Here are some tough **PyMongo (MongoDB with Python)** interview questions, categorized based on different aspects of PyMongo:

**Basic PyMongo Questions**

1. **How do you establish a connection to a MongoDB database using PyMongo?**
2. **How does PyMongo handle authentication, and how can you authenticate against a specific database?**
3. **What are the different ways to insert documents into a MongoDB collection using PyMongo?**
4. **How do you retrieve all documents from a collection using PyMongo?**
5. **How do you update a document in PyMongo, and what is the difference between update\_one() and update\_many()?**
6. **How do you delete documents using PyMongo? Explain delete\_one() vs. delete\_many().**

**Intermediate PyMongo Questions**

1. **What is the purpose of find\_one\_and\_update() and how does it differ from update\_one()?**
2. **How can you use indexing in MongoDB via PyMongo to improve query performance?**
3. **What are the different types of cursors in PyMongo, and how do they work?**
4. **How do you handle transactions in PyMongo?**
5. **What is the difference between insert\_one() and insert\_many()? What happens if one document fails in insert\_many()?**
6. **What are MongoDB aggregation pipelines, and how do you implement them using PyMongo?**
7. **Explain the concept of GridFS in MongoDB and how to use it with PyMongo.**
8. **What is bulk\_write() in PyMongo, and when should you use it?**
9. **How do you ensure that a PyMongo connection is closed properly?**
10. **How does PyMongo handle ObjectId, and how can you convert it to a string?**

**Advanced PyMongo Questions**

1. **What are some potential performance issues when using PyMongo, and how can you optimize queries?**
2. **Explain how to use explain() in PyMongo to analyze query performance.**
3. **What is the difference between w=1, w=0, and w=majority in MongoDB write concern?**
4. **How do you implement sharding in MongoDB, and how does PyMongo interact with sharded clusters?**
5. **How do you use change streams in PyMongo to listen for real-time updates to a collection?**
6. **How does PyMongo handle connection pooling, and how can you configure it for better performance?**
7. **Explain the difference between a capped collection and a regular collection in MongoDB. How do you create a capped collection using PyMongo?**
8. **How do you perform a full-text search using PyMongo?**
9. **What are collation settings in PyMongo, and how can they be useful?**
10. **How can you prevent SQL injection-like attacks in PyMongo when accepting user input for queries?**
11. **How do you implement an upsert operation in PyMongo?**
12. **How can you use PyMongo with an ORM like MongoEngine? What are the pros and cons?**
13. **How do you handle time-series data in MongoDB using PyMongo?**
14. **What is the bson module in PyMongo, and why is it important?**

Would you like detailed answers or code snippets for any of these questions? 🚀

4o

**You said:**

Would you like detailed answers or code snippets for any of these questions? 🚀 - yes please

**ChatGPT said:**

ChatGPT

Great! I'll provide **detailed answers with code snippets** for some of the most important **PyMongo** interview questions. 🚀

**1. How do you establish a connection to a MongoDB database using PyMongo?**

To connect to MongoDB using PyMongo, use the MongoClient class.

**Code Example:**

python

CopyEdit

from pymongo import MongoClient

# Connect to the MongoDB server (default localhost:27017)

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

# Select a database

db = client["mydatabase"]

# Check available databases

print(client.list\_database\_names())

✅ **Key Points:**

* MongoClient() establishes a connection.
* By default, it connects to localhost:27017.
* Use client["db\_name"] to access a specific database.

**2. How does PyMongo handle authentication?**

MongoDB can require authentication, and PyMongo supports connecting with credentials.

**Code Example (Username & Password Authentication)**

python

CopyEdit

from pymongo import MongoClient

# Connecting with authentication

client = MongoClient("mongodb://username:password@localhost:27017/")

# Access the database

db = client["secured\_db"]

✅ **Key Points:**

* MongoDB authentication depends on the authentication database (admin by default).
* You can specify the database in the connection URI:

python

CopyEdit

client = MongoClient("mongodb://user:pass@localhost:27017/mydatabase")

* If authentication is required, ensure **role-based access control (RBAC)** is properly configured.

**3. What are the different ways to insert documents in PyMongo?**

There are three main ways:

1. insert\_one() – Insert a single document.
2. insert\_many() – Insert multiple documents.
3. bulk\_write() – Efficient batch operations.

