# CS 340 README – AnimalShelter Python class

## About

The purpose of the AnimalShelter Python class is to provide an interface between the MongoDB database – AAC – and an external application. It is currently set up for a local Python development. In interacting with the database, the AnimalShelter class connects using a locally set up username and password, and has functionality to insert, find, update, and delete documents.

## Motivation

This project exists to be a reusable interface with the AAC database, allowing multiple users to create different applications from the same database. One such application will be a dashboard that allows users to sign into the database and query it using a simple interface. The AnimalShelter class will allow for abstracting the pyMongo logic and syntax and interface with the database away from the end user of the application.

## Usage

The usage of the AnimalShelter class is simple. First import the class:

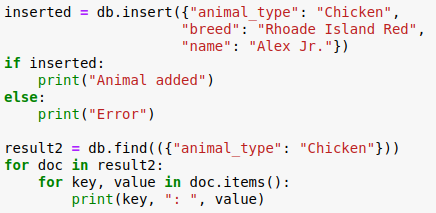


An AnimalShelter() object can be instantiated using your Mongo username and password:

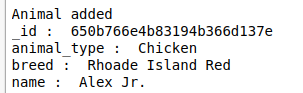


You can then access the insert() and find() functions. Each function takes a dictionary as an argument. The insert() function returns ‘true’ if it is successful, ‘false’ if not and find() returns a list of dictionaries containing the information from each document found:

Sample code:



Result:

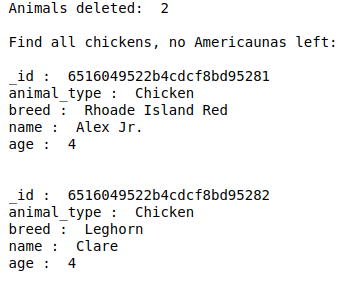
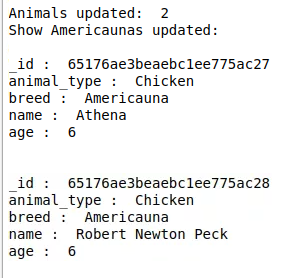


Additionally, there are update() and delete() functions. Update() takes two dictionaries as input – one for filtering documents to update and a second with the information to update. Delete() takes one dictionary as input to identify which documents to delete. Both have an optionally Boolean representing if the user wants to update/delete all the documents that meet the criteria or just one. Update() has this set to True by default, delete() has it set to false. This is designed to ensure that deleting many documents must be an intentional act. Both methods return an int representing the number of documents affected.

Sample code:



Result:



## Installation

The tools used on this project were MongoDB, Spyder IDE, and Jupyter Notebook. MongoDB was used to import the dataset and set up a username and password. Spyder was used to develop the AnimalShelter class. Jupyter Notebook was used for testing. Python libraries used were pymongo – MongoClient - for interacting with the database and collections.abc – Mapping – for validating data is input as dictionaries.

## Getting Started

The class uses the Model View Controller (MVC) software design pattern. This is a pattern where the model holds the data, the view presents the data to the end user, and the controller is the glue that holds the two together. For this application the model is the database, the view is the dashboard presented to the end user, and the controller is the AnimalShelter class.

The class interacts with the MongoDB database. This database has been imported to MongoDB and requires a username and password to gain read/write access:

Loading the database:

**A computer screen with white text

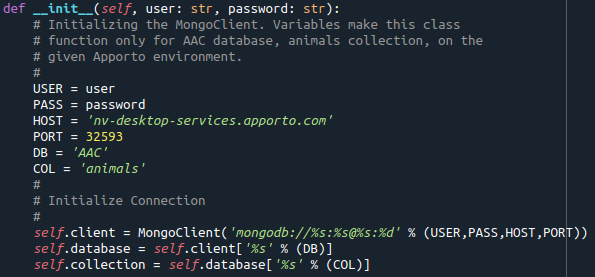
Description automatically generated**

New user login and verification:

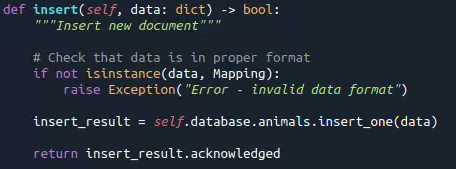
A screenshot of a computer screen

Description automatically generated

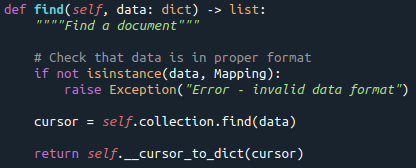
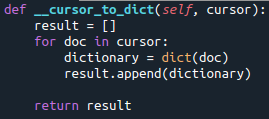
In addition, pymongo requires input for host, port, database, and collection. These are all set up in the \_\_init\_\_ constructor:



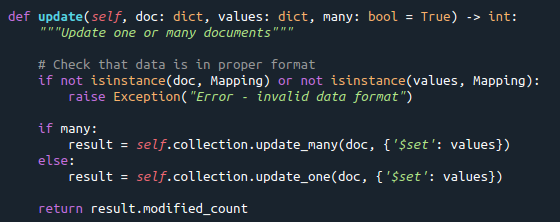
The insert(), find(), update(), and delete() functions are simple methods that take dictionaries as parameters and return a result. All have data-type checks to ensure input is in the correct format. The insert() method uses the ‘acknowledged’ parameter to return ‘true’ if successful, ‘false’ if not:



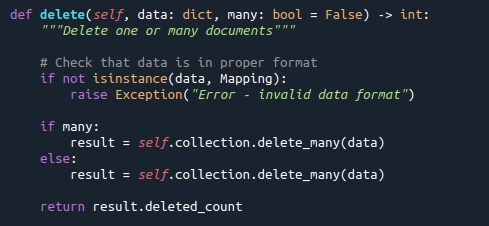
The find() method returns a list of dictionaries. The query through pyMongo returns a cursor object. That cursor is run through a private method to turn it into a list of dictionaries:

The update() function uses the ‘modified\_count’ value to return the number of documents modified by the pyMongo command:



The delete() function uses the ‘deleted\_count’ value to return the number of documents modified by the pyMongo command:



## Contact

David France

[david.france@snhu.edu](mailto:david.france@snhu.edu)