DATABASE SYSTEMS

Database programming in a web environment

Database System Course

TEL AVIV UNIVERSITY

AGENDA FOR TODAY

- The final project
- Advanced Mysql
- Database programming
- Recap: DB servers in the web
- Web programming architecture
- HTTP on a need-to-know basis.
- How to use web APIs

THE FINAL PROJECT

- **Project goal**
  - Building your very own web application
  - Design a database, optimize it and compose several complex queries
  - Data will be obtained from the world wide web
- **Requirements**
  - Coding in Python, or in PHP if you wish. No other languages allowed
  - Teams of 4-5 (send me your names)
  - The web application will be deployed and run on university servers.

THE FINAL PROJECT STEP BY STEP

1.Assemble a team

2.Find the API you like

3.Get a general idea

4.Design the database

5.Fetch the data

6.Compose queries

7.Optimize

8.Build a UI

9.Test on UNI servers

10.Write the docs

11. Submit!

THE FINAL PROJECT

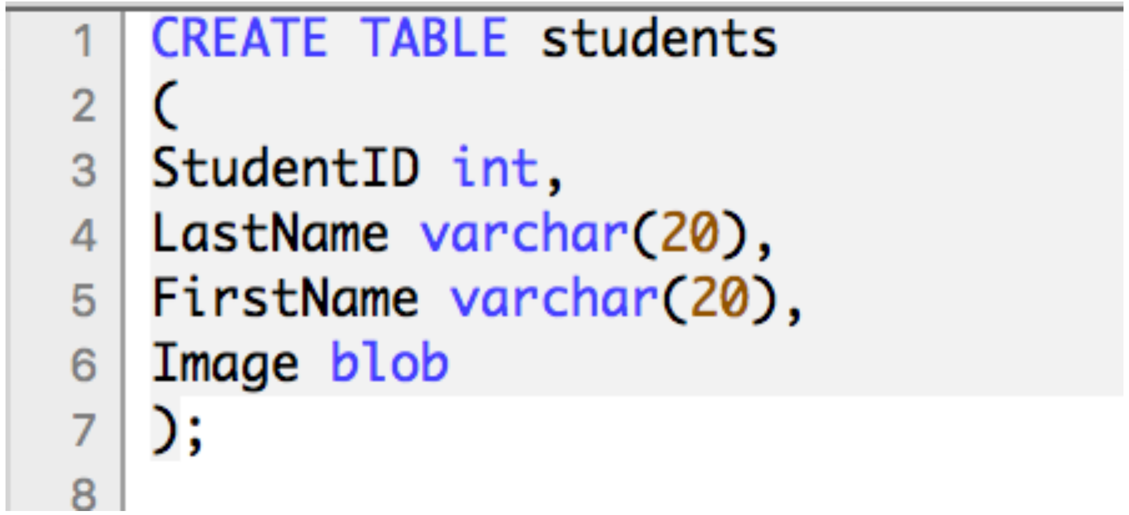
- **Important tips**
  - Read the project document and the grading guide carefully!
  - Working in a group of 4-5 is not easy. Plan and divide the tasks efficiently
  - APIs have requests limits. Start using them early to fetch enough data.
  - Your application should not rely on users contribution for its main functions.
  - Constantly test your code on the university servers, don't leave it to the last minute.
  - Focus on the DB design, optimizations and interesting queries, rather on the UI.
  - Get the bonus! (+10 points)

AGENDA FOR TODAY

- The final project
- **Advanced Mysql**
- Database programming
- Recap: DB servers in the web
- Web programming architecture
- HTTP on a need-to-know basis.
- How to use web APIs

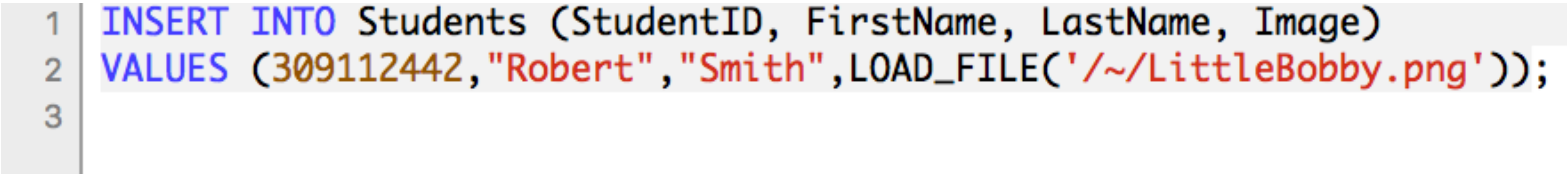
ADAVANCED MYSQL

- **More than just SELECT**
  - CREATE



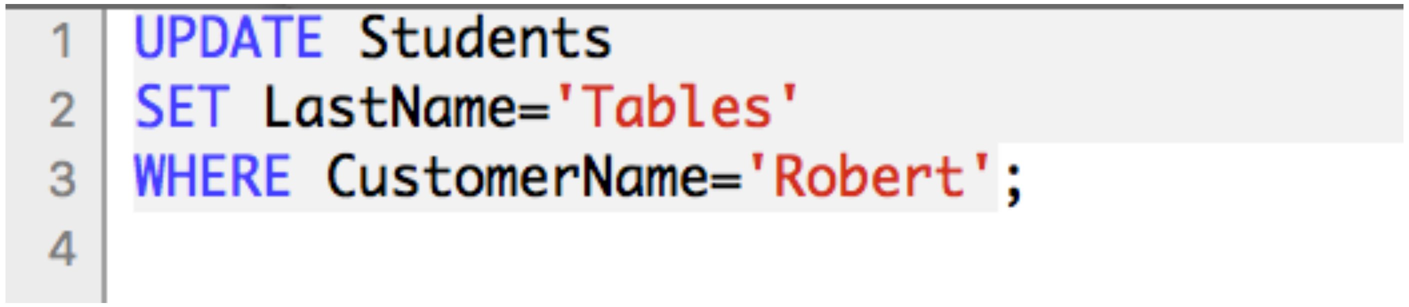
```
CREATE TABLE students
(
StudentID int,
LastName varchar(20),
FirstName varchar(20),
Image blob
);

