# CS 340 README

## About the Project/Project Title

This project was commissioned by the Grazioso Salvare company to help them identify shelter dogs in the Austin, TX area for search-and-rescue training. They want the software to be free and open source so that similar organizations in other areas can make use of its features as well.

## Motivation

The motivation behind the project is that we want to make it easier for companies like Grazioso Salvare to help the dogs and people around them. It helps the dogs in the shelter system by making it easier to find ones that would qualify for search-and-rescue training programs or any other kind of service dog training programs. Finally, with being able to identify dogs for service training programs easier, more dogs will get trained. This leads to service dogs being available when the need arises such as during natural disasters.

## Getting Started

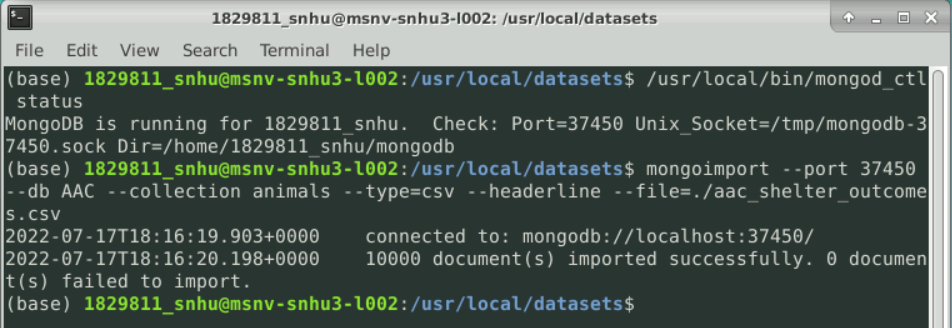
In order to get started with this project, you will first need to follow the instructions in the installation section to get MongoDB and Python installed. Please see the below steps to recreate the database and Python module.

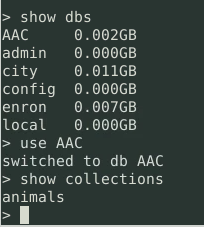
* DATABASE SETUP
  + To set up the database, you will need the appropriate csv file with the animal information. For this, you can use the Austin Animal Center (AAC) Outcomes data set (aac\_shelter\_outcomes.csv).
  + In order to create the database, you need to open the terminal and navigate to the directory where the csv file is located.

EX code:  
cd /usr/local/datasets/

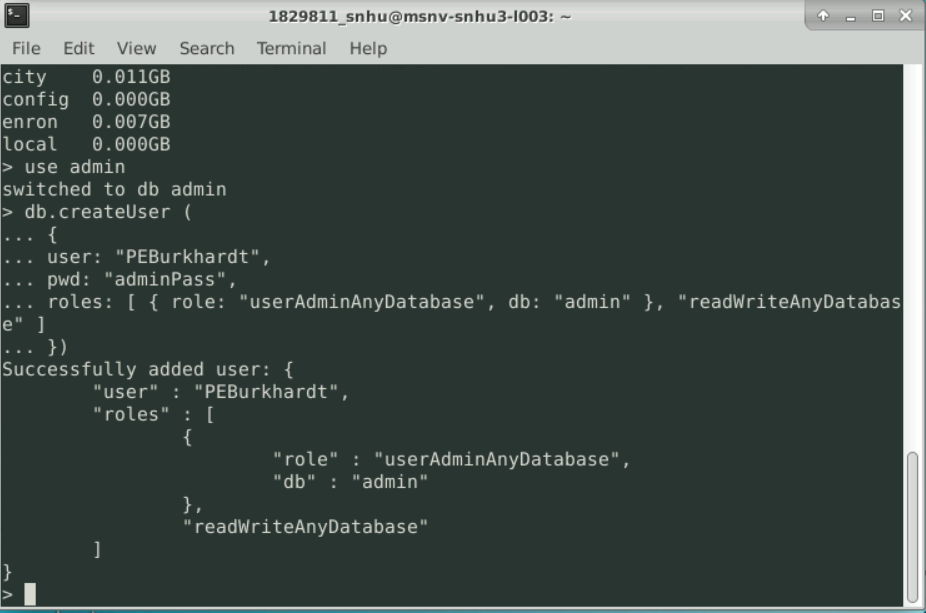
* + Next, you import the csv and create the database via the following command:

mongoimport --port PORTNUMBER --db DBNAME --collection COLLECTIONNAME --type=csv --headerline --file=./aac\_shelter\_outcome.csv  
Replace PORTNUMBER, DBNAME, and COLLECTIONNAME with your port number and desired names. You should get a log stating the connection to the MongoDB instance and how many documents were imported successfully and/or failed to import.



