**MongoDB and Python**

MongoDB is a cross-platform, document-oriented database that works on the concept of collections and documents. MongoDB offers high speed, high availability, and high scalability.  
The next question which arises in the mind of the people is “Why MongoDB”?  
**Reasons to opt for MongoDB :**

1. It supports hierarchical data structure
2. It supports associate arrays like Dictionaries in Python.
3. Built-in Python drivers to connect python-application with Database. Example- PyMongo
4. It is designed for Big Data.
5. Deployment of MongoDB is very easy.

**MongoDB vs RDBMS**

Table

Description automatically generated

1. First start MongoDB from command prompt using :  
   **Method 1:**

mongod

See port number by default is set 27017 (last line in above image).  
Python has a native library for MongoDB. The name of the available library is “PyMongo”. To import this, execute the following command:

|  |
| --- |
| **from** pymongo **import** MongoClient |

1. **Create a connection :** The very first after importing the module is to create a MongoClient.

|  |
| --- |
| **from** pymongo **import** MongoClient  client **=** MongoClient() |

1. After this, connect to the default host and port. Connection to the host and port is done explicitly. The following command is used to connect the MongoClient on the localhost which runs on port number 27017.

|  |
| --- |
| client **=** MongoClient(‘host’, port\_number)  example:**-** client **=** MongoClient(‘localhost’, 27017) |

1. It can also be done using the following command:

|  |
| --- |
| client **=** MongoClient(“mongodb:**//**localhost:27017**/**”) |

1. **Access DataBase Objects :**To create a database or switch to an existing database we use:  
   **Method 1 : Dictionary-style**

|  |
| --- |
| mydatabase **=** client[‘name\_of\_the\_database’] |

1. **Method2 :**

|  |
| --- |
| mydatabase **=** client.name\_of\_the\_database |

1. If there is no previously created database with this name, MongoDB will implicitly create one for the user.  
   Note : The name of the database fill won’t tolerate any dash (-) used in it. The names like my-Table will raise an error. So, underscore are permitted to use in the name.
2. **Accessing the Collection :**Collections are equivalent to Tables in RDBMS. We access a collection in PyMongo in the same way as we access the Tables in the RDBMS. To access the table, say table name “myTable” of the database, say “mydatabase”.  
   **Method 1:**

|  |
| --- |
| mycollection **=** mydatabase[‘myTable’] |

1. Method 2 :

|  |
| --- |
| mycollection **=** mydatabase.myTable |

1. >MongoDB store the database in the form of dictionaries as shown:>

record = {

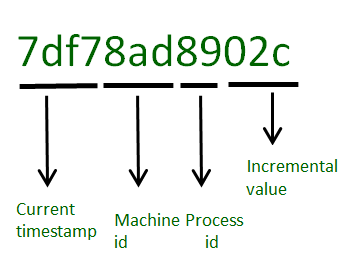
title: 'MongoDB and Python',

description: 'MongoDB is no SQL database',

tags: ['mongodb', 'database', 'NoSQL'],

viewers: 104

}

1. ‘\_id’ is the special key which get automatically added if the programmer forgets to add explicitly. \_id is the 12 bytes hexadecimal number which assures the uniqueness of every inserted document.  
   
2. **Insert the data inside a collection :**  
   Methods used:

insert\_one() or insert\_many()

We normally use insert\_one() method document into our collections. Say, we wish to enter the data named as record into the ’myTable’ of ‘mydatabase’.

|  |
| --- |
| rec **=** myTable.insert\_one(record) |

The whole code looks likes this when needs to be implemented.

|  |
| --- |
| # importing module  **from** pymongo **import** MongoClient    # creation of MongoClient  client**=**MongoClient()  # Connect with the portnumber and host  client **=** MongoClient(“mongodb:**//**localhost:27017**/**”)   # Access database  mydatabase **=** client[‘name\_of\_the\_database’]  # Access collection of the database  mycollection**=**mydatabase[‘myTable’]   # dictionary to be added in the database  rec**=**{  title: 'MongoDB and Python',  description: 'MongoDB is no SQL database',  tags: ['mongodb', 'database', 'NoSQL'],  viewers: 104  }  # inserting the data in the database  rec **=** mydatabase.myTable.insert(record) |

1. **Querying in MongoDB :**There are certain query functions which are used to filter the data in the database. The two most commonly used functions are:
   1. find()  
      find() is used to get more than one single document as a result of query.

|  |
| --- |
| **for** i **in** mydatabase.myTable.find({title: 'MongoDB and Python'})      print(i) |

* 1. This will output all the documents in the myTable of mydatabase whose title is ‘MongoDB and Python’.
  2. **count()**  
     count() is used to get the numbers of documents with the name as passed in the parameters.

|  |
| --- |
| print(mydatabase.myTable.count({title: 'MongoDB and Python'})) |

* 1. This will output the numbers of documents in the myTable of mydatabase whose title is ‘MongoDB and Python’.

1. These two query functions can be summed to give a give the most filtered result as shown below.

|  |
| --- |
| print(mydatabase.myTable.find({title: 'MongoDB and Python'}).count()) |

* 1. **To print all the documents/entries inside ‘myTable’ of database ‘mydatabase’ :** Use the following code:

|  |
| --- |
| **from** pymongo **import** MongoClient    **try**:      conn **=** MongoClient()      print("Connected successfully!!!")  **except**:      print("Could not connect to MongoDB")    # database name: mydatabase  db **=** conn.mydatabase    # Created or Switched to collection names: myTable  collection **=** db.myTable    # To find() all the entries inside collection name 'myTable'  cursor **=** collection.find()  **for** record **in** cursor:      print(record) |