Grazioso Salvare's CRUD Python Module README Template

About the Project

This project is titled "Grazioso Salvare's CRUD Python Module". Our project is to create a full stack software application for our client, Grazioso Salvare. The full stack development includes creating a Python module with CRUD (create, read, update, and delete) functionalities that can connect the database and client-facing web application dashboard that we will create for our client. The purpose of the CRUD Python module is to allow our client to manipulate the animal data that is imported from the Austin Animal Center's database (AAC). The AAC is a non-profit agency that is working with our client.

Installation

The tools required to complete this project is a Terminal application, MongoDB version 4.2, PyMongo, and Jupyter Notebook. Here is why each tool is required and directions on how to install them:

1. The Terminal application is used to host MongoDB. The Linux terminal was used for this project.
2. MongoDB is used as our NoSQL database tool. More specifically, we are using version 4.2 of the application. This version can be downloaded by going to THIS link and selecting the download option that matches your operating platform.
3. PyMongo is the official MongoDB driver for Python. More information on it and how to install it can be found HERE.
4. Jupyter Notebook is the Python IDE used in this project to create the Python file that contains our development code and Python Notebook file that contains the test code. This IDE can be downloaded HERE.

Usage

This section provides an overview of how the CRUD functionalities work and can be used in a Python module. More specifically, it contains an example code, test code, and screenshots of both.

Code Example

The Python code example in "Screenshot #1" below breaks down how to insert a document, and then create methods for each of the CRUD functionalities. The code starts with including the PyMongo driver. This driver, then, must import the MongoClient class that allows the connection to MongoDB. Next, "from bson.objectid import ObjectId" is added to allow us to query each item in our document by its object ID, and then convert the data to and from BSON types.

The body of the code includes an "AnimalShelter" class that passes through an object. This class is used to initialize the MongoClient via authentication using a unique port number, and then create four methods; one for each of the CRUD functionalities.

The first method developed is the create functionality in CRUD. This method allows us to insert a document using the insert() function. The method returns "True" if the document is inserted successfully and "False" if it does not.

The second method is for the read functionality. This allows us to search for documents using a specified search parameter and the find() function. The method returns the result of the search in cursor if the search was successful. Otherwise, it returns an error message.

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

The third method is the update functionality. This allows us to locate documents using a specified search parameter, and then update all of the documents that match the search parameter to the specified updated data using the update_many() function. If successful, the method returns the number of documents that were updated successfully and prints the data that was updated. Otherwise, it returns an error message.

The last method is the delete functionality. This allows us to locate documents using a specified search parameter, and then delete all of the documents that match the search parameter using the delete_many() function. If successful, the method returns the number of documents that were deleted successfully and prints the data that was deleted. Otherwise, it returns an error message.

