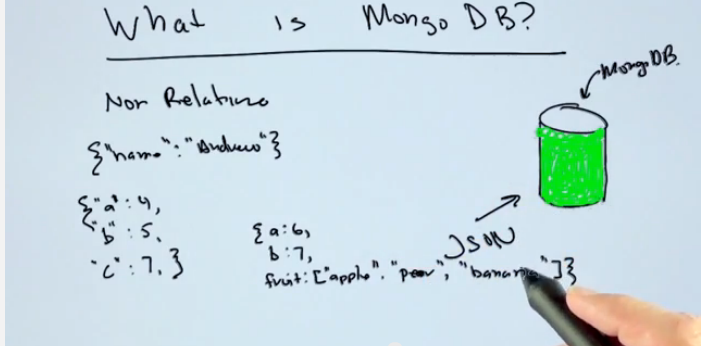
# Video - What is MongodB

## Mongodb

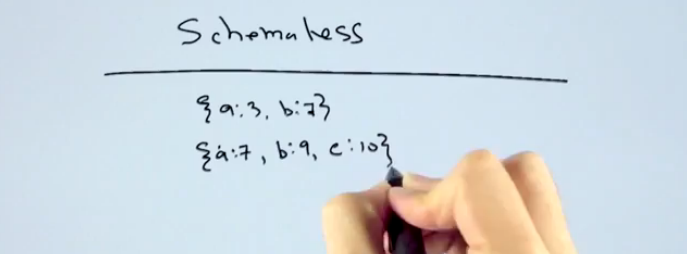
* is non-relational datastore for json documents.
* non relational means "it doesnt store data in tables", like relation db does.
* it stores json documents.
* it is easy to programm.

### Example of json document (with and without hierarchy)



### Schema-less

Same Document can have collection in different hierarchical order.



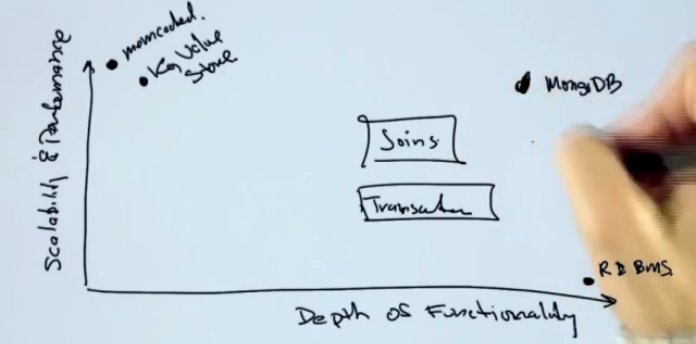
### Quiz: What is MongoDB?

Which of the following statements are true about MongoDB?

Top of Form

MongoDB is document oriented.MongoDB supports Joins.MongoDB is schemalessMongoDB supports SQL.

# Video - Mongo Relative to Relational



Memcache programs and Key-value store programs are scalable and give a good performance, but offers almost NO functionality.

RDBMS example: db2, oracle, mysql

## Whats missing in MongoDb

* doesnt support join. FYI Join reduces scaling.
* Doesn't support transaction across multiple documents. This can be achieved by doing multiple updates on document atomically.

## Quiz

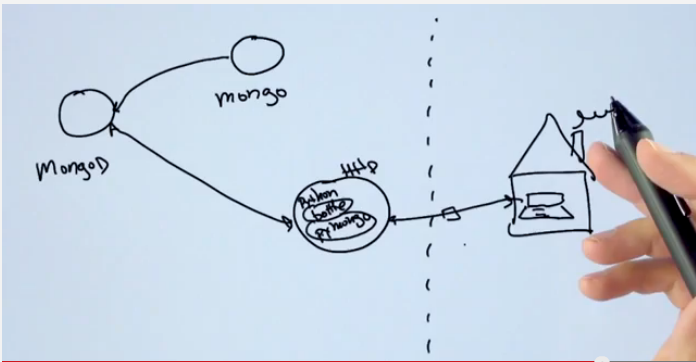
Which features did MongoDB omit in order to retain scalability?

