DATA8005 – Distributed Data Management

# Python and PyMongo Lab

## Overview

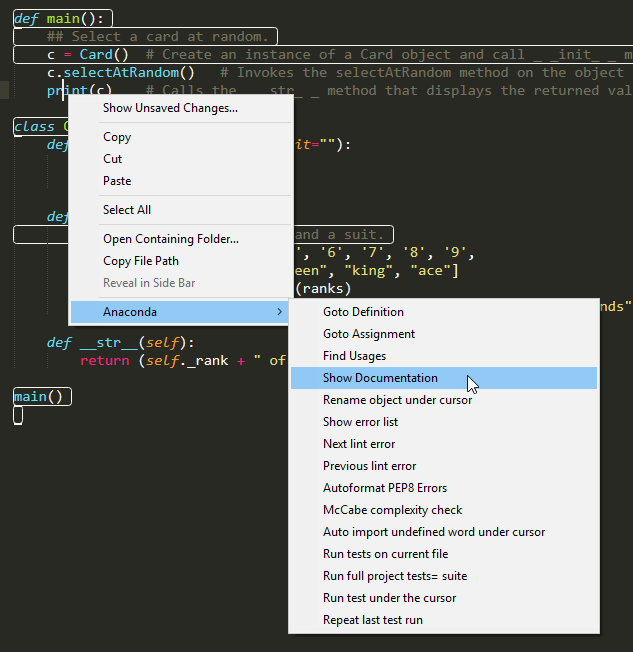
The purpose of this lab is primarily to get you back being comfortable with Python, if you are not as it stands. Once you have built up and run a basic Python script, we will use the PyMongo library to access a MongoDB database. By the end of the lab document (and after completing the exercises), you will have seen your first example of aggregation (the count of documents matching a filter expression).

## Part 1

Create a text file called ddm\_lab1.py somewhere on your H: drive, e.g. H:\DATA8005\Scripts.

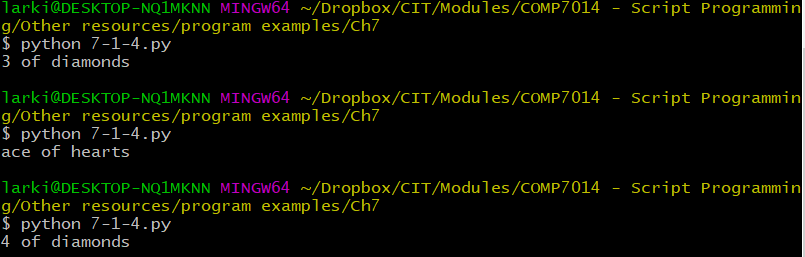
You can edit the file using whatever IDE or text editor you like. PyCharm should be available in the labs, and that would give you advanced editing capabilities as well as debugging. However, some advanced text editors feature support for Python, such as Sublime Text, which has syntax highlighting. If you want code completion in Sublime Text 3, the Anaconda package includes that capability (<http://damnwidget.github.io/anaconda/>).

 *Example of Python syntax highlighting in Sublime Text 3*

 *Anaconda in action (right-clicking on function in this case)*

Don’t worry if you are using Sublime Text and can’t get Anaconda installed.

Then run from command line, switching to the correct director (example below shows Git Bash being used – should be available in the labs if you prefer unix/linux style output and commands; you can always use the Command Prompt).