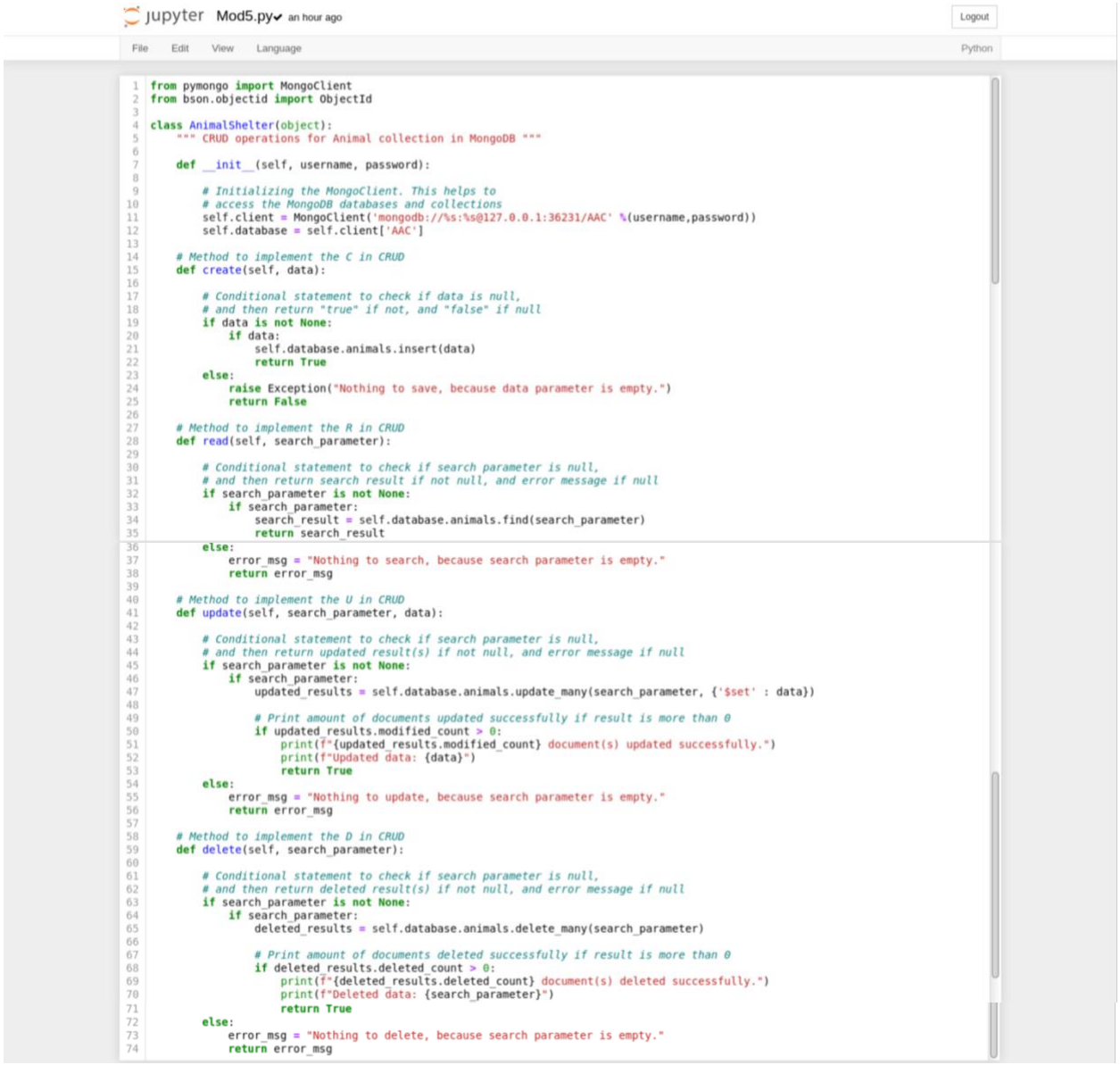
Tests

"Screenshot #2" below displays the commands and executions of the Python testing script. The steps to create and run the test for the example code is as followed:

1. Import the "AnimalShelter" class that was created in the example code.
2. Define a data dictionary that includes all of the information tied to a unique ID. The "data" dictionary that is used specifies the details of an animal from the AAC database, such as animal type, breed, color, etc.
3. Define a search parameter using a key and entry from the dictionary created in step 2. Our search parameter uses the animal ID defined in our "data" dictionary.
4. Instantiate an object of the "AnimalShelter" class and pass through the username and password of the unique user account that has access to the MongoDB database in use. Our example code uses the "AAC" database that can be accessed using the "aacuser" account.
5. Call the create method that was created in the example code, and print the status of the data insert in the form of "True" or "False".
6. Call the read method that was created in the example code and print the result.
7. Define the data that needs to be updated. Our example specifies that the data type "breed" needs to be updated to "Husky". Then, we call the update method that was created in the example code and print the results.
8. Call the delete method that was created in the example code and print the results.
9. Run the test code to print items mentioned in steps 5 through 8. Inputting "data" should output information of the animal entered in the "data" dictionary, and the unique Object ID that is created for this entry.

