Version: 1.0

Date: 07/31/2022

CS 340 MongoCRUD README

About the Project/Project Title

MongoCRUD provides a simple way to interface with a mongo database to perform basic CRUD (Create, Read, Update, Delete) actions, built on top of the pymongo library.

Motivation

MongoCRUD was created to serve as middleware layer between a database and a data visualization display, easily passing data requests through pymongo and catching associated errors. The project has been generalized to be able to work in a variety of applications.

Getting Started

MongoCRUD is built in Python 3 using the pymongo library. To start using MongoCRUD you will need import both into your environment. See examples in the installation guide below.

Installation

venv – *recommended* – create a new environment to isolate projects and avoid version confusion

```
python -m venv PATH
```

pymongo - **required** – use pip to install from pyPI

```
pip install pymongo
or
python -m pip install pymongo
```

mongo_crud – **required** – currently not available on pyPI

ensure a local copy of mongo_crud.py in your workspace folder

mongoDB – **required** – database engine

A free local instance of a mongoDB server can be created using MongoDB Community Server
https://www.mongodb.com/try/download/community
and following instructions in the associated documentation

Usage
Check the doc (thanks pdoc)!

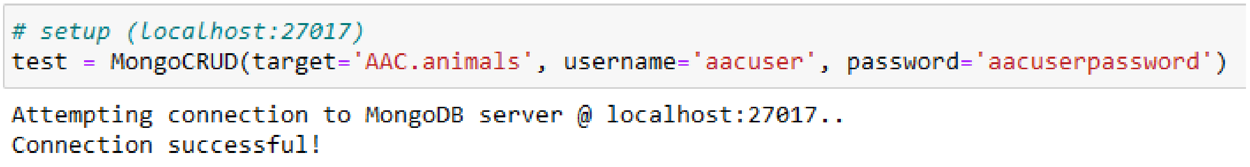
MongoCRUD is designed to setup a workflow quickly and easily with just a few lines of code, so that your focus can be on the project itself.

Note: The following assumes that you have already set up and are running a local mongo database server and have enabled read/write authentication for a database. If not, see the appendix for more info.

Code Example

1. **Create an instance of MongoCRUD. The constructor requires both the target database/collection as well as authentication information.**

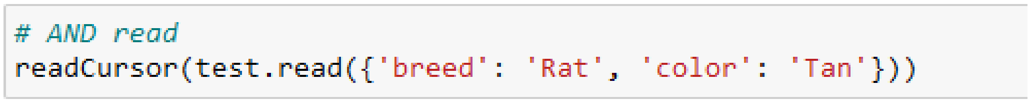
   In this example we are using the database name 'AAC', collection name 'animals', and the 'aacuser' credentials. We set this user up in the database via the mongo shell beforehand.



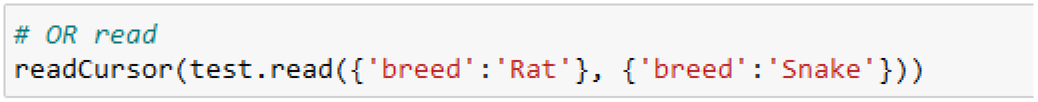
```
# setup (localhost:27017)
test = MongoCRUD(target='AAC.animals', username='aacuser', password='aacuserpassword')
```

```
Attempting connection to MongoDB server @ localhost:27017..
Connection successful!
```

2. **Perform READ operations by passing key/values to search for as argument parameters to the .read() function.**

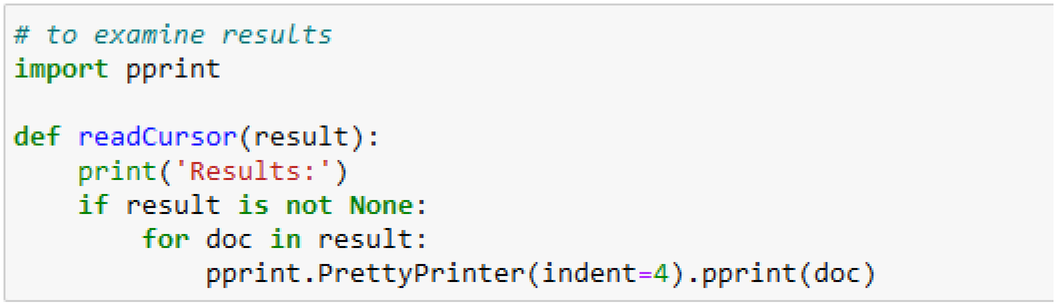


```
# AND read
readCursor(test.read({'breed': 'Rat', 'color': 'Tan'}))
```



```
# OR read
readCursor(test.read({'breed':'Rat'}, {'breed':'Snake'}))
```