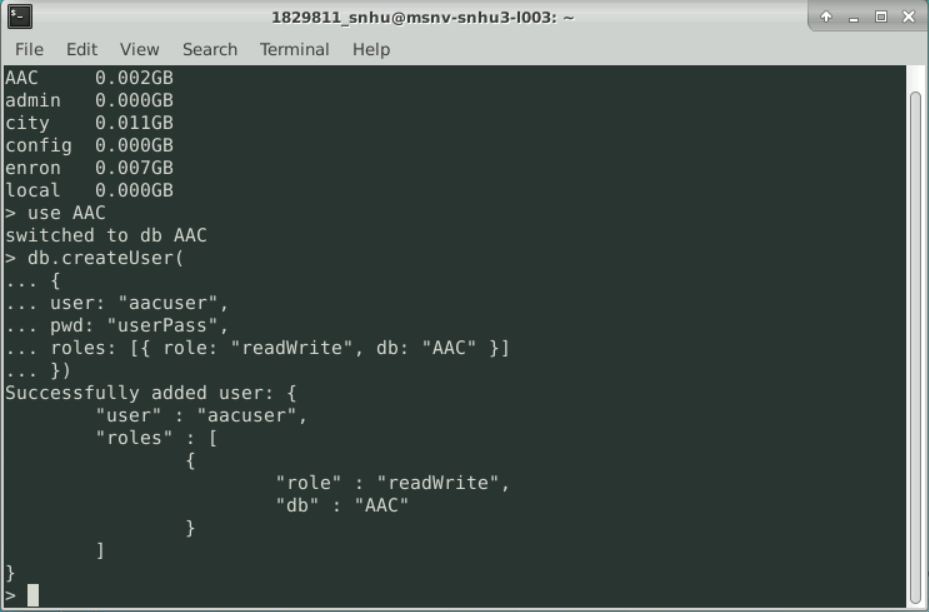
* + From here you can start your MongoDB instance and get into the command line using the commands:  
    /usr/local/bin/mongod\_ctl start-noauth  
    and  
    mongo
  + Ensure that the database was created/imported correctly by using the commands below (for the example, the database name was AAC and the collection name was animals):  
    show dbs  
    use AAC  
    show collections  
    
* USER ACCOUNT SETUP
  + ADMIN ACCOUNT
    - Start the Mongo instance using the mongo command.
    - Navigate to the *admin* database using the use admin command.
    - From here use the following command to add a user with administrative privileges:

db.createUser({   
 user: “admin”,   
 pwd: “password”,   
 roles: [{ role: “userAdminAnyDatabase”, db: “admin”}, “readWriteAnyDatabase” ]  
})

You can see the example below where the username is “PEBurkhardt” and the password is “adminPass”.  


* + READ/WRITE USER ACCOUNT
    - For a regular user account, you follow the same steps as above, except the roles will be different. In this case, we made a user account for someone using the *AAC* database we created previously. For this one, use the command:

db.createUser({   
 user: “aacuser”,   
 pwd: “password”,   
 roles: [{ role: “readWrite”, db: “AAC”}]  
})



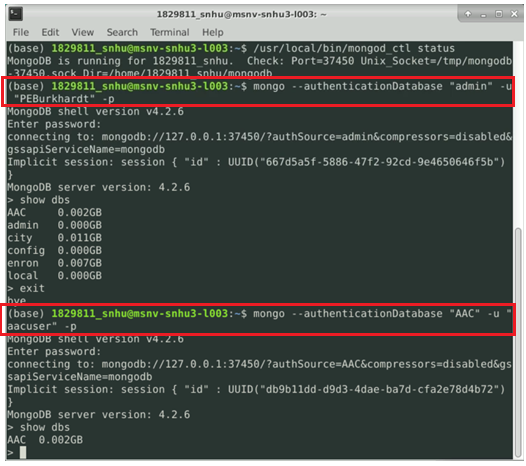
* LOGGING INTO THE DATABASE
  + In order to log into the database as a user, first exit the database using the exit command.
  + From here, restart the database using the following commands:   
    /usr/local/bin/mongod\_ctl stop