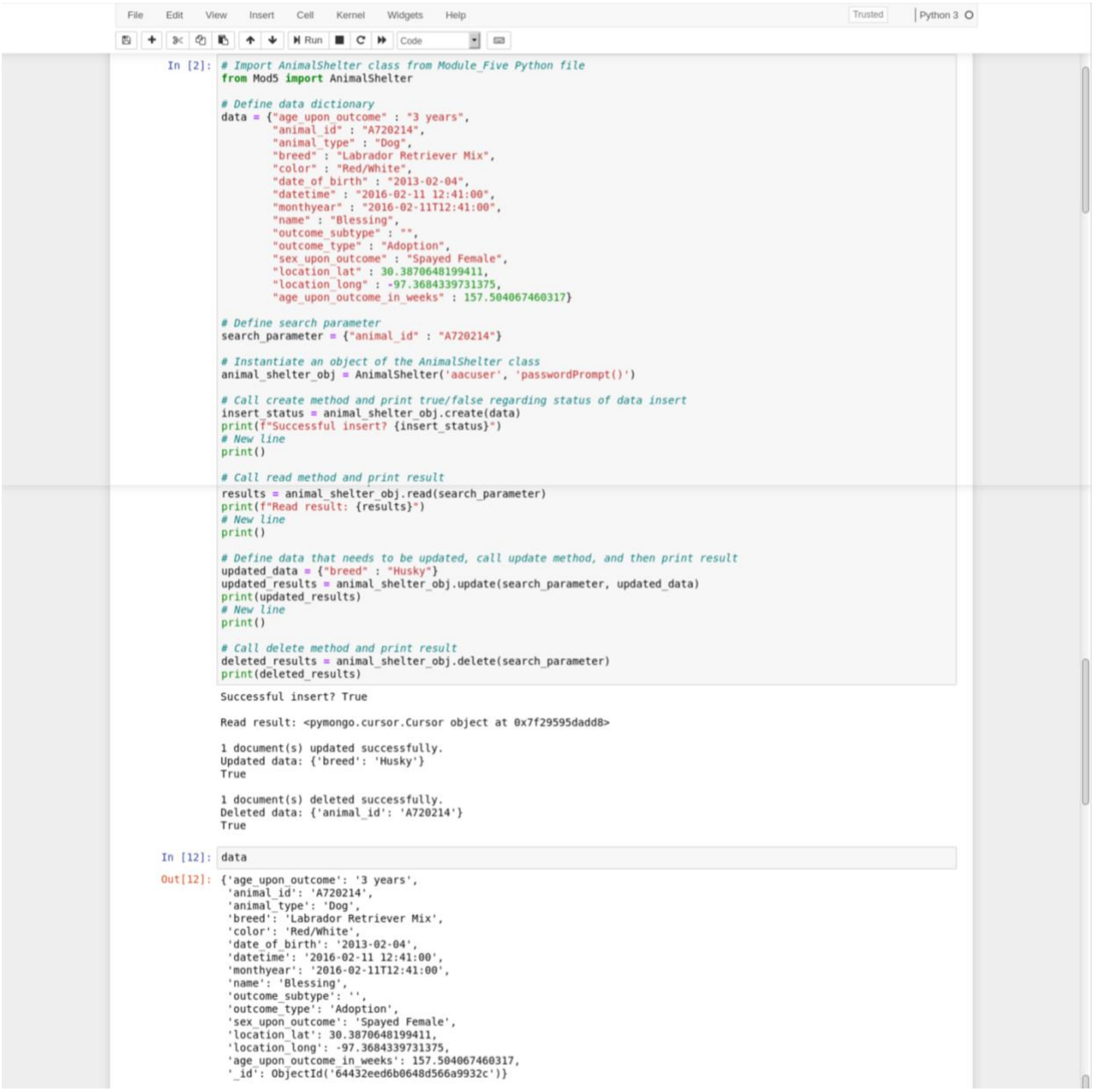
Screenshots

Screenshot #1: Code Example



Screenshot #2: Test Code

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.



```
In [2]:
# Import AnimalShelter class from Module_Five Python file
from Mod5 import AnimalShelter

# Define data dictionary
data = {"age_upon_outcome" : "3 years",
        "animal_id" : "A720214",
        "animal_type" : "Dog",
        "breed" : "Labrador Retriever Mix",
        "color" : "Red/White",
        "date_of_birth" : "2013-02-04",
        "datetime" : "2016-02-11 12:41:00",
        "monthyear" : "2016-02-11T12:41:00",
        "name" : "Blessing",
        "outcome_subtype" : "",
        "outcome_type" : "Adoption",
        "sex_upon_outcome" : "Spayed Female",
        "location_lat" : 30.3870648199411,
        "location_long" : -97.3684339731375,
        "age_upon_outcome_in_weeks" : 157.504067460317}

# Define search parameter
search_parameter = {"animal_id" : "A720214"}

# Instantiate an object of the AnimalShelter class
animal_shelter_obj = AnimalShelter('aacuser', 'passwordPrompt()')

# Call create method and print true/false regarding status of data insert
insert_status = animal_shelter_obj.create(data)
print(f"Successful insert? {insert_status}")
# New line
print()

# Call read method and print result
results = animal_shelter_obj.read(search_parameter)
print(f"Read result: {results}")
# New line
print()

# Define data that needs to be updated, call update method, and then print result
updated_data = {"breed" : "Husky"}
updated_results = animal_shelter_obj.update(search_parameter, updated_data)
print(updated_results)
# New line
print()

# Call delete method and print result
deleted_results = animal_shelter_obj.delete(search_parameter)
print(deleted_results)

Successful insert? True

Read result: <pymongo.cursor.Cursor object at 0x7f29595dadd8>

1 document(s) updated successfully.
Updated data: {'breed': 'Husky'}
True

1 document(s) deleted successfully.
Deleted data: {'animal_id': 'A720214'}
True

In [12]:
data

Out[12]:
{'age_upon_outcome': '3 years',
 'animal_id': 'A720214',
 'animal_type': 'Dog',
 'breed': 'Labrador Retriever Mix',
 'color': 'Red/White',
 'date_of_birth': '2013-02-04',