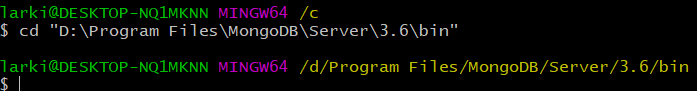


## Part 2

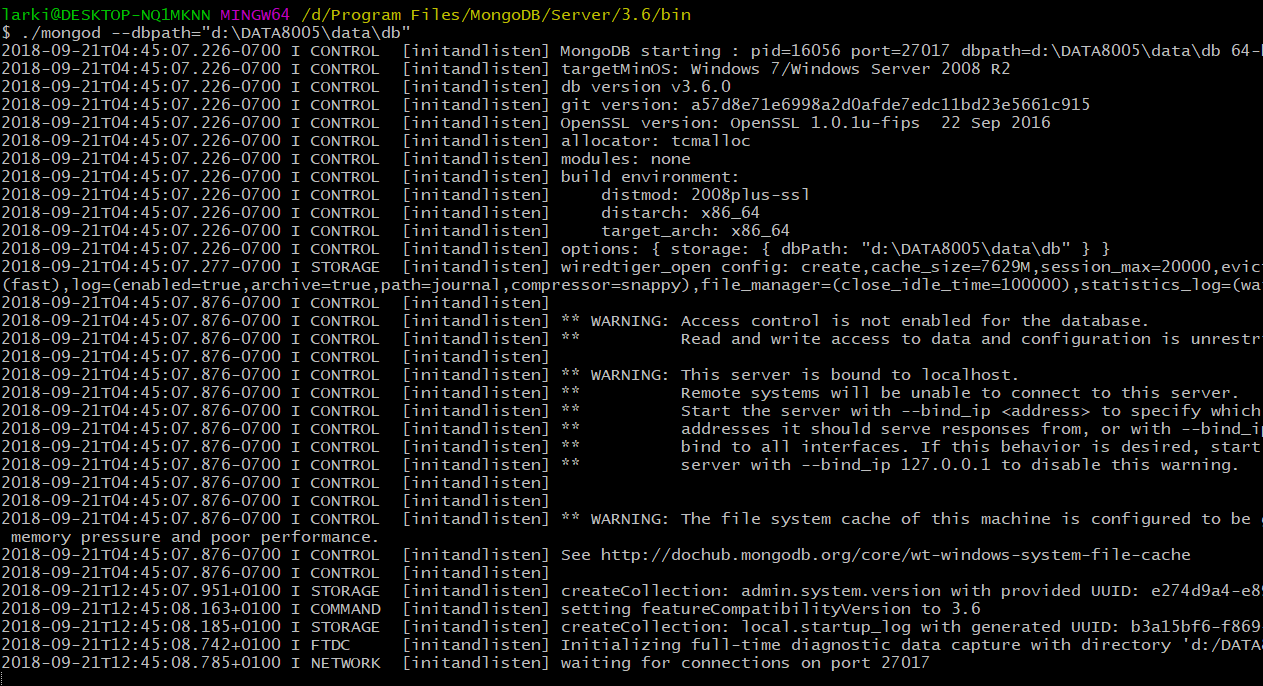
Let’s get a simple MongoDB database and collection created.

Note: depending on how MongoDB was installed, it may already be running as a service, particularly if you installed it on your laptop. If that is the case, you won’t need to start it manually like this.

Somewhere on your PC or laptop should be a MongoDB folder. Using your command line interface, change directory to the bin directory, e.g.



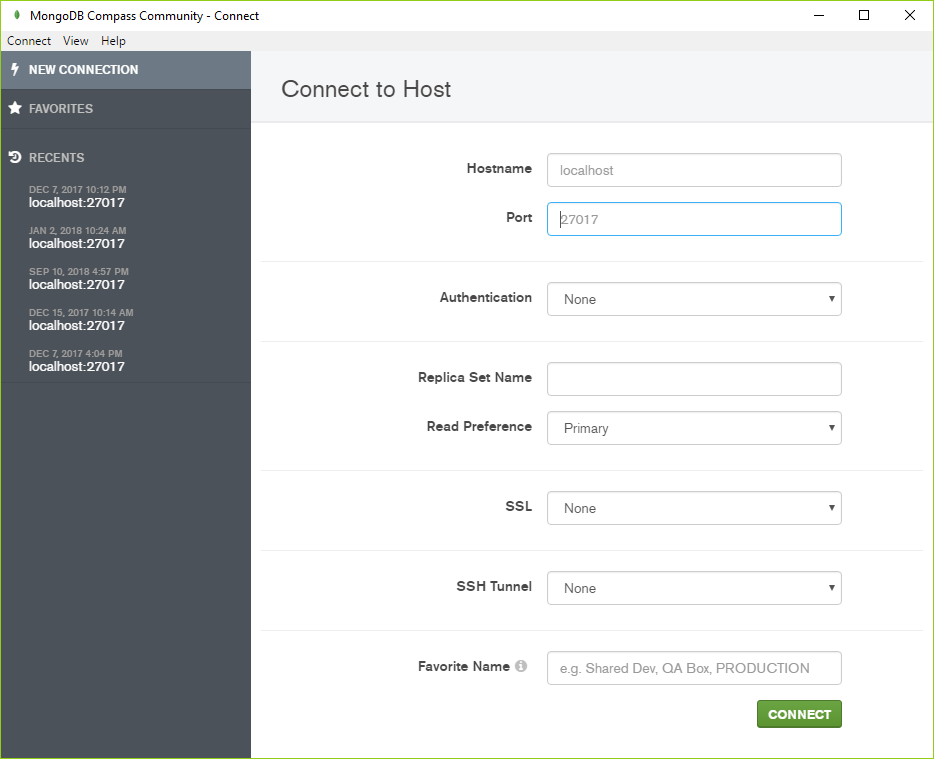
1. Create your data folder on the H: drive (e.g. as H:\DATA8005\data).
2. Within the data folder, create a db folder.
3. Now use the mongod executable to start the MongoDB service with your data/db folder as a parameter:



