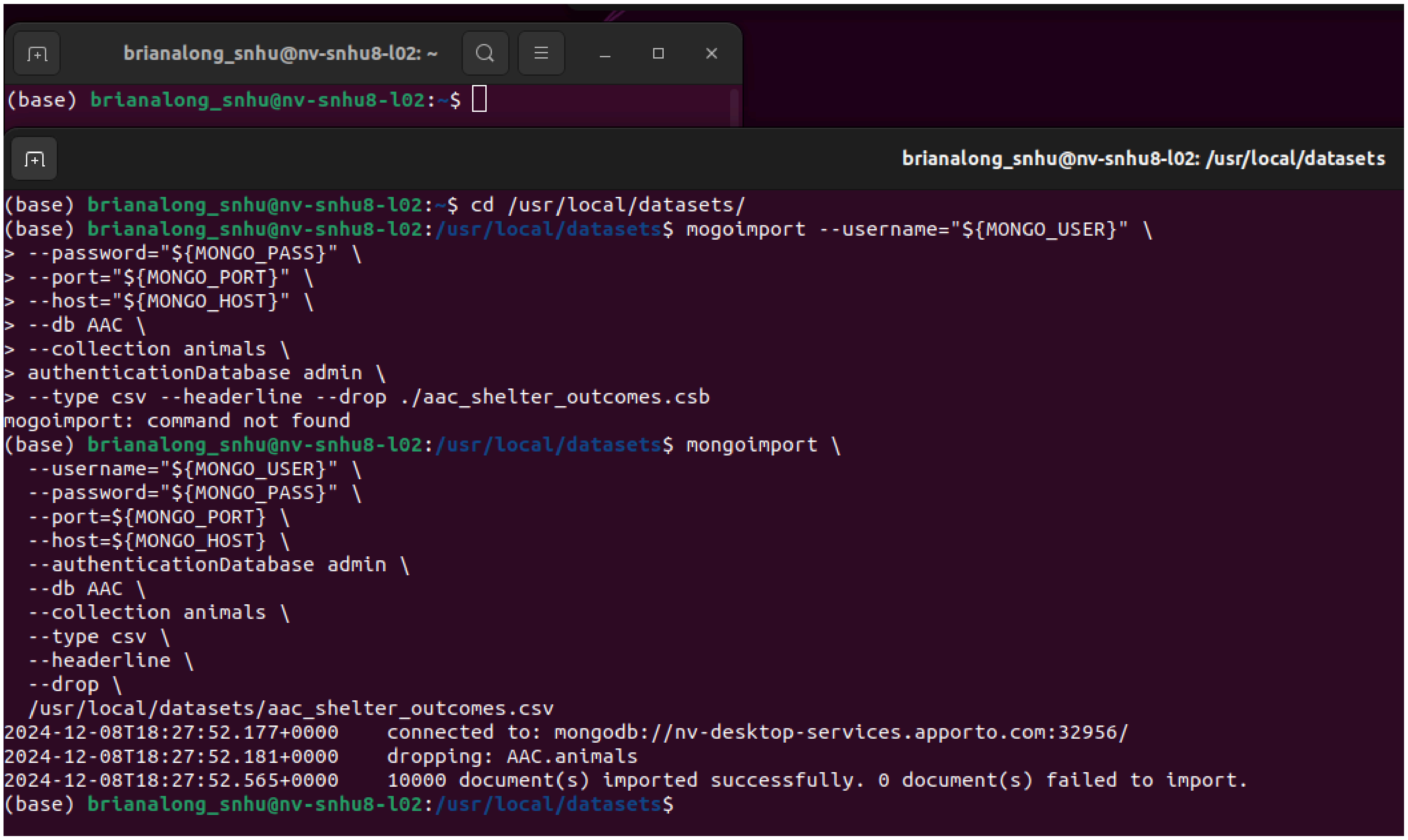
Briana Long
CS340
Project one

1. Upload the Austin Animal Center Outcomes data set into MongoDB by inserting a CSV file using the appropriate MongoDB import tool. The data set is located in the Supporting Materials section. Complete the import using the mongoimport tool, and take screenshots of both the import command and its execution. You will include these screenshots in your README file later.


brianalong_snhu@nv-snhu8-l02: ~
brianalong_snhu@nv-snhu8-l02: /usr/local/datasets

