Project One README

About the Project/Project Title
PyMongo CRUD implementation

Motivation
The purpose of the project is to allow for database interactions in Python through MongoDB by integrating CRUD functionality into our AnimalShelter class.

Getting Started
To get started with the code ensure your authentication and database information is in the crud.py file. The database and user were created in the MongoDB shell using the aac_shelter_outcomes.csv and the mongoimport command. The C and R portions were developed using Python's own error handling tools and the pymongo module. The only issue encountered was in getting the error handling to work properly, I attempted to implement the MongoDB error handling, but I was getting errors that I could not figure out, so I decided to switch to standard exception handling.

Installation
The tools used were a Linux distribution, MongoDB, Python, and Jupyter Notebooks.

Usage

Code Example

```
from pymongo import MongoClient
from bson.objectid import ObjectId

class AnimalShelter(object):
  """ CRUD operations for Animal collection in MongoDB """

  # Create CRUD function
  # data = dictionary of key/value pairs
  # Returns a boolean depending on the success of the function
  def create(self, data):
    try:
      result = self.database.animals.insert_one(data)
      return result.acknowledged
    except Exception as e:
      #print("An error occured during the write ::", e)
      return False

  # Read CRUD function
  # data = dictionary of key/value pair(s)
  # Returns list of objects
  def read(self, data):
    try:
      return list(self.database.animals.find(data))
    except Exception as e:
      #print("An error occured during the read ::", e)
```

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.

```
        return ([], [])

# Update CRUD function
# data = dictionary of key/value pair(s) to find
# mod = dictionary of key/value pair(s) to modify
# Returns number of modified objects
def update(self, data, mod):
    try:
        result = self.database.animals.update_many(data, {"$set": mod})
        return result.modified_count
    except Exception as e:
        #print("An error occured during the write ::", e)
        return 0

# Delete CRUD function
# data = dictionary of key/value pair(s)
# Returns number of deleted objects
def delete(self, data):
    try:
        result = self.database.animals.delete_many(data)
        return result.deleted_count
    except Exception as e:
        #print("An error occured during the delete ::", e)
        return 0

def __init__(self):
    # Initializing the MongoClient. This helps to
    # access the MongoDB databases and collections.
    # This is hard-wired to use the aac database, the
    # animals collection, and the aac user.
    # Definitions of the connection string variables are
    # unique to the individual Apporto environment.
    #
    # You must edit the connection variables below to reflect
    # your own instance of MongoDB!
    #
    # Connection Variables
    #
    USER = 'aacuser'
    PASS = 'SNHU1234'
    HOST = 'nv-desktop-services.apporto.com'
    PORT = 32398
    DB = 'AAC'
    COL = 'animals'
    #
    # Initialize Connection
    #
```

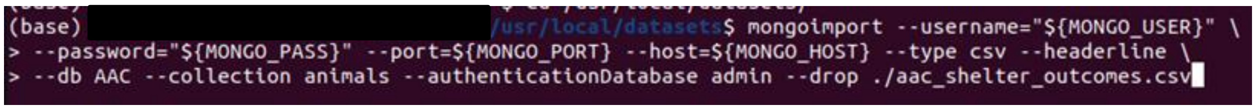
Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

self.client = MongoClient('mongodb://%s:%s@%s:%d' % (USER,PASS,HOST,PORT))
self.database = self.client['%s' % (DB)]
self.collection = self.database['%s' % (COL)]

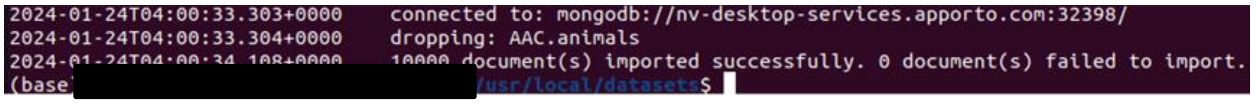
Tests
Eight tests were made using a special breed value that would not bog down the results of the read tests. The first test was for the create function to successfully create an animal in the database, this returns True. The second test attempts to read the newly created data successfully and returns a list of all the data found. The third test attempts to modify the newly created object from having a color value of orange to green and returns the number of entries modified. The fourth test attempts the same as the third but fails because there is nothing to modify therefore returning 0. The fifth test attempts to make a duplicate insert and returns False. The sixth test attempts to look up a breed that does not exist and returns an empty list. The seventh test deletes our test object from the database and returns the number of objects deleted (1). The final test attempts to delete nothing and returns 0.