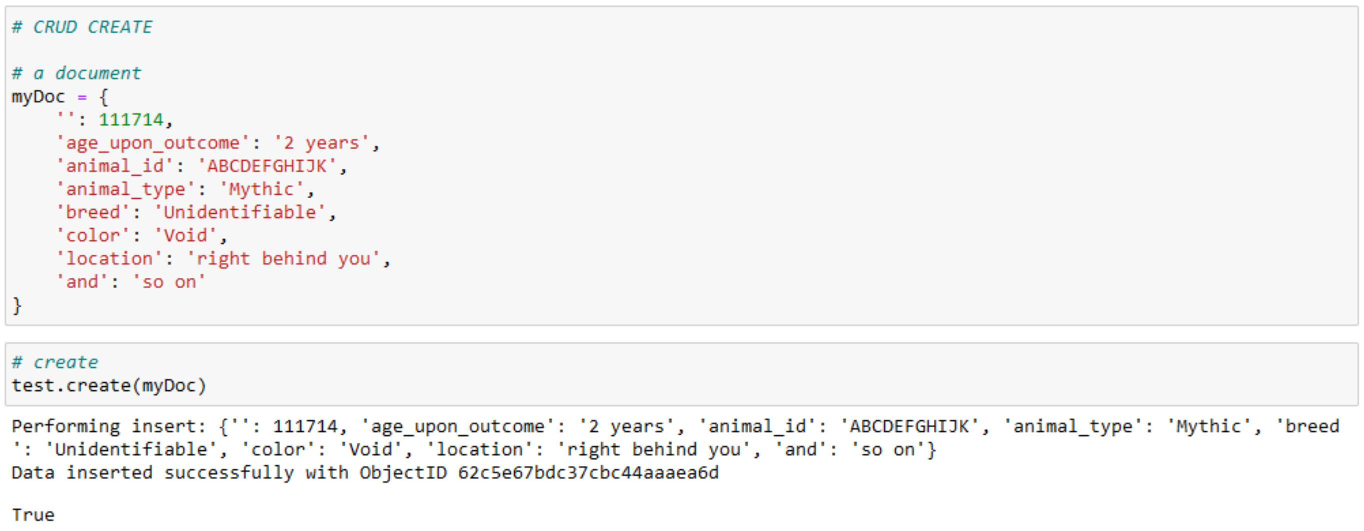
Results are returned in a pymongo cursor, an iterable that can be displayed like



```
# to examine results
import pprint

def readCursor(result):
    print('Results:')
    if result is not None:
        for doc in result:
            pprint.PrettyPrinter(indent=4).pprint(doc)
```

```
Performing query: {'$or': [{'breed': 'Rat'}, {'breed': 'Snake'}]}
Results:
{   '': 1872,
    '_id': ObjectId('62c347a9130859071f874577'),
    'age_upon_outcome': '2 years',
    'age_upon_outcome_in_weeks': 105.938392857143,
    'animal_id': 'A715156',
    'animal_type': 'Other',
    'breed': 'Rat',
    'color': 'Tan',
    'date_of_birth': '2013-11-01',
    'datetime': '2015-11-12 13:39:00',
    'location_lat': 30.4474019382656,
    'location_long': -97.3604392019443,
    'monthyear': '2015-11-12T13:39:00',
    'name': '',
    'outcome_subtype': 'Partner',
    'outcome_type': 'Transfer',
    'sex_upon_outcome': 'Unknown'}
{   '': 3014,
    '_id': ObjectId('62c347a9130859071f8749ee'),
    'age_upon_outcome': '10 months',
    'age_upon_outcome_in_weeks': 43.8142857142857,
    'animal_id': 'A670435',
    'animal_type': 'Other',
    'breed': 'Rat',
    'color': 'Black/White',
    'date_of_birth': '2013-03-08',
    'datetime': '2014-01-08 16:48:00',
    'location_lat': 30.658697880893,
    'location_long': -97.4475783107949,
    'monthyear': '2014-01-08T16:48:00',
    'name': '',
    'outcome_subtype': 'Partner',
    'outcome_type': 'Transfer',
    'sex_upon_outcome': 'Intact Male'}
{   '': 5001,
    '_id': ObjectId('62c347a9130859071f8751b0'),
    'age_upon_outcome': '1 year',
    'age_upon_outcome_in_weeks': 52.2377976190476,
    'animal_id': 'A715120',
    'animal_type': 'Other',
    'breed': 'Snake',
    'color': 'Brown/Tan',
    'date_of_birth': '2014-10-31',
    'datetime': '2015-10-31 15:57:00',
    'location_lat': 30.2726694681691,
    'location_long': -97.5360323953878,
```