```

  - INSERT



```
INSERT INTO Students (StudentID, FirstName, LastName, Image)
VALUES (309112442,"Robert","Smith",LOAD_FILE('/~/LittleBobby.png'));

```

  - UPDATE

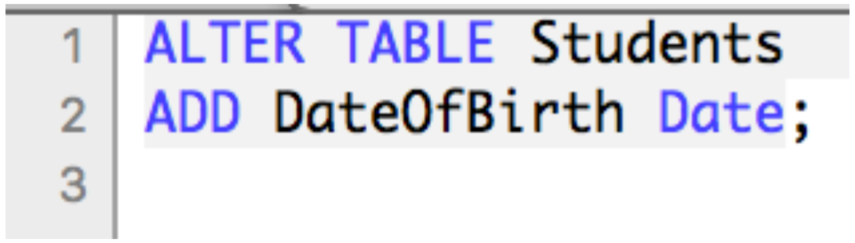


```
UPDATE Students
SET LastName='Tables'
WHERE CustomerName='Robert';

```

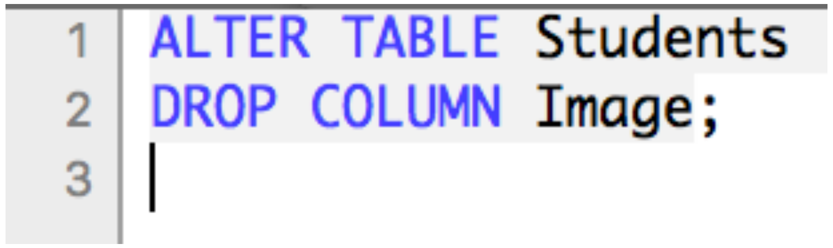
ADAVANCED MYSQL

More than just SELECT

- ALTER

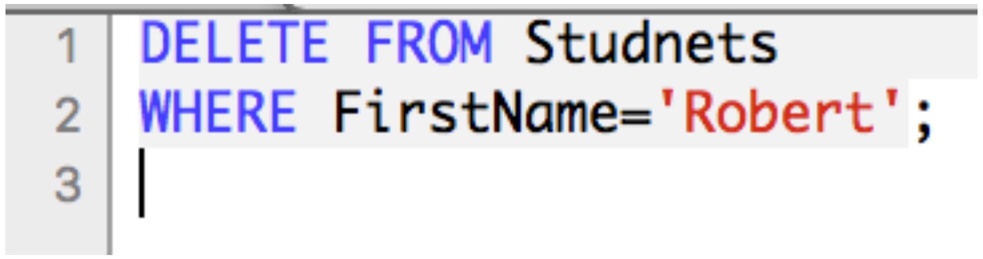


```
ALTER TABLE Students
ADD DateOfBirth Date;

```



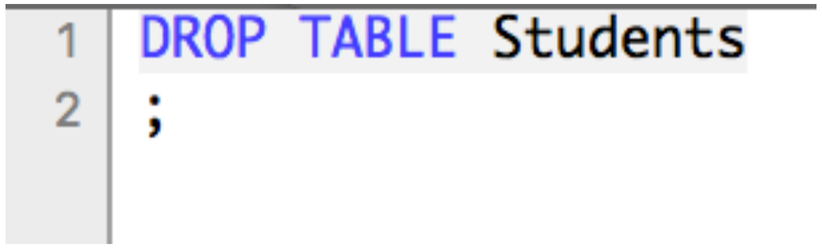
```
ALTER TABLE Students
DROP COLUMN Image;

```

- DELETE



```
DELETE FROM Studnets
WHERE FirstName='Robert';