Top of Form

JoinsIndexesSecondary IndexesTransactions across multiple collections

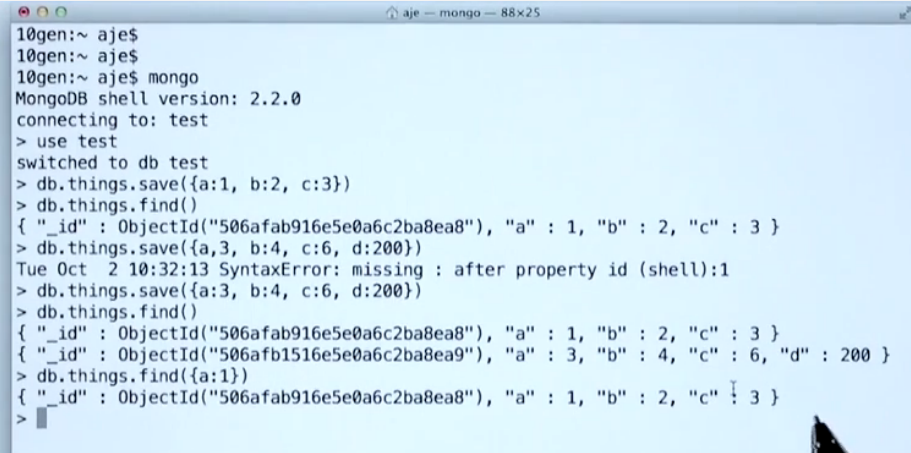
Bottom of Form

# Video - Overview of building an app with mongodB



* M**Mongod** is database server running on a box.
* **Mongoshell** interacts with mongod for CRUD operations.
* On http server, we are going to use **python**.
* We will be using **python bottle framework** and **py-mongo** which will help us connect to mongodB.

# Video - Quick Introduction to Mongo Shell



Which of the following expressions are valid JSON documents?

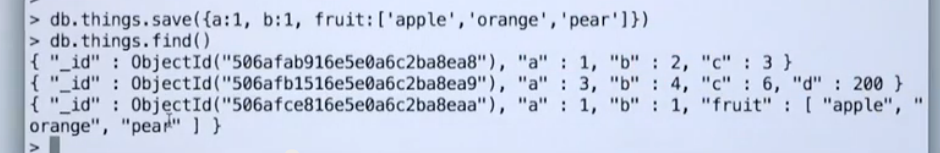
Top of Form

{a:1, b:2, c:3}{a,1; b,4, c,6}{a:1; b:1; c:4}(A,1; b:2; c,4}

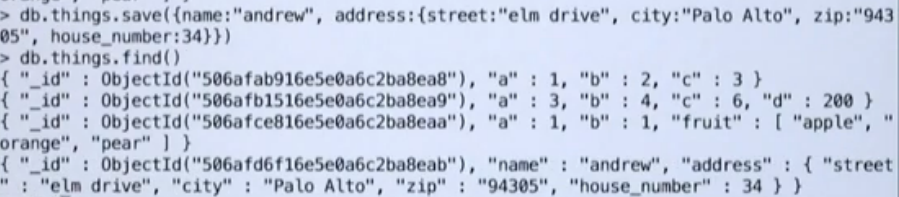
Bottom of Form

## Video - Json Introduced

### Adding json array inside a json object



## Storing json object inside a json object



Which of the following are valid JSON documents?

Top of Form

{a:1, b:2, c: 3}

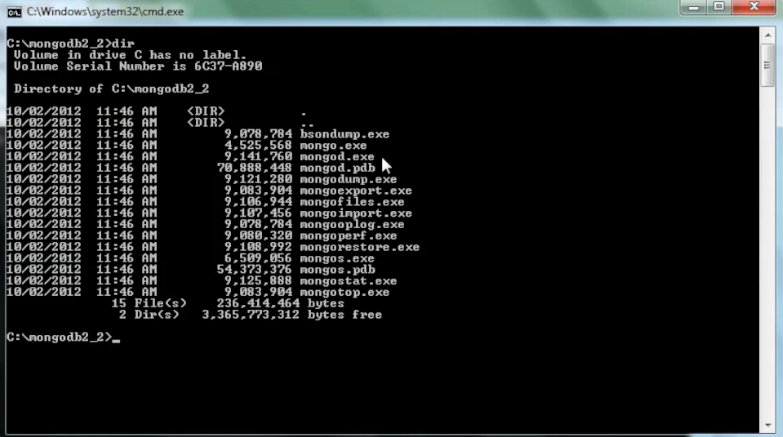
{a:1, b:2, c:[1,2,3,4,5]}

{a:1, b:{}, c: [ { a:1, b:2}, 5, 6]}

{ }

Bottom of Form

# Video - Install Mongodb Windows



mongod.ex : database

mongo.exe: adminsitrative shell

The **mongo.exe <mongo>** shell will connect to **mongod.exe** running on the localhost interface and port 27017 by default.

mongos.exe has all of the features of **mongos** on Unix-like platforms and is completely compatible with the other builds of **mongos**. In addition, **mongos.exe** provides several options for interacting with the Windows platform itself.

## --dbpath

To specify a [dbpath](http://docs.mongodb.org/manual/reference/configuration-options/#dbpath) for **mongod** to use as a data directory, use the [*--dbpath*](http://docs.mongodb.org/manual/reference/program/mongod/#cmdoption-mongod--dbpath) option. The following invocation will start a **mongod** instance and store data in the /srv/mongodb path

mongod --dbpath /srv/mongodb/

## Start database

For example: **mongod --dbpath "\Program Files (x86)\mongodb-2.4.4\data\db"**

## mongod version

mongod --version

## Connect database

D:\Program Files (x86)\mongodb-2.4.4\bin>mongo localhost

MongoDB shell version: 2.4.4

connecting to: localhost

## Adding collection to Db

**>** db.users.insert**({**username:"mkyong",password:"123456"**})**

**>** db.users.find**()**

**{** "\_id" : ObjectId**(**"504f45cd17f6c778042c3c07"**)**, "username" : "mkyong", "password" : "123456" **}**

1. show dbs – List all databases.
2. use db\_name – Switches to db\_name.
3. show collections – List all tables in the current selected database.

