# CS 340 README Template

## About the Project/Project Title

*The CRUD project is to give access to the aac database in mongodb. This project currently allows create and read access by using the class AnimalShelter in the python file.*

## Motivation

*This project has been created to allow users to stop using the mongo shell to create and query for records. This will give database read and write access in python to use in other programs or code snippets. Ultimately, we will be using this to allow users to submit and read records through a user interface.*

## Getting Started

*To get started you will need to download the CRUD\_Python\_Module.py file and put it into your project directory. Then simply add the following code to the top of your python file that will be using the library.*

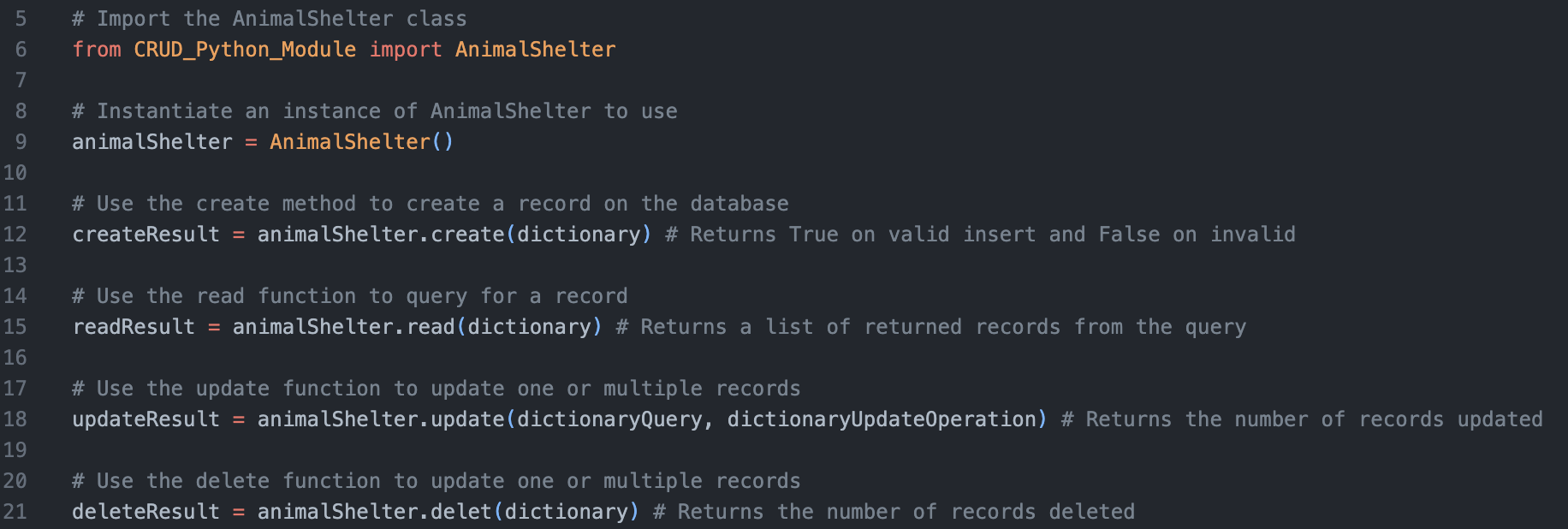


## Installation

*The first file you will need is the CRUD\_Python\_Module.py file. It is the main file with the AnimalShelter class and when downloaded, simply add it to your project to be able to import it to your code. It is currently not hosted anywhere and will not be available to simply install onto your machine or environment. In addition to that, you will also need pymongo installed as this library relies on MongoDB connection.*

## Usage

### Code Example



### Tests

*To run tests, you simply need to run bits of code like the ones shown above. Each return something different. There is a screenshot of our tests we created to test each function at the bottom of the ReadMe file. Create returns True or False. Read returns a list of the returned records. Update and delete return the number of records affected.*

*As you test, we recommend that you print out the results so you can see what is being returned. As they are boolean and list types, be sure to cast it as a string before printing. Update and delete will return in string types.*