/usr/local/bin/mongod\_ctl start

* + You can then log into the database using the following command:

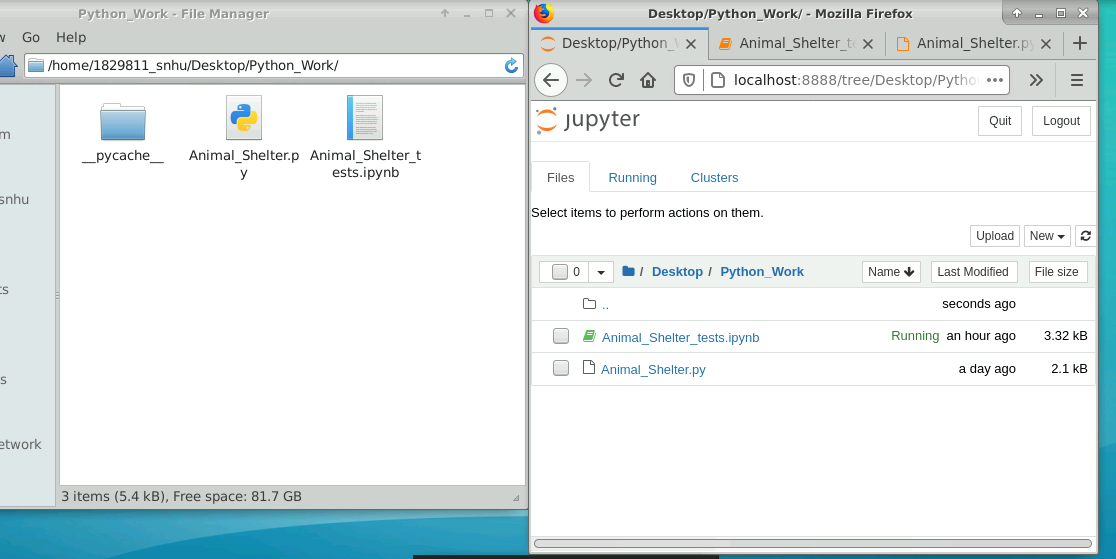
mongo --authenticationDatabase "DBNAME" -u "USERNAME" -p

Replace DBNAME with the database used when creating the user account and replace USERNAME with the username. You can see in the below image the commands used to log into the admin account and user account that we created in previous steps.



One thing to note is that the admin can see all of the databases when you use the show dbs command while the aacuser account can only see the database that they have privileges for.

* + For any issues creating a user account, see the [MongoDB documentation for creating users.](https://www.mongodb.com/docs/v4.2/tutorial/create-users/)

* PYTHON MODULE SETUP
  + Pull the Animal\_Shelter.py and Animal\_Shelter\_tests.ipynb from the GitHub repository or your cloned version of the repository.
  + Navigate to these files in a program like Jupyter Notebook and open them.  
    
  + Once they are open, you can make changes to the .py file and test the changes/functions in the .ipynb file.
  + NOTE: you don’t NEED to use Jupyter Notebook, but it makes editing and testing the project files easier.

