

Open Platform Software

開放平台軟體

Assignment #7: Chat Room (3)

Semester: 106-2

Offered: 2018

Dr. Bo-Hao Chen

Department of Computer Science and Engineering

Yuan Ze University, Taoyuan, Taiwan

bhchen at saturn.yzu.edu.tw

What is MySQL?

- MySQL is a popular open-source relational database management system (RDBMS) that is developed, distributed and supported by Oracle Corporation.
- Like other relational systems, MySQL stores data in tables and uses structured query language (SQL) for database access.

What is MongoDB?

- MongoDB is an open-source, non-relational database developed by MongoDB, Inc. MongoDB stores data as documents in a binary representation called BSON (Binary JSON).
- Related information is stored together for fast query access through the MongoDB query language.

Terminology and Concepts

- Many concepts in MySQL have close analogs in MongoDB. The table below outlines the common concepts across MySQL and MongoDB

MySQL	MongoDB
ACID Transactions	ACID Transactions*
Table	Collection
Row	Document
Column	Field
Secondary Index	Secondary Index
JOINS	Embedded documents \$lookup & \$graphLookup
GROUP_BY	Aggregation Pipeline

Install MongoDB Enterprise on Windows

- Download MongoDB Enterprise Server
 - <https://www.mongodb.com/download-center#enterprise>

Enterprise Server | Ops Manager | Compass | Connector for BI

Release: 3.6.5

Platforms: Windows x64

Download

[Current Releases & Packages](#)
[Development Releases](#)
[Archived Releases](#)
[Release Notes](#)
[Documentation](#)

Download Now

Business email
[email address]

First Name
[first name]

Last Name
[last name]

Business phone
[phone number]

Company
university

DBA
[dropdown]

China
[dropdown]

☒ Check here to indicate that you have read and agree to the terms of the Customer Agreement.

Submit

Download MongoDB Enterprise Server

- <http://gofile.me/3trS5/b8NioPPQq>
- <http://gofile.me/3trS5/uqgeYZwn4>

Install MongoDB Enterprise

Please follow the installation guide to setup MongoDB Enterprise.

Installation Guide

Alternatively, you can download and install our individual Enterprise packages below:

archive

msi

Close

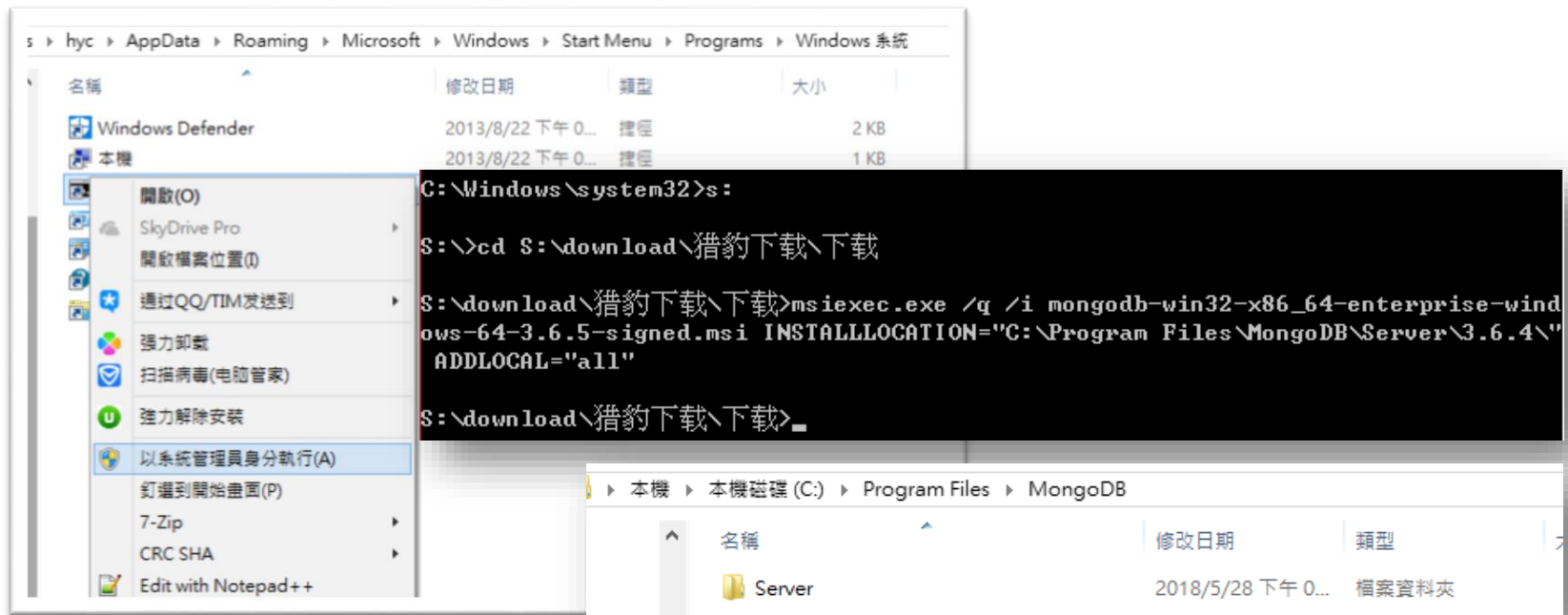


mongodb-
win32-x86_64-
enterprise-
windows-64-3.6.