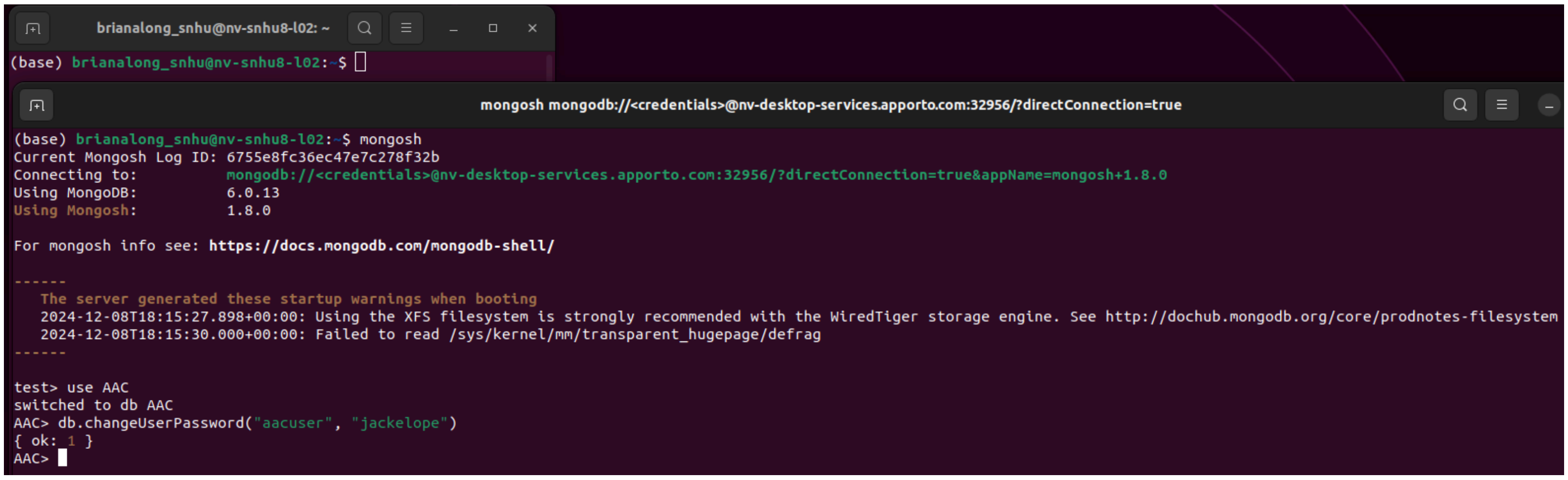
```
(base) brianalong_snhu@nv-snhu8-l02:~$ 
```

```
(base) brianalong_snhu@nv-snhu8-l02:~$ cd /usr/local/datasets/
(base) brianalong_snhu@nv-snhu8-l02:/usr/local/datasets$ mogoimport --username="${MONGO_USER}" \
> --password="${MONGO_PASS}" \
> --port="${MONGO_PORT}" \
> --host="${MONGO_HOST}" \
> --db AAC \
> --collection animals \
> authenticationDatabase admin \
> --type csv --headerline --drop ./aac_shelter_outcomes.csb
mogoimport: command not found
(base) brianalong_snhu@nv-snhu8-l02:/usr/local/datasets$ mongoimport \
  --username="${MONGO_USER}" \
  --password="${MONGO_PASS}" \
  --port=${MONGO_PORT} \
  --host=${MONGO_HOST} \
  --authenticationDatabase admin \
  --db AAC \
  --collection animals \
  --type csv \
  --headerline \
  --drop \
  /usr/local/datasets/aac_shelter_outcomes.csv
2024-12-08T18:27:52.177+0000    connected to: mongodb://nv-desktop-services.apporto.com:32956/
2024-12-08T18:27:52.181+0000    dropping: AAC.animals
2024-12-08T18:27:52.565+0000    10000 document(s) imported successfully. 0 document(s) failed to import.
(base) brianalong_snhu@nv-snhu8-l02:/usr/local/datasets$
```

2. Create a user account in the mongo shell to ensure user authentication to the database and collection you created. Be sure to take a screenshot of the mongo shell execution command screen that shows your login process. You will include this screenshot in your README file later.

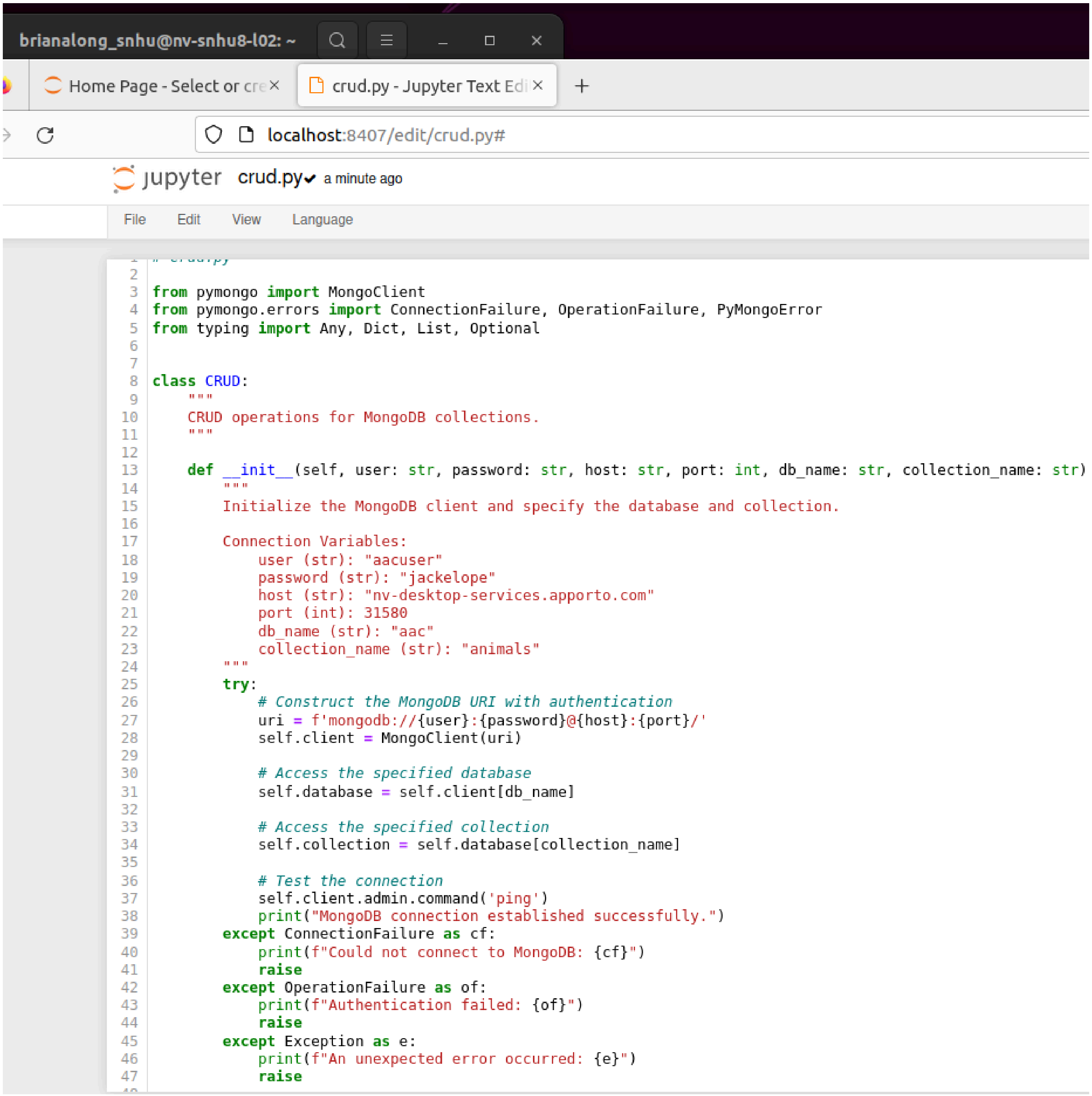
```
AAC> db.createUser({
...   user: "aacuser",
...   pwd: "jackelope",
...   roles: [{ role: "readWrite", db: "AAC" }]
```

