# AnimalShelter Python CRUD class

## About the Project/Project Title

The Animal Shelter module is a Python class designed to perform basic CRUD operations within the Mongo database framework. Its methods allow the instantiating class to access and maintain the database without the need to lengthy and repetitive in-line database calls.

## Motivation

The Animal Shelter class was created to support an AAC shelter in the maintenance of their local database, but is designed to be useful for any small organization that uses the Mongo format. Planned updates to the project include the creation of more advanced method functions (such as indexing), as well as adding support for multiple logins.

## Getting Started

To get started with the AnimalShelter class, the local database administrator must add a set of user credentials to the AAC database with read/write privileges. The default values for these credentials can be found in the image of the class’s Init method, but this can be changed to reflect your local database’s values. You will also need to obtain the connection URL for your local system’s database, and replace the HOST value with this address. Once these steps have been completed, you should be able to instantiate and run the AnimalShelter class.

In setting up the read and write sections of the class, the most difficult step is verifying database connection. Because of the way that Python handles instances of MongoClient, it can be difficult to identify when the connection has not been made successfully. Calls that result in a failed connection may simply exit without result, rather than crashing or returning an error. For this reason, connection should be verified by calling the *verifyme()* method before beginning operations. This method returns a true/false result that can be easily used to determine whether the database connection is correctly functioning.

## Installation

The AnimalShelter class is designed to work with MongoDB. Prior to installation, the user should verify that MongoDB is installed on their local network, and that it is functioning correctly. To install the class, you will need the following information:

Username: This will be determined by the database administrator. Default value is ‘aacuser’ but the class can be instantiated with ‘USER=\_\_\_’ to reflect your local database setup.

Password: Also determined by the database administrator. The default value is ‘userpass’ but the class can be instantiated with the argument ‘PASS=\_\_\_’ to reflect your local account.

Host: This is the address at which your local MongoDB server can be reached. When the class is instantiated, use the argument ‘HOST=\_\_\_’ with the address of your local server to enable connection.

Port: Similar to above. Instantiate the class with ‘PORT=\_\_\_’ adjusted for your local server port.

Database: The name of the database to access on the Mongo server. The default value is ‘AAC’. Change this value by instantiating with ‘DB=\_\_\_’.

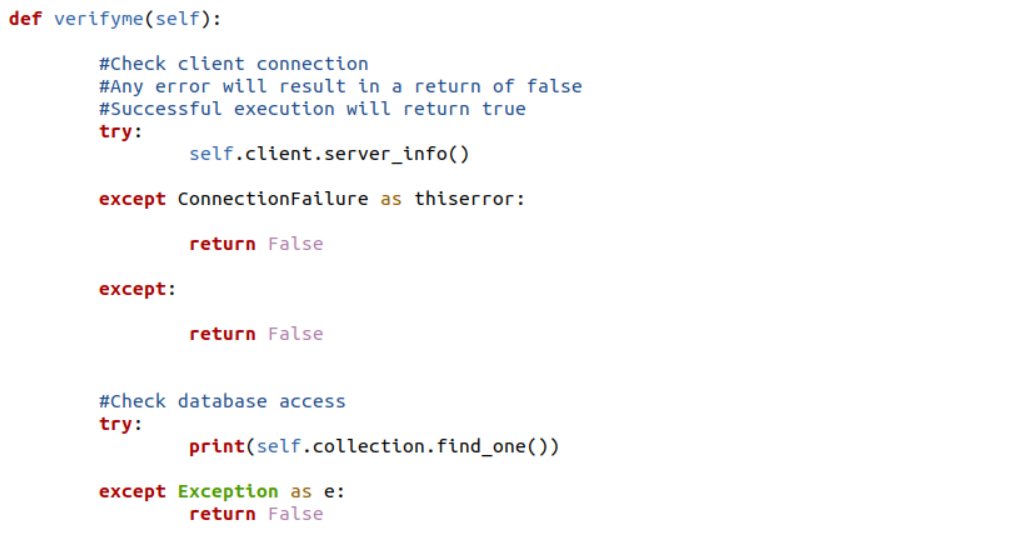
Collection: This is the name of the collection within the Mongo database that the program will access. This is changed by instantiating with the argument ‘COL=\_\_\_’. Default is ‘animals’.

