



## Overview

As you have learned, CRUD functionality is essential for interacting with databases. In Module Two, you gained practice using CRUD commands within the mongo shell. In order to develop a web application that connects a client-side user interface (such as a dashboard) to a database, it will help to develop a portable Python module that enables CRUD functionality for this data connection. In this milestone, you will begin creating a Python module that enables **create** and **read** functionality. You will finish developing the **update** and **delete** functionality for your Project One submission. This Python module will eventually be used to connect the user interface component to the database component of your dashboard in Project Two.

Note: This milestone requires you to use the “aacuser” account and password that you set up back in the Module Three milestone. If you did not successfully complete that milestone, follow the steps in Part II of the Module Three milestone to set up the “aacuser” account before beginning this milestone.

## Prompt

After completing the readings for this module, you will implement the fundamental operations of creating and reading documents (the C and R of CRUD) in Python. You will use the PyMongo driver to create CRUD functional access to your document collection.

1. Upload the Austin Animal Center (AAC) Outcomes data set into MongoDB by **importing a CSV file using the appropriate MongoDB import tool**. This file is in the `/usr/local/datasets/` directory in Apporto and the filename is “aac\_shelter\_outcomes.csv”. Use the database name “AAC” and collection name “animals”. Complete the import using the mongoimport tool and **take screenshots** of both the import command and its execution.

Note: If you completed the Module Three milestone, you have already completed this step.

2. Next, you must develop a Python module in a PY file, using object-oriented programming methodology, to enable **create** and **read** functionality for the database. To support code reusability, your Python code needs to be importable as a module by other Python scripts.

**Develop** a CRUD class that, when instantiated, provides the following functionality:

**a. A method that inserts a document into a specified MongoDB database and collection**

- i. Input -> argument to function will be set of key/value pairs in the data type acceptable to the MongoDB driver insert API call
- ii. Return -> “True” if successful insert, else “False”

**b. A method that queries for documents from a specified MongoDB database and specified collection**

- i. Input -> arguments to function should be the key/value lookup pair to use with the MongoDB driver find API call
- ii. Return -> result in cursor if successful, else MongoDB returned error message

Important: Be sure to use **find()** instead of **find\_one()** when developing your method.

As you develop your code, be sure to **use industry standard best practices** such as proper naming conventions, exception handling, and in-line comments. This will ensure that your code is easy to read and reusable for future projects.

TIP: Use the following sample code to get started. Note that the authentication to MongoDB is in the initialization method for the CRUD class.

**Example Python Code to Insert a Document**

```

from pymongo import MongoClient
from bson.objectid import ObjectId

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

    def __init__(self):
        # Initializing the MongoClient. This helps to
        # access the MongoDB databases and collections.
        self.client = MongoClient('mongodb://%s:%s@localhost:YOUR_PORT_NUMBER' % (username, password))
        self.database = self.client['project']

    # Complete this create method to implement the C in CRUD.
    def create(self, data):
        if data is not None:
            self.database.animals.insert(data) # data should be dictionary
        else:
            raise Exception("Nothing to save, because data parameter is empty")

    # Create method to implement the R in CRUD.

```

3. Finally, create a Python testing script that imports your CRUD Python module to call and test the create and read instances of CRUD functionality. Be sure to use the username and password for the “aacuser” account for authentication when instantiating the class. This script should be created in a separate Jupyter Notebook IPYNB file, and should import and instantiate an object from your CRUD library to effect changes in MongoDB. After creating your script, execute it in Jupyter Notebook and take screenshots of the commands and their execution.

## Guidelines for Submission

For your submission, you must include the code files for your **Python module (PY file)** and your **Python testing script (IPYNB file)**. You must also submit a **Microsoft Word document** with your **screenshots** from Step 3.

Module Four Milestone Rubric

Criteria	Proficient (100%)	Needs Improvement (70%)	Not Evident (0%)	Value
<b>Create Function</b>	Develops a method that inserts a document into a specified MongoDB database and collection and applies industry standard best practices such as naming conventions, exception handling, and in-line comments	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include developing the create function or applying industry standard best practices	Does not attempt criterion	30

Criteria	Proficient (100%)	Needs Improvement (70%)	Not Evident (0%)	Value
Read Function	Develops a method that queries for documents from a specified MongoDB database and specified collection and applies industry standard best practices such as naming conventions, exception handling, and in-line comments	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include developing the read function or applying industry standard best practices	Does not attempt criterion	30
Python Testing Script	Creates a Python testing script that imports a CRUD Python module to call and test the create and read instances of CRUD functionality	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include creating the Python testing script	Does not attempt criterion	40
Total:				100%