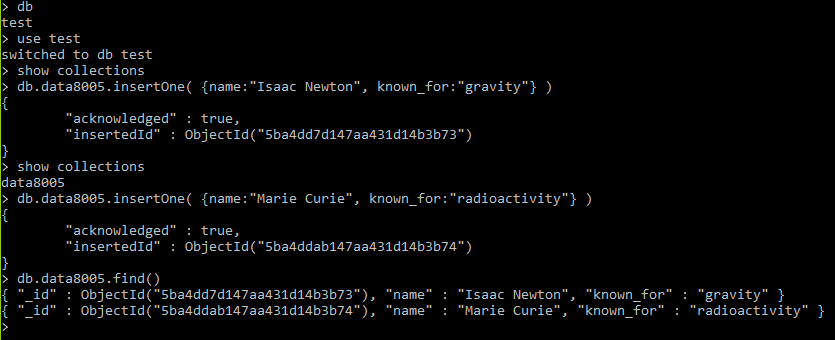
1. That will hog that command line process, so open another command line window and enter the following (you may need to change back to the mongodb bin folder if it’s not on your PATH, e.g. if you can’t use the mongo command from anywhere):  
   mongo (or ./mongo if on Linux or Mac)

You should see a > prompt. (this might not work so well with Git Bash)

Note: there are GUI tools you can use like Compass, but the instructions here will be command-line based.



1. Type each of the following commands in sequence:



We now have a collection, data8005, with two documents, for Isaac Newton and Marie Curie. Now we are ready to code some Python to access this database.

## Part 3

A quick tutorial on PyMongo is available at <http://api.mongodb.com/python/current/tutorial.html>. However, let’s build our own script and use our own database.

We are using the pymongo library and this may need to be installed. An easy way to do this is to use Python’s own “easy\_install”, i.e.

python -m easy\_install pymongo

Assuming that works, let’s go back to ddm\_lab1.py.

Add the following code:

import pymongo

from pymongo import MongoClient

client = MongoClient() # This assumes a default host of

# localhost and the default port of 27017

db = client.test

collection = db.data8005

# Now let's find the Marie Curie document

marie\_curie = collection.find\_one({"name" : "Marie Curie"})

print(marie\_curie)

Then run it and you should see output like this:



Now let us add an update to change what she is known for to “radioactivity,polonium,radium”.

Add the following to the end of your script and re-run:

collection.update\_one({"name":"Marie Curie"},   
 {"$set":{"known\_for":"radioactivity,polonium,radium"}})

marie\_curie = collection.find\_one({"name" : "Marie Curie"})

print(marie\_curie)



Notice how we had to use find\_one again to retrieve the updated document before we print.

The marie\_curie object is a dict(ionary). We can access dict fields directly like this:

print(marie\_curie["name"])

Go ahead and add it and re-run.

Next let us try some filtering. There are several ways to filter documents. We will use one that works for MongoDB 3.6 and later versions.

print(collection.count\_documents({"$expr":{"$gt":[{"$strLenCP":"$name"},5]}}))

This uses the count\_documents function to do the same as COUNT in SQL. It expects a filter as the first parameter. We use $expr to denote a following expression. $gt means greater than with a comparison of the first parameter to the second. The first parameter to $gt is {"$strLenCP":"$name"}, which gets the length of the name field (CP refers to *code points* in the Unicode codespace and so only valid unicode will be counted). Any name of 6 characters or more will be counted.

print(collection.count\_documents({"$expr":{"**$gte**":[{"$strLenCP":"$name"},12]}}))

The code above will only count Isaac Newton and not Marie Curie because “Marie Curie” is not greater than **or equal to** 12 in length.

Now you have the basics of connecting to MongoDB and running some basic queries. Now for some exercises (sample solutions to which will be put on Blackboard by the end of the week).

## Exercises

1. Go to <https://stackoverflow.com/questions/10610131/checking-if-a-field-contains-a-string> and pick a solution that will help you get the number of people with a double-a in their name.
2. Add 2 new fields to the documents: year\_of\_birth and year\_of\_death (find the values on Wikipedia). See <https://docs.mongodb.com/manual/reference/operator/update/set/> for a way to do this.
3. **Replace** the Marie Curie document with a new document as follows:

{ first\_name: “Marie”,  
 last\_name: “Curie”,  
 known\_for: [“radioactivity”,”polonium”,”radium”] }  
See <http://api.mongodb.com/python/current/api/pymongo/collection.html> for find and replace functions.  
Note: you should comment out your previous Marie Curie update code. Also, should you re-run the script after your update, your original collection.find\_one({"name" : "Marie Curie"}) will return a None/null and result in errors, so you should comment that out too along with the prints of Marie Curie data and the $strLenCP code.

1. With exercise 3 complete (and you might as well comment out the replacement code), retrieve the new Marie Curie document and neatly list what she is known for as follows:  
     
   Known For  
   ---------  
   Radioactivity  
   Polonium  
   Radium

Hint: you have already accessed a document field (you printed Marie Curie’s name). We replaced the old known\_for field with an array field. This becomes a list in Python. So how do you loop through a list and print the capitalized version of each element in the list? And I suggest you retrieve the document by first name and last name.

*Next week we will work on aggregation and we will import a fairly large dataset containing artists and artworks to work with. After that we will explore how to split the data across multiple nodes (sharding).*

*You will be given some exercises like these as part of your first assignment (in your own time). They will be worth 10% of your overall module grade (10 of the 50 marks for assignment 1).*