Screenshots

Import of CSV



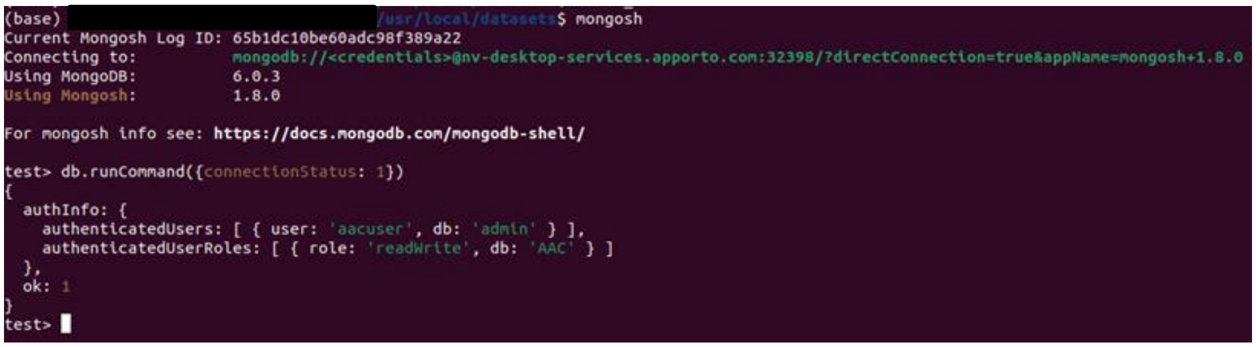
```
(base) [redacted] /usr/local/datasets$ mongoimport --username="${MONGO_USER}" \
> --password="${MONGO_PASS}" --port=${MONGO_PORT} --host=${MONGO_HOST} --type csv --headerline \
> --db AAC --collection animals --authenticationDatabase admin --drop ./aac_shelter_outcomes.csv
```



```
2024-01-24T04:00:33.303+0000    connected to: mongodb://nv-desktop-services.apporto.com:32398/
2024-01-24T04:00:33.304+0000    dropping: AAC.animals
2024-01-24T04:00:34.108+0000    10000 document(s) imported successfully. 0 document(s) failed to import.
(base [redacted] /usr/local/datasets$
```

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]().

Authentication of new user “aacuser”

