# CS 340 README Template

## About the Project/Project Title

animalshelter is a wrapper for pymongo aimed for specific access to the Austin Animal Center’s (AAC) animal database stored within a MongoDB database. This class, which includes CRUD functionality, is used as middleware between AAC’s database and Grazioso Salvare's web interface.

## Motivation

The client Grazioso Salvare is a rescue animal company that trains search and rescue dogs. Grazioso Salvare has partnered with the animal shelter, Austin Animal Center, who has a number of animals and data about those animals stored in a MongoDB database. Grazioso Salvare needs to interact with data about AAC’s animals through a web interface that connects to AAC’s MongoDB database.

## Getting Started

To import and use animalshelter in a python script, ensure animalshelter.py is either in the local directory of the script or in your python-specified third-party module directory. Review the *Installation* and *Usage* sections to install necessary dependencies and starting code examples. animalshelter only connects to the AAC database -> animal collection within MongoDB, and **requires a username and password to already be set up within the database by an administrator.** Check with administration to verify the correct port number and enter it as shown in the *Usage* section if it differs from the default 54161.

*Common issues:*

The username/password is not entered correctly (or does not exist) if an authorization error is returned.

The port number is likely incorrect if a WinError "No connection could be made because the target machine actively refused it." is returned.

*Default third-party directory:*

Windows: <install directory>/Python/Python<version>/Lib/site-packages

Mac / Linux: /usr/local/lib/python<version>/dist-packages

*Brief usage descriptions:*

To **import** -

From animalshelter import AnimalShelter

The **create** method inserts a new document into the database. A list of documents may be used and sub documents are also acceptable.

The **read** method queries the database. Pass a dict parameter as a filter. Ex. query = {'name': 'shadow'} returns all documents whose name is shadow. Multiple key:value pairs are acceptable.

## Installation

*Dependencies required to use animalshelter*

python – to run animalshelter.py python script

pip – to install additional libraries

pymongo – to interact with MongoDB

*Pymongo Installation*

*Open terminal or cmd and type:*

pip install pymongo

*If pip is not installed, type:*

python get-pip.py

*If pip is not in the PATH, type:*

python -m pip install pymongo

*NOTE: Python needs to be in the environment table PATH or navigate to python install directory before entering commands.*

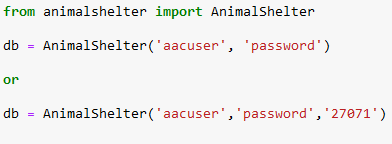
## Usage

***Accessing the database:***

AnimalShelter(String <username>, String <password>, String <port>=54161

Connect to the database by creating a new class object and supplying username and password.

Port number is optional with default of 54161.



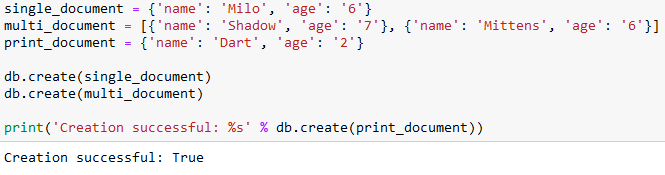
***Creating documents:***

bool create(dict <document>)

bool create(list <document\_list>)

Inserts a document or list of documents into the database.

Returns True or False depending on whether the insert into the database was successful.

******

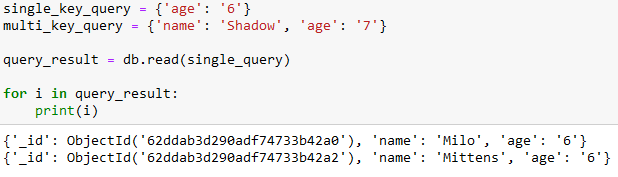
***Reading documents:***

pymongo.cursor.Cursor read(dict <filter>)

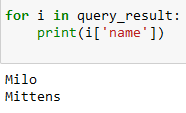
Retrieves a document that matches the passed filter.

Mutliple dict keys can be used.

Returns a pymongo cursor that can be iterated to access all matching documents.