3. Perform CREATE operations by passing key/values to search for as argument parameters to the .create() function.

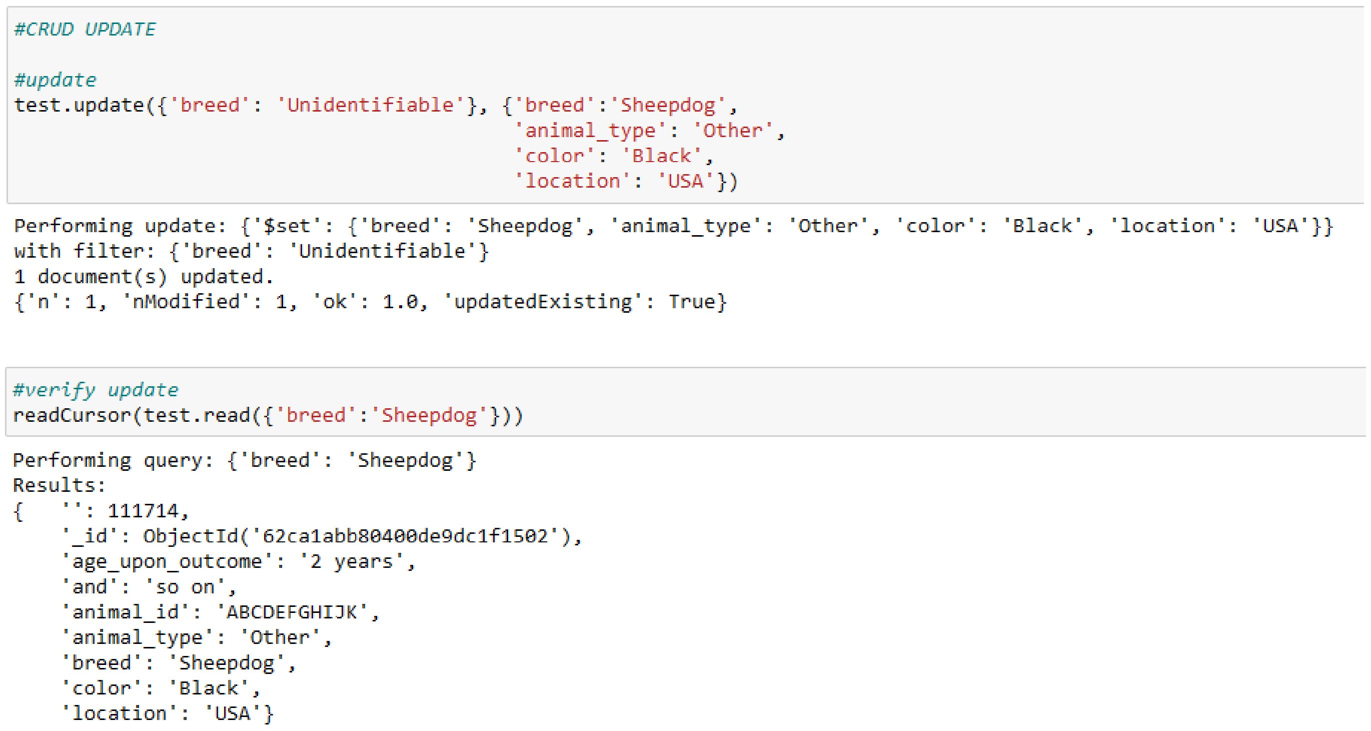


```
# CRUD CREATE

# a document
myDoc = {
    '': 111714,
    'age_upon_outcome': '2 years',
    'animal_id': 'ABCDEFGHIJK',
    'animal_type': 'Mythic',
    'breed': 'Unidentifiable',
    'color': 'Void',
    'location': 'right behind you',
    'and': 'so on'
}
```

```
# create
test.create(myDoc)
```

```
Performing insert: {'': 111714, 'age_upon_outcome': '2 years', 'animal_id': 'ABCDEFGHIJK', 'animal_type': 'Mythic', 'breed
': 'Unidentifiable', 'color': 'Void', 'location': 'right behind you', 'and': 'so on'}
Data inserted successfully with ObjectID 62c5e67bdc37cbc44aaaea6d

True
```