## Usage/Code Examples

The AnimalShelter class includes the following methods:

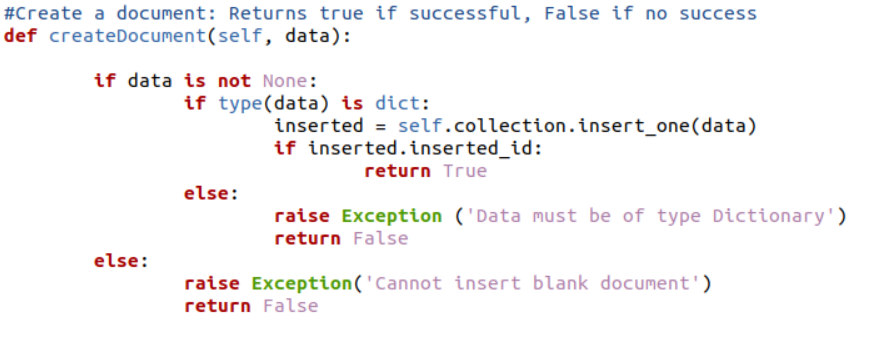
*verifyme()*

Checks database access. Returns True if successful, or False if unsuccessful.



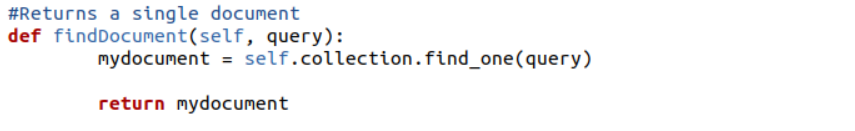
*createDocument()*

This method must be passed a dictionary as an argument. Returns True if the insertion was successful, or False if it was not.



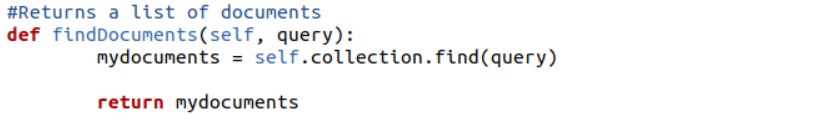
*findDocument()*

This method takes a SQL query as argument, and returns a single document (the first to match the given query)



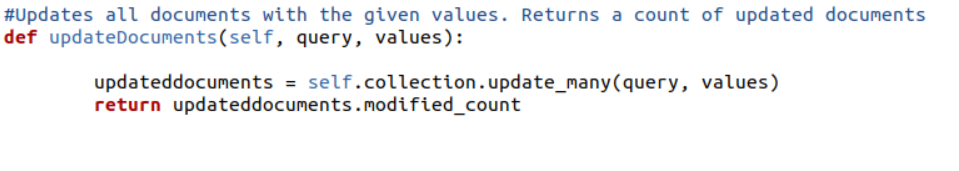
*findDocuments()*

This method takes a SQL query, and returns a list of documents (all documents that match the given query)



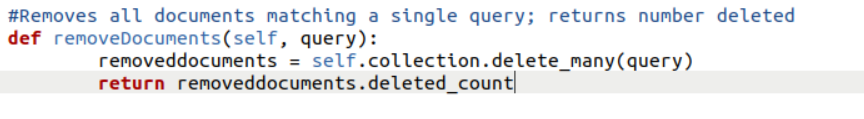
*updateDocuments()*

This method takes two arguments: The first is a standard SQL query, indicating which documents should be updated. The second is a modification query, such as {$set{animal\_name:Puddles}}. Documents matching the first query will be updated using the second. The method returns a count indicating how many documents were modified.



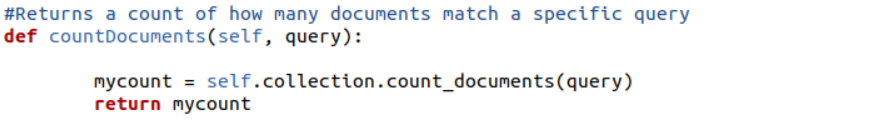
*removeDocuments()*

This method removes all documents that match the given query. It returns a count indicating how many documents were removed.