... })



```
brianalong_snhu@nv-snhu8-l02: ~
(base) brianalong_snhu@nv-snhu8-l02:~$

mongosh mongodb://<credentials>@nv-desktop-services.apporto.com:32956/?directConnection=true
(base) brianalong_snhu@nv-snhu8-l02:~$ mongosh
Current Mongosh Log ID: 6755e8fc36ec47e7c278f32b
Connecting to:          mongodb://<credentials>@nv-desktop-services.apporto.com:32956/?directConnection=true&appName=mongosh+1.8.0
Using MongoDB:          6.0.13
Using Mongosh:          1.8.0

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

------
   The server generated these startup warnings when booting
   2024-12-08T18:15:27.898+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2024-12-08T18:15:30.000+00:00: Failed to read /sys/kernel/mm/transparent_hugepage/defrag
------

test> use AAC
switched to db AAC
AAC> db.changeUserPassword("aacuser", "jackelope")
{ ok: 1 }
AAC>
```

3.Next, you must develop a Python module in a PY file, using object-oriented programming methodology, to enable CRUD functionality for the database. Other Python scripts must be able to import your Python code, so it must support code reusability.

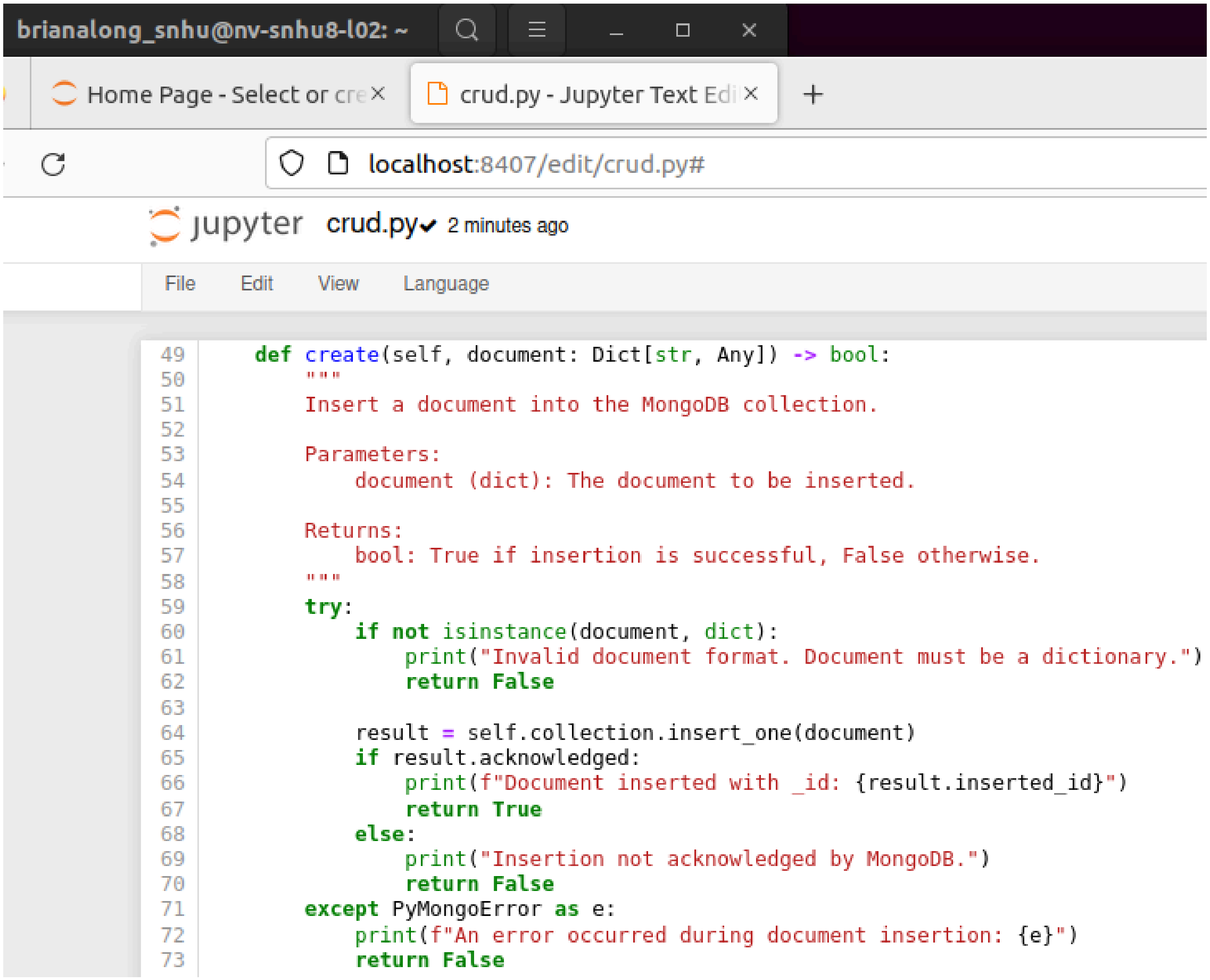

brianalong_snhu@nv-snhu8-l02: ~
Home Page - Select or cre
crud.py - Jupyter Text Edi
localhost:8407/edit/crud.py#
jupyter crud.py a minute ago
File Edit View Language

```python

from pymongo import MongoClient
from pymongo.errors import ConnectionFailure, OperationFailure, PyMongoError
from typing import Any, Dict, List, Optional


class CRUD:
    """
    CRUD operations for MongoDB collections.
    """

    def __init__(self, user: str, password: str, host: str, port: int, db_name: str, collection_name: str)
        """
        Initialize the MongoDB client and specify the database and collection.

        Connection Variables:
            user (str): "aacuser"
            password (str): "jackelope"
            host (str): "nv-desktop-services.apporto.com"
            port (int): 31580
            db_name (str): "aac"
            collection_name (str): "animals"
        """
        try:
            # Construct the MongoDB URI with authentication
            uri = f'mongodb://{user}:{password}@{host}:{port}/'
            self.client = MongoClient(uri)

            # Access the specified database
            self.database = self.client[db_name]