4. Perform UPDATE operations by pass a key/value filter to select the documents to update, and a second update information key/value argument to update the document fields, to the .update() function.



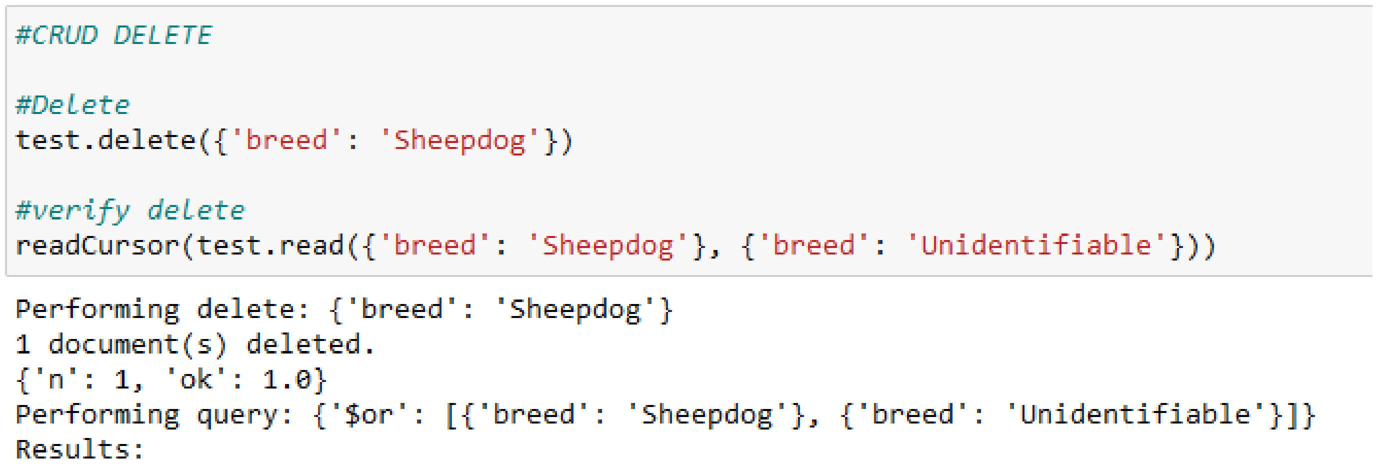
```
#CRUD UPDATE

#update
test.update({'breed': 'Unidentifiable'}, {'breed':'Sheepdog',
                                          'animal_type': 'Other',
                                          'color': 'Black',
                                          'location': 'USA'})
```

```
Performing update: {'$set': {'breed': 'Sheepdog', 'animal_type': 'Other', 'color': 'Black', 'location': 'USA'}}
with filter: {'breed': 'Unidentifiable'}
1 document(s) updated.
{'n': 1, 'nModified': 1, 'ok': 1.0, 'updatedExisting': True}
```

```
#verify update
readCursor(test.read({'breed':'Sheepdog'}))
```