Install MongoDB Enterprise

- Change to the directory containing the .msi installation binary of your choice and invoke:

```
msiexec.exe /q /i mongodb-win32-x86_64-enterprise-windows-64-3.6.5-signed.msi INSTALLLOCATION="C:\Program Files\MongoDB\Server\3.6.4\" ADDLOCAL="all"
```



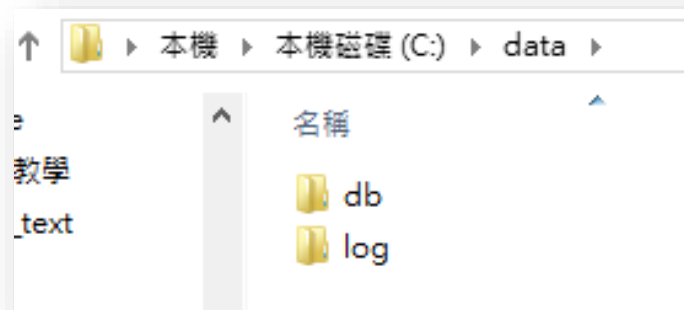
Set up the MongoDB environment

- MongoDB requires a **data directory to store all data**.
 - MongoDB's default data directory path is the absolute path `\data\db` on the drive from which you start MongoDB.
 - Create this folder by running the following command in a Command Prompt:

```
md \data\db
```

- MongoDB requires a **log directory to record logs**.

```
md \data\log
```



Start MongoDB

- Add system path

```
C:\Users\hyc>mongod
```

- or

```
C:\Users\hyc>"C:\Program Files\MongoDB\Server\3.6.4\bin\mongod.exe"
```

```
C:\Users\hyc>mongod
2018-05-28T16:11:13.255+0800 I CONTROL [initandlisten] MongoDB starting : pid=9152 port=27017 dbpath=C:\data\db\ 64-bit host=WIN-L5EQM1SC8NB
2018-05-28T16:11:13.256+0800 I CONTROL [initandlisten] targetMinOS: Windows 7/...
```

Verify that MongoDB has started successfully

- Verify that MongoDB has started successfully by checking the process output for the following line:

- 1.

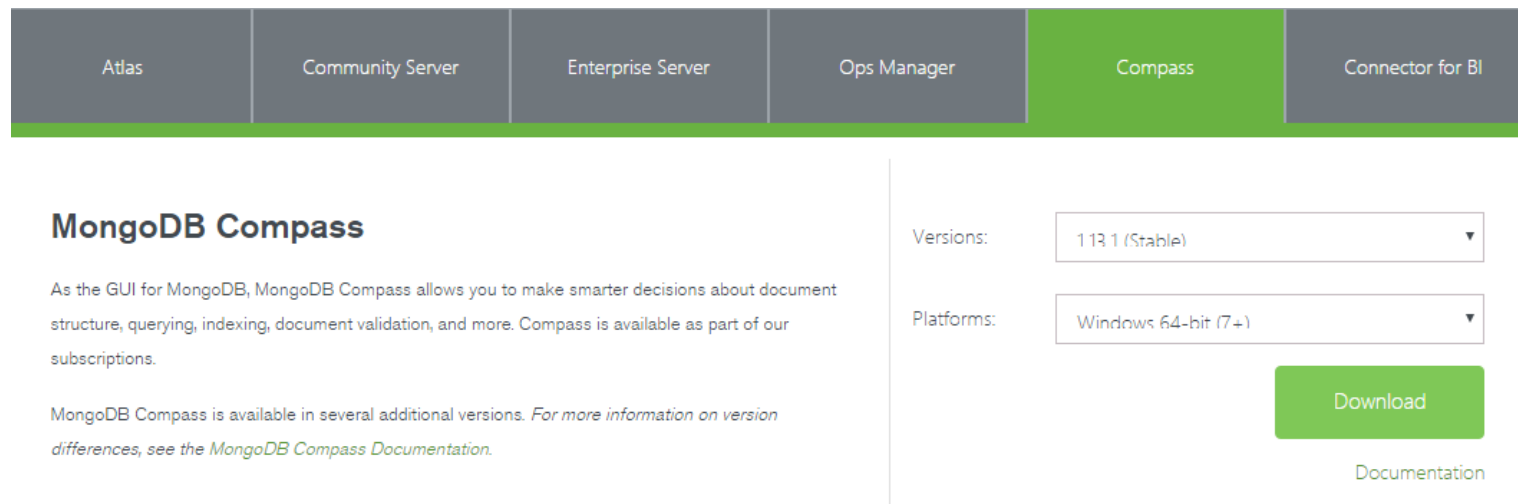
```
diagnostic data capture with directory 'C:/data/db/diagnostic.data'  
2018-05-28T16:11:13.938+0800 I NETWORK [initandlisten] waiting for connections  
on port 27017
```

- 2.