            # Access the specified collection
            self.collection = self.database[collection_name]

            # Test the connection
            self.client.admin.command('ping')
            print("MongoDB connection established successfully.")
        except ConnectionFailure as cf:
            print(f"Could not connect to MongoDB: {cf}")
            raise
        except OperationFailure as of:
            print(f"Authentication failed: {of}")
            raise
        except Exception as e:
            print(f"An unexpected error occurred: {e}")
            raise
```

Develop a CRUD class that, when instantiated, provides the following functionality:
A *Create* method that inserts a document into a specified MongoDB database and collection

Input -> argument to function should be the key/value lookup pair to use with the MongoDB driver find API call.

Return -> “True” if successful insert, else “False”.


brianalong_snhu@nv-snhu8-l02: ~
Home Page - Select or cre
crud.py - Jupyter Text Edi
localhost:8407/edit/crud.py#
jupyter crud.py 2 minutes ago
File Edit View Language

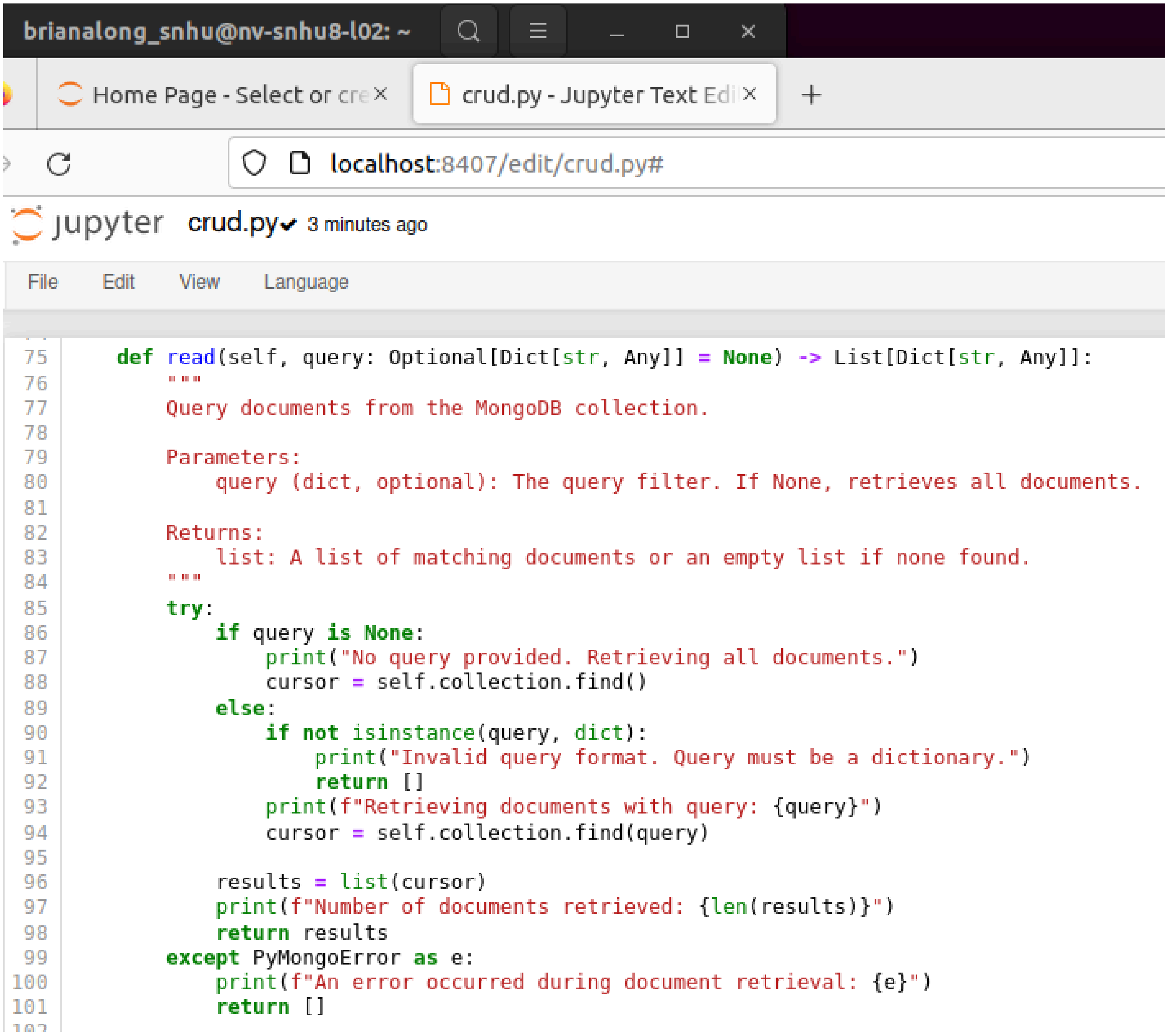
```python
    def create(self, document: Dict[str, Any]) -> bool:
        """
        Insert a document into the MongoDB collection.

        Parameters:
            document (dict): The document to be inserted.

        Returns:
            bool: True if insertion is successful, False otherwise.
        """
        try:
            if not isinstance(document, dict):
                print("Invalid document format. Document must be a dictionary.")
                return False

            result = self.collection.insert_one(document)
            if result.acknowledged:
                print(f"Document inserted with _id: {result.inserted_id}")
                return True
            else:
                print("Insertion not acknowledged by MongoDB.")
                return False
        except PyMongoError as e:
            print(f"An error occurred during document insertion: {e}")
            return False
```

A *Read* method that queries for document(s) from a specified MongoDB database and specified collection

Input -> arguments to function should be the key/value lookup pair to use with the MongoDB driver find API call.

