**Grazioso Salvare**Allow customer to identify dogs that are candidates for search-and-rescue training.

**About Project**

The customer has teamed up with the Austin Animal Shelter to find suitable dogs for search-and-rescue training. The purpose of this project is to provide the customer with CRUD (Create, Read, Update, Delete) functionality in shelter database to find these candidates. This will give the customer the ability to add new animals to the database, find, update, and delete existing database records. This functionality will be provided via the Python programming language and utilizes the MongoDB NoSQL database.

**Motivation**

This project is designed to give the customer an easy way of accessing and manipulating animal information. The project does this by providing an easy-to-use and flexible Python module. This module will allow the customer to very easily locate animals that are prime candidates to search-and-rescue training. Using standardized inputs across all methods make it easy for the customer to search and manipulate data.

**Prerequisites/Requirements**

Jupyter Notebook

Python3

MongoDB NoSQL Server

MongoImport tool

Dataset Of Animals

**Getting Started**

First, PyMongo is used in this project to interact with the Mongo database and in general contains tools for working with MongoDB. The reason for using it in this project is because it is an industry standard and most often recommended when working with Mongo databases in Python.

To get started, the requirements must be met (not included in this file), we will start by creating an admin user and standard user which will be used by the Python script for interacting with the data and the admin user for creating the standard user and importing the dataset into the database.

Below are the screenshots for this process.

This is the creation of the admin account:

Graphical user interface, text

Description automatically generated

This is the creation of the standard account:

Text

Description automatically generated with low confidence

This shows successful login by both accounts:

Text

Description automatically generated

Now that the users have been created and login has been shown, we need to import the dataset into the database using the mongoimport tool. Below is the illustration of this importing using the new admin user we previously created.

Graphical user interface

Description automatically generated with medium confidence

**Usage**

The examples below assume that a blank Jupyter Notebook has been created. Following creating a new notebook start out by importing the module, then we create a new object and finally using the new object we perform various actions using class methods.

**Import Module**

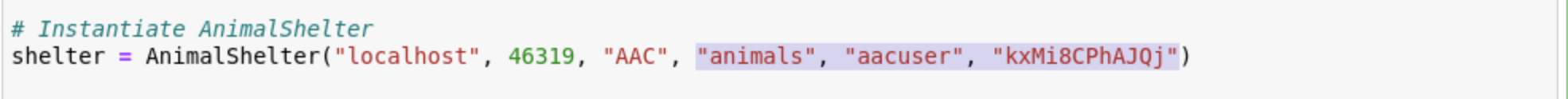
*from animalShelter import AnimalShelter*



**Create New Object**

To create the new object we must provide the connection information to the MongoDB including the port number, database name, collection name and the user information of the standard user created earlier.

*shelter = AnimalShelter(“localhost”, 46319, “AAC”, "animals", "aacuser", "kxMi8CPhAJQj")*



**Perform Database Operations**

This is not required but we specify the data which we will use in the examples below to perform the database operations provided by the module.

*data = {*

*"1": 10012,*

*"age\_upon\_outcome" : "11 years",*

*"animal\_id" : "A10012",*

*"animal\_type" : "Dog",*

*"breed" : "Beagle",*

*"color" : "Black/White/Tan",*

*"date\_of\_birth" : "2012-04-05",*

*"datetime" : "08:43:03",*

*"monthyear" : "2023-02-05",*

*"name" : "Max",*

*"outcome\_subtype" : "",*

*"outcome\_type" : "Purchase",*

*"sex\_upon\_outcome" : "Neutered Male",*

*"location\_lat" : 35.159870,*

*"location\_long" : -84.875504,*

*"age\_upon\_outcome\_in\_weeks" : 5*

*}*

*animal = {"animal\_id": "A10012"}*

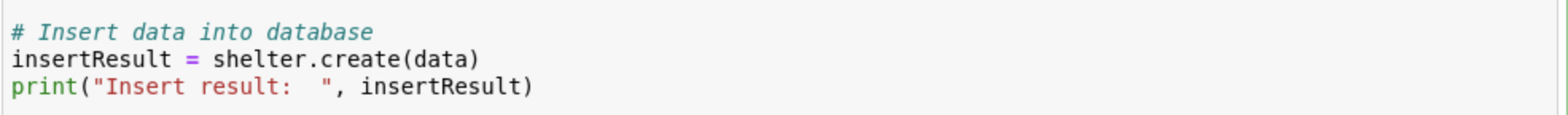
Graphical user interface

Description automatically generated with medium confidence

**Insert Record**

The insert takes an entire record as the input which can be seen above in the test data. If the insert was successful True is returned and if unsuccessful, False is returned. Below shows the syntax for inserting a record. The print function is not required but, in the example, it is used to show the successful insert.

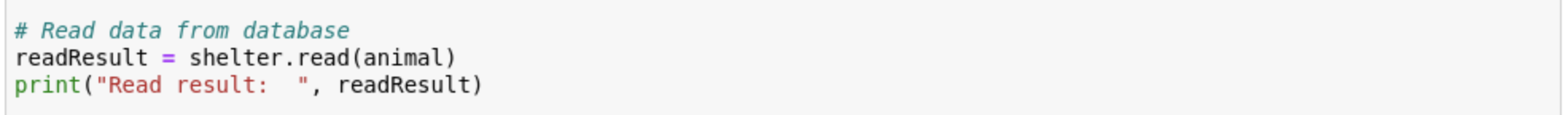
*insert = shelter.create(data)*



**Read Record**

The read method takes as input key/value pairs of the criteria to be searched. If the read was successful, it will return a cursor of the data and if unsuccessful it displays an error message. As with the above, the print function is not required.

*read = shelter.read({“animal\_id”: “A10012”})*

**

**Update Record(s)**

The update method takes two inputs, both of which are key/value pairs. The first set of inputs is the filter which is used to determine which record or records is to be updated and the second set of key/value pairs is the updated information which will replace the current data on the record or records. If successful, the record or records which was updated will be returned in JSON format and if no records were found or updated a message stating no records were updated will be returned and if the function resulted in an error, it will be returned,.

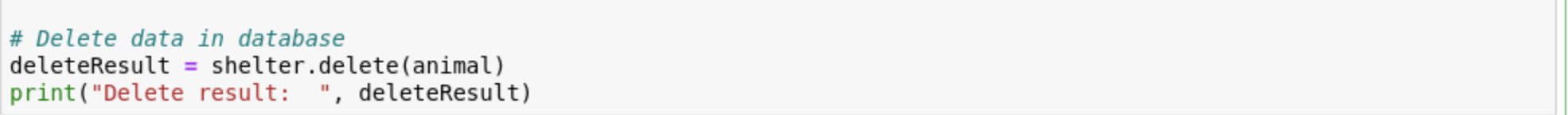
*update = shelter.update({“animal\_id”: “A10012”}, {“name”:”Maximus”})*

**

**Delete Record(s)**

The delete method takes as input, key/values pairs of the record or records needing to be deleted. If successful, the result is the response given by the server on the delete and if unsuccessful, the result will be an error message. Note – It is possible to return a successful and not delete any records. If the data specified does not match any records than an error message may not be returned.

*delete = shelter.delete({“animal\_id”: “A10012”})*

**

**Complete Example**

Below is a screenshot that shows all methods in use and the results of functionality. From the explanations given for each method, it can be determined that the results from each method was a success.

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

**Contact**

Eric Wallace