## Installation

*List the tools you need to use the software and how to install them.*

The tools needed for this project are as follows:

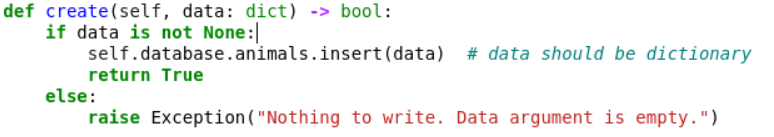
* MongoDB (v4.2.6) – Chosen for its ease of use and ease of scalability.
  + [MongoDB installation.](https://www.mongodb.com/docs/v4.2/installation/)
* Python (v3.6.9) – Chosen because of its ease of use for beginners and the fact that it is free and open source.
  + [Python installation.](https://wiki.python.org/moin/BeginnersGuide/Download)
* PyMongo (v3.10.1) – Chosen because it is the official MongoDB driver for Python applications interfacing with MongoDB databases. Easy to use and built specifically for Python + MongoDB.
  + Open the command prompt or terminal once Python has been installed
  + Run the command:  
    pip install pymongo==3.10.1

## Usage

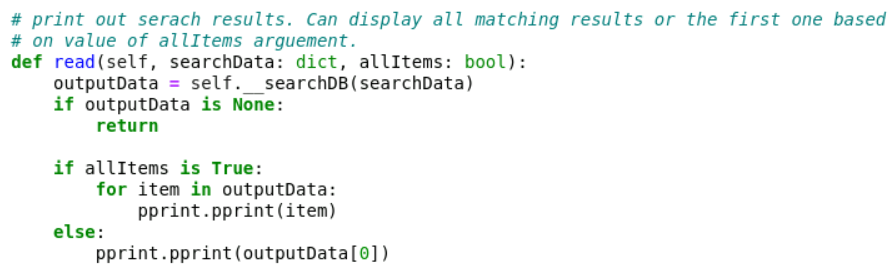
The main part of this project acts as a CRUD (create, read, update, delete) application that interfaces with the MongoDB database.

### Code Example

* CREATE
  + This portion of the module is used to add entries to the database. It takes a dictionary containing the data to be added as an argument.



* READ
  + This portion of the module is used to read entries from the database. You can either read a whole list of entries matching the search query or read the first entry in the list of matching results. You select the output based on the second argument of the read() command. True for outputting all matches, False for outputting only the first match.



* UPDATE
  + This portion of the module is used for updating documents within the database. It takes two arguments, both being dictionaries. The first one is the parameters you wish to search for and the second one is the parameters which you wish to update. The function has handling for “None” arguments and if the search comes back empty. Also, it will ask the user if they wish to delete multiple or single documents if more than one meet the search criteria.



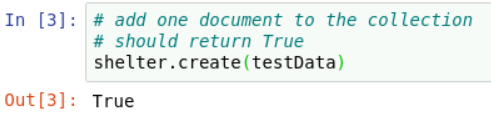
* DELETE
  + This portion of the module is used for deleting documents out of the database. It is modeled and performs almost exactly the same as the update function except it only takes one argument. That argument is the search parameters for the documents(s) you wish to delete.

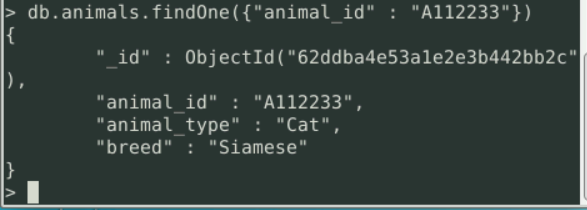


### Tests

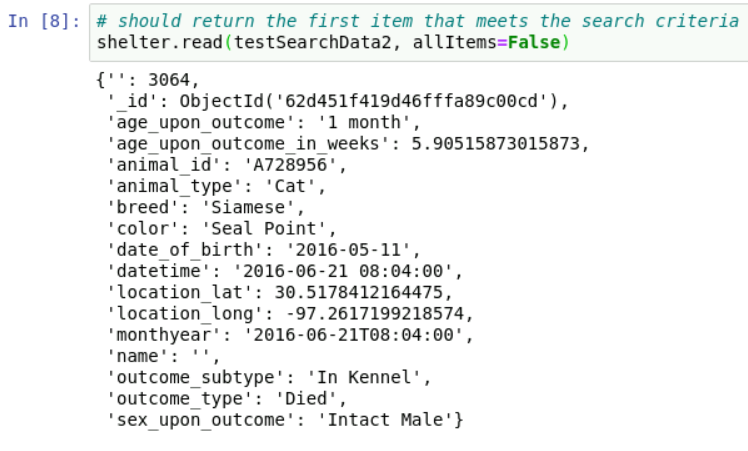
NOTE: Ensure that the MongoDB instance is running BEFORE testing.