Return -> result in a list if the command is successful, else an empty list.

Important: Be sure to use find() instead of find_one() when developing your method. Hint: You will have to work with the MongoDB cursor returned by the find() method.

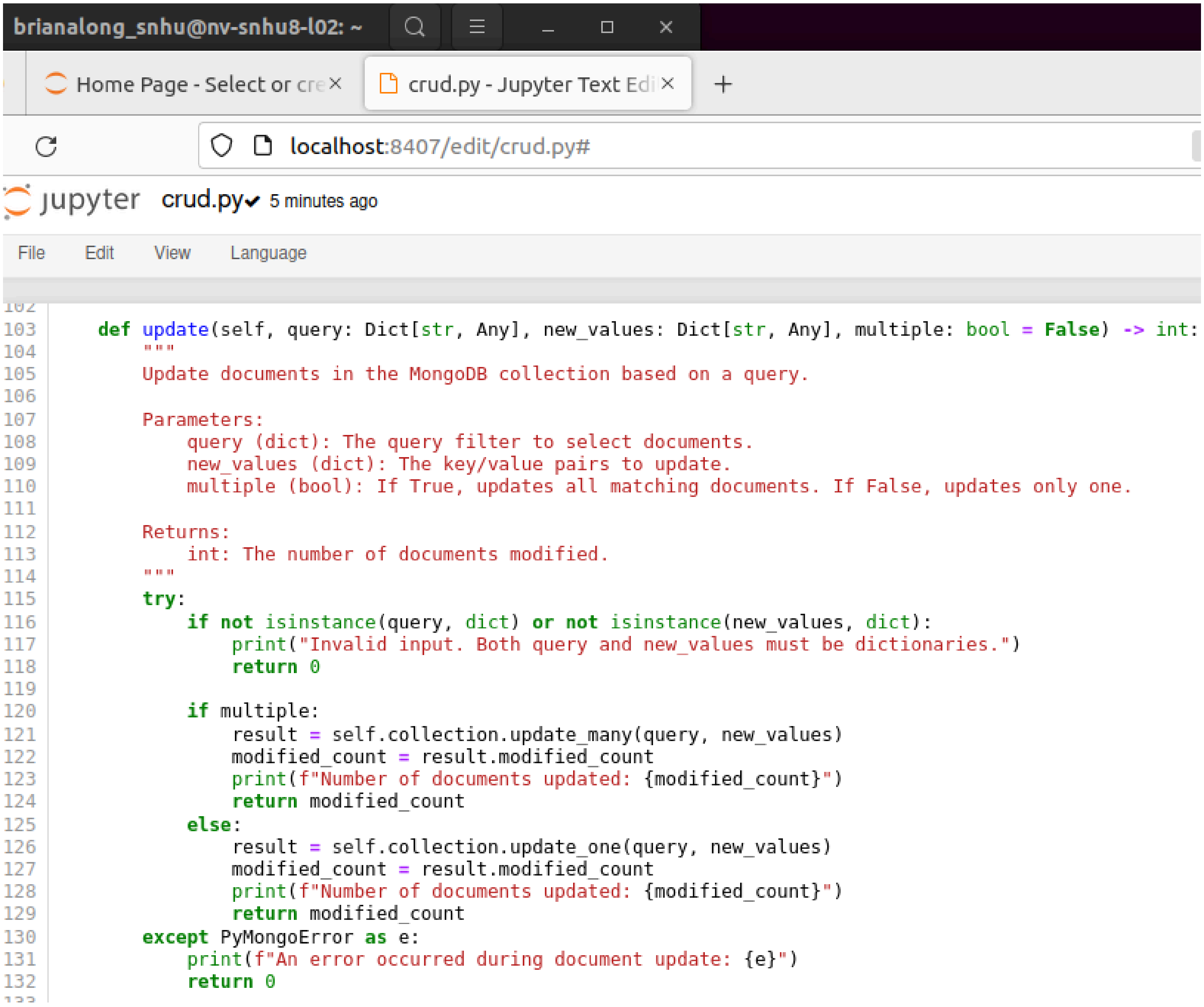

brianalong_snhu@nv-snhu8-l02: ~
Home Page - Select or cre
crud.py - Jupyter Text Edi
localhost:8407/edit/crud.py#
jupyter
crud.py 3 minutes ago
File
Edit
View
Language

```
    def read(self, query: Optional[Dict[str, Any]] = None) -> List[Dict[str, Any]]:
        """
        Query documents from the MongoDB collection.

        Parameters:
            query (dict, optional): The query filter. If None, retrieves all documents.

        Returns:
            list: A list of matching documents or an empty list if none found.
        """
        try:
            if query is None:
                print("No query provided. Retrieving all documents.")
                cursor = self.collection.find()
            else:
                if not isinstance(query, dict):
                    print("Invalid query format. Query must be a dictionary.")
                    return []
                print(f"Retrieving documents with query: {query}")
                cursor = self.collection.find(query)

            results = list(cursor)
            print(f"Number of documents retrieved: {len(results)}")
            return results
        except PyMongoError as e:
            print(f"An error occurred during document retrieval: {e}")
            return []
```

An *Update* method that queries for and changes document(s) from a specified MongoDB database and specified collection

Input -> arguments to function should be the key/value lookup pair to use with the MongoDB driver Find API call. The last argument to function will be a set of key/value pairs in the data type acceptable to the MongoDB driver update_one() or update_many() API call.