*countDocuments()*

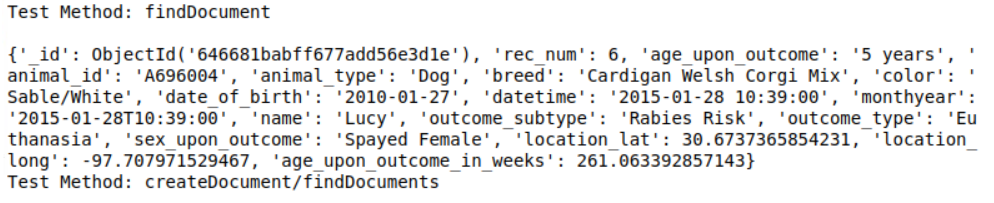
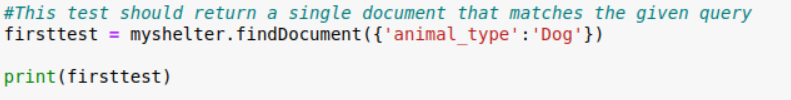
This method returns a count indicating how many documents match a given query. Note that this method returns *only* a number – not the documents themselves.



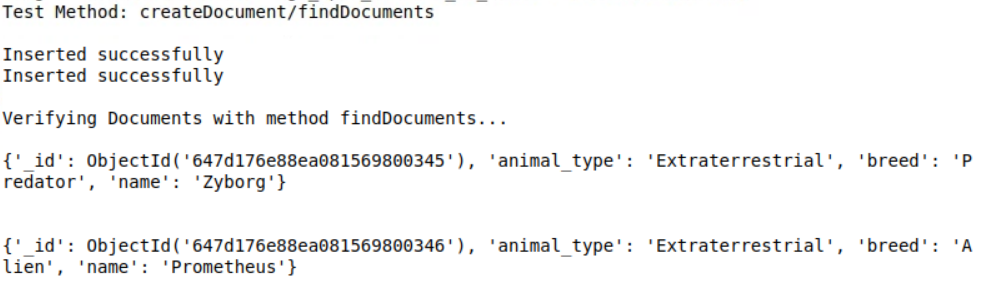
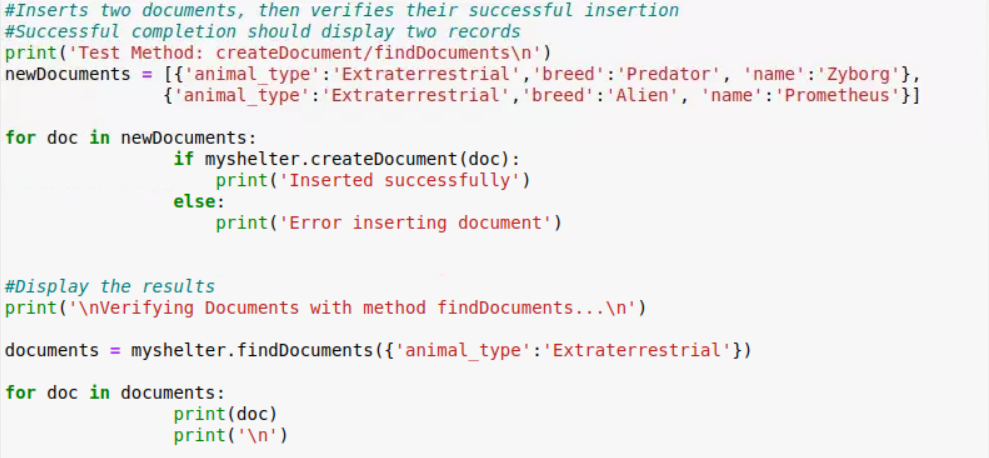
### Tests

The following are examples of tests for each class method, and the expected outcome.