Accessing elements within cursor:

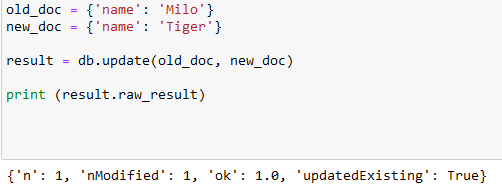


**Updating documents:**

pymongo.results update(dict <filter>, dict <updated data>)

Updates one or more documents that match the filter with the new data.

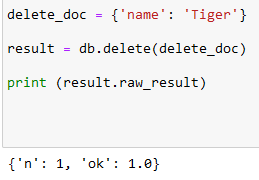
Retains other data in document that was not updated. (Uses {$set: {<new data>})



**Deleting documents:**

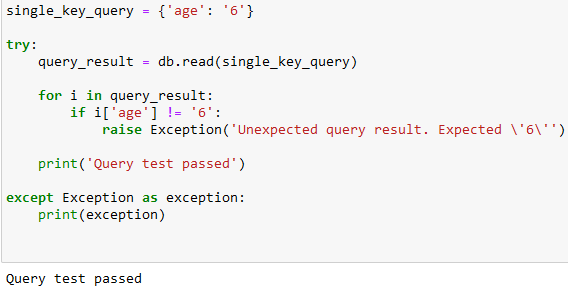
pymongo.result delete(dict <filter>)

Deletes one or more documents that matches filter.



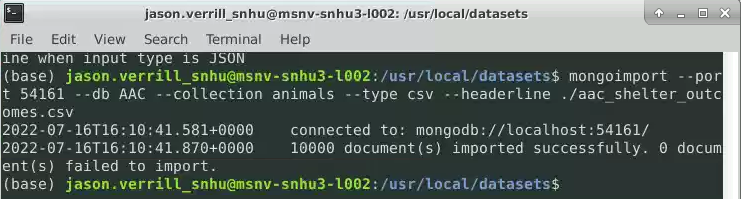
## Sample Test Case:

Create query for all animals of age '6' and test expected age result.



**Screenshots**

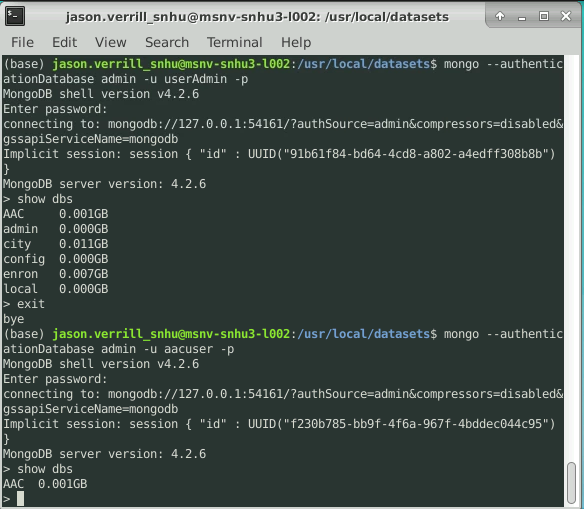
**Importing AAC database**



**Create admin account**

## 

**Admin vs. User Login**



## Contact

Jason Verrill

**Reflection of Challenges**

The most difficult challenge during this project was the learning curve of Dash's components and syntax. Particularly, understanding where the callbacks got its input and output, how multiple callbacks could not use the same output, and how the callbacks returned its data was the most challenging. In addition, working with pandas' dataframes was difficult as well, especially since I was unable to use print statements to understand the data that was inside the dataframe and how it was organized. To overcome these challenges, I reviewed a vast number of sources that explained Dash and how it worked and found creative ways to view dataframe objects. I also worked the project in multiple environments to see how the code's results would change. For example, I found that the geo-map would not appear on my personal computer, but that with the same code, it worked properly on the virtual server. In many other cases, it was the opposite, where code would function locally, but not on the virtual server. Some of these issues were not resolved, but I suspect that the issues were related to browser settings, and library version differences.