Tutorial 2 – Python with MongoDB

MongoDB is one of the best NoSQL document-oriented databases. If you are working on a project in Python and planning to use MongoDB on the backend, then you are perfectly at the right place. In today's post, we'll teach the basics of Python MongoDB. And will also add a few database programming samples. The ready to use coding snippets will help you in ramping up quickly.

In this Python MongoDB programming tutorial, you'll get to learn where to use MongoDB, and how to create a database. Additionally, we'll give Python code for selecting, inserting, and deleting records from the tables. In fact, we thought to bring every essential that can quickly get you on board. So enjoy this post and we hope to see you writing better database programming scripts in Python. Let's read a few words about MongoDB next.

Both MySQL and MongoDB are open source databases. But unlike MySQL, MongoDB is a document-oriented database. It stores all data as documents. There are many benefits of the MongoDB data modelling approach.

- **Documents function as the independent units.** So it's easier to distribute them which leads to an increase in performance.
- **MongoDB programming is easy.** No need to do the complex translation of objects between application and SQL queries.
- **Flexible schema** makes it easier to store the unstructured data.
- **It uses a document-based query language** to create dynamic queries. It's as powerful as SQL.

Learn Python MongoDB Programming

In this section, we'll discuss the essential Python MongoDB commands to connect and access the database. First of all, let's how to make a connection to MongoDB because after connecting to it only we can perform an access operation on it.

1- How To Connect To MongoDB?

To make a connection to the database we require creating a Mongo Client against the running the <*Mongod*> instance. For this, we'll provide the arguments indicating the host and port where the database is running.
If the MongoDB server is running locally <*default port for MongoDB is 27017*>, then you can proceed with the following command.

```
from pymongo import MongoClient

con = MongoClient('localhost', 27017)
```

For instance, if you are working in a large hybrid setup where the application server runs on a separate machine. Then, you'll need to provide the IP address of that machine while creating the Mongo Client.

```
from pymongo import MongoClient

con = MongoClient('192.168.1.2', 27017)
```

Consequently, if you like to connect on the default <host/port>, then please give the below command.

```
con = MongoClient()
```

Once you run through the above steps, you'll all set to perform the operations on the database.

2- How To Create A Database In MongoDB?

MongoDB voluntarily creates a database as you start to use it. So for the testing purpose, you can execute the below step for DB creation.

```
db = con.testdb
```

Another approach is to use the dictionary-style access for DB creation. See the below code.

```
db = client['testdb']
```

3- How To Access A Collection In MongoDB?

A collection is a group of documents stored in the database. It's the same as a table in RDBMS.

Also, we can access a MongoDB collection in the same way as we did while accessing the database at the last point.

```
my_coll = db.coll_name

#OR do it in the dictonary-style.

my_coll = db['coll_name']
```

4- How To Add Documents To A Collection?

MongoDB models data in JSON format. And it uses the dictionary to store the records as.

```
emp_rec = {'name':emp_name, 'address':emp_addr, 'id':emp_id}
```

Also, to work with collections, Python MongoDB module exposes a set of methods.

For example, the <*insert()*> method, you may like to call it from your code to add documents to a collection.

```
rec_id = my_coll.insert(emp_rec)
```

Since the <*insert()*> method returns a new record, which you can save to a variable <*rec_id*> for using it later.

5- How To Query Data In A Collection?

The Python MongoDB driver also gives you a method <*find()*> to query data from any MongoDB collection. Also, on top of it, you can run the <*pretty()*> method to format the query results.
Here is the code for you to follow.

```
testdb.coll_name.find()
```

While we are providing the <*pretty()*> method example, but use it only if required.

```
testdb.coll_name.find().pretty()
{
"_id" : ObjectId("7abf53ce1220a0213d"),
"name" : emp_name,
"address" : emp_addr,
"id" : emp_id
}
```

Probably, you've taken a note that the <*_id*> is an auto-generated value. Also, it's unique for a particular document.
Finally, you would have to close the open MongoDB connection after completing the transactions. Call the method as given below.

```
con.close()
```

6- How To Update Data In A Collection?

To modify a collection, use any of the following Python MongoDB methods.

- <*update_one()*>,
- <*update_many()*>.

While you may call any of the above methods to update a collection, but you would've to use the <*$set*> macro to change values.
Also, note that we are storing the output into a variable.

```
ret = db.my_coll.update_one(
    {"name": "Post"},
    {
        "$set": {
            "Category": "Programming"
        },
        "$currentDate": {"lastModified": True}
    }
)
```

Consequently, you can verify the result by issuing the following step.

```
ret.modified_count
```

7- How To Remove Data From A Collection?

Same like the update, here is a list of methods to delete the documents.

- *<delete_one()>*,
- *<delete_many()>*.

Check out the below code snippet for removing more than one documents.

```
ret = db.posts.delete_many({"category": "general"})
```

Also, you can call the following method to print the no. of deleted records.

```
ret.deleted_count
```

8. Python MongoDB Demo Script

Since we promised to give you a full database programming script, so here is the Python script to cover the essential Python MongoDB commands.

Also, the script should now be easy for you to understand, as we've already explained the individual commands above.

Sample Code: Python MongoDB Script

```
from pymongo import MongoClient

#connect to the MongoDB.
conn = MongoClient('localhost',27017)

#Access the testdb database and the emp collection.
db = conn.testdb.emp

#create a dictionary to hold emp details.

#create dictionary.
emp_rec = {}

#set flag variable.
flag = True

#loop for data input.
while (flag):
   #ask for input.
   emp_name,emp_addr,emp_id = input("Enter emp name, address
and id: ").split(',')
   #place values in dictionary.
   emp_rec = {'name':emp_name,'address':emp_addr,'id':emp_id}
   #insert the record.
   db.insert(emp_rec)
   #Ask from user if he wants to continue to insert more 
documents?
   flag = input('Enter another record? ')
   if (flag[0].upper() == 'N'):
      flag = False

#find all documents.
ret = db.find()

print()
print('+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-')

#display documents from collection.
for record in ret:
      #print out the document.
      print(record['name'] + ',',record['address'] + 
',',record['id'])

print()

#close the conn to MongoDB
conn.close()
```