Visualize and explore

- The GUI for MongoDB, Visually explore your data.
- View and optimize your query performance.
- <https://www.mongodb.com/download-center?jmp=hero#compass>



The screenshot shows the MongoDB download center interface. At the top, there is a navigation bar with six tabs: Atlas, Community Server, Enterprise Server, Ops Manager, Compass (which is highlighted in green), and Connector for BI. Below the navigation bar, the main content area is divided into two columns. The left column is titled 'MongoDB Compass' and contains a paragraph describing it as the GUI for MongoDB, allowing users to make smarter decisions about document structure, querying, indexing, document validation, and more. It also mentions that Compass is available as part of several subscriptions. The right column contains two dropdown menus: 'Versions' with '11.1 (Stable)' selected, and 'Platforms' with 'Windows 64-bit (7+)' selected. Below these dropdowns is a green 'Download' button and a link to 'Documentation'.

MongoDB Compass

As the GUI for MongoDB, MongoDB Compass allows you to make smarter decisions about document structure, querying, indexing, document validation, and more. Compass is available as part of our subscriptions.

MongoDB Compass is available in several additional versions. *For more information on version differences, see the [MongoDB Compass Documentation](#).*

Versions: 11.1 (Stable) ▼

Platforms: Windows 64-bit (7+) ▼

[Download](#)

[Documentation](#)

Visualize and explore

MongoDB Compass - Connect

Connect View Help

CREATE ATLAS CLUSTER

NEW CONNECTION

FAVORITES

RECENTS

MAY 28, 2018 4:14 PM
localhost:27017

4 MINUTES AGO
localhost:27017

Connect to Host

Hostname hostname

Port Port

SRV Record ☐

Authentication

Replica Set Name

Read Preference

SSL

SSH Tunnel

Favorite Name ⓘ

CONNECT

Visualize and explore

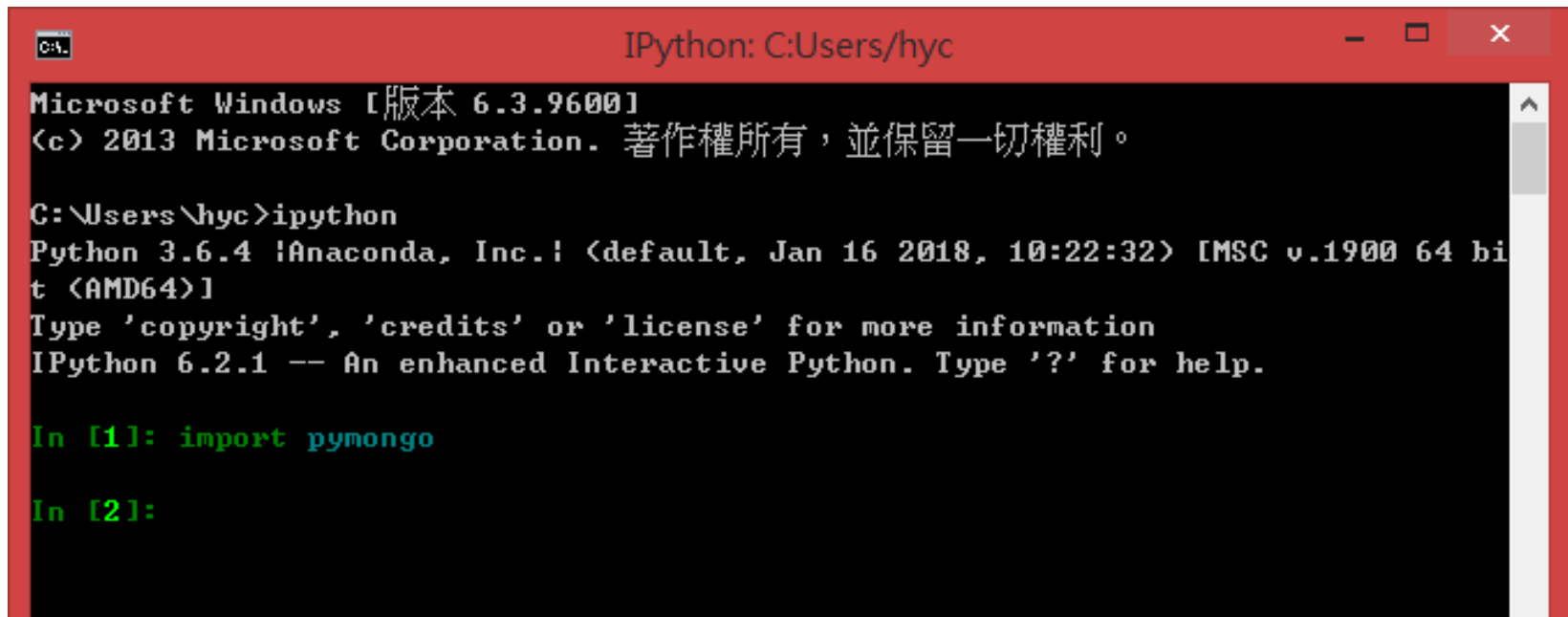
The screenshot shows the MongoDB Compass application window titled "MongoDB Compass - localhost:27017". The interface includes a sidebar on the left with a "My Cluster" section showing "4 DBS" and "2 COLLECTIONS". A search filter is present, and a list of databases is displayed: ChatRoom, admin, config, and local. A red arrow points from the word "Database" to this list. The main panel shows the "Databases" tab with a "CREATE DATABASE" button highlighted by a red arrow and the text "New Database". Below this is a table listing the databases.

