

[Practice](#)[GATE CS](#)[Placements](#)[Videos](#)[Contribute](#)[Login/Register](#)**Quick Links for Python**[Recent Articles](#)[MCQ / Quizzes](#)[Practice Problems](#)**Basics**[Introduction](#)[New Generation Language](#)[Keywords, Set 1 Set 2](#)[Explore More...](#)**Variables**[Variables, Expressions & Functions](#)[Global and Local Variables](#)[Type Conversion](#)[Explore More...](#)**Operators**[Increment and Decrement Operator](#)[Teranry Operator & Divison Operator](#)[Logical and Bitwise Not Operators on Boolean](#)[Any & ALL](#)

Operator Functions Set 1 & Set 2
Data Types
Introduction
Arrays Set 1, Set 2
String Methods Set 1, Set 2, Set 3
String Template Class & String Formatting using %
List Methods Set 1, Set 2, Set 3
Tuples & Sets
Dictionary Methods Set 1, Set 2
ChainMap
Explore More...
Control Flow
Loops and Control Statements
Counters & Accessing Counters
Iterators & Iterator Functions Set 1, Set 2
Generators
Explore More...
Functions
Function Decorators
Returning Multiple Values
Yield instead of Return
Python Closures & Coroutine
Explore More...
Modules
Introduction

Numeric Functions & Logarithmic and Power functions
Calender Functions Set 1, Set 2
Complex Numbers Introduction & Important functions
Explore More...
Object Oriented Concepts
Class, Object and Members
Data Hiding and Object Printing
Inheritance, Subclass and super
Class method vs static method & Class or Static Variables
Explore More...
Exception Handling
Exception Handling
User-Defined Exceptions
Built-in Exceptions
Libraries and Functions
Timeit
Numpy Set 1, Set 2
Get and Post
import module & reload module
Collection Modules Deque, Namedtuple & Heap
Explore More...
Machine Learning with Python
Classifying data using Support Vector Machines(SVMs) in Python

K means Clustering
How to get synonyms/antonyms from NLTK WordNet in Python?
Explore More...
Misc
Sql using Python & MongoDB and Python
Json formatting & Python Virtual environment
Metaprogramming with Metaclasses in Python
Python Input Methods for Competitive Programming
Explore More...
Applications and Projects
Creating a proxy webserver Set 1, Set 2
Send Message to FB friend
Twitter Sentiment Analysis & Whatsapp using Python
Desktop Notifier & Junk File Organizer
Explore More...

MongoDB and Python

Prerequisite : [MongoDB : An introduction](#)

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"? There are several reasons for this. These are listed below.

1. It supports hierarchical data structure (Please refer [docs](#) for details)
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.

While comparing MongoDB with RDBMS, we got to the following conclusion.

RDBMS	MongoDB
Table	Collection
Row	JSON Document
Index	Index
Join	Embedding & Linking
Partition	Shard

PyMongo Installation

Python has a native library for MongoDB. The name of the available library is “PyMongo”. To import this, execute the following command:

```
import pymongo
```

Create a connection

The very first after importing the module is to create a MongoClient.

```
from pymongo import MongoClient
client = MongoClient()
```

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)
```

It can also be done using the following command:

```
client = MongoClient("mongodb://localhost:27017/")
```

Accessing the database

Using MongoDB, we can use dictionary style access to access databases on the MongoClient instances.

```
mydatabase = client['name_of_the_database']
```

If there is no previously created database with this name, MongoDB will implicitly create one for the user. Note that the name of the database will 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.

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 “my-database”.

```
mycollection = mydatabase['myTable']
```

MongoDB store the database in the form of dictionaries as shown:

```
{
  title: 'MongoDB and Python',
  description: 'MongoDB is no SQL database',
  tags: ['mongodb', 'database', 'NoSQL'],
  viewers: 104
}
```

‘_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.

7df78ad8902c

Current timestamp	Machine id	Process id	Incremental value
----------------------	---------------	---------------	----------------------

Insert the data using insert method()

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

```
rec = mydatabase.myTable.insert(record)
```

Recommended: Please try your approach on {IDE} first, before moving on to the solution.

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)
```

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()
2. count()

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)
```

This will output all the documents in the myTable of mydatabase whose title is 'MongoDB and Python'.

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'}))
```

This will output the numbers of documents in the myTable of mydatabase whose title is 'MongoDB and Python'.

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())
```

This article is contributed by **Rishabh Bansal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner Company Wise Coding Practice

GBlog Python

Recommended Posts:

MongoDB : An introduction

SQL using Python

Defining Clean Up Actions in Python

Extraction of Tweets using Tweepy

Python program to check if a string is palindrome or not

(Login to Rate and Mark)

3.6

Average Difficulty : **3.6/5.0**
Based on **3** vote(s)



Add to TODO List

Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!

@geeksforgeeks, Some rights reserved

Contact Us!

About Us!

Careers!

Privacy Policy