To test the Python module, use the Animal\_Shelter\_tests.ipynb file.

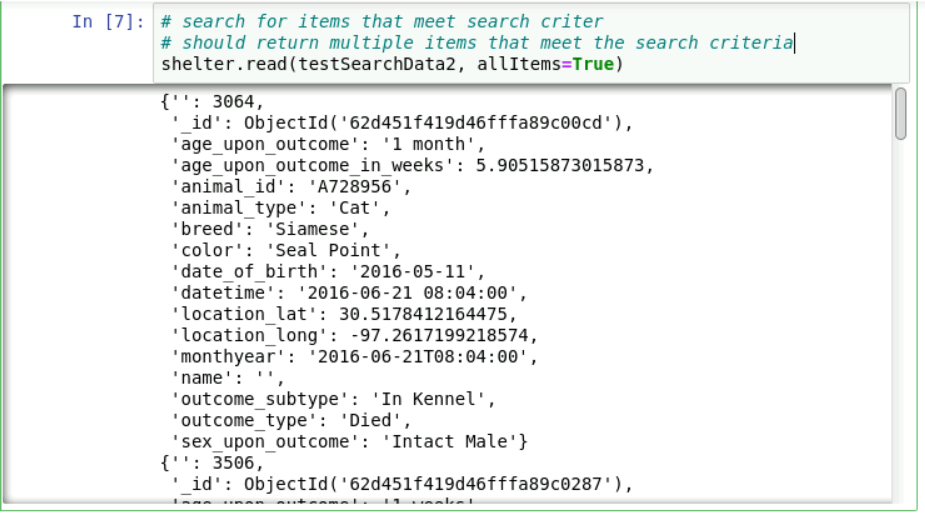
* Initialize the class and test variables:  
  NOTE: You can use an account that you set up during the user account setup section OR you can start the MongoDB instance in “noauth” mode and omit the username and password.  
  
* Select the cells you containing the tests you wish to run and hit the “>| Run” button. It is advised that they are tested in the order they are in or else unexpected results may occur. An example is running the test to display “data not found” output after you have run the test where the data is created. The previous test will return the created data rather than the expected error output.
* EXAMPLE: running the create() test with the test variable initialized in a previous step.  
    
    
  MongoDB query showing the database entry:

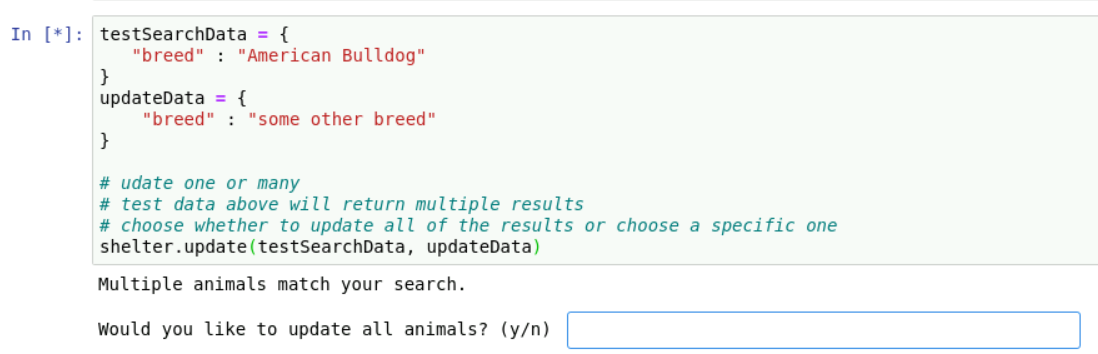
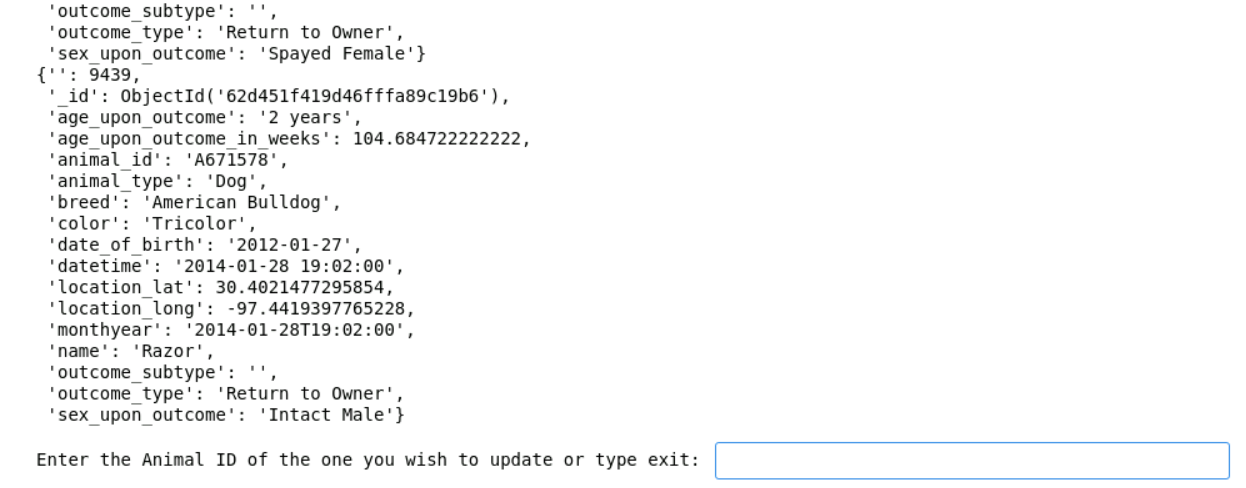