Database Name	Storage Size	Collections	Indexes
ChatRoom	36.0KB	1	1
admin	16.0KB	0	1
config	24.0KB	0	2
local	36.0KB	1	1

Install pymongo

- Pymongo is a library used to manipulate MongoDB in python

```
C:\Users\hyc> pip install pymongo
```



```
IPython: C:\Users\hyc
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\hyc>ipython
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bi
t (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import pymongo

In [2]:
```

Pymongo: establish connection

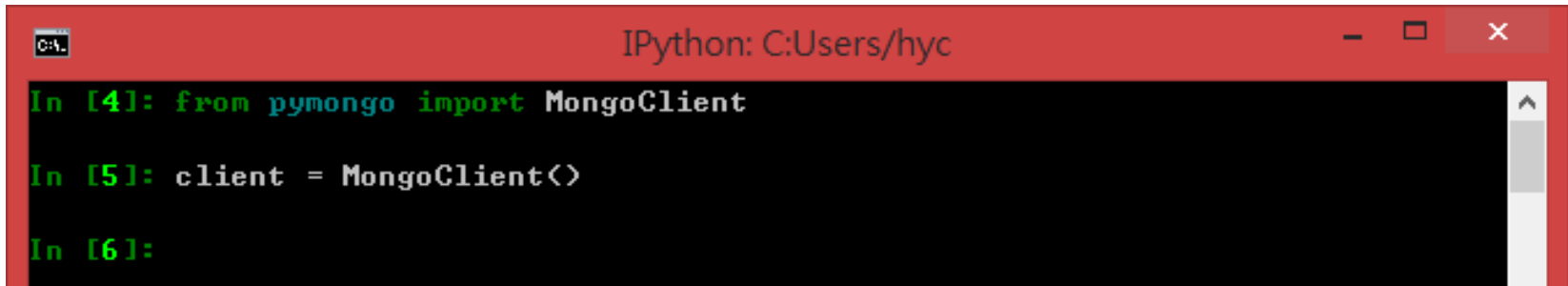
```
from pymongo import MongoClient  
client = MongoClient()
```

- or

```
client = MongoClient("localhost", 27017)
```

- or

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



The screenshot shows an IPython terminal window with a red title bar. The title bar contains the text "IPython: C:\Users\hyc" and standard window control buttons (minimize, maximize, close). The terminal has a black background with green text. It shows three input prompts: "In [4]:", "In [5]:", and "In [6]:". The first two prompts have corresponding code lines entered: "from pymongo import MongoClient" and "client = MongoClient()". The third prompt is empty.