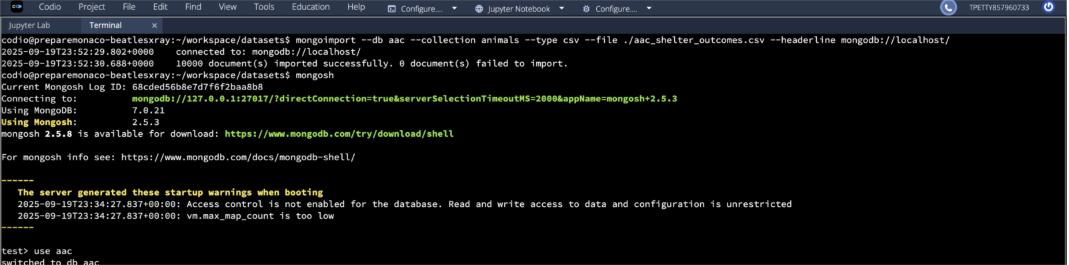
*Additionally, below is shown a complete dictionary with all the attributes needed to create a new animal record. Note that rec\_num needs to be unique. Included in the screenshot below is an example of a query dictionary. You do not need all the attributes when calling the read function to query.*

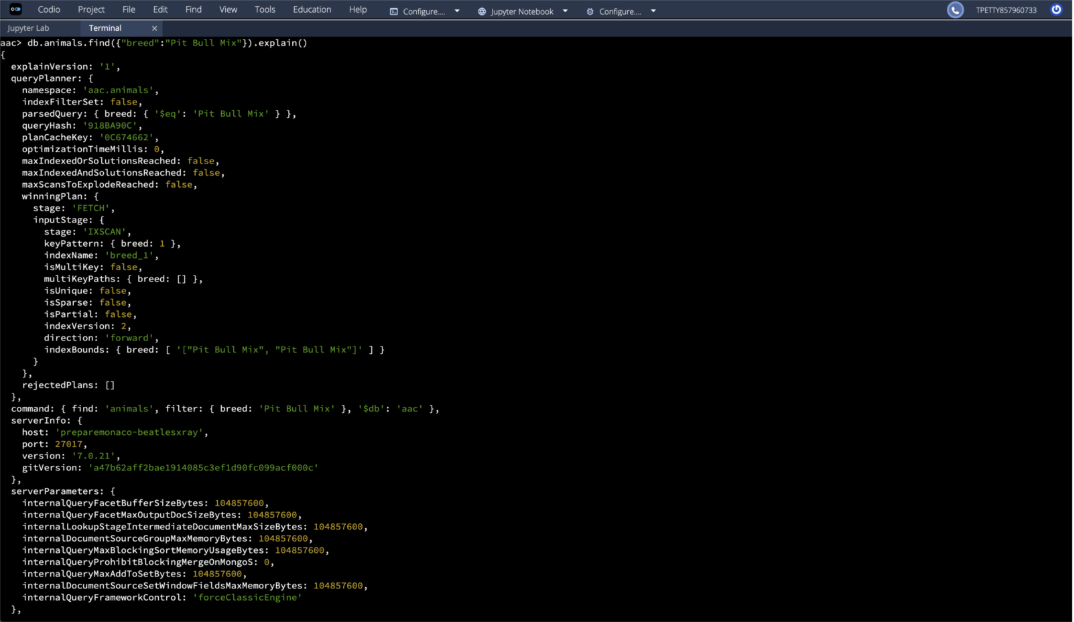
### Screenshots

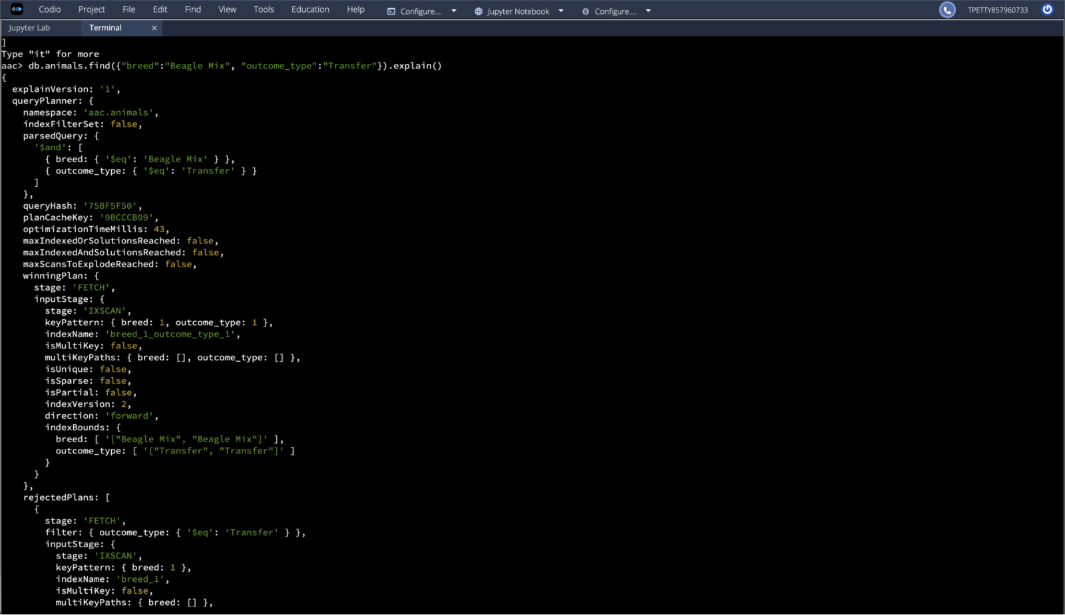


Functional Operations

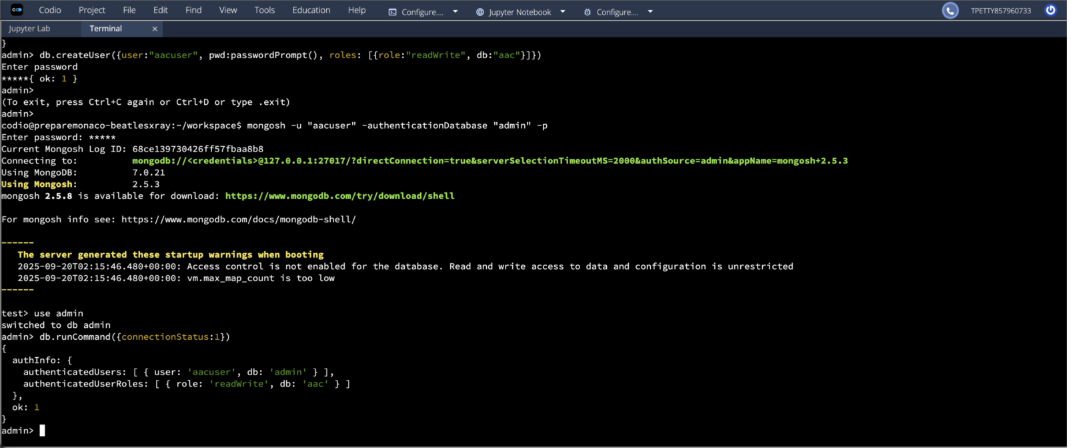
*MongoDB import execution*





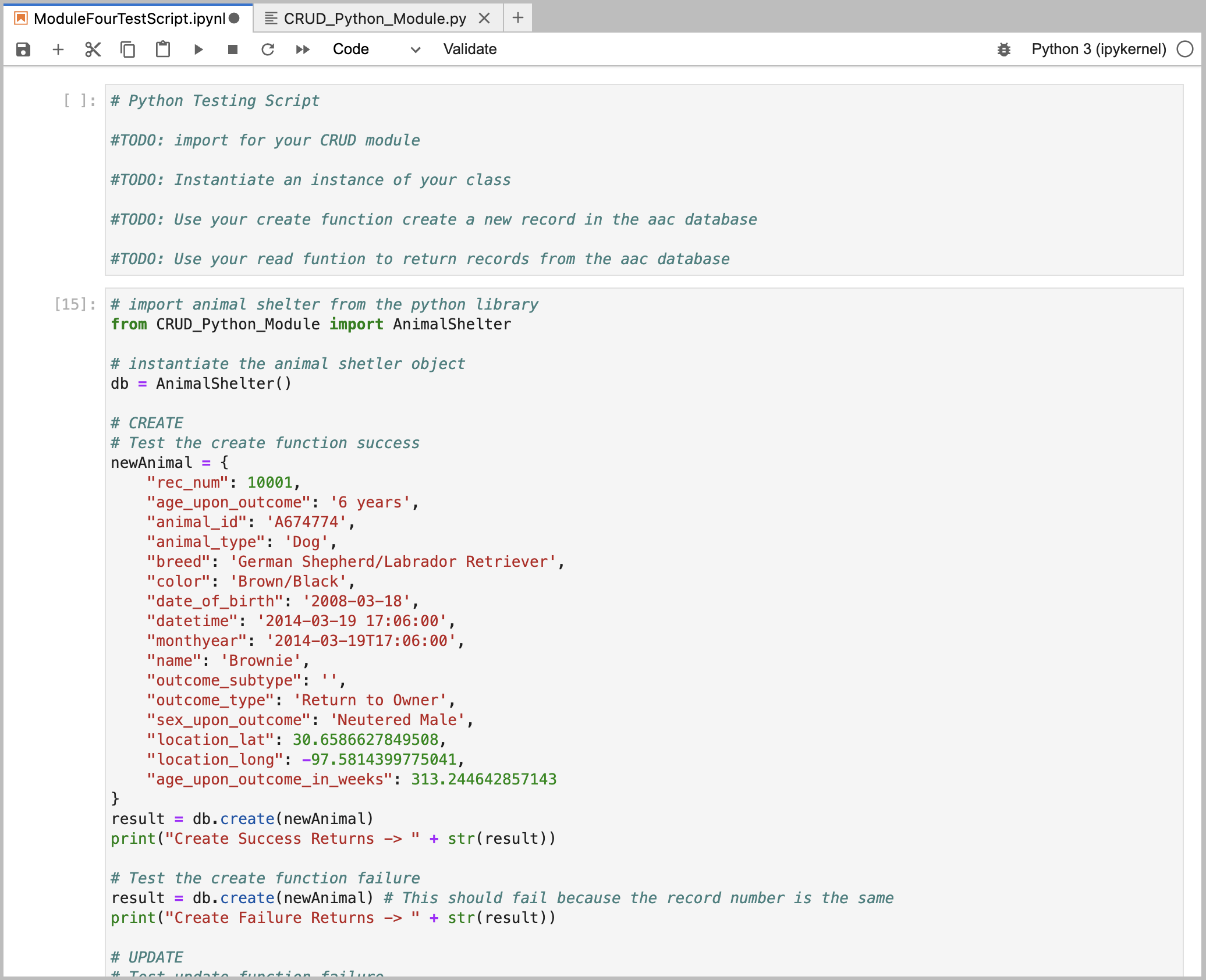


