Carmen Kingery

Hari Paruchuri

Client/Server Development

Project Two README

## 10/19/2022

## About the Project/Project Title

What we have here is a simple Python project that provides CRUD (create, read, update, and delete) functions for an example animal shelter database using MongoDB, using Dash for the UI. We currently have all four functions, the create, read, update, and delete functions, available. However, we are open to adding some more tools if requests were made! You can use just the CRUD script on its own by downloading the .py file. You may also use the UI portion of the project by downloading the .ipynb file. The User Interface portion currently only utilizes the read() function of the project. It takes your desired search input from a dropdown menu and displays the results in an interactive table and pie chart, as well as displays the location of a selected item (an animal in our example dataset) on a geolocation map.

## Motivation

What good is a database if you can’t access any of the information? Or update the information? Wouldn’t removing unnecessary documents and inserting new documents also be nice? These are the situations CRUD functions are made for and the exact things we aim to address in this project. In this project we have worked out the code for each function and combined them, as well as other necessary documents, in a single folder for your convenience. We have also included the start of a nice User Interface that utilizes these CRUD functions on your database.

## Getting Started

To get a local copy up and running, follow these simple example steps:

1. Download the “myPyFiles” folder we have made available. This folder will contain all the necessary files needed to run this project.
2. Install a Python IDE, Dash library, and MongoDB if you do not already have them.
3. Import your database into MongoDB.
4. If you are working straight from the database without a user interface, make sure you navigate to the file directory you saved the myPyFiles to and you may begin maintaining your database! Otherwise, simply import animalShelter.py from the myPyFiles into your user interface code and continue coding!

## Installation

Tools you need to use the software and how to install them.

* MongoDB can be installed by following the instructions on their website: <https://www.mongodb.com/docs/manual/installation/>
* A Python IDE such as Jupyter Notebooks, which we used for this project, can be installed following this link: <https://jupyter.org/install>
* Dash can be installed by following the instructions in this link <https://dash.plotly.com/installation>

## Usage

### Code Example

The code itself is really simple. The point of this project is that all these functions are all in one place and you don’t have to waste time thinking about how to make them yourself.

For the create function we require an input in the form of a dictionary, or else it will throw an error back saying you gave it nothing to create. Therefore, once you input a document, the function will take it and insert() it within a specific database folder “animals”. If the insert was successful it will return “True”. Here is an example of the code:

*Graphical user interface, text, application

Description automatically generated* For the read function we require at least a key/value pair in order to look up the document you are searching for, otherwise the function will once again throw back an error message saying you didn’t provide what is needed to look up anything. When the information is properly given the function searches the “animals” database using the find() tool. If it was successful in finding something it will return the curser, loop through the curser to pull out the document, and then print it to the screen; otherwise, it will send back a message that nothing was found. Here is an example of the code:

Graphical user interface, text

Description automatically generated

For the update function we two dictionary type inputs: a key/value pair to look up the item you wish to update, as well as the update information. Without one or either of these inputs, the function will return an error message saying one of the input parameters are empty. Once you have given the appropriate information the function will search through the database for your document. If the search was successful, it will continue on to update() the document, otherwise it will send back a message saying it could not find your document. Upon completing a successful update, the function searches for your document again, iterates through the returned curser to extract the document from it, and prints it to the screen. Here is an example of the code:

Text

Description automatically generated

For the delete function we once again require at least a key/value pair to look up the item you wish to remove from the database. Without that key/value pair, the function will give you an error message “because the data parameter is empty”. Upon given a proper document to search for, it will find() your document and, iterating through the curser, will save the document to return to the caller after deletion. If the search was successful, the function will then remove the document from the database, print the document that was deleted to the screen, and print how many documents were deleted. Here is an example of the code:

Graphical user interface, text, application, email

Description automatically generated

### Tests

To test the CRUD functions first we need to start up mongodb in our terminal using no-auth to begin with. Now we need to create an instance of the class. You do this by simply calling the class, as shown below:

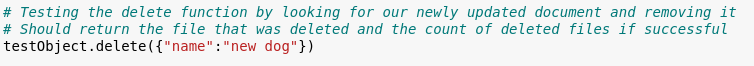


 Now to test the create function all we have to do is call the function with our newly created object by calling testObject.create. We also need to provide an example document for it to insert. Once we have called it, we expect to return “True” for a successful insert:

We test the read function the same way but this time we only need a key/value pair as input. Once we call the function we expect to see our searched for item is returned to us:



To test the update function, we need to call testOject.update() and provide a look up key as well as a key/value to update: {name: new dog, breed: beagle}. It should return our updated document to us:

To test the delete function, we similarly need to call testObject.detete() and provide a key/value pair for it to look up the item we want to delete. It should, once again, return our deleted document, as well as the number of items deleted:

Next, we need to test to make sure the UI is operating properly. To do this we simply run the ipynb file and open the output in a new window. If functioning properly it should look like this:

Chart

Description automatically generatedGraphical user interface, table

Description automatically generated

Graphical user interface, text, application

Description automatically generatedChart

Description automatically generatedNow we will test that the filters work by selecting each one individually. Here is the Water filter:

Graphical user interface, text, application, email

Description automatically generatedChart

Description automatically generatedHere is the Mountain and Wilderness filter:

Graphical user interface, text, application, email

Description automatically generatedHere is the Disaster and Individual Tracking filter:

Chart

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated And here is the Reset filter:

Graphical user interface

Description automatically generated

### Screenshots

Text

Description automatically generatedFirst you need to upload your data set into MongoDB. For this example, we uploaded the Austin Animal Center Outcomes CVS data set:

Next, you will need to make a user and an admin account to access your database in MongoDB. Once you do this, here is an example of what the login screen will look like:

Text

Description automatically generated

Next you should check to make sure the Python script is successfully connecting to your data base. Here is the output of what a successful run of the tests should look like:

A picture containing text

Description automatically generated

## Contact

If you have any questions you can reach me at:

Carmen Kingery: carmen\_kingery@snhu.edu