**Code Example:**

python

CopyEdit

# Insert one document

doc = {"name": "Alice", "age": 25, "city": "New York"}

db.users.insert\_one(doc)

# Insert multiple documents

docs = [

{"name": "Bob", "age": 30, "city": "London"},

{"name": "Charlie", "age": 28, "city": "Berlin"},

]

db.users.insert\_many(docs)

✅ **Key Points:**

* \_id is automatically assigned unless specified.
* insert\_many() executes all insertions as a batch.

**4. How do you retrieve documents from a MongoDB collection using PyMongo?**

Use find\_one() for a single document and find() for multiple.

**Code Example:**

python

CopyEdit

# Retrieve one document

user = db.users.find\_one({"name": "Alice"})

print(user)

# Retrieve all documents

for user in db.users.find():

print(user)

✅ **Key Points:**

* find\_one() returns a single document.
* find() returns a cursor that needs iteration.

**5. What is the difference between update\_one() and update\_many()?**

* update\_one() updates the **first matching document**.
* update\_many() updates **all matching documents**.

**Code Example:**

python

CopyEdit

# Update a single document

db.users.update\_one({"name": "Alice"}, {"$set": {"age": 26}})

# Update multiple documents

db.users.update\_many({"city": "New York"}, {"$set": {"city": "San Francisco"}})

✅ **Key Points:**

* Use $set to update only specified fields.
* Always **check matched & modified count**:

python

CopyEdit

result = db.users.update\_one({"name": "Alice"}, {"$set": {"age": 26}})

print(result.modified\_count)

**6. How do you handle transactions in PyMongo?**

Transactions ensure **atomicity** for multiple operations in **replica sets**.

**Code Example (Transaction with with block)**

python

CopyEdit

with client.start\_session() as session:

with session.start\_transaction():

db.users.insert\_one({"name": "Eve", "age": 24}, session=session)

db.orders.insert\_one({"user": "Eve", "product": "Laptop"}, session=session)

✅ **Key Points:**

* Transactions **only work in replica sets or sharded clusters**.
* If an error occurs, the entire transaction rolls back.

**7. How do you use aggregation pipelines in PyMongo?**

Aggregation is used for complex queries like grouping, filtering, and computing statistics.

**Code Example (Aggregation Pipeline)**

python

CopyEdit

pipeline = [

{"$match": {"city": "New York"}},

{"$group": {"\_id": "$city", "average\_age": {"$avg": "$age"}}}

]

result = db.users.aggregate(pipeline)

for doc in result:

print(doc)

✅ **Key Points:**

* $match filters data.
* $group performs aggregation operations.

**8. How do you use indexes in PyMongo?**

Indexes improve query performance.

**Creating an Index**

python

CopyEdit

db.users.create\_index([("name", 1)]) # 1 for ascending, -1 for descending

**Checking Indexes**

python

CopyEdit

print(db.users.index\_information())

✅ **Key Points:**

* Use indexes to **speed up queries**.
* Create indexes on **fields used in queries**.

**9. How do you use Change Streams in PyMongo for real-time updates?**

Change Streams allow listening to **real-time changes** in a MongoDB collection.

**Code Example (Listening for Changes)**

python

CopyEdit

with db.users.watch() as stream:

for change in stream:

print("Change detected:", change)

✅ **Key Points:**

* Requires **replica set** or **sharded cluster**.
* Can be used for **real-time analytics**.

**10. How do you perform an upsert operation in PyMongo?**

An **upsert** updates a document if it exists or inserts it if it doesn’t.

**Code Example:**

python

CopyEdit

db.users.update\_one(

{"name": "Frank"},

{"$set": {"age": 40, "city": "Paris"}},

upsert=True

)

✅ **Key Points:**

* If "Frank" exists, it updates.
* If "Frank" does **not** exist, it **inserts** a new document.

**11. What is bulk\_write() in PyMongo?**

bulk\_write() performs multiple operations efficiently.

**Code Example:**

python

CopyEdit

from pymongo import InsertOne, UpdateOne, DeleteOne

operations = [

InsertOne({"name": "Grace", "age": 29}),

UpdateOne({"name": "Alice"}, {"$set": {"age": 27}}),

DeleteOne({"name": "Charlie"})

]

db.users.bulk\_write(operations)

✅ **Key Points:**

* Supports **batch operations** for speed.
* Useful for **large datasets**.

**12. How do you ensure proper connection handling in PyMongo?**

Use **connection pooling** and **proper cleanup**.

**Code Example:**

python

CopyEdit

client = MongoClient("mongodb://localhost:27017/", maxPoolSize=50)

# Close connection

client.close()

✅ **Key Points:**

* Use **maxPoolSize** for optimal performance.
* Always **close connections** when done.