* EXAMPLE: reading the first entry from that database that meets the search criteria:

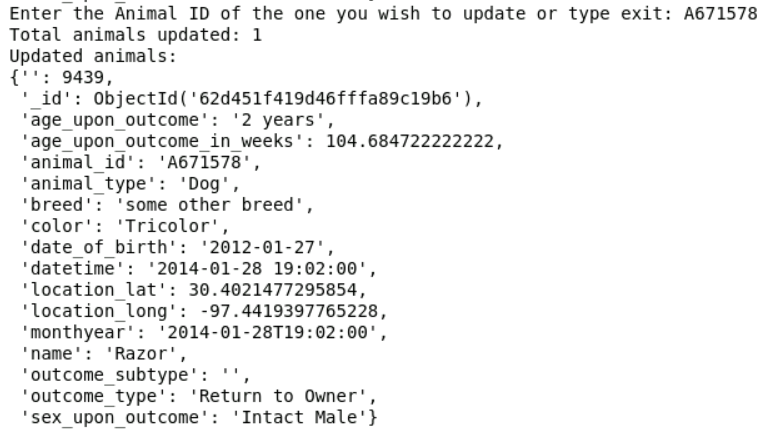


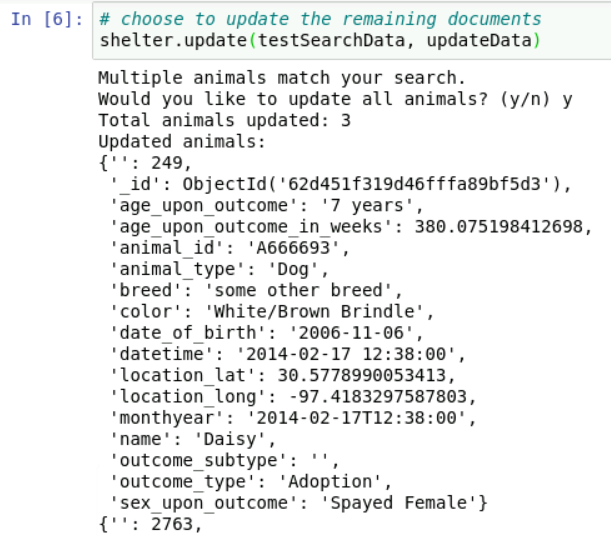
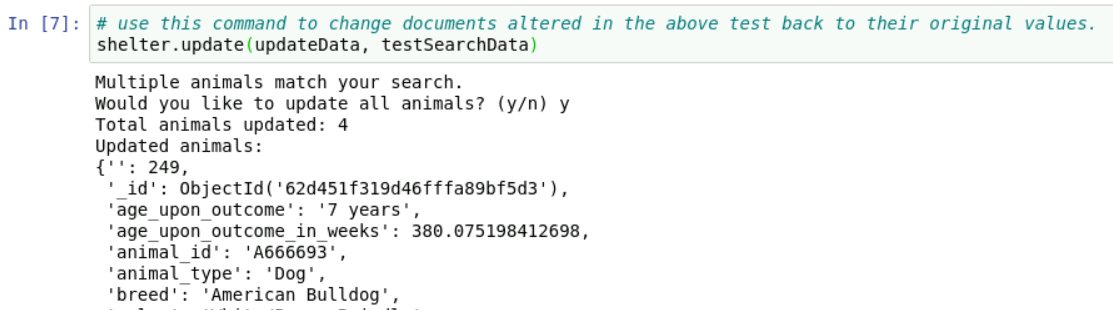
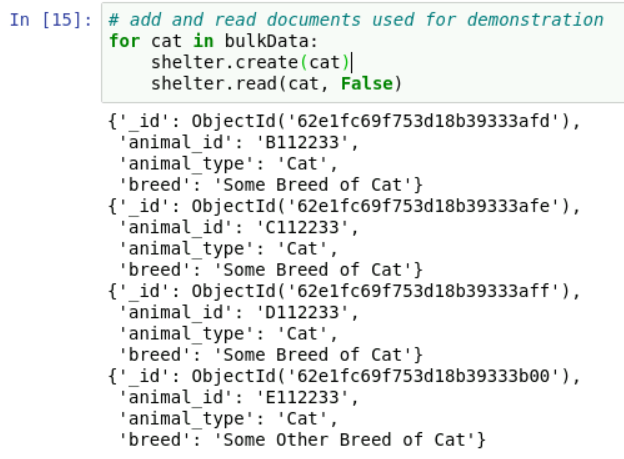
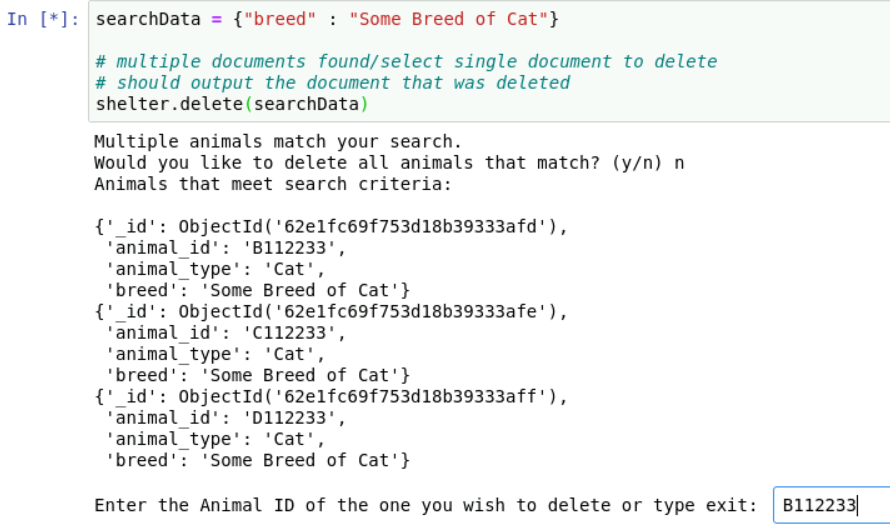
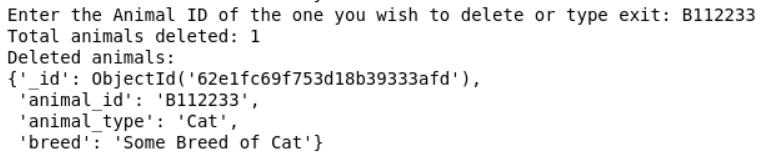