Return -> The number of objects modified in the collection.

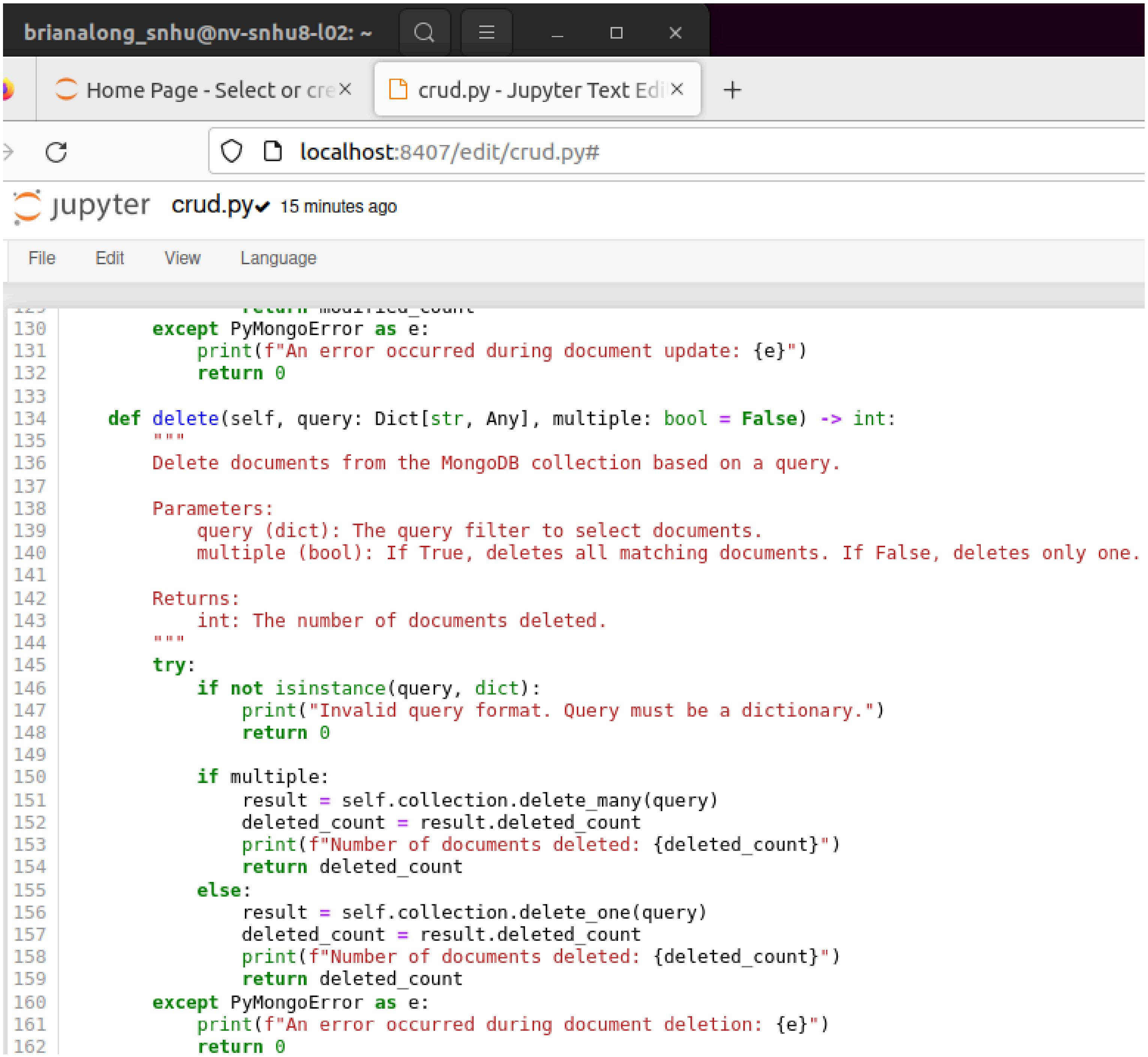

brianalong_snhu@nv-snhu8-l02: ~
Home Page - Select or cre
crud.py - Jupyter Text Edi
localhost:8407/edit/crud.py#
jupyter crud.py 5 minutes ago
File Edit View Language

```
    def update(self, query: Dict[str, Any], new_values: Dict[str, Any], multiple: bool = False) -> int:
        """
        Update documents in the MongoDB collection based on a query.

        Parameters:
            query (dict): The query filter to select documents.
            new_values (dict): The key/value pairs to update.
            multiple (bool): If True, updates all matching documents. If False, updates only one.

        Returns:
            int: The number of documents modified.
        """
        try:
            if not isinstance(query, dict) or not isinstance(new_values, dict):
                print("Invalid input. Both query and new_values must be dictionaries.")
                return 0

            if multiple:
                result = self.collection.update_many(query, new_values)
                modified_count = result.modified_count
                print(f"Number of documents updated: {modified_count}")
                return modified_count
            else:
                result = self.collection.update_one(query, new_values)
                modified_count = result.modified_count
                print(f"Number of documents updated: {modified_count}")
                return modified_count
        except PyMongoError as e:
            print(f"An error occurred during document update: {e}")
            return 0
```

A *Delete* method that queries for and removes document(s) from a specified MongoDB database and specified collection

Input -> arguments to function should be the key/value lookup pair to use with the MongoDB driver find API call.

Return -> The number of objects removed from the collection.