## Insert a record

To insert a record, uses db.tablename.insert({data}) or db.tablename.save({data}), both works.

**>** db.users.save**({**username:"google",password:"google123"**})**

**>** db.users.find**()**

**{** "\_id" : ObjectId**(**"504f45cd17f6c778042c3c07"**)**, "username" : "mkyong", "password" : "123456" **}**

**{** "\_id" : ObjectId**(**"504f48ea17f6c778042c3c0a"**)**, "username" : "google", "password" : "google123" **}**

## Update a record

To update a record, uses db.tablename.update({criteria},{$set: {new value}}). In below example, the password of username : “mkyong” is updated.

**>** db.users.update**({**username:"mkyong"**}**,**{**$set:**{**password:"hello123"**}})**

**>** db.users.find**()**

**{** "\_id" : ObjectId**(**"504f48ea17f6c778042c3c0a"**)**, "username" : "google", "password" : "google123" **}**

**{** "\_id" : ObjectId**(**"504f45cd17f6c778042c3c07"**)**, "password" : "hello123", "username" : "mkyong" **}**

## Find Records

To find or query records, uses db.tablename.find({criteria}).

### List all records from table “users”.

db.users.find**()**

**>** db.users.find**()**

**{** "\_id" : ObjectId**(**"504f48ea17f6c778042c3c0a"**)**, "username" : "google", "password" : "google123" **}**

**{** "\_id" : ObjectId**(**"504f45cd17f6c778042c3c07"**)**, "password" : "hello123", "username" : "mkyong" **}**

### Find records where username is “google”

db.users.find**({**username:"google"**})**

**>** db.users.find**({**username:"google"**})**

**{** "\_id" : ObjectId**(**"504f48ea17f6c778042c3c0a"**)**, "username" : "google", "password" : "google123" **}**

### Find records where username’s length is less than or equal to 2

db.users.find**({**$where:"this.username.length<=2"**})**

Find records where username field is existed.

db.users.find**({**username:**{**$exists : **true}})**

## Delete Record

To delete a record, uses db.tablename.remove({criteria}). In below example, the record of username “google” is deleted.

**Note**  
To delete all records from a table, uses db.tablename.remove().  
To drop the table, uses db.tablename.drop().

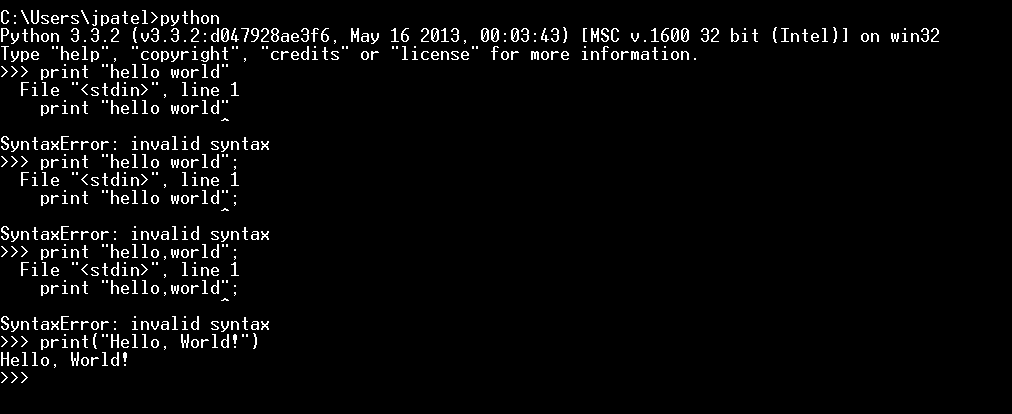
**>** db.users.remove**({**username:"google"**})**

**>** db.users.find**()**

**{** "\_id" : ObjectId**(**"504f45cd17f6c778042c3c07"**)**, "password" : "hello123", "username" : "mkyong" **}**

## Video - Installing Python windows

Add " D:\Program Files (x86)\Python33;" to "Path" in "Environment Variable"



# Video - Installing Bottle python web framework

1. Download and install file by **running** "ez\_setup.py" (http://bottlepy.org/docs/0.11/)

2. Install Bottle framework

**D:\Program Files (x86)\Python33\Scripts**>easy\_install.exe -U bottle

Searching for bottle

Reading https://pypi.python.org/simple/bottle/

Best match: bottle 0.11.6

Downloading https://pypi.python.org/packages/source/b/bottle/bottle-0.11.6.tar.gz#md5=0bafdc4e13ea2b1a3bddf36b5af108c4

Processing bottle-0.11.6.tar.gz

Writing c:\users\jpatel\appdata\local\temp\easy\_install-bhln29\bottle-0.11.6\setup.cfg

Running bottle-0.11.6\setup.py -q bdist\_egg --dist-dir c:\users\jpatel\appdata\local\temp\easy\_install-bhln29\bottle-0.1

zip\_safe flag not set; analyzing archive contents...