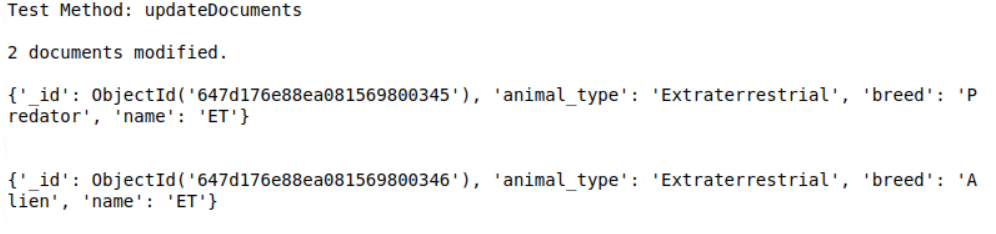
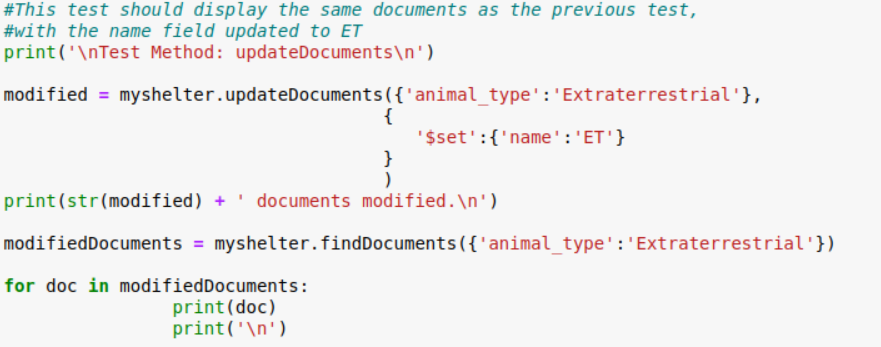
Test for method findDocument()



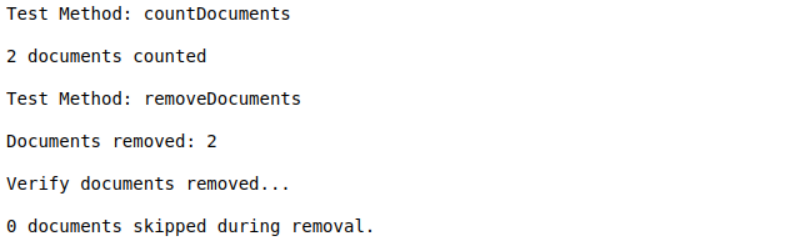
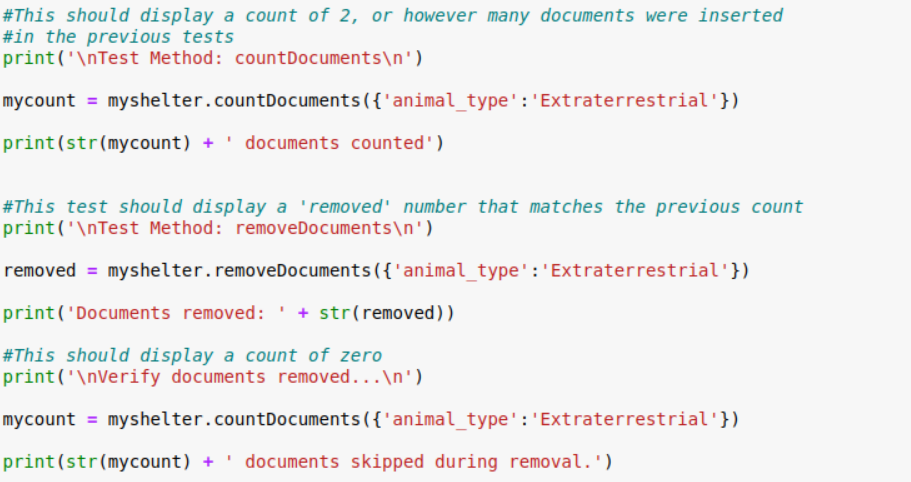
Test for methods findDocuments() and createDocuments()



### Test for method updateDocuments()

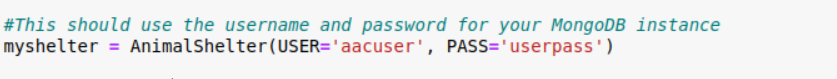


Test demonstrating removeDocuments() and countDocuments() functions



### Screenshots

Sample class instantiation, with username and password included (see section on installation for a list of arguments that can be used).



## Contact

Joshua Morris