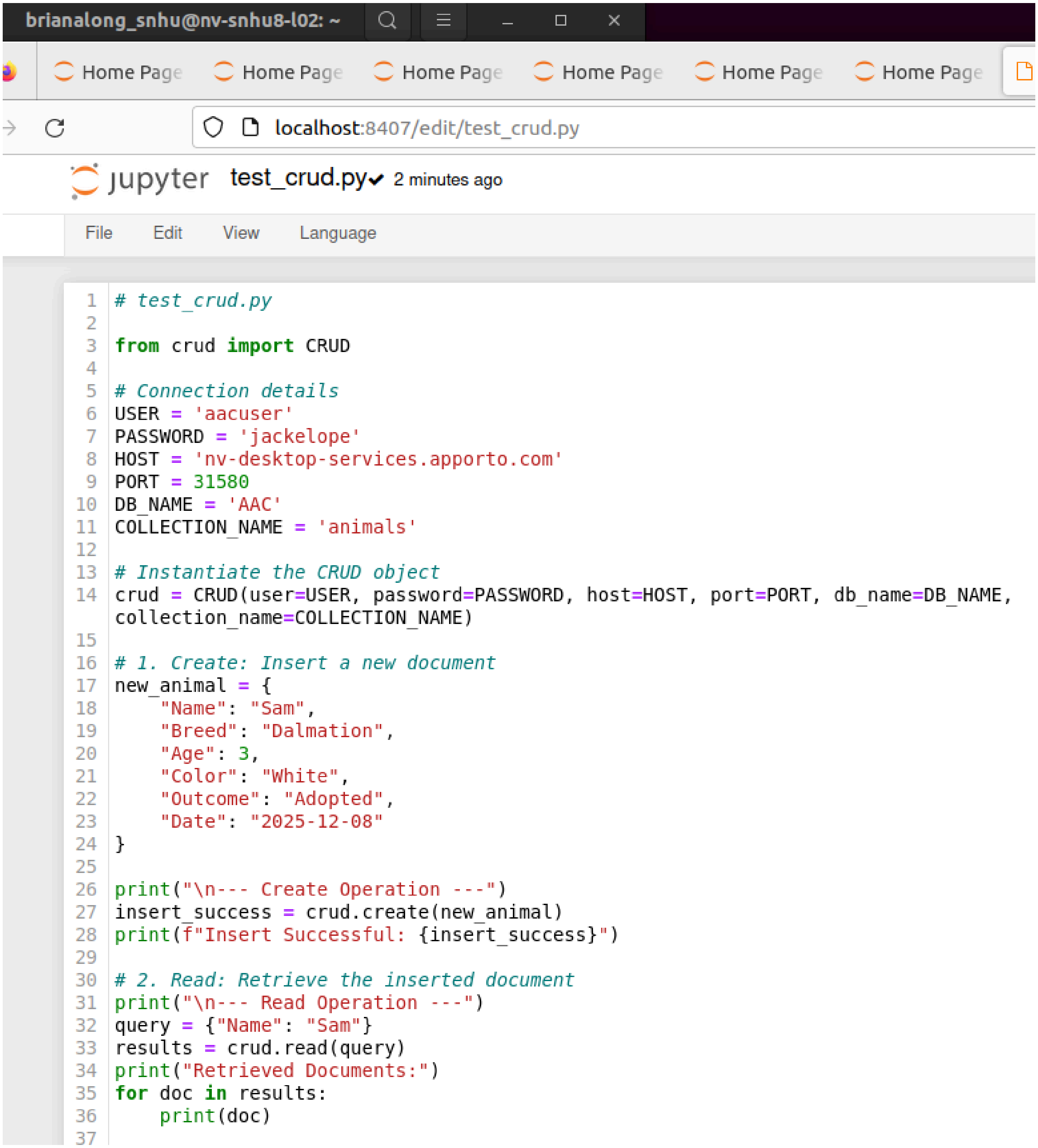


brianalong_snhu@nv-snhu8-l02: ~
Home Page - Select or cre
crud.py - Jupyter Text Edi
localhost:8407/edit/crud.py#
jupyter crud.py 15 minutes ago
File Edit View Language