\_\_pycache\_\_.bottle.cpython-33: module references \_\_file\_\_

\_\_pycache\_\_.bottle.cpython-33: module references \_\_path\_\_

Adding bottle 0.11.6 to easy-install.pth file

Installing bottle.py script to D:\Program Files (x86)\Python33\Scripts

Installed d:\program files (x86)\python33\lib\site-packages\bottle-0.11.6-py3.3.egg

Processing dependencies for bottle

Finished processing dependencies for bottle

**D:\Program Files (x86)\Python33\Scripts>**

## Check if framework works

Save the programm and run it

from bottle import route, run, template

@route('/hello/:name')

def index(name='World'):

return template('<b>Hello {{name}}</b>!', name=name)

run(host='localhost', port=8080)

Output

http://localhost:8080/hello/jaishriram

jaishriram

# Video - Installing PyMongo

## Output

D:\Program Files (x86)\mongodb-2.4.4\data\education\m101P\week1\lesson\_files>python python\_getting\_started.py

Hello

mkyong

## Python code

import pymongo

from pymongo import MongoClient

#connect to dB

connection = MongoClient('localhost', 27017)

#localhost: name of dB

dB = connection.localhost

#users: name of collection inside "databse: localhost"

users = dB.users

item = users.find\_one()

print( item['username'] )

## dB

> db.users.find()

{ "\_id" : ObjectId("51ca5081c0df442471537436"), "password" : "hello123", "username" : "mkyong" }

{ "\_id" : ObjectId("51cce7bfa75b499eed37dfd5"), "username" : "google", "password" : "google123" }

# Video - Hello World Mongo Style

## Return a Single Document from a Collection

With the [db.collection.findOne()](http://docs.mongodb.org/manual/reference/method/db.collection.findOne/#db.collection.findOne) method you can return a single document from a MongoDB collection. The [findOne()](http://docs.mongodb.org/manual/reference/method/db.collection.findOne/#db.collection.findOne) method takes the same parameters as [find()](http://docs.mongodb.org/manual/reference/method/db.collection.find/#db.collection.find), but returns a document rather than a cursor.

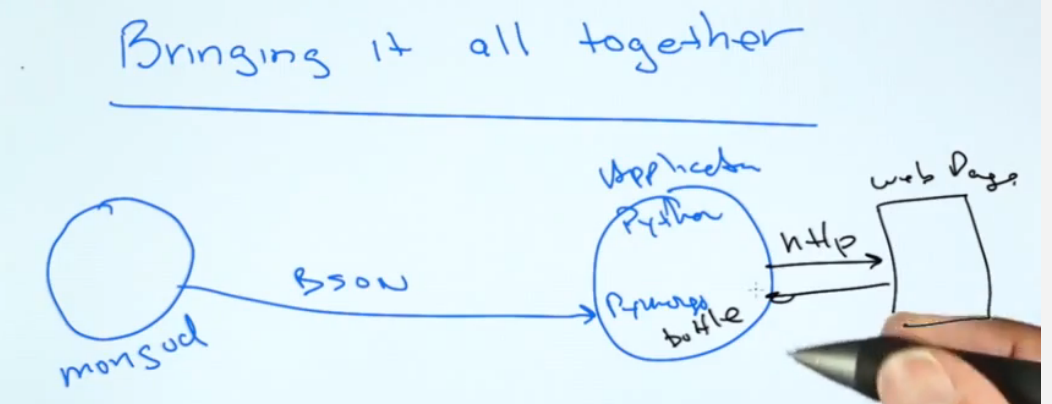
To retrieve one document from the testData collection, issue the following command:

db.testData.findOne()

|  |
| --- |
| > db.users.find()  { "\_id" : ObjectId("51ca5081c0df442471537436"), "password" : "hello123", "username" : "mkyong" }  { "\_id" : ObjectId("51cce7bfa75b499eed37dfd5"), "username" : "google", "password" : "google123" }  > db.users.findOne()  {  "\_**id**" : ObjectId("51ca5081c0df442471537436"),  "password" : "hello123",  "username" : "mkyong"  }  > var j = db.users.findOne() - **Stores Object Id**  > j  {  "\_id" : ObjectId("51ca5081c0df442471537436"),  "password" : "hello123",  "username" : "mkyong"  }  > j.username = jatin  Thu Jun 27 19:05:30.139 JavaScript execution failed: ReferenceError: jatin is not defined  > j.username = "jatin"  jatin  > j  {  "\_id" : ObjectId("51ca5081c0df442471537436"),  "password" : "hello123",  "username" : "jatin"  }  **> db.users.save(j)**  > db.users.findOne()  {  "\_id" : ObjectId("51ca5081c0df442471537436"),  "password" : "hello123",  "username" : "jatin"  }  > db.users.find()  { "\_id" : ObjectId("51ca5081c0df442471537436"), "password" : "hello123", "username" : "jatin" }  { "\_id" : ObjectId("51cce7bfa75b499eed37dfd5"), "username" : "google", "password" : "google123" } |
| > db.users.save - **Searches for object id and updates if suitable match is found**  function ( obj ){  if ( obj == null || typeof( obj ) == "undefined" )  throw "can't save a null";  if ( typeof( obj ) == "number" || typeof( obj) == "string" )  throw "can't save a number or string"  if ( typeof( obj.\_id ) == "undefined" ){  obj.\_id = new ObjectId();  return this.insert( obj );  }  else {  return this.**update**( { \_id : obj.\_id } , obj , true );  }  }  > |