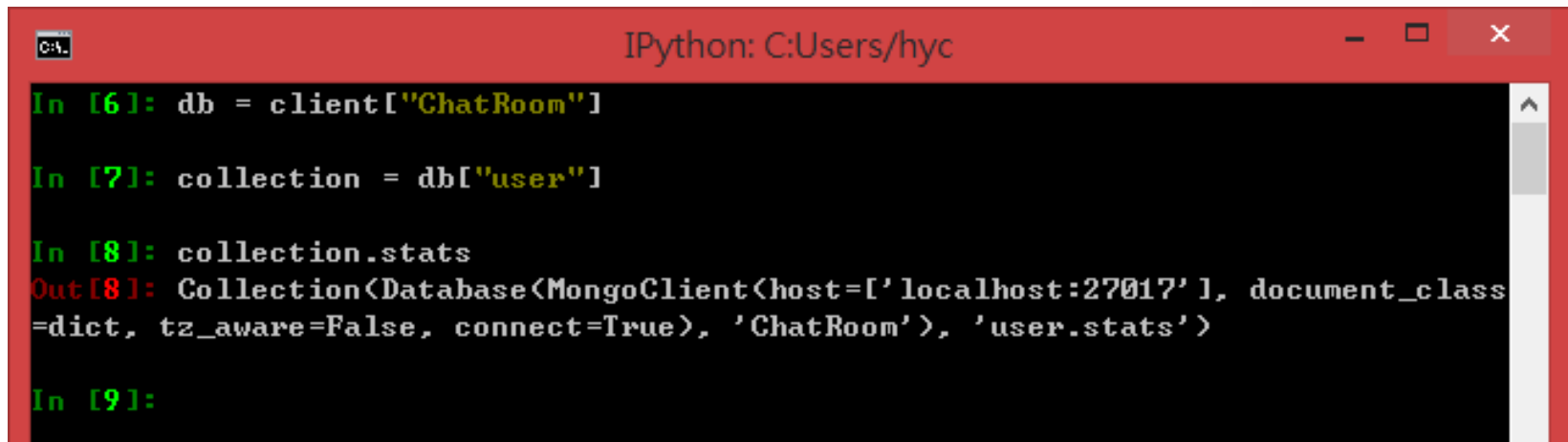
```
IPython: C:\Users\hyc  
In [4]: from pymongo import MongoClient  
In [5]: client = MongoClient()  
In [6]:
```

Pymongo: database and collection

```
# connection
db = client["ChatRoom"]
collection = db["user"]

# test if connection success
print(collection.stats)
```

SQL	NOSQL
server	server
database	database
table	collection
row	dictionary



```
IPython: C:Users/hyc

In [6]: db = client["ChatRoom"]

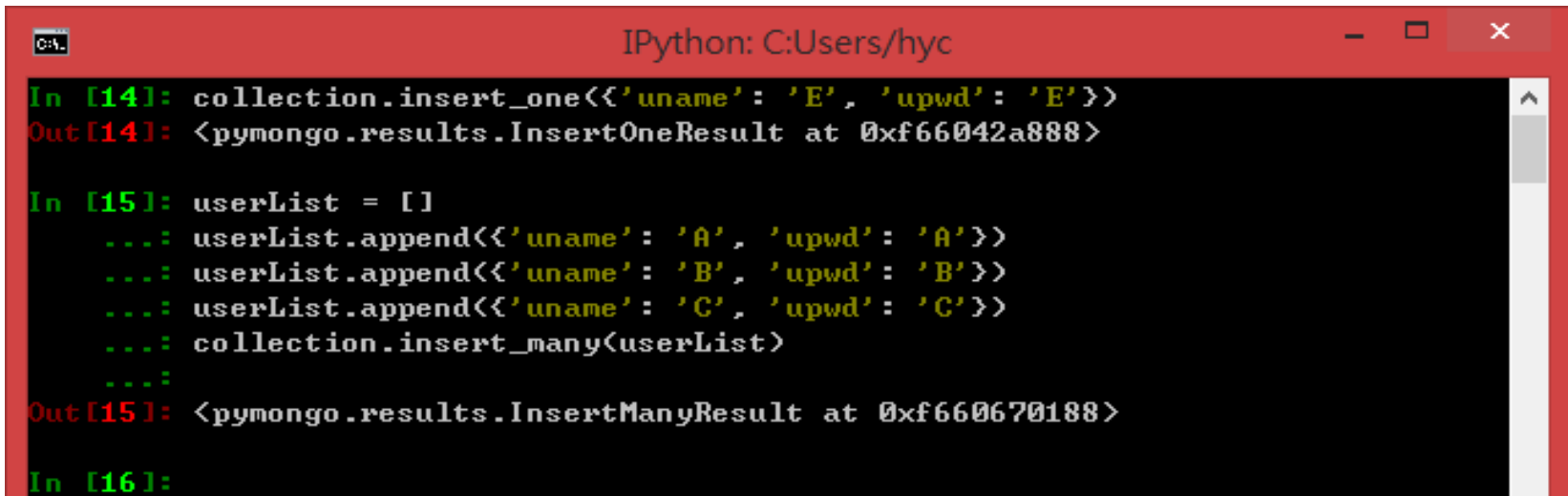
In [7]: collection = db["user"]

In [8]: collection.stats
Out[8]: Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'ChatRoom'), 'user.stats')

In [9]:
```


Pymongo: insert

```
# insert one
collection.insert_one({'uname': 'E', 'upwd': 'E'})
# insert many
userList = []
userList.append({'uname': 'A', 'upwd': 'A'})
userList.append({'uname': 'B', 'upwd': 'B'})
userList.append({'uname': 'C', 'upwd': 'C'})
collection.insert_many(userList)
```