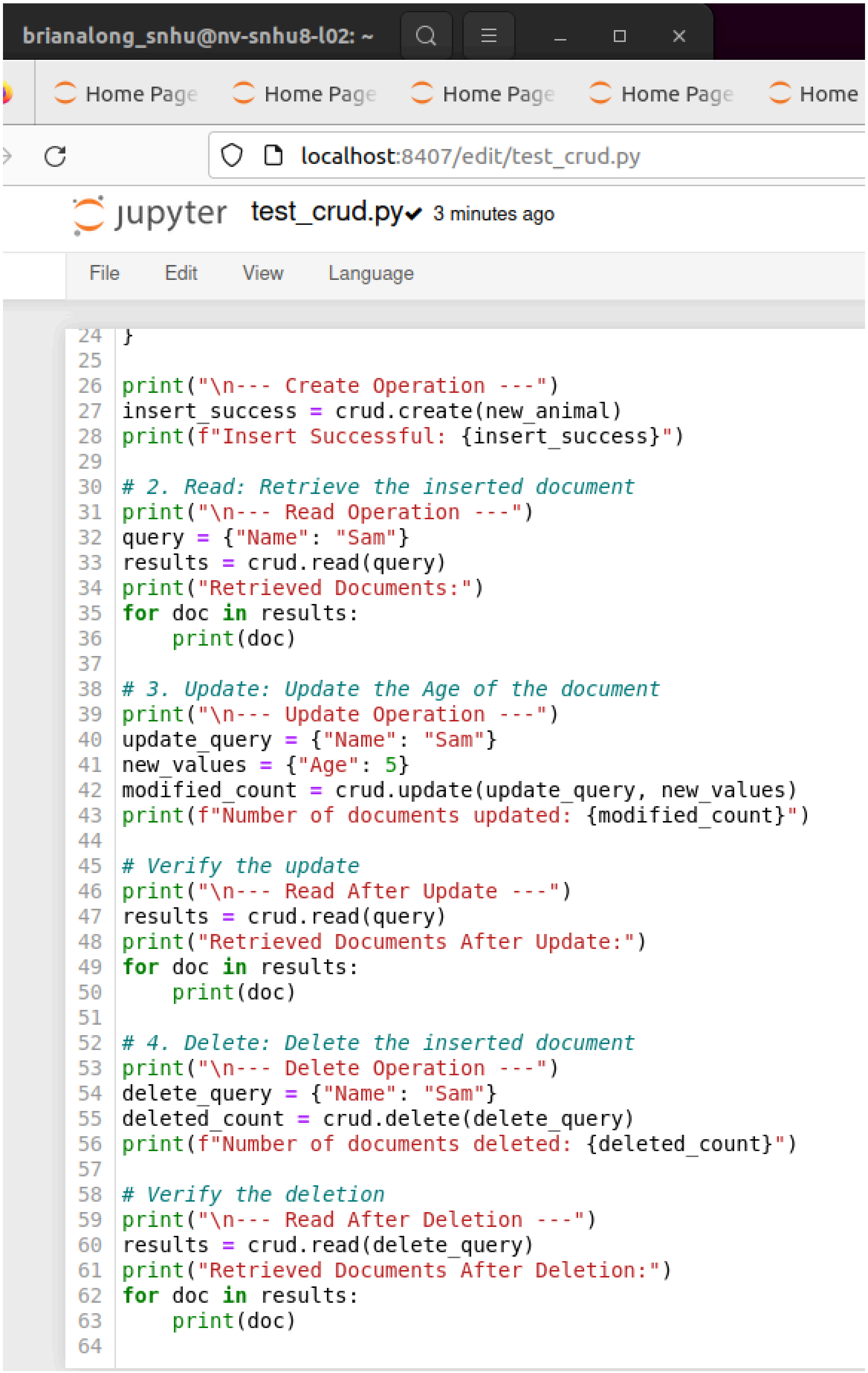
```
        except PyMongoError as e:
            print(f"An error occurred during document update: {e}")
            return 0

    def delete(self, query: Dict[str, Any], multiple: bool = False) -> int:
        """
        Delete documents from the MongoDB collection based on a query.

        Parameters:
            query (dict): The query filter to select documents.
            multiple (bool): If True, deletes all matching documents. If False, deletes only one.

        Returns:
            int: The number of documents deleted.
        """
        try:
            if not isinstance(query, dict):
                print("Invalid query format. Query must be a dictionary.")
                return 0

            if multiple:
                result = self.collection.delete_many(query)
                deleted_count = result.deleted_count
                print(f"Number of documents deleted: {deleted_count}")
                return deleted_count
            else:
                result = self.collection.delete_one(query)
                deleted_count = result.deleted_count
                print(f"Number of documents deleted: {deleted_count}")
                return deleted_count
        except PyMongoError as e:
            print(f"An error occurred during document deletion: {e}")
            return 0
```

Finally, you must test your Python module to make sure it works. To do this testing, create a Python script that imports your CRUD Python module to call and test all instances of CRUD functionality. Be sure to create this script in a separate Jupyter Notebook (IPYNB) file and import and instantiate an object from your CRUD library to effect changes in MongoDB. Be sure to use the username and password for the “aacuser” account for authentication when instantiating the class. After creating your script, execute it in Jupyter Notebook and take screenshots of the commands and their execution. You will include these screenshots in your README file later.

Note: If you completed the Module Four Milestone, you have already begun this work. Expand your script to call and test the Update and Delete functionality.



```python
# test_crud.py

from crud import CRUD

# Connection details
USER = 'aacuser'
PASSWORD = 'jackelope'
HOST = 'nv-desktop-services.apporto.com'
PORT = 31580
DB_NAME = 'AAC'
COLLECTION_NAME = 'animals'

# Instantiate the CRUD object
crud = CRUD(user=USER, password=PASSWORD, host=HOST, port=PORT, db_name=DB_NAME,
collection_name=COLLECTION_NAME)

# 1. Create: Insert a new document
new_animal = {
    "Name": "Sam",
    "Breed": "Dalmation",
    "Age": 3,
    "Color": "White",
    "Outcome": "Adopted",
    "Date": "2025-12-08"
}

print("\n--- Create Operation ---")
insert_success = crud.create(new_animal)
print(f"Insert Successful: {insert_success}")

# 2. Read: Retrieve the inserted document
print("\n--- Read Operation ---")
query = {"Name": "Sam"}
results = crud.read(query)
print("Retrieved Documents:")
for doc in results:
    print(doc)

```



```python
}

print("\n--- Create Operation ---")
insert_success = crud.create(new_animal)
print(f"Insert Successful: {insert_success}")

# 2. Read: Retrieve the inserted document
print("\n--- Read Operation ---")
query = {"Name": "Sam"}
results = crud.read(query)
print("Retrieved Documents:")
for doc in results:
    print(doc)

# 3. Update: Update the Age of the document
print("\n--- Update Operation ---")
update_query = {"Name": "Sam"}
new_values = {"Age": 5}
modified_count = crud.update(update_query, new_values)
print(f"Number of documents updated: {modified_count}")

# Verify the update
print("\n--- Read After Update ---")
results = crud.read(query)
print("Retrieved Documents After Update:")
for doc in results:
    print(doc)

# 4. Delete: Delete the inserted document
print("\n--- Delete Operation ---")
delete_query = {"Name": "Sam"}
deleted_count = crud.delete(delete_query)
print(f"Number of documents deleted: {deleted_count}")

# Verify the deletion
print("\n--- Read After Deletion ---")
results = crud.read(delete_query)
print("Retrieved Documents After Deletion:")
for doc in results:
    print(doc)

```