# Video - Hello World on Server

**Pymongo communicates with mongod using BSON**

****

|  |  |
| --- | --- |
| Ouput | Code |
| Browser  http://localhost:8082/  jatin | import pymongo  import bottle  from pymongo import MongoClient  @bottle.route('/')  def index():  #connect to dB  connection = MongoClient('localhost', 27017)  #localhost: name of dB  dB = connection.localhost  #users: name of collection inside "database: localhost"  users = dB.users  item = users.find\_one()  return '<b>Hello %s!</b>' %item['username']    bottle.run(host='localhost', port=8082) |

# Video - Mongo is Schemaless

|  |  |
| --- | --- |
| Each row can follow any schema | Update an object |
| > db.users.find()  > db.users.save({name:"jatin patel", city\_of\_birth:"Mumbai"})  > db.users.save({name:"p patel", city\_of\_birth:"Ahmedabad", favourite\_color:"I dont know"})  > db.users.find().pretty()  {  "\_id" : ObjectId("51ccf564ff56ba712a40bd5b"),  "name" : "jatin patel",  "city\_of\_birth" : "Mumbai"  }  {  "\_id" : ObjectId("51ccf585ff56ba712a40bd5c"),  "name" : "p patel",  "city\_of\_birth" : "Ahmedabad",  "favourite\_color" : "I dont know"  } | > j = db.users.findOne({name:"jatin patel"})  {  "\_id" : ObjectId("51ccf564ff56ba712a40bd5b"),  "name" : "jatin patel",  "city\_of\_birth" : "Mumbai"  }  > j.favourite\_color = "Dont care to remeber"  Dont care to remeber  > j  {  "\_id" : ObjectId("51ccf564ff56ba712a40bd5b"),  "name" : "jatin patel",  "city\_of\_birth" : "Mumbai",  "favourite\_color" : "Dont care to remeber"  }  > db.users.save(j)  > db.users.find().pretty()  {  "\_id" : ObjectId("51ccf585ff56ba712a40bd5c"),  "name" : "p patel",  "city\_of\_birth" : "Ahmedabad",  "favourite\_color" : "I dont know"  }  {  "\_id" : ObjectId("51ccf564ff56ba712a40bd5b"),  "name" : "jatin patel",  "city\_of\_birth" : "Mumbai",  "favourite\_color" : "Dont care to remeber"  } |

# Video - JSON revisited

Json has only 2 data structures

Top level has to be dictionary

1. Arrays - list of things

For e.g. **[**'1' , '2' ,...**]**

1. Dictionaries - Associative maps

Key value pair

**{**username:"jd", password:"heheh"**}**

## Quiz: JSON Subdocuments

Write a JSON document with a single key, "address" that has as it value another document with the keys 'street\_address', 'city', 'state', 'zipcode', with the following values: 'street\_address' is "23 Elm Drive", 'city' is "Palo Alto", 'state' is "California", 'zipcode' is "94305"

Answer

Key should have a single quote and value double quote

{'address': {'street\_address': "23 Elm Drive", 'city' : "Palo Alto", 'state': "California", 'zipcode': "94305"} }

# Quiz: Blog in Relational Tables

let’s assume that our blog can be modeled with the following relational tables:

authors:

author\_id,

name,

email,

password

posts:

post\_id,

author\_id

title,

body,

publication\_date

comments:

comment\_id,

name,

email,

comment\_text

post\_comments:

post\_id,

comment\_id

tags

tag\_id

name

post\_tags

post\_id

tag\_id

In order to display a blog post with its comments and tags, how many tables will need to be accessed?

Top of Form

2356Video

# Video - Modeling our Blog in Mongo

Given the document schema that we proposed for the blog, how many collections would we need to access to display the blog home page?

Top of Form

0124

# Quiz: Intro to Schema Design

In which scenario is it impossible to embed data within a document (you must put the data in it a separate collection). Check all that apply.

Top of Form

The data would be duplicated across multiple objects within a collection.You need an index on the data element.The embedded data could exceed the 16MB document limit within MongoDBThe data is not isomorphic.