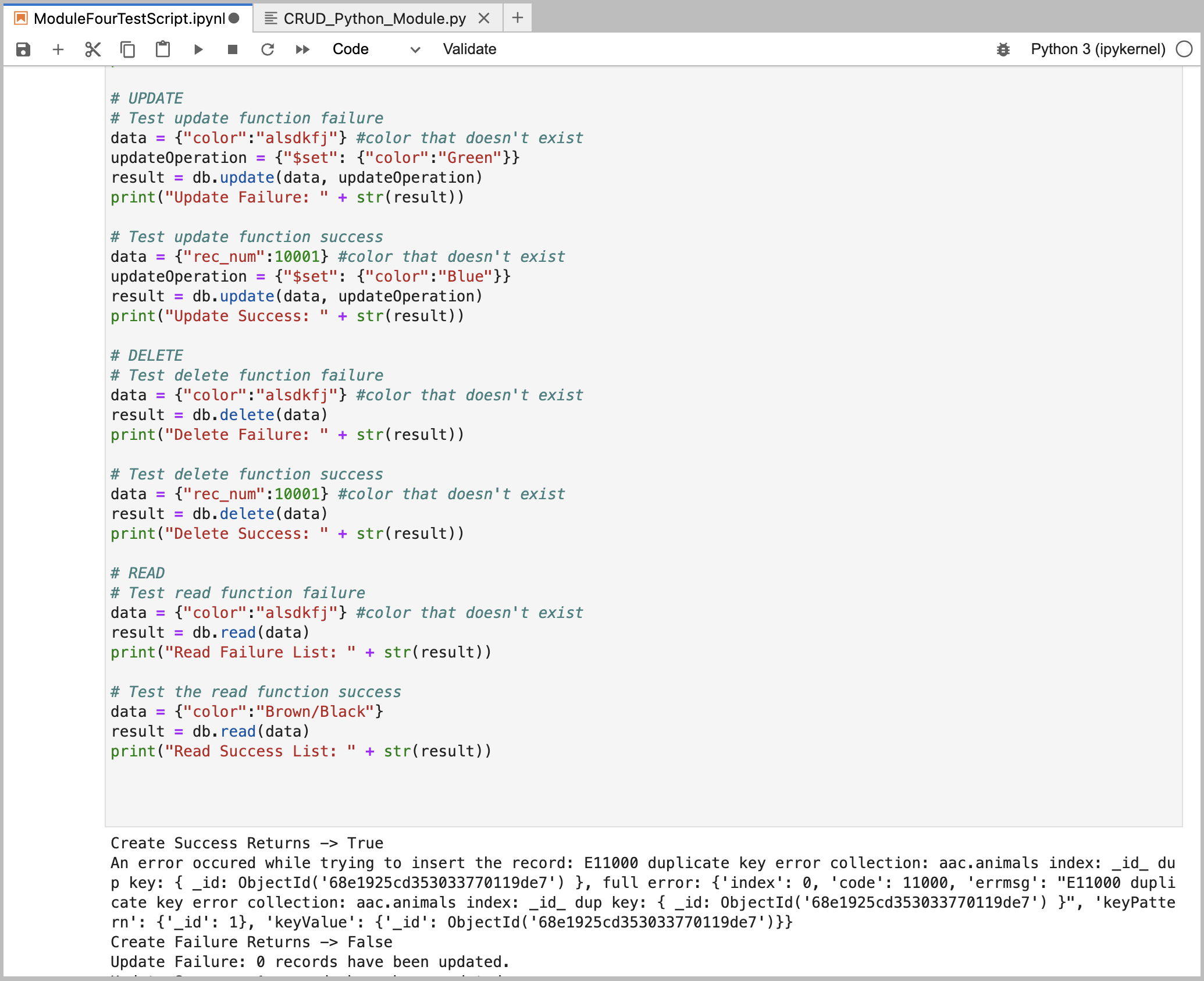
*User authentication execution*

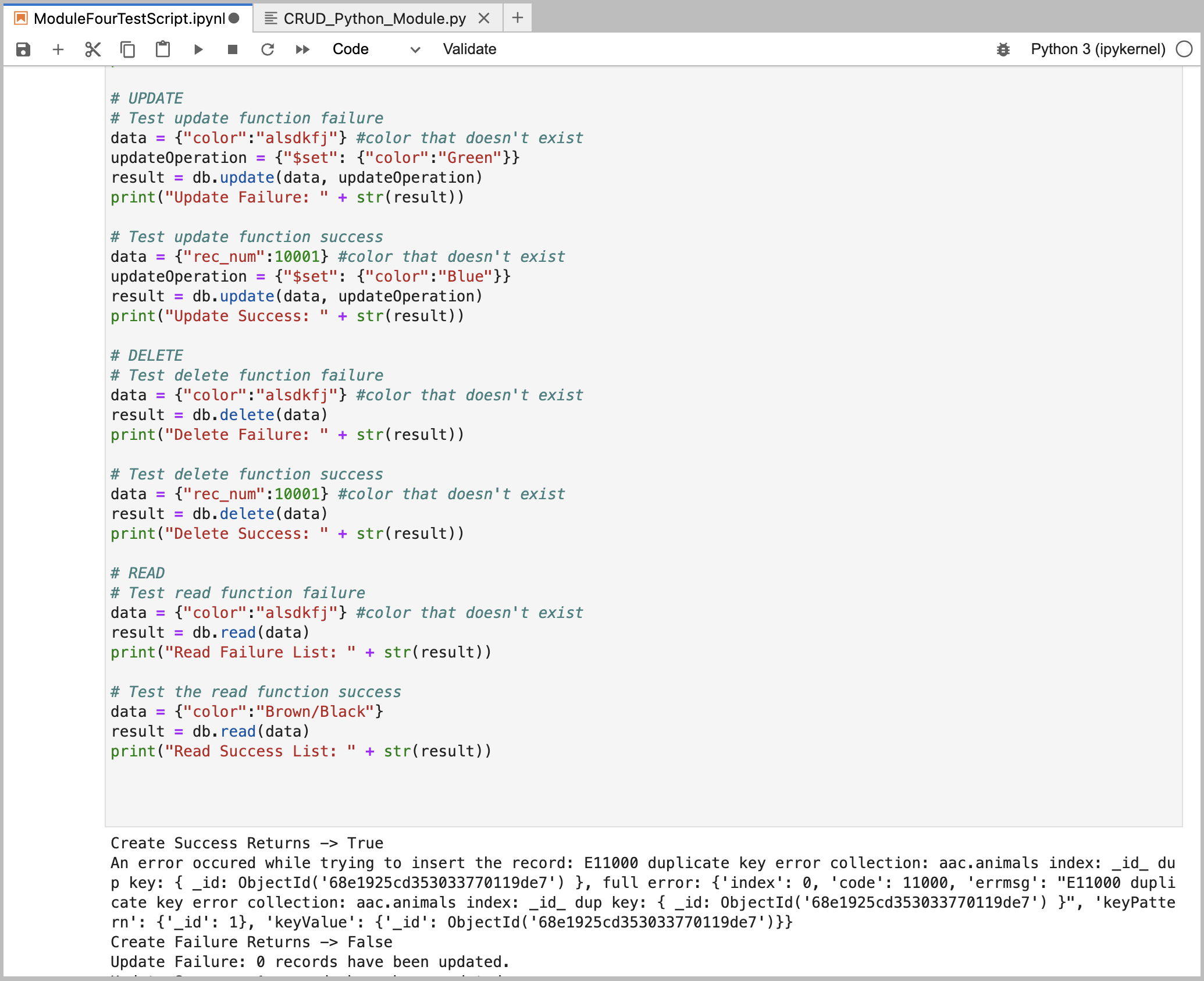


*CRUD Functionality Test*

*These screenshots show an example of how you can test the functionality of the python library.*







## Dashboard Documentation

*To access the information, we were requested to create a dashboard with a few things.*

* *Interactive filter that affects the whole dashboard*
* *An interactive data table*
* *An interactive chart*
* *An interactive map*

*Essentially, all items on the dashboard are to be filtered by the type of dog some one is looking