 'datetime': '2016-02-11 12:41:00',
 'monthyear': '2016-02-11T12:41:00',
 'name': 'Blessing',
 'outcome_subtype': '',
 'outcome_type': 'Adoption',
 'sex_upon_outcome': 'Spayed Female',
 'location_lat': 30.3870648199411,
 'location_long': -97.3684339731375,
 'age_upon_outcome_in_weeks': 157.504067460317,
 '_id': ObjectId('64432eed6b0648d566a9932c')}
```

Getting Started

Follow these example steps to get a local copy up and running:

1. Ensure that all tools mentioned in the “Installation” section above are installed and set up correctly.
2. Start the Terminal application.
3. Identify the data set that you would like to use, figure out where it is located in your directory, and then start Mongo without authentication by entering: /usr/local/bin/mongod_ctl start-noauth
4. Copy the unique port number that is given (ex: +++ Starting MongoDB (NOAUTH): Port=27908), and then enter it into the mongoimport function to load the data set into the new MongoDB

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

server environment. More information on mongoimport can be found HERE. Our example is as followed, where “AAC” is our database name and “animals” is the collection name:

```
File  Edit  View  Search  Terminal  Help
(base) quynh-nhuho_snhu@msnv-snhu3-l002:~$ /usr/local/bin/mongo_ctl start-noauth
bash: /usr/local/bin/mongo_ctl: No such file or directory
(base) quynh-nhuho_snhu@msnv-snhu3-l002:~$ /usr/local/bin/mongod_ctl start-noauth
+++ Starting MongoDB (NOAUTH): Port=36231  Unix_Socket=/tmp/mongodb-36231.sock Dir=/home/quynh-nhuho_snhu/mongodb
(base) quynh-nhuho_snhu@msnv-snhu3-l002:~$ cd /usr/local/datasets
(base) quynh-nhuho_snhu@msnv-snhu3-l002:/usr/local/datasets$ /usr/local/bin/mongod_ctl start-noauth
MongoDB is already running for quynh-nhuho_snhu.  Please stop the database first before running with start-noauth
(base) quynh-nhuho_snhu@msnv-snhu3-l002:/usr/local/datasets$ mongoimport --port 36231 --db AAC --collection animals --type=csv --headerline ./aac_shelter_outc
omes.csv
2023-03-16T23:40:39.337+0000    connected to: mongodb://localhost:36231/
2023-03-16T23:40:39.578+0000    10000 document(s) imported successfully. 0 document(s) failed to import.
```