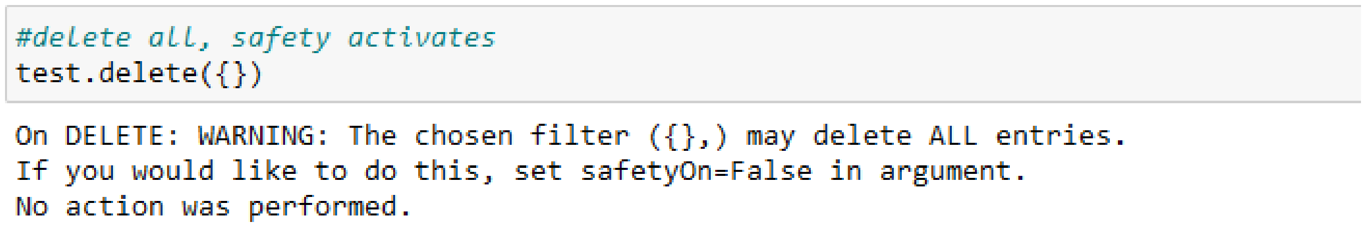
```
Performing query: {'breed': 'Sheepdog'}
Results:
{   '': 111714,
    '_id': ObjectId('62ca1abb80400de9dc1f1502'),
    'age_upon_outcome': '2 years',
    'and': 'so on',
    'animal_id': 'ABCDEFGHIJK',
    'animal_type': 'Other',
    'breed': 'Sheepdog',
    'color': 'Black',
    'location': 'USA'}
```

5. **Perform DELETE operations by passing a set of key/values to search for to the .delete() function. Files found matching the parameters will be deleted from the database.**



```
#CRUD DELETE

#Delete
test.delete({'breed': 'Sheepdog'})

#verify delete
readCursor(test.read({'breed': 'Sheepdog'}, {'breed': 'Unidentifiable'}))
```

```
Performing delete: {'breed': 'Sheepdog'}
1 document(s) deleted.
{'n': 1, 'ok': 1.0}
Performing query: {'$or': [{'breed': 'Sheepdog'}, {'breed': 'Unidentifiable'}]}
Results:
```



```
#delete all, safety activates
test.delete({})
```

```
On DELETE: WARNING: The chosen filter ({},) may delete ALL entries.
If you would like to do this, set safetyOn=False in argument.
No action was performed.
```

Development Process

Development of this project was done primarily using documentation for PyMongo (https://pymongo.readthedocs.io/en/stable/) and for MongoDB (https://www.mongodb.com/docs/). PyMongo was chosen for this project in part because of its simple setup and great support for this application.

Some challenges encountered for the READ function:

-Understanding how results are given in cursor. I initially was focused on getting a number from the cursor for how many hits were found for a given query, which does not exist. Instead, a separate request would be needed.

-Building the OR request for multiple parameters. My original issue stemmed from mistakenly passing a string of the desired $or command instead of sending a dictionary as the pymongo API requires.

The CREATE function development went much smoother, and most of the work is done by the pymongo library for us here as we are essentially just passing along the document to insert then verifying the insertion.

Roadmap/Features

MongoCRUD currently supports a single target database/collection at localhost:27017 (mongo default) with basic CRUD capabilities. The following additions are planned in future releases.

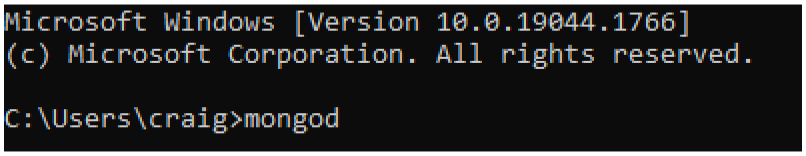
Change MongoDB server target
Change database/collection
Pass projection parameters on READ requests

Any comments or additional info, or collab requests, find me at:

Contact
Craig O'Loughlin
craig.oloughlin@snhu.edu
https://github.com/oloughlinc

Appendix A: MongoDB Server Setup (local)

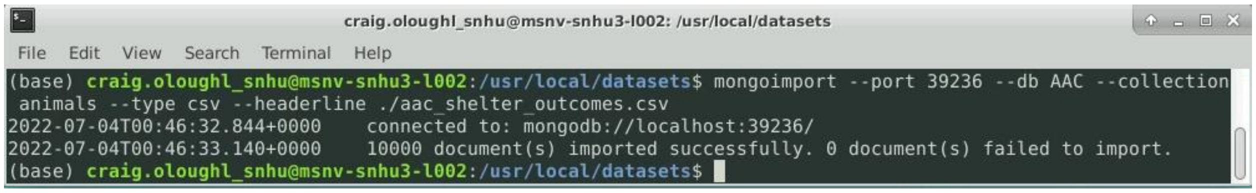
1. **Start a mongo server by running the *mongod* command. Example on Windows (mongo installation location added to PATH already):**



```
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\craig>mongod
```