for. That type is determined by a preset list of attributes set for certain rescue dogs. That way a user can simply click “Water Rescue” and see dogs that excel at that.*

*\*A screen recording of how the dashboard works is provided separately.*

## Dashboard Tools

*MongoDB was used as the backend for the dashboard and the specifics of how that works is shown above. We even have examples of how to connect to the database and parse it in python above.*

*We did use Jupyter Dash framework for the front end as well. It makes it easy to connect to the database and create the front end all in one place. We simply imported the AnimalShelter object and started querying the database to populate the tables and graphs. It is awesome because we can use HTML within the Dash framework as well, making it an easy choice to read.*

## Dashboard Steps to Completion

*To complete the dashboard portion of the project, I first had to create a table, graph, and map on the dashboard. Once that was in place, we simply needed to use the HTML id tags to see when the filter was being changed so that I could update the charts dynamically. I created more queries to re-retrieve the filtered data and apply it to the dashboard. After that, it was just a matter of testing to ensure it worked effortlessly.*

## Dashboard Challenges

*I had a few challenges to overcome. The most prominent of which was the misspelling of HTML ids. The methods used to update the graphs and tables on the dashboard based on the filter are watched for changes specific to an HTML tag. I had missed spelling it at one point and it was tough to find. I would watch out for that as you examine and use the code for your own purposes. One typo could throw you very far off.*

## Contact

Your name: Ty Petty