# CS 340 README

Client/Server Interaction

The objective of this project is to import a database from Austin Animal Center, using MongoDB, that can interact with client-side Python code. Implementations include the following:

* Importing database
* Creating/authenticating users
* Indexing
* Basic Python CRUD

Inspiration for this project comes from current trends in development environments and patterns of use. Code reusability is extremely advantageous. By developing the code used for one project to be easily adapted for other projects, you have created an advantage for later projects. With CRUD functionality being perhaps the most used function in coding, it would make sense to allow for reusable code.

Necessary steps to get this up and running locally are as follows:

1. Import the AAC database

**A picture containing text, screenshot, font

Description automatically generated**

1. Create a new user with the appropriate roles

A picture containing text, screenshot, font

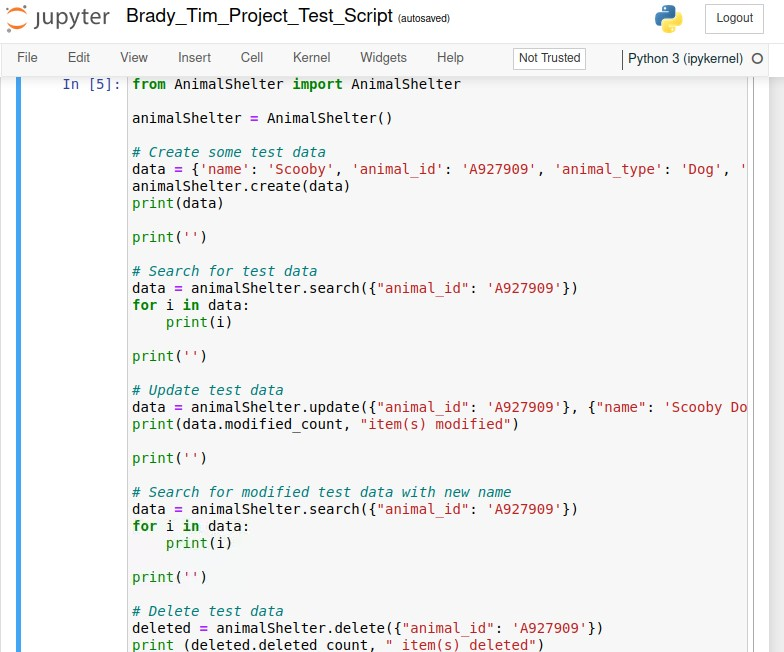
Description automatically generated

1. Create a .py file to store your CRUD functions for easy reusability

A screenshot of a computer program

Description automatically generated with medium confidence

1. Call your functions in the IDE of your choice

Calling all 4 CRUD methods to create an item, search for that item, modify that item, and then delete the item

Output from CRUD tests. A screenshot of a computer

Description automatically generated with medium confidence

Tools Used:

* MongoDB
  + MongoDB was used to create the database and users. A file from AAC was imported into MongoDB where indexes were setup. A basic read/write user was also created and authenticated.
* Jupyter Notebook
  + Jupyter Notebook was used to create the code for the .py file that holds the methods as well as the ipynb file that tests those methods. An additional ipynb file was created for the dash using Jupyter as well.

Libraries Used:

* MongoClient
* ObjectId

The AAC database is vast and holds many different animals. Using this program, you can quickly sift through the data to find any animal that meets certain criteria. You can also add new animals to the database. A dashboard has also been implemented that will allow users access to an organized table of all animals with filtering capabilities, a map showing each animals location, and a graph showing the makeup of all breeds selected.

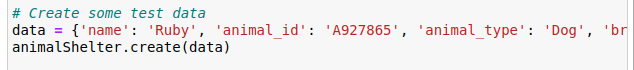
Adding a new dog named ‘Ruby’ to the existing database.

* + First we setup our function to create a new object in our .py file

A screen shot of a computer program

Description automatically generated with low confidence

* + Then we call that function in our Python script



Searching for the new dog ‘Ruby’

* + First create the search function in the .py file

A screenshot of a computer program

Description automatically generated with medium confidence

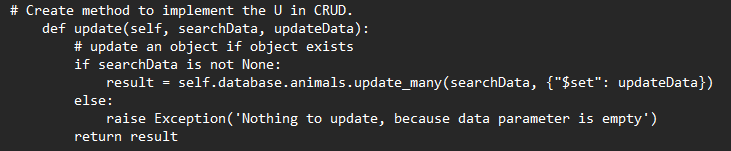
* + Then call that function in the Python script

*A picture containing text, font, screenshot, line

Description automatically generated*

Updating info for a dog that was named ‘Scooby’ but should be named ‘Scooby Dooby’

* + Create the function in the .py file

**

* + Then call the function in the Python script



Deleting the dog named ‘Scooby Dooby’ based on it’s ID.

* + First create the function in the .py file

*A screen shot of a computer program

Description automatically generated with low confidence*

* + Then call the function in your python file

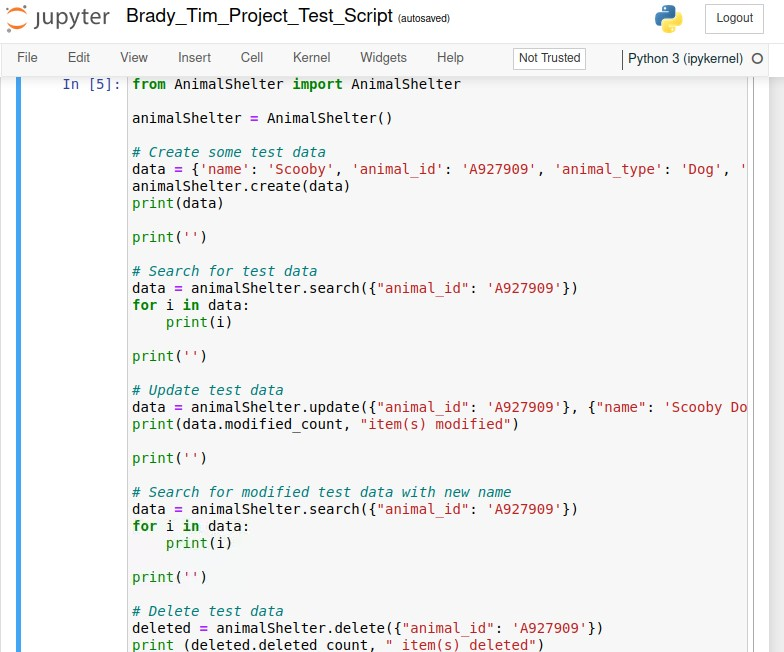


The above instructions will help you setup tests in your IDE to check that the functions within your .py file are working as intended. In the tests we created fictional objects and then called those objects. The result, in our search for Ruby, is at the bottom of the screenshot below where it shows all stored info for Ruby. Additional screenshots show all four CRUD functions examples of their results.

### Screenshots

A screenshot of a computer program

Description automatically generated with low confidence



A screenshot of a computer

Description automatically generated with medium confidence

**Dashboard Screenshots**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**Dashboard Demo**

Below is a short video that demonstrates the capabilities of the dashboard. Features being demonstrated are as follows:

* Implementation of Grazioso and developer logos.
* Table organizing animal data with filtering capabilities.
* Dynamic updating of a pie chart and map pin with the selection of an animal.

[](https://player.vimeo.com/video/837345617?app_id=122963)

## Contact

Tim Brady

Timothy.brady1@snhu.edu