A screenshot of an IPython terminal window titled "IPython: C:Users/hyc". The window has a red title bar and a black background with green and red text. It shows the execution of pymongo insert operations. The first command (In [14]) is collection.insert_one({'uname': 'E', 'upwd': 'E'}), which returns a pymongo.results.InsertOneResult object. The second command (In [15]) is a multi-line block where a list named userList is created, three dictionary objects are appended to it (each with 'uname' and 'upwd' keys), and then collection.insert_many(userList) is called. This returns a pymongo.results.InsertManyResult object. The third command (In [16]) is partially visible at the bottom.

```
IPython: C:Users/hyc

In [14]: collection.insert_one({'uname': 'E', 'upwd': 'E'})
Out[14]: <pymongo.results.InsertOneResult at 0xf66042a888>

In [15]: userList = []
...:     userList.append({'uname': 'A', 'upwd': 'A'})
...:     userList.append({'uname': 'B', 'upwd': 'B'})
...:     userList.append({'uname': 'C', 'upwd': 'C'})
...:     collection.insert_many(userList)
...:
Out[15]: <pymongo.results.InsertManyResult at 0xf660670188>

In [16]:
```

Visualize

The screenshot shows the MongoDB Compass interface for a local database. The left sidebar shows the 'My Cluster' section with 'localhost:27017' and 'STANDALONE'. The 'ChatRoom' database is selected, and the 'user' collection is highlighted. The main panel displays the 'ChatRoom.user' collection with 5 documents. The documents are listed in a table view, showing fields like '_id', 'uname', and 'upwd'. The documents are:

- `{ "_id": ObjectId("5b0bc278e55ca91098a090b6"), "uname": "A", "upwd": "A" }`
- `{ "_id": ObjectId("5b0bc278e55ca91098a090b7"), "uname": "B", "upwd": "B" }`
- `{ "_id": ObjectId("5b0bc278e55ca91098a090b8"), "uname": "C", "upwd": "C" }`
- `{ "_id": ObjectId("5b0bc278e55ca91098a090b9"), "uname": "D", "upwd": "D" }`
- `{ "_id": ObjectId("5b0bc278e55ca91098a090ba"), "uname": "E", "upwd": "E" }`

The word 'data' is written in red next to the list of documents. The interface also shows a filter bar with the filter `{ field: 'value' }` and a 'FIND' button. The status bar at the bottom indicates 'Displaying documents 1 - 5 of 5'.

Pymongo: query

```
# find_one() returns a single document matching a query (or None if  
# there are no matches).
```

```
collection.find_one({'uname':'A'})  
collection.find_one({'uname':'A','upwd':'A'})
```

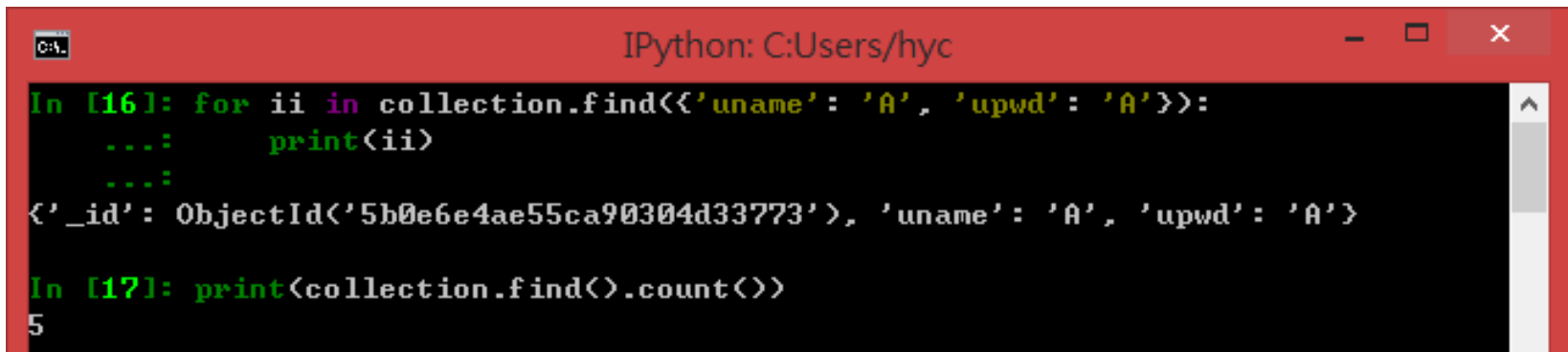
```
# find by id
```

```
from bson.objectid import ObjectId  
collection.find_one({'_id':ObjectId('5b0e6e4ae55ca90304d33773')})
```

```
In [27]: collection.find_one({'uname':'A'})  
Out[27]: {'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'}  
  
In [28]: collection.find_one({'uname':'A','upwd':'A'})  
Out[28]: {'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'}  
  
In [29]: from bson.objectid import ObjectId  
  
In [30]: collection.find_one({'_id':ObjectId('5b0e6e4ae55ca90304d33773')})  
Out[30]: {'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'}  
>
```

Pymongo: query

```
for ii in collection.find({'uname': 'A', 'upwd': 'A'}):  
    print(ii)  
  
print(collection.find().count())
```