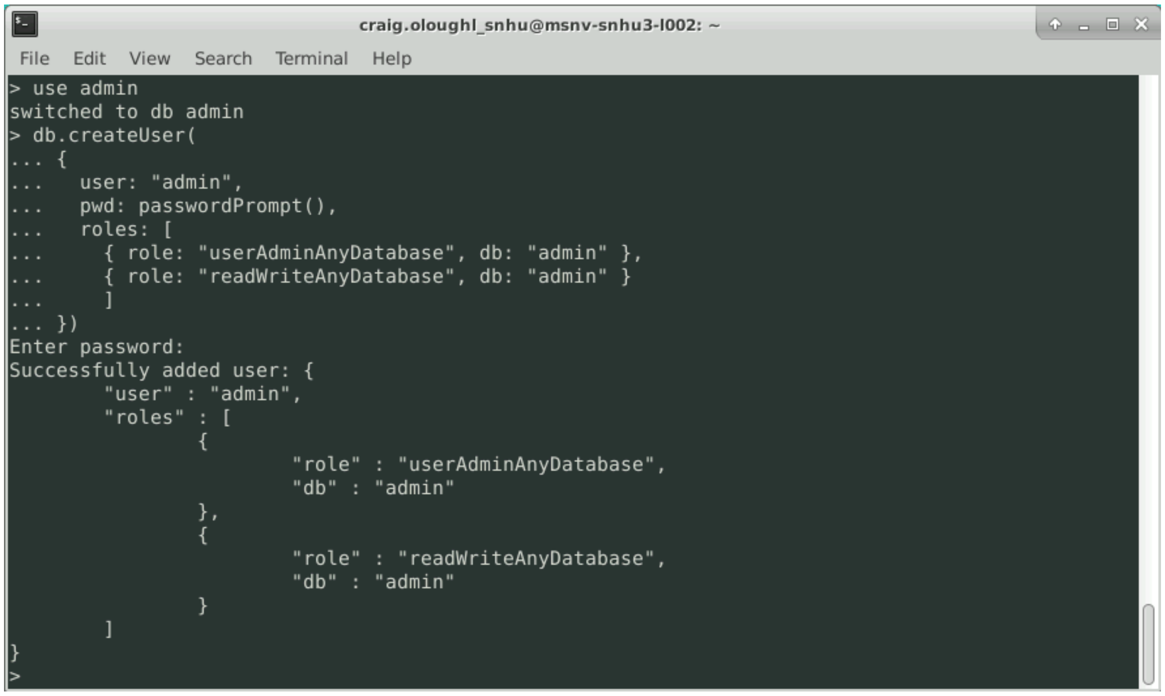
This will start a new mongod instance on default port 27017 of your machine. If the port is different make a note of it.

2. **Use the mongoimport tool to load a new database from a file (you may also manually create a database in the mongo shell). Below we use mongoimport to load a .csv file into a new collection 'animals'.**


craig.oloughl_snhu@msnv-snhu3-l002: /usr/local/datasets
File Edit View Search Terminal Help