```

- DROP



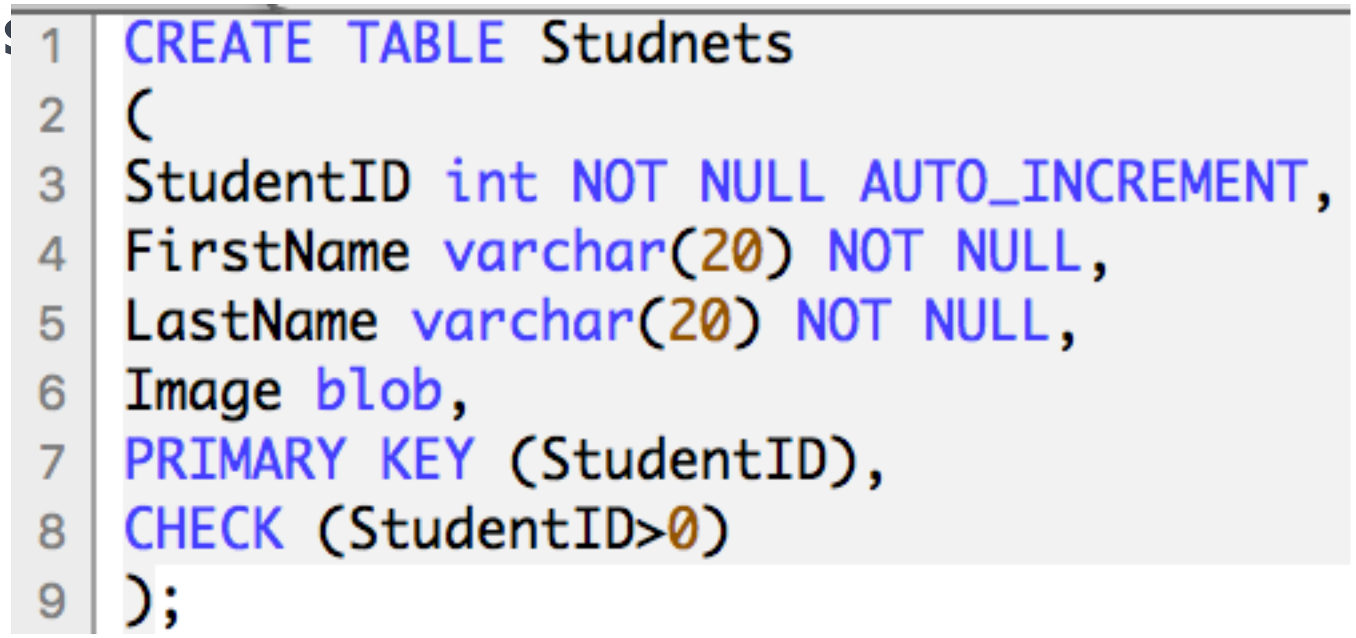
```
DROP TABLE Students
;
```

ADAVANCED MYSQL

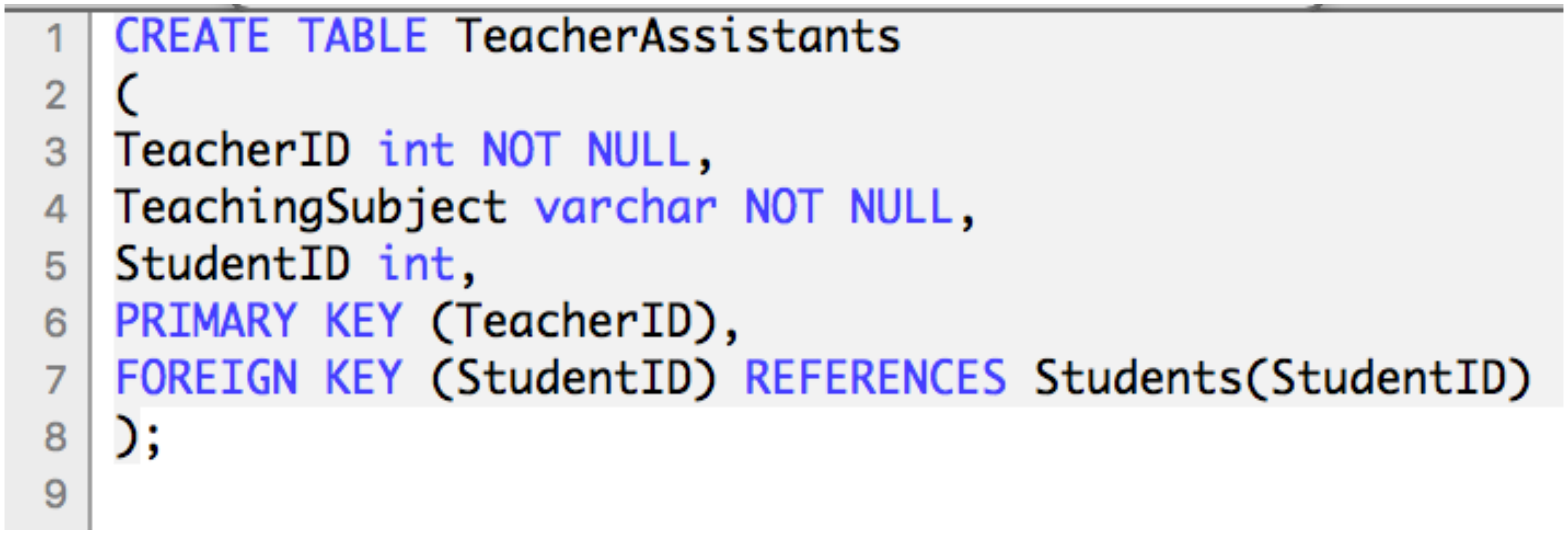
- **Creating tables:**
  - **Field constraints:**
    - NOT NULL - Indicates that a column cannot store NULL value
    - UNIQUE - Ensures that each row for a column must have a unique value
    - PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have a unique identity which helps to find a particular record in a table more easily and quickly
    - FOREIGN KEY - Ensure the referential integrity of the data in one table to match values in another table
    - CHECK - Ensures that the value in a column meets a specific condition
    - DEFAULT - Specifies a default value for a column

ADAVANCED MYSQL

- **Creating tables**
  - **Constraints**



```sql
CREATE TABLE Studnets
(
StudentID int NOT NULL AUTO_INCREMENT,
FirstName varchar(20) NOT NULL,
LastName varchar(20) NOT NULL,
Image blob,
PRIMARY KEY (StudentID),
CHECK (StudentID>0)
);
```