The screenshot shows an IPython terminal window with a red title bar. The window title is "IPython: C:Users/hyc". The terminal output shows the execution of two code blocks. The first block, labeled "In [16]:", is a for loop that iterates over the results of a MongoDB query and prints each document. The output shows a single document with an ObjectId and the fields 'uname' and 'upwd'. The second block, labeled "In [17]:", is a print statement that outputs the count of documents found by the query.

```
IPython: C:Users/hyc  
In [16]: for ii in collection.find(<<'uname': 'A', 'upwd': 'A'>>):  
...:     print(ii)  
...:  
<'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'>  
In [17]: print(collection.find(<>).count(<>))  
5
```

Pymongo: query

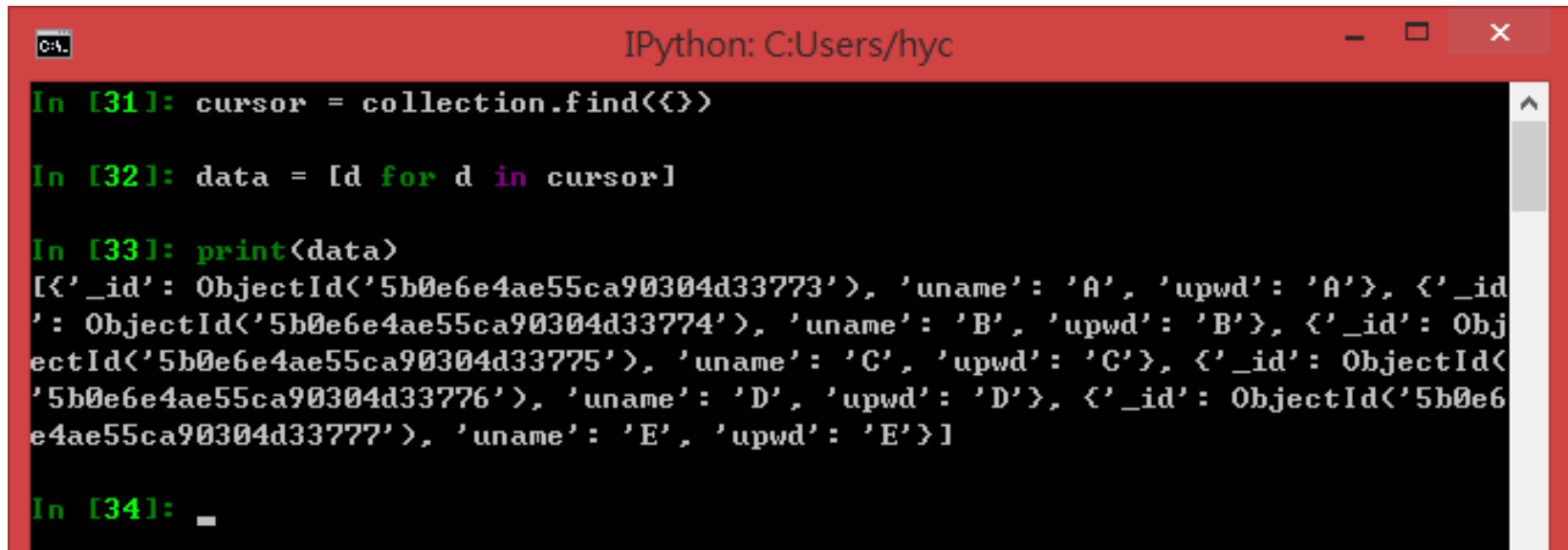
```
for ii in collection.find().sort('uname'):  
    print(ii)
```

```
for ii in collection.find().sort('uname',pymongo.DESCENDING):  
    print(ii)
```

```
In [18]: for ii in collection.find().sort('uname'):  
...:     print(ii)  
...:  
<'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'>  
<'_id': ObjectId('5b0e6e4ae55ca90304d33774'), 'uname': 'B', 'upwd': 'B'>  
<'_id': ObjectId('5b0e6e4ae55ca90304d33775'), 'uname': 'C', 'upwd': 'C'>  
<'_id': ObjectId('5b0e6e4ae55ca90304d33776'), 'uname': 'D', 'upwd': 'D'>  
<'_id': ObjectId('5b0e6e4ae55ca90304d33777'), 'uname': 'E', 'upwd': 'E'>  
  
In [19]: for ii in collection.find().sort('uname',pymongo.DESCENDING):  
...:     print(ii)  
...:  
<'_id': ObjectId('5b0e6e4ae55ca90304d33777'), 'uname': 'E', 'upwd': 'E'>  
<'_id': ObjectId('5b0e6e4ae55ca90304d33776'), 'uname': 'D', 'upwd': 'D'>  
<'_id': ObjectId('5b0e6e4ae55ca90304d33775'), 'uname': 'C', 'upwd': 'C'>  
<'_id': ObjectId('5b0e6e4ae55ca90304d33774'), 'uname': 'B', 'upwd': 'B'>  
<'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'>
```