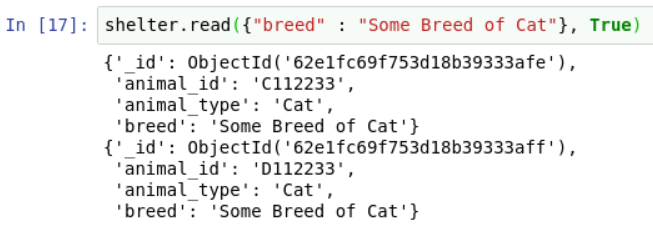
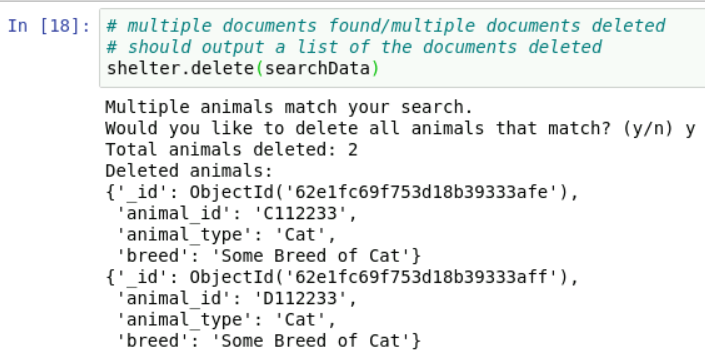
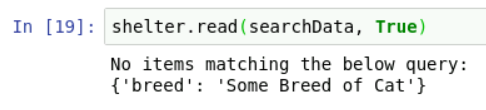
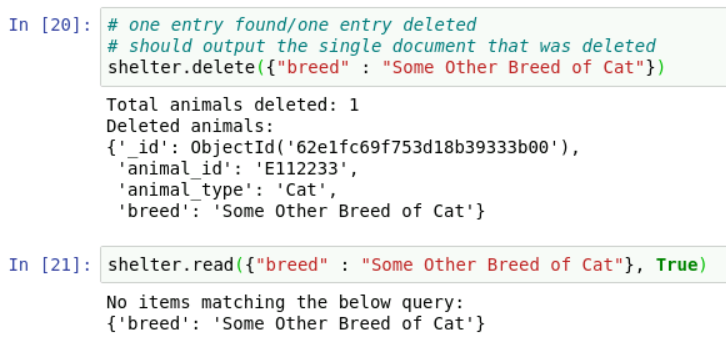
* EXAMPLE: reading multiple entries from the database (note the multiple IDs and scroll bar, indicating there are multiple entries that meet the criteria of the search):