# Quiz - intro to python

* Python is readable, helps with GC and can be dynamically typed.

# Video - python Lists

**python list are like json list**

|  |  |
| --- | --- |
| >>> a = ['apple1', "apple2", 'apple3']  >>> a  ['apple1', 'apple2', 'apple3']  >>> print(a)  ['apple1', 'apple2', 'apple3'] | >>> c = [1, ['apple1', 'apple2'], 3]  >>> c  [1, ['apple1', 'apple2'], 3]  >>> print(c)  [1, ['apple1', 'apple2'], 3]  >>> print(c[2])  3  >>> print(c[1])  ['apple1', 'apple2'] |
| >>> b = [1,2,3]  >>> b  [1, 2, 3]  **Integers doesn't need any quotes** |  |

Write the code to initialize a list with the items "hammer", "nail" and "wall" and assign the list to the variable named "things".

things = ["hammer","nail","wall"]

# Video - Python lists, Manipulating

|  |  |
| --- | --- |
| >>> c  [1, ['apple1', 'apple2'], 3]  >>> c.append("antartica")  >>> c  [1, ['apple1', 'apple2'], 3, 'antartica']  >>> c[1] = "One"  >>> c  [1, 'One', 3, 'antartica'] | >>> dir(c)  ['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_delitem\_\_', '\_\_dir\_\_', '\_\_doc\_\_',  getitem\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_iadd\_\_', '\_\_imul\_\_', '\_\_init\_\_', '\_\_iter\_\_', '\_\_le\_\_',  duce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_reversed\_\_', '\_\_rmul\_\_', '\_\_setattr\_\_', '\_\_setitem\_  'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort'] |

Write the code to append the item "hammer" onto a list named things. Please use double quotes at this time.

things.append("hammer")

# Video - Python Lists, Slice Operator

|  |  |
| --- | --- |
| >>> a=["One", "Two", "Three", "Four", "Five"]  >>> a  ['One', 'Two', 'Three', 'Four', 'Five']  >>> a[0:3]  ['One', 'Two', 'Three']  >>> a[2:3]  ['Three']  >>> a[2:]  ['Three', 'Four', 'Five'] | >>> a[:7]  ['One', 'Two', 'Three', 'Four', 'Five']  >>> a[:9]  ['One', 'Two', 'Three', 'Four', 'Five']  >>> a[:6]  ['One', 'Two', 'Three', 'Four', 'Five']  >>> a[:]  ['One', 'Two', 'Three', 'Four', 'Five'] |

## Quiz:

things = ['apples', 'orange', 'pear', 'grape', 'kiwi']. What is the slice notation that will return the sublist ['orange', 'pear']?

things[1:3]

# Video - Python lists, Inclusion

|  |  |
| --- | --- |
| >>> a  ['One', 'Two', 'Three', 'Four', 'Five']  >>> 'One' in a  True  >>> 'One ' in a  False  >>> One in a  Traceback (most recent call last):  File "<stdin>", line 1, in <module>  NameError: name 'One' is not defined  >>> if 'One' in a  File "<stdin>", line 1  if 'One' in a | >>> if 'One' in a:  ... print("There is one")  ...  There is one  >>> |

Quiz

given a python list called "fruit", write an if statement to check whether "apple" is in the list.

Please use double quotes at this time.

if "apple" in fruit":

# Video - Python Dicts / Dictionary

## Note: Python does not retain order inside dictionary.

|  |  |
| --- | --- |
| Dictionary example >>> g = {'name':'Jatinkumar patel', 'city\_of\_birth':'Mumbai'}  >>> g  {'name': 'Jatinkumar patel', 'city\_of\_birth': 'Mumbai'} | Retrieving all keys >>> g.keys()  dict\_keys(['name', 'city\_of\_birth']) |
| Find element by key >>> g['name']  'Jatinkumar patel' | Checking 'keys ' in dictionary >>> 'name' in g  True  >>> 'city\_of\_birth' in g  True |
| Delete key-value in dictionary >>> del(g['name'])  >>> g  {'city\_of\_birth': 'Mumbai'}  >>> |  |

Quiz

Initialize a new dict named "colors" with the following key values pairs: sky is blue, sea is blue. earth is brown. *Note: Please preserve the order of these keys when you enter your answer.*

colors = {'sky': 'blue', 'sea' : 'blue', 'earth': 'brown'}

# Video - Dicts and Lists together

|  |  |
| --- | --- |
| Mix >>> a ={'name': 'jatinkumar patel', 'interest':['cycling', 'running', 'golf']}  >>> a  {'interest': ['cycling', 'running', 'golf'], 'name': 'jatinkumar patel'}  >>> a['interest']  ['cycling', 'running', 'golf']  >>> a['interest'][0]  'cycling' | Append >>> a['interest'].append('dancing')  >>> a  {'interest': ['cycling', 'running', 'golf', 'dancing'], 'name': 'jatinkumar patel'} |