Pymongo: query

```
# query all data
cursor = collection.find({})
data = [d for d in cursor]
print(data)
```

A screenshot of an IPython terminal window titled "IPython: C:Users/hyc". The terminal shows a series of commands and their output. The first command is `cursor = collection.find({})`. The second command is `data = [d for d in cursor]`. The third command is `print(data)`, which outputs a list of five dictionaries. Each dictionary represents a document in the MongoDB collection, with fields `_id`, `uname`, and `upwd`. The `_id` field is an `ObjectId` string. The `uname` and `upwd` fields are strings. The output is as follows:

```
In [31]: cursor = collection.find({})
In [32]: data = [d for d in cursor]
In [33]: print(data)
[{'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'}, {'_id': ObjectId('5b0e6e4ae55ca90304d33774'), 'uname': 'B', 'upwd': 'B'}, {'_id': ObjectId('5b0e6e4ae55ca90304d33775'), 'uname': 'C', 'upwd': 'C'}, {'_id': ObjectId('5b0e6e4ae55ca90304d33776'), 'uname': 'D', 'upwd': 'D'}, {'_id': ObjectId('5b0e6e4ae55ca90304d33777'), 'uname': 'E', 'upwd': 'E'}]
In [34]: _
```

Pymongo: update

```
temp = collection.find_one({'uname' : 'A'})
temp['upwd'] = 'K'
```

```
# update method 1
collection.save(temp)
# update method 2
temp2 = temp.copy()
collection.update(temp, temp2)
```

```
In [34]: temp = collection.find_one({'uname': 'A'})
In [35]: temp['upwd'] = 'K'
In [36]: collection.save(temp)
S:\software\anaconda\Scripts\ipython:1: DeprecationWarning: save is deprecated.
Use insert_one or replace_one instead
Out[36]: ObjectId('5b0e6e4ae55ca90304d33773')
In [37]: temp2 = temp.copy()
...: collection.update(temp, temp2)
...:
S:\software\anaconda\Scripts\ipython:2: DeprecationWarning: update is deprecated.
Use replace_one, update_one or update_many instead.
Out[37]: {'n': 1, 'nModified': 0, 'ok': 1.0, 'updatedExisting': True}
In [38]: print(collection.find_one({'uname': 'A'}))
{'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'K'}
```

Pymongo: delete

```
collection.remove({'uname':'E'})
collection.delete_one({'uname':'D'})
collection.delete_many({'uname':'C'})
print([d for d in collection.find({})])
```

```
# delete all data
collection.delete_many({})
```

```
In [46]: collection.remove({'uname':'E'})
...: collection.delete_one({'uname':'D'})
...: collection.delete_many({'uname':'C'})
...: print([d for d in collection.find({})])
...:
S:\software\anaconda\Scripts\ipython:1: DeprecationWarning: remove is deprecated
. Use delete_one or delete_many instead.
[{'_id': ObjectId('5b0e9f12e55ca90f58a8fbcd'), 'uname': 'A', 'upwd': 'K'}, {'_id':
'_id': ObjectId('5b0e9f12e55ca90f58a8fbcd'), 'uname': 'B', 'upwd': 'B'}]

In [47]: collection.delete_many({})
Out[47]: <pymongo.results.DeleteResult at 0xf660940bc8>

In [48]: print([d for d in collection.find({})])
[]
```


Assignment

服務器端介面(1分)

The screenshot shows a web browser window titled "Chat Application". At the top, there are input fields for "NickName" (containing "G") and "Password" (with a masked character). Below these is a yellow "Add" button. A list of users (A, B, C, D, E) is displayed, each with a checkbox to its left. At the bottom of the window is a yellow "Del" button.

服務器可以增加用戶，
並及時更新下方列表(2
分)

客戶端間正常通訊(3分)

服務器可以批量刪除用戶(2分)

客戶端端介面(1分)

客戶端及時顯示在線人數(2分)

The screenshot shows a web browser window titled "Chat Application". At the top, a green banner displays "目前聊天室有2人". Below this is a login section with "NickName" (containing "A") and "Password" (masked) fields, a yellow "Login" button, and a "change password" section with a "password display as dot" toggle (checked) and an "update password" button. The main chat area contains messages: "Welcome to chat room! A", "Now Lets Chat, A", "SYSTEM: B in the chat room", "hello B:You", "B:hi A [08:38:17]", "SYSTEM: C in the chat room", and "[SYSTEM: C leave the chat room]". At the bottom is a text input field with the placeholder "输入发送内容，空时无法发送" and a yellow "Send" button.

客戶端可修改密碼並可成功用
新密碼登錄(2分)

用戶離開通知，並更新在線人數(2分)

使用mongodb作為數據庫(3分)

上週完成的加2.5分