5. Create a user administrator account by following steps 1-5 of the “Procedure” section in the MongoDB version 4.2 manual for enabling access control HERE. Once you have successfully logged into the account, enter “show dbs” as a command in the Mongo shell. This will return all of the databases that are linked to your port number and MongoDB directory. Our screenshot below starts at logging in using our newly created admin account. Once logged in and the command to show databases has been entered, we can see that 7 databases can be found in our directory, including the “AAC” database that we are working with on this project. Now, we want to enter “exit” into the command line to exit the Mongo shell. Then, enter “/usr/local/bin/mongod_ctl stop” into the Terminal shell to log out of the admin account and prepare for the next step.

```
(base) quynh-nhuho_snhu@msnv-snhu3-l001:~$ /usr/local/bin/mongod_ctl start
+++ Starting MongoDB: Port=36231  Unix_Socket=/tmp/mongodb-36231.sock Dir=/home/quynh-nhuho_snhu/mongodb
(base) quynh-nhuho_snhu@msnv-snhu3-l001:~$ mongo --authenticationDatabase "admin" -u "myUserAdmin" -p
MongoDB shell version v4.2.6
Enter password:
connecting to: mongodb://127.0.0.1:36231/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("3f4b1ead-8165-4791-a613-363f687f35d5") }
MongoDB server version: 4.2.6
> show dbs
AAC     0.003GB
admin   0.000GB
city    0.011GB
config  0.000GB
enron   0.007GB
local   0.000GB
test    0.000GB
> exit
bye
(base) quynh-nhuho_snhu@msnv-snhu3-l001:~$ /usr/local/bin/mongod_ctl stop
```

6. Create a new user account to access the data set in the database created in step 4, by following steps 6 and 7 in the link mentioned in step 5. Our example user account is the “aacuser” account. Once you have successfully logged into the account, enter “show dbs” in the command line. This should return only the database(s) that is/are linked to this user account. Our screenshot below starts at logging in using our newly created “aacuser” account. Once logged in and the command to show databases has been entered, we can see that only the “AAC” database is displayed as expected.

```
(base) quynh-nhuho_snhu@msnv-snhu3-l001:~$ /usr/local/bin/mongod_ctl start
+++ Starting MongoDB: Port=36231  Unix_Socket=/tmp/mongodb-36231.sock Dir=/home/quynh-nhuho_snhu/mongodb
(base) quynh-nhuho_snhu@msnv-snhu3-l001:~$ mongo --authenticationDatabase "AAC" -u "aacuser" -p
MongoDB shell version v4.2.6
Enter password:
connecting to: mongodb://127.0.0.1:36231/?authSource=AAC&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("9d53e7f7-a239-463c-a6ad-8b2d8f712933") }
MongoDB server version: 4.2.6
> show dbs
AAC  0.003GB
```

7. Enter “use AAC” in the Mongo shell – where “AAC” should be replaced with the name of your database.
8. Now, open Jupyter Notebook and create a new Python file by clicking on the “New” drop-down menu on the top right side of the home page, and then select “Text File” from the drop-down. This will open a new text file. Rename this text file, then enter “.py” at the end of the name to convert the text file to a Python file.
9. Enter the code example from the “Usage” section above into this Python file.
10. Once code has been entered into the Python file, create a new Jupyter Notebook IPYNB file to test the code. This can be done by returning to the Jupyter home page, navigating back to the

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

"New" drop-down menu, and then selecting "Python 3". Make sure that this test file is located in the same directory as the Python file that was created.

11. Once the Notebook file has been created, enter the example test code from the "Usage" section above.
12. Once the test code has been entered, click "Run" in the Notebook file's toolbar. Then, data can be inputted to test the code. The input data that we used for our example test code is "data", the name of our data dictionary.
13. The final step is to ensure that the test code's outputs match the expectation of the development code.

Contact

Quinnie Ho

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.