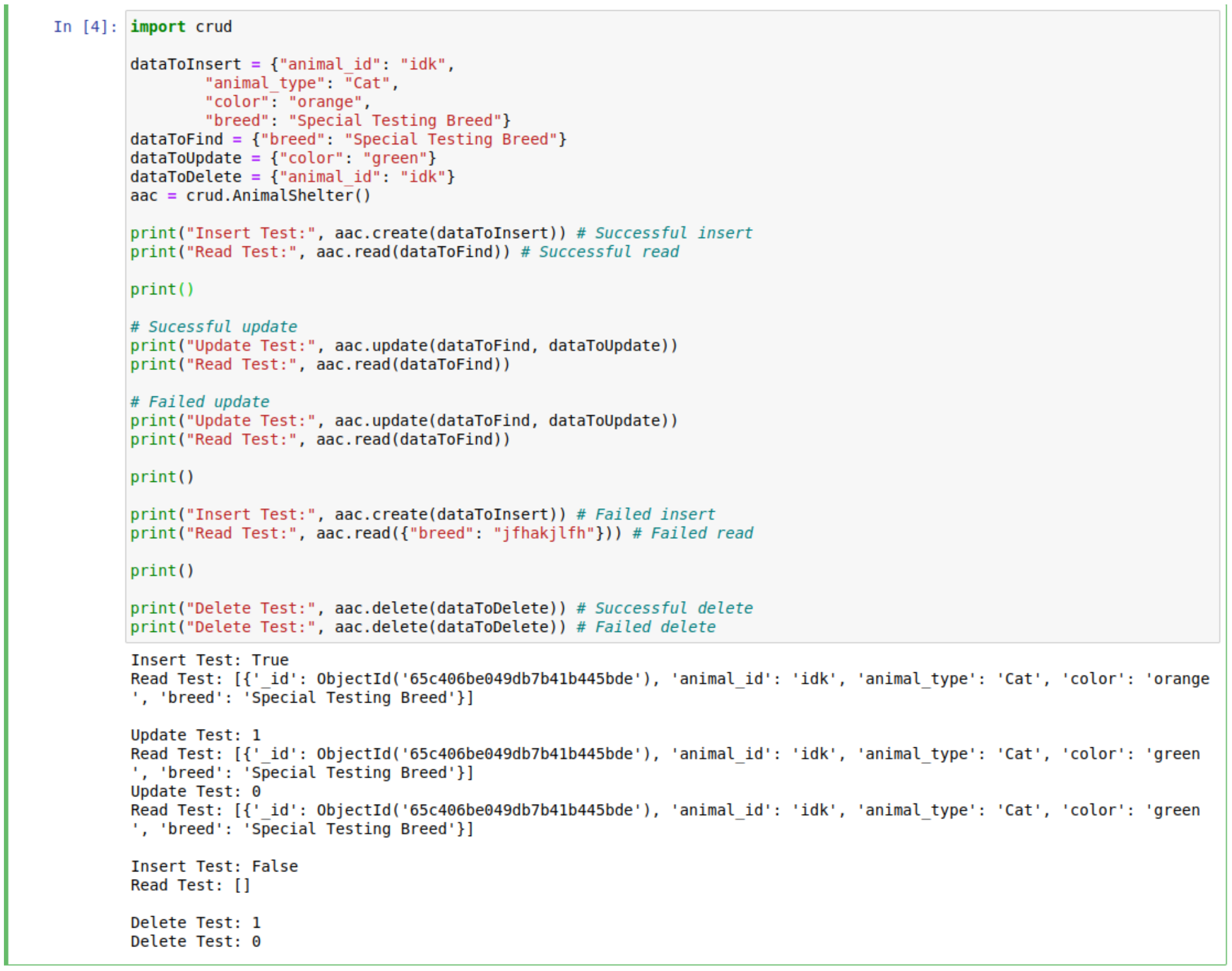


```
(base) [redacted] /usr/local/datasets$ mongosh
Current Mongosh Log ID: 65b1dc10be60adc98f389a22
Connecting to:          mongodb://<credentials>@nv-desktop-services.apporto.com:32398/?directConnection=true&appName=mongosh+1.8.0
Using MongoDB:          6.0.3
Using Mongosh:          1.8.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

test> db.runCommand({connectionStatus: 1})
{
  authInfo: {
    authenticatedUsers: [ { user: 'aacuser', db: 'admin' } ],
    authenticatedUserRoles: [ { role: 'readWrite', db: 'AAC' } ]
  },
  ok: 1
}
test> 
```

Tests



```
In [4]: import crud

        dataToInsert = {"animal_id": "idk",
                "animal_type": "Cat",
                "color": "orange",
                "breed": "Special Testing Breed"}
        dataToFind = {"breed": "Special Testing Breed"}
        dataToUpdate = {"color": "green"}
        dataToDelete = {"animal_id": "idk"}
        aac = crud.AnimalShelter()

        print("Insert Test:", aac.create(dataToInsert)) # Successful insert
        print("Read Test:", aac.read(dataToFind)) # Successful read

        print()

        # Sucessful update
        print("Update Test:", aac.update(dataToFind, dataToUpdate))
        print("Read Test:", aac.read(dataToFind))

        # Failed update
        print("Update Test:", aac.update(dataToFind, dataToUpdate))
        print("Read Test:", aac.read(dataToFind))

        print()

        print("Insert Test:", aac.create(dataToInsert)) # Failed insert
        print("Read Test:", aac.read({"breed": "jfhakjlfh"})) # Failed read

        print()

        print("Delete Test:", aac.delete(dataToDelete)) # Successful delete
        print("Delete Test:", aac.delete(dataToDelete)) # Failed delete

        Insert Test: True
        Read Test: [{'_id': ObjectId('65c406be049db7b41b445bde'), 'animal_id': 'idk', 'animal_type': 'Cat', 'color': 'orange
        ', 'breed': 'Special Testing Breed'}]

        Update Test: 1
        Read Test: [{'_id': ObjectId('65c406be049db7b41b445bde'), 'animal_id': 'idk', 'animal_type': 'Cat', 'color': 'green
        ', 'breed': 'Special Testing Breed'}]
        Update Test: 0
        Read Test: [{'_id': ObjectId('65c406be049db7b41b445bde'), 'animal_id': 'idk', 'animal_type': 'Cat', 'color': 'green
        ', 'breed': 'Special Testing Breed'}]

        Insert Test: False
        Read Test: []

        Delete Test: 1
        Delete Test: 0
```

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.