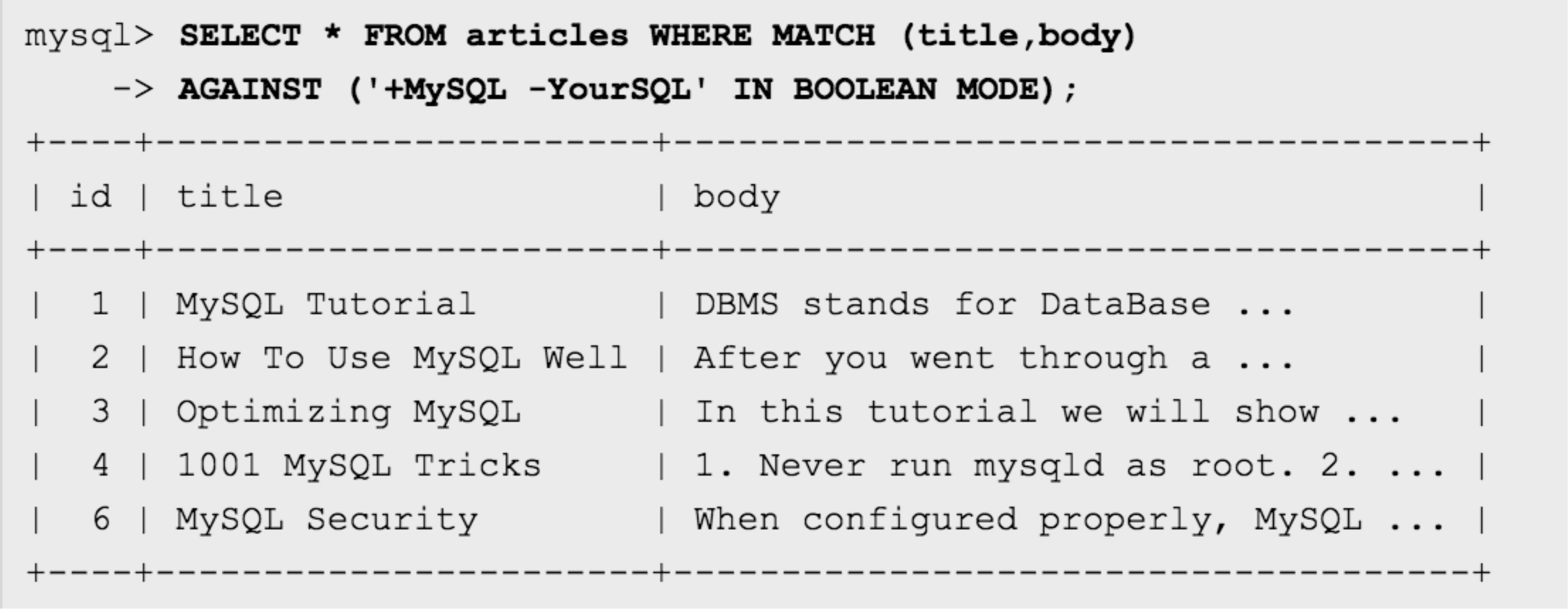


```sql
CREATE TABLE TeacherAssistants
(
TeacherID int NOT NULL,
TeachingSubject varchar NOT NULL,
StudentID int,
PRIMARY KEY (TeacherID),
FOREIGN KEY (StudentID) REFERENCES Students(StudentID)
);
```

ADAVANCED MYSQL

Full Text search: MATCH … AGAINST

- Please don’t use “…LIKE “%MySQL%”.
- + for AND
- - for NOT,
- nothing for OR



```
mysql> SELECT * FROM articles WHERE MATCH (title,body)
    -> AGAINST ('+MySQL -YourSQL' IN BOOLEAN MODE);
+----+-----------------------+-------------------------------------+
| id | title                 | body                                |
+----+-----------------------+-------------------------------------+
|  1 | MySQL Tutorial        | DBMS stands for DataBase ...        |
|  2 | How To Use MySQL Well | After you went through a ...        |
|  3 | Optimizing MySQL      | In this tutorial we will show ...   |
|  4 | 1001 MySQL Tricks     | 1. Never run mysqld as root. 2. ... |
|  6 | MySQL Security        | When configured properly, MySQL ... |
+----+-----------------------+-------------------------------------+
```

ADAVANCED MYSQL

MySQL Optimizations

- **Storage Engines (Database Engine):**
  - The underlying software performing CRUD operations: Create, Read, Update, Delete.
  - The storage engine implements the data structures and memory usage strategy
  - Many of the modern DBMS support multiple storage engines within the same database
  - MySQL support InnoDB and MyISAM

ADAVANCED MYSQL

MySQL Optimizations (Storage Engines)

- InnoDB:
  - The default general-purpose MySQL storage engine
  - ACID Compliant:
    - **A**tomicity: A transaction (i.e., set of DB operations) is atomic
    - **C**onsistency: Any given database transaction must change affected data only in allowed ways (Triggers, Constraints)
    - **I**solation: Concurrent transactions are isolated from one to another
    - **D**urability: The ability of the system to recover committed transaction updates if either the system or the storage media fails
  - Main Features:
    - ✦ Takes care of data integrity
    - ✦ Row-level locking

ADAVANCED MYSQL

MySQL Optimizations (Storage Engines)

- MyISAM (Indexed Sequential Access Method)
  - **Storage paradigm:**
    - ✦ Each entry points to a record in the data file, and the pointer is offset from the beginning of the file
    - ✦ This way records can be quickly read, especially when the format is FIXED
    - ✦ Inserts are easy too, because new rows are appended to the end of the data file
    - ✦ However, delete and update operations are more problematic: deletes must leave an empty space, or the rows' offsets would change; the same goes for updates, as the length of the rows becomes shorter;
  - **Main features:**
    - ✦ Non Transactional (Does not support foreign keys)
    - ✦ Fits for Read Mostly environments (because of the table level locking mechanism)