Initialize a new dict with a single key, "animals" whose value is the list "dog", "cat", "zebra" and assign the entire expression to variable named "things"

things={"animals":["dog", "cat", "zebra"]}

# Video - Python for loop with lists

|  |  |
| --- | --- |
| #Initialize array  array1 = ['One', 'Two', 'Three']  array2 = []  for item in array1:  print(item)  array2.append(item)    print(array2)  **O/P**  One  Two  Three  ['One', 'Two', 'Three'] | Quiz  What does the following code print?  sum = 0  numbers = [1,2,3,5,8]  for i in numbers:  sum = sum + i  print i  8 |

# Video - Python for loops with dicts

|  |  |
| --- | --- |
| #Initialize dictionary  person={'name':'hiral', 'favourite\_color':"red", "hair":"black"}  print(person)  for key in person:  print("key: "+key+", value: "+person[key])  o/p  {'name': 'hiral', 'hair': 'black', 'favourite\_color': 'red'}  key: name, value: hiral  key: hair, value: black  key: favourite\_color, value: red | Quiz: people = {'name':'Bob', 'hometown': "Palo Alto", 'favorite\_color': 'red'}  for item in people:  if (item == 'favorite\_color'):  print people[item]  red |

# Video - Python: Combining Dicts and Lists

|  |  |
| --- | --- |
| Code  person={'name':'hiral',  'favourite\_color':"red",  "hair":"black",  "interests":["hiking","research","studying"]}  for key in person:  if(key == 'interests'):  print("key: "+key)  for interest in person[key]:  print(interest)  else:  print("key: "+key+", value: "+person[key]) Output key: favourite\_color, value: red  key: hair, value: black  key: name, value: hiral  key: interests  hiking  research  studying | Quiz obj = {'a':1,'b': 2, 'c': [1, 3, 5]}  sum = 0  if 'c' in obj:  for n in obj['c']:  sum = sum + n  print sum  9 |

# Video - Python - while loop

**i++ is not legal in python**

|  |
| --- |
| #Initialize array  array1 = ['One', 'Two', 'Three']  i = 0  while(i < len(array1)):  print(array1[i])  i = i+1 |

# Video - Python function calls

In python, every function name starts with "def functionName (args):"

|  |
| --- |
| #Initialize array  array1 = ['One', 'Two', 'Three', 'One', 'Four', 'Three', 'One', 'Two']  **def analyze\_list(list):**    #Initialize dictionary  counts\_of\_each\_instance = {}    for item in list:  if item in counts\_of\_each\_instance:  counts\_of\_each\_instance[item] = counts\_of\_each\_instance[item] + 1;  else:  counts\_of\_each\_instance[item] = 1    return counts\_of\_each\_instance    variable1 = analyze\_list(array1)  print(variable1) o/p {'Three': 2, 'One': 3, 'Four': 1, 'Two': 2} |

# Video - Python - Exception Handling

|  |  |
| --- | --- |
| import sys  try:  print(5/0)  except:  print("exception ", sys.exc\_info()[0])    print("but life goes on") | Output exception <class 'ZeroDivisionError'>  but life goes on |

# Video - Bottle framework: URL Handler

|  |
| --- |
| import bottle  @bottle.route('/')  def homepage():  return "Hello World"    @bottle.route('/testpage')  def homepage():  return "Hello World - test page"  bottle.debug(true);  bottle.run(host='localhost', port=8082) |

# Video - Bottle framework: using views

|  |
| --- |
| Output  Welcome Andrew   * apple * baanananana * orange * peach   **Note: View (\*.tpl) should be inside a child directory 'views'** |
| Code: D:\Program Files (x86)\mongodb-2.4.4\data\education\m101P\week1\lesson\_files\bottle framework using views  import bottle  @bottle.route('/')  def homepage():  mythings = ['apple','baanananana','orange','peach']  #return bottle.template('view\_python\_bottle\_viewController', usernameInHtml="Andrew", thingsInHtml=mythings) - **APPROACH 1**    # passing parameter in DICTIONARY  return bottle.template('view\_python\_bottle\_viewController',{'usernameInHtml':"Andrew", 'thingsInHtml':mythings}) - **APPROACH2**    bottle.debug(True);  bottle.run(host='localhost', port=8082) |
| view\_python\_bottle\_viewController.tpl  <html>  <head>  <title>Hello world</title>  </head>  <body>  <p>  Welcome {{usernameInHtml}}  </p>  <ul>  %for fruits in thingsInHtml:  <li>{{fruits}}</li>  %end  </ul>  </body>  </html> |

