharm, but many other IDEs would work as well
  + [Pycharm Download](https://www.jetbrains.com/pycharm/download/#section=windows)
* A MongoDB that is up and running that you would like to connect to
  + [Purchase a MongoDB](https://www.mongodb.com/pricing)
* Although not necessary to run the project, access to Jupyter Notebook is recommended to effectively test to ensure the python module has successfully connected to the mongo db
  + [Jupyter Notebook Download](https://jupyter.org/install)
* Dash was used for the framework to build the web application
  + [Dash Enterprise](https://plotly.com/dash/)

## Usage

### Code Examples

* **A picture containing text, screenshot, font, line

  Description automatically generated**The following code is an example of creating an AnimalShelter object with the given mongo user and mongo password as parameters:
* **A picture containing text, screenshot, font, line

  Description automatically generated**The following code is an example of how to implement the create method to add a document to the collection
* **A screenshot of a computer

  Description automatically generated with medium confidence**The following code is an example of how to implement the read method to search and print all the found documents with a given data parameter:
* **A screenshot of a computer program

  Description automatically generated with low confidence**The following code is an example of how to implement the update method to change elements within an existing document:
* **A screenshot of a computer program

  Description automatically generated with low confidence**The following code is an example of how to implement the delete method with a key/value pair parameter to delete all documents that match the search criteria:
* A picture containing text, screenshot, font

  Description automatically generatedAn example of how the mongo host, port, database, and collection names can all be adjusted to your specific db:

### Tests

**A screenshot of a computer program

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with medium confidenceA picture containing text, screenshot, font, line

Description automatically generated**The following screenshot is an example of how you can test to make sure the python module is working properly, as well as how to implement and use the built-in methods to perform all the CRUD functionality. The example was created within Jupyter Notebook.

### Application Example

1. Starting State:

Provided below is an example of how the Python CRUD module can be applied to use with frameworks such as Dash, Plotly, Pandas, and Numpy to build a web application that interacts with the mongo database. This dashboard allows a user to interact and visualize the database with ease. The screenshot below shows the starting state of the dashboard. Notice the 4 buttons that are available for the user to select, as well as the two widgets at the bottom of the page that display useful data extracted from the database.

A screenshot of a computer

Description automatically generated with low confidence

1. Water Rescue Filter Example

A screenshot of a computer

Description automatically generated with low confidenceProvided below is how the dashboard dynamically updates after the user has selected the ‘water rescue’ button. The data table updates and shows what rescue animals fit the water rescue criteria as well as the pie chart showing what percentage of dog breeds make up the result. When a row is selected within the data table the geolocation chart will dynamically update as well to show the location of where the rescue animal was found.

1. Mountain or Wilderness Rescue Filter Example

Provided below is how the dashboard dynamically updates after the user has selected the ‘mountain or wilderness rescue’ button. The data table updates and shows what rescue animals fit the specific rescue criteria as well as the pie chart showing what percentage of dog breeds make up the result.

A picture containing text, screenshot, diagram, map

Description automatically generated

1. Disaster Rescue or Individual Tracking Filter Example

A picture containing text, screenshot, diagram, map

Description automatically generatedProvided below is how the dashboard dynamically updates after the user has selected the ‘disaster rescue or individual tracking’ button. The data table updates and shows what rescue animals fit the specific rescue criteria as well as the pie chart showing what percentage of dog breeds make up the result.

1. Reset Button Example

Provided below is how the dashboard dynamically updates after the user has selected the ‘reset’ button. It essentially returns the default starting state of the dashboard.

A screenshot of a computer

Description automatically generated with low confidence

### Challenges Encountered

1. Challenge: One challenge I encountered during the testing process was accidentally deleting man of the documents contained within the database. I was testing to ensure the delete function worked correctly, but in this process, I was left with very few documents. I did not realize this until coming back to work on this project later and realizing that my queries were not working properly; this was because that was hardly any data to query.

Solution: My solution to this problem was to re-import a fresh version of the csv file into the mongo database. This fixed all the query problems I was having. Another suggestion would be to have a duplicate database to complete any necessary testing to ensure no sensitive data is being deleted/manipulated.

1. Challenge: I encountered some issues when trying to implement various queries, filters, or conditional statements with some of the csv rows and I had no idea why they weren’t responding as expected. This was because the imported csv utilized some acronyms with some of the data. For example, when trying to query the dob breed ‘Doberman Pinscher’ I was getting 0 results. This was because the name was shortened to ‘Doberman Pinsch’.

Solution: My solution here was to go to the source csv file and investigate to see exactly how the data was entered and troubleshoot accordingly based on those findings. Through this method is how I found they used many acronyms for longer names/descriptions for the rescue animals in the csv file.