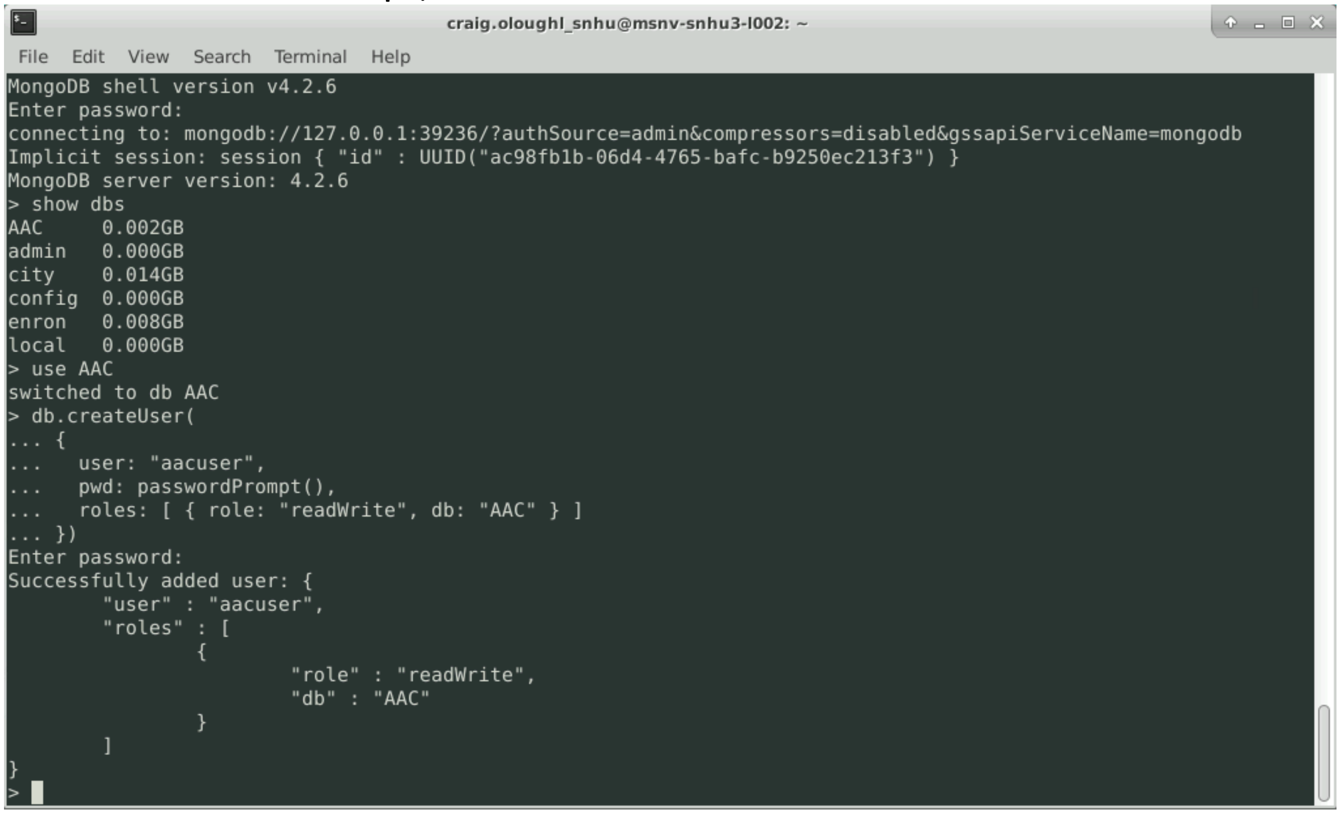
```
(base) craig.oloughl_snhu@msnv-snhu3-l002:/usr/local/datasets$ mongoimport --port 39236 --db AAC --collection
 animals --type csv --headerline ./aac_shelter_outcomes.csv
2022-07-04T00:46:32.844+0000    connected to: mongodb://localhost:39236/
2022-07-04T00:46:33.140+0000    10000 document(s) imported successfully. 0 document(s) failed to import.
(base) craig.oloughl_snhu@msnv-snhu3-l002:/usr/local/datasets$
```

3. **From the mongo shell, we can create an 'administrator' role with access to Admin functions on any database.**