# Video - Bottle framework: Handling form content

|  |  |
| --- | --- |
| POST method  Welcome Andrew   * apple * baanananana * orange * peach   Top of Form  What is your favourite food ?  Bottom of Form | Output  Your favourite food is apple |

|  |
| --- |
| import bottle  @bottle.route('/')  def homepage():  mythings = ['apple','baanananana','orange','peach']  # passing parameter in DICTIONARY  return bottle.template('form\_post',{'usernameInHtml':"Andrew", 'thingsInHtml':mythings})    **@bottle.post('/favourite\_fruit')**  def favouritefood():  fruit\_selected = **bottle.request.forms.get("fruit")**    if(fruit\_selected == None or fruit\_selected == ""):  fruit\_selected = "No Fruit selected"    return bottle.template('form\_selection.tpl',{'fruit\_selectedInHtml':fruit\_selected})  bottle.debug(True);  bottle.run(host='localhost', port=8082) |
| |  |  | | --- | --- | | <html>  <head>  <title>Hello world</title>  </head>  <body>  <p>  Welcome {{usernameInHtml}}  </p>  <ul>  %for fruits in thingsInHtml:  <li>{{fruits}}</li>  %end  </ul>    </p>  <form action="/favourite\_fruit" method="POST">  What is your favourite food ?  <input type="text" name="fruit" size="40"><br>  <input type="submit" value="Submit">  </form>    </body>  </html> | <html>  <head>  <title>Fruit selection confirmation</title>  </head>  <body>  <p>  Your favourite food is {{fruit\_selectedInHtml}}  </p>  </body>  </html> | |

# Video - Bottle framework: using cookies

|  |
| --- |
| import bottle  @bottle.route('/')  def homepage():  mythings = ['apple','baanananana','orange','peach']    # passing parameter in DICTIONARY  return bottle.template('form\_post',{'usernameInHtml':"Andrew", 'thingsInHtml':mythings})    @bottle.post('/favourite\_fruit')  def favouritefood():  fruit\_selected = bottle.request.forms.get("fruit")    if(fruit\_selected == None or fruit\_selected == ""):  fruit\_selected = "No Fruit selected"    bottle.response.set\_cookie("fruit\_cookie", fruit\_selected);  bottle.redirect("/show\_fruit")    @bottle.route('/show\_fruit')  def showfavouritefood():  fruit\_selected = bottle.request.get\_cookie("fruit\_cookie")  return bottle.template('form\_selection.tpl',{'fruit\_selectedInHtml':fruit\_selected})    bottle.debug(True);  bottle.run(host='localhost', port=8082) |

# Video - saving data - (pymongo driver)

|  |
| --- |
| Output > db.people.find().pretty()  {  "\_id" : ObjectId("51d09fe608ba821c38789aa8"),  "role" : "Professor",  "name" : "Hiral patel",  "address" : {  "address1" : "The white House",  "state" : "Oregon"  }  } |
| import pymongo  def main():    connection = pymongo.MongoClient("mongodb://localhost")  db = connection.m101  people = db.people  person = {'name': 'Hiral patel', 'role':'Professor',  'address':{'address1':'The white House',  'state': 'Oregon'  }  }    people.insert(person)  main() |

# Video - Pymongo exception processing

|  |
| --- |
| Output 1 D:\Program Files (x86)\mongodb-2.4.4\data\education\m101P\week1\lesson\_files>python mongo\_exception.py  {'role': 'Professor', 'name': 'Hiral patel'}  about to insert person  {'role': 'Professor', 'name': 'Hiral patel', '\_id': ObjectId('51d0a79f08ba821de8df0f36')}  **about to insert person**  **insert failed: <class 'pymongo.errors.DuplicateKeyError'>** |
| import pymongo  import sys  def main():    connection = pymongo.MongoClient("mongodb://localhost")    db = connection.m101  people = db.people    person = {'name': 'Hiral patel', 'role':'Professor'}  print(person)  print("about to insert person")    try:  people.insert(person)  except:  print("insert failed: ", sys.exc\_info()[0])    print(person)  print("about to insert person")    try:  people.insert(person)  except:  print("insert failed: ", sys.exc\_info()[0])  main() |
| Change required try:  people.insert(person)  except:  print("insert failed: ", sys.exc\_info()[0])    **person = {'name': 'Hiral patel', 'role':'Professor'}**    print(person)  print("about to insert person")    try:  people.insert(person)  except:  print("insert failed: ", sys.exc\_info()[0]) |

## Homework: Homework 1.1

Install MongoDB on your computer and run it on the standard port.

Download the [HW1 tarball (mac)](https://education.10gen.com/static/m101p-june-2013/handouts/hw1.1b0437e07670.tar) or [zipfile (windows)](https://education.10gen.com/static/m101p-june-2013/handouts/hw1.96978f031d01.zip), expand it as follows:

Mac Users

tar -xvf hw1.tar

Windows Users

You probably don't have tar installed so right click on the hw1.zip file and choose "extract all"

Use mongorestore to restore the dump into your running mongod. Do this by opening a terminal window (mac) or cmd window (windows) and navigating to the directory so that the dump directory is directly beneath you. Now type

mongorestore dump

Note you will need to have your path setup correctly to find mongorestore.

Now, using the Mongo shell, perform a findone on the collection called hw1 in the database m101. That will return one document. Please provide the value corresponding to the "answer" key from the document returned.

Bottom of Form

Bottom of Form