* EXAMPLE: running the update function when multiple entries meet the search criteria prompts the user with a question on whether or not they want to update all entries or just one. Selecting just one will further prompt for the Animal ID of the one the user wishes to update or to type “exit” to exit out of the function:  
    
    
  “n” was entered into the box:  
  

Upon entering a valid ID, that animal’s record will be displayed with the updated information along with a conformation message about the total number of animals

updated:  


* EXAMPLE: running the same command but entering “y” for the prompt asking if the user wishes to update all entries will update all the entries and display the updated entries along with the count of how many animals were updated:  
    
    
    
    
    
  NOTE: run the command under that one and enter “y” for the prompt to change all of the updates from the previous two commands back to their original values. This is accomplished using the same update() command, but swapping the arguments.  
  
* EXAMPLE: The delete() command works exactly the same way as the update() command. For this one we will add multiple documents to use for a demonstration.:  
    
    
    
    
    
    
  We can first try deleting a single cat from the ones we added:  
    
  After entering the ID:  
    
    
  We can confirm the delete by running our read() command, which should only return two documents:  
    
    
    
    
    
    
    
    
    
    
    
  Running the delete command again, but selection “y” will result in the other two documents being deleted:  
    
    
  Running the search again should result in “no items matching”:  
    
    
  Finally, if you run the delete() command and only one document in the database matches the search criteria, it will be automatically deleted:  
  

## Roadmap/Features (Optional)

* TO DO:
  + ~~Implement update() functionality.~~
  + ~~Implement delete() functionality.~~
* TO FIX:
  + ~~Roll realAll() and readOne() into a single read() command where a second argument determines whether a list of entries or a single entry is returned.~~

## Contact

Your name: Preston Burkhardt