craig.oloughl_snhu@msnv-snhu3-l002: ~
File Edit View Search Terminal Help

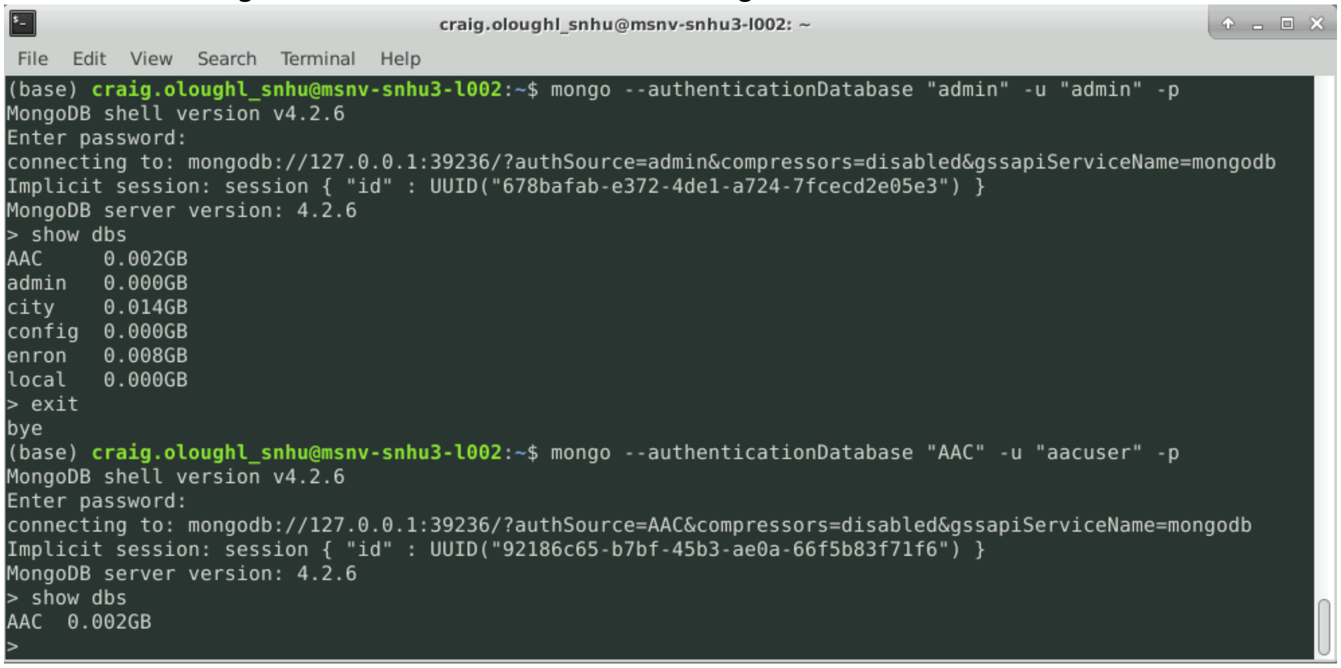
```
> use admin
switched to db admin
> db.createUser(
... {
...    user: "admin",
...    pwd: passwordPrompt(),
...    roles: [
...      { role: "userAdminAnyDatabase", db: "admin" },
...      { role: "readWriteAnyDatabase", db: "admin" }
...      ]
... })
Enter password:
Successfully added user: {
        "user" : "admin",
        "roles" : [
                {
                        "role" : "userAdminAnyDatabase",
                        "db" : "admin"
                },
                {
                        "role" : "readWriteAnyDatabase",
                        "db" : "admin"
                }
        ]
}
>
```

4. **We can also create a 'user' account with read/write access only on a specific database. In this case, we use the database created in step 2, 'AAC'.**


craig.oloughl_snhu@msnv-snhu3-l002: ~
File Edit View Search Terminal Help

```
MongoDB shell version v4.2.6
Enter password:
connecting to: mongodb://127.0.0.1:39236/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("ac98fb1b-06d4-4765-bafc-b9250ec213f3") }
MongoDB server version: 4.2.6
> show dbs
AAC     0.002GB
admin   0.000GB
city    0.014GB
config  0.000GB
enron   0.008GB
local   0.000GB
> use AAC
switched to db AAC
> db.createUser(
... {
...   user: "aacuser",
...   pwd: passwordPrompt(),
...   roles: [ { role: "readWrite", db: "AAC" } ]
... })
Enter password:
Successfully added user: {
        "user" : "aacuser",
        "roles" : [
                {
                        "role" : "readWrite",
                        "db" : "AAC"
                }
        ]
}
> 
```

5. **And we can then login to either user or admin from the mongo shell**


craig.oloughl_snhu@msnv-snhu3-l002: ~
File Edit View Search Terminal Help

```
(base) craig.oloughl_snhu@msnv-snhu3-l002:~$ mongo --authenticationDatabase "admin" -u "admin" -p
MongoDB shell version v4.2.6
Enter password:
connecting to: mongodb://127.0.0.1:39236/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("678bafab-e372-4de1-a724-7fcecd2e05e3") }
MongoDB server version: 4.2.6
> show dbs
AAC     0.002GB
admin   0.000GB
city    0.014GB
config  0.000GB
enron   0.008GB
local   0.000GB
> exit
bye
(base) craig.oloughl_snhu@msnv-snhu3-l002:~$ mongo --authenticationDatabase "AAC" -u "aacuser" -p
MongoDB shell version v4.2.6
Enter password:
connecting to: mongodb://127.0.0.1:39236/?authSource=AAC&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("92186c65-b7bf-45b3-ae0a-66f5b83f71f6") }
MongoDB server version: 4.2.6
> show dbs
AAC  0.002GB
>
```