ADAVANCED MYSQL

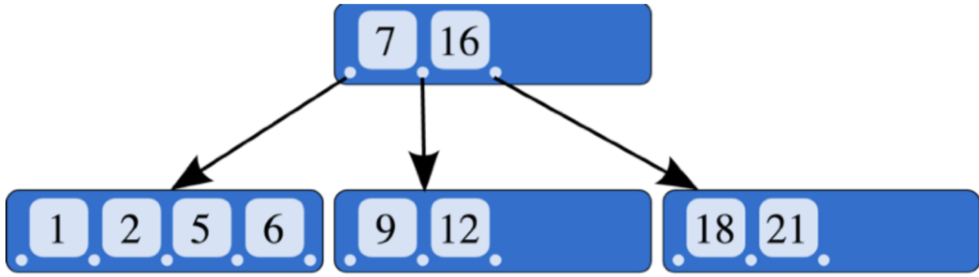
MySQL Optimizations: Indexing

- If you don't use indexes:
  - Your DB is small (or)
  - Your DB is slow
- Indexes are used to find rows with specific column values quickly
- Can be single or multi-column.
- Can use only part of the data:
- **Examples:**
  - CREATE INDEX last ON Students (LastName)
  - CREATE INDEX full_name ON Students (FirstName, LastName)
  - CREATE INDEX part_of_name ON Students (LastName(5));

ADAVANCED MYSQL

MySQL Optimizations: Indexing

- Without an index, MySQL must begin with the first row and then read through the entire table to find the relevant rows
- Updates cost more…
- Storing indexes:
  - **B-tree** (Search, insert and delete are O(log(n))
  - R-tree (Spatial Data)
  - Hash tables
  - Inverted lists (mapping words/numbers to DB entries)
  - FULLTEXT



DB DESIGN: TIPS AND TRICKS

- **Schema Design:** You will have/already had a dedicated class on DB design principles, so please don't worry.

  1. **Use primary keys**:
     - They have special indexes in InnoDB for fast lookups
     - If your table is big and important, but does not have an obvious column or set of columns to use as a primary key:
       - Create a separate column with auto-increment values to use as the primary key.
       - These unique IDs can serve as pointers to corresponding rows in other tables when you join tables using foreign keys.
  2. **Use foreign keys**:
     - Mostly for data integrity
     - For optimisation: Large tables Vs. Many small tables
       - Consider splitting your less-frequently used data into separate tables
       - Each small table can have a primary key for fast lookups of its data, and you can query just the set of columns that you need using a join operation.
       - Queries might perform less I/O and take up less cache memory because the

DB DESIGN: TIPS AND TRICKS

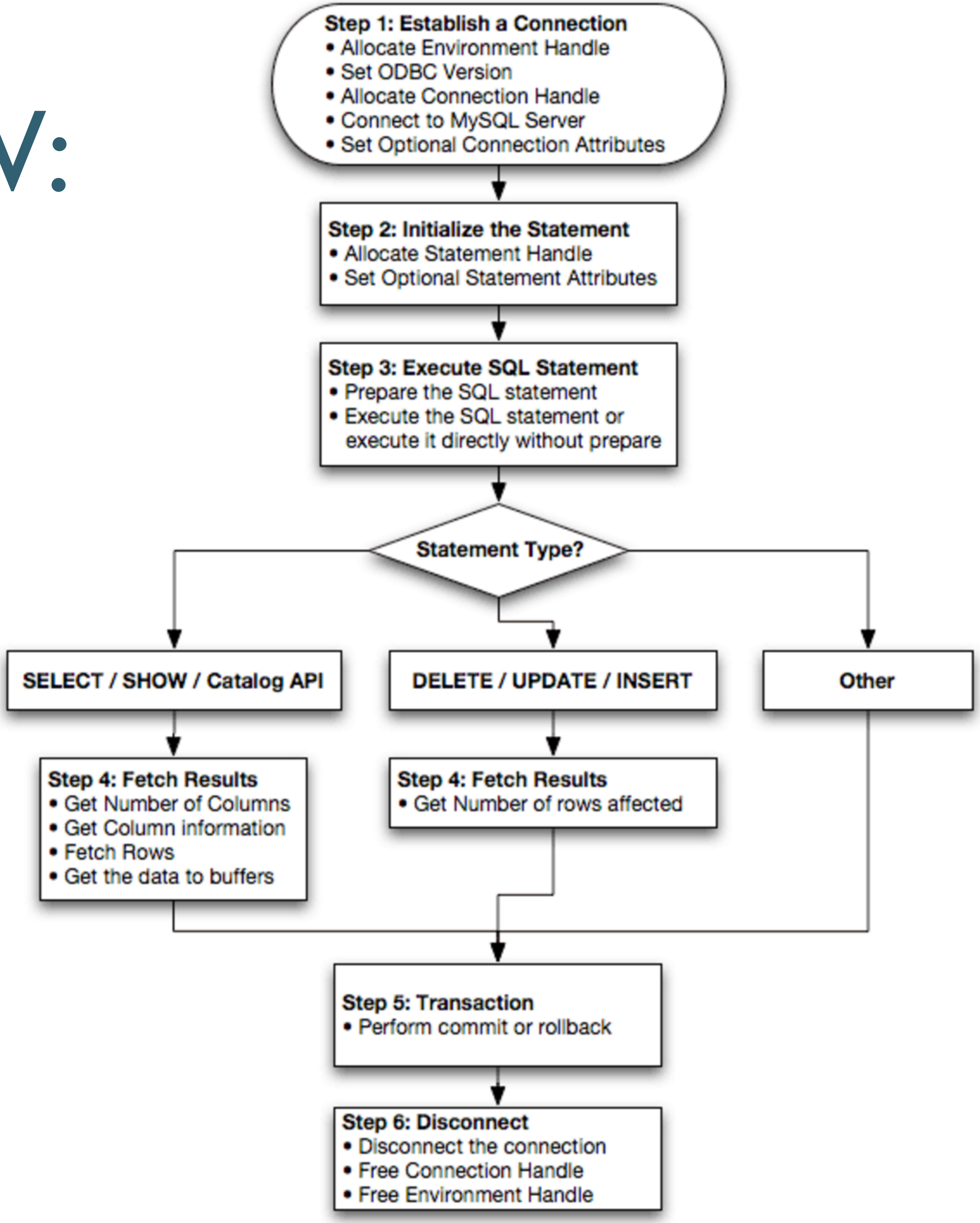
- **Schema Design:** You will have/already had a dedicated class on DB design principles don’t worry

3. **Use indexes *when appropriate***:
    - They take more storage and update costs more
    - Multi column Vs. Single column: It depends on the query (‘Or’ vs. ‘And’)
    - For full text search use a reverse index.
    - Rebuild indexes after your DB is stable.
4. **Choose a storage engine**
5. **Use correct data types**:
    - Smallest as possible to minimize disk space
6. **Use “NOT NULL” as often as possible**
    - Enabling better use of indexes and eliminating overhead for testing whether each value is NULL
7. **Normalization ?**(Avoiding redundant data by using unique IDs)
    - To save disk space, do it. For fast retrieval: Don’t.

AGENDA FOR TODAY

- The final project
- Advanced Mysql
- **Database programming**
- Recap: DB servers in the web
- Web programming architecture
- HTTP on a need-to-know basis.
- How to use web APIs

WORKFLOW:



Step 1: Establish a Connection
- Allocate Environment Handle
- Set ODBC Version
- Allocate Connection Handle
- Connect to MySQL Server
- Set Optional Connection Attributes

Step 2: Initialize the Statement
- Allocate Statement Handle
- Set Optional Statement Attributes

Step 3: Execute SQL Statement
- Prepare the SQL statement
- Execute the SQL statement or execute it directly without prepare

Statement Type?

SELECT / SHOW / Catalog API

DELETE / UPDATE / INSERT

Other

Step 4: Fetch Results
- Get Number of Columns
- Get Column information
- Fetch Rows
- Get the data to buffers

Step 4: Fetch Results
- Get Number of rows affected

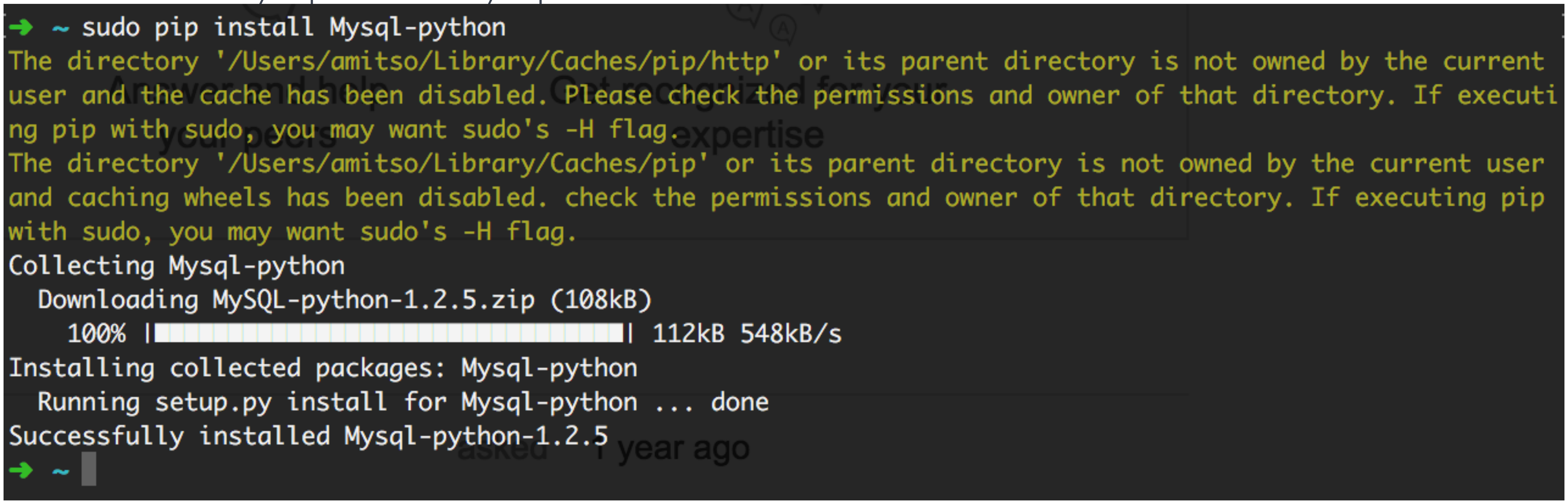
Step 5: Transaction
- Perform commit or rollback

Step 6: Disconnect
- Disconnect the connection
- Free Connection Handle
- Free Environment Handle

DB PROGRAMMING HELLOWORLD

Using a mysqlDB (python 2.7x) or MySQLClient (python 3.x)

- Install mysqlDB or mysqlclient via PIP

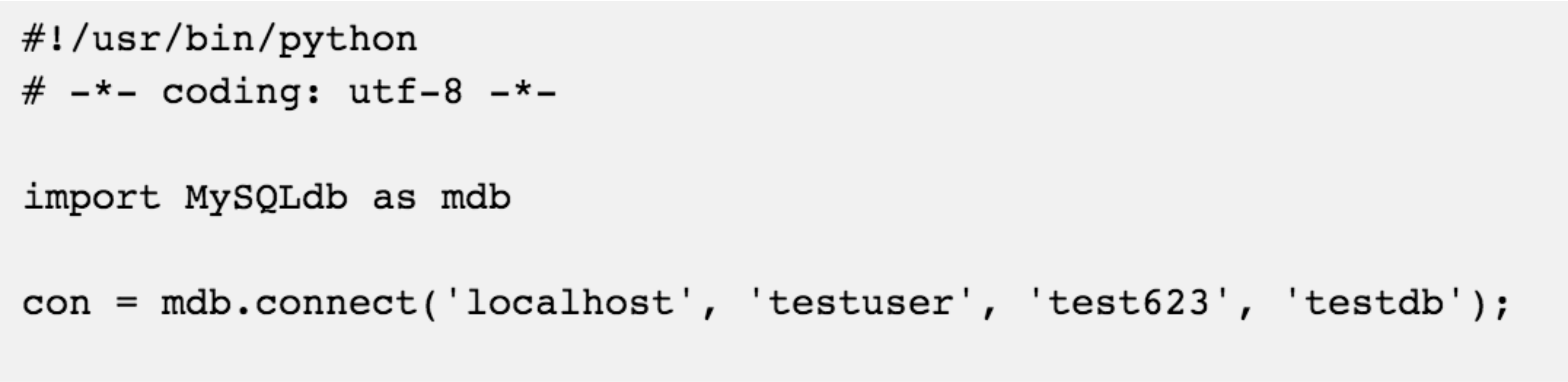


```
➜ ~ sudo pip install Mysql-python
The directory '/Users/amitso/Library/Caches/pip/http' or its parent directory is not owned by the current
user and the cache has been disabled. Please check the permissions and owner of that directory. If executi
ng pip with sudo, you may want sudo's -H flag.
The directory '/Users/amitso/Library/Caches/pip' or its parent directory is not owned by the current user
and caching wheels has been disabled. check the permissions and owner of that directory. If executing pip
with sudo, you may want sudo's -H flag.
Collecting Mysql-python
  Downloading MySQL-python-1.2.5.zip (108kB)
    100% |████████████████████████████████| 112kB 548kB/s
Installing collected packages: Mysql-python
  Running setup.py install for Mysql-python ... done
Successfully installed Mysql-python-1.2.5
➜ ~ 
```

1.THE CONNECTOR

Using a mysqlDB (python 2.7x) or mysqlclient (python 3.x)

- In your Python script:
  1. Import MySQLdb
  2. Create a connector to the DB with: server name, user, password , DB name



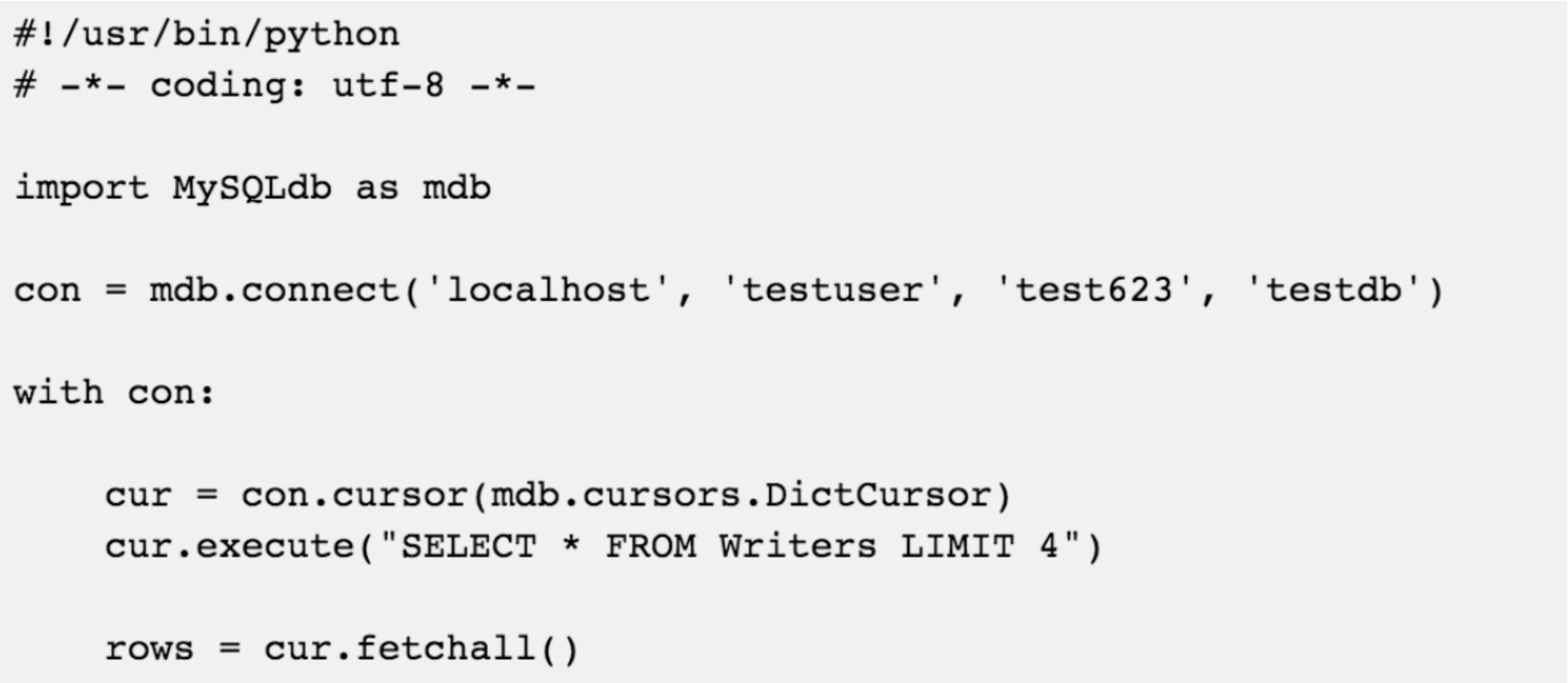
```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb');
```

2. THE CURSOR

Using a mysqlDB (python 2.7x) or mysqlclient (python 3.x)

- In your Python script:

1. Create a **cursor** (*cur = con.cursor()*)
2. **Execute** a query (*cur.execute("<YOURSQL_QUERY>")*)
3. **Fetch** the rows in the results (*rows=cur.fetchall()*)

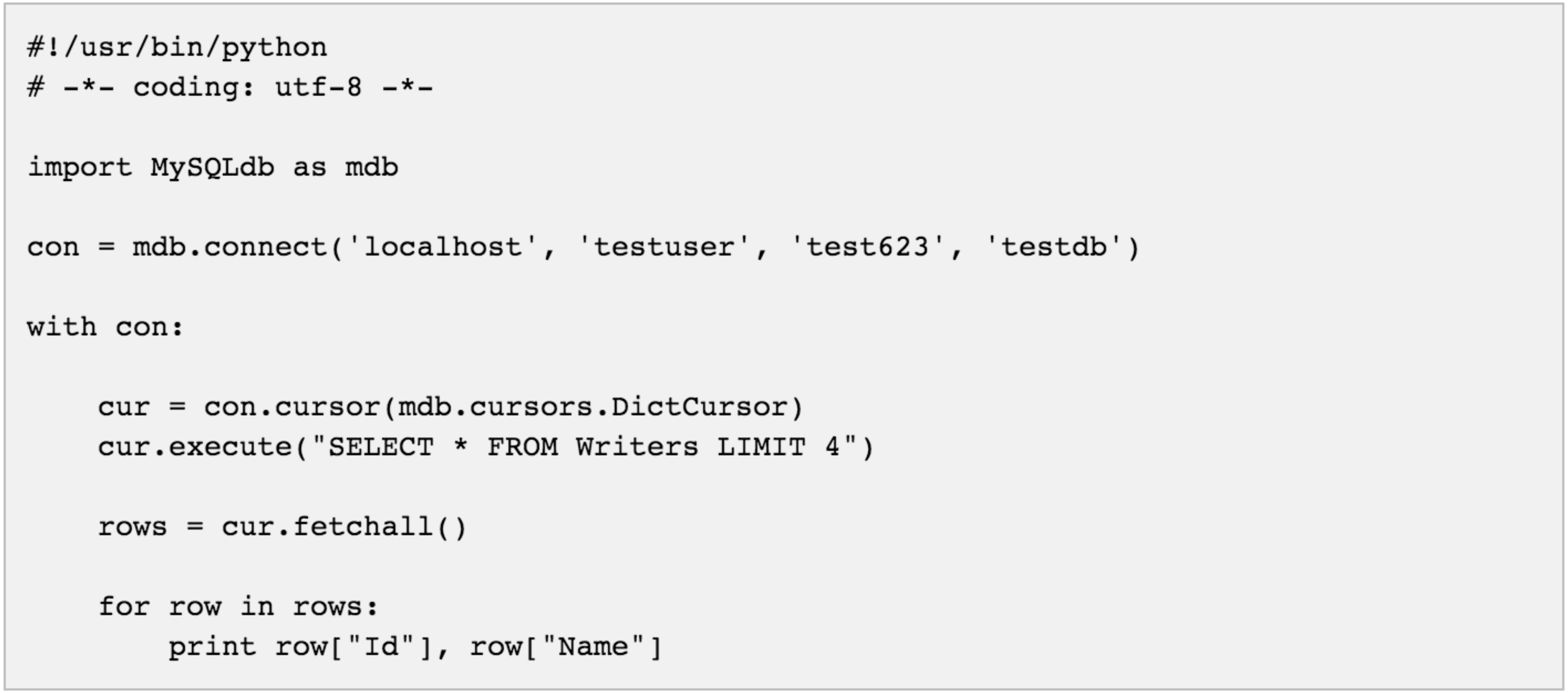


```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb')

with con:

    cur = con.cursor(mdb.cursors.DictCursor)
    cur.execute("SELECT * FROM Writers LIMIT 4")

    rows = cur.fetchall()
```

3.1 FETCH ALL

Using a mysqDB (python 2.7x) or mysqlclient (python 3.x)

- In your Python script:
    1. Working the results:
        1. Reference by position (*row[0], row[1]*)
        2. *Reference by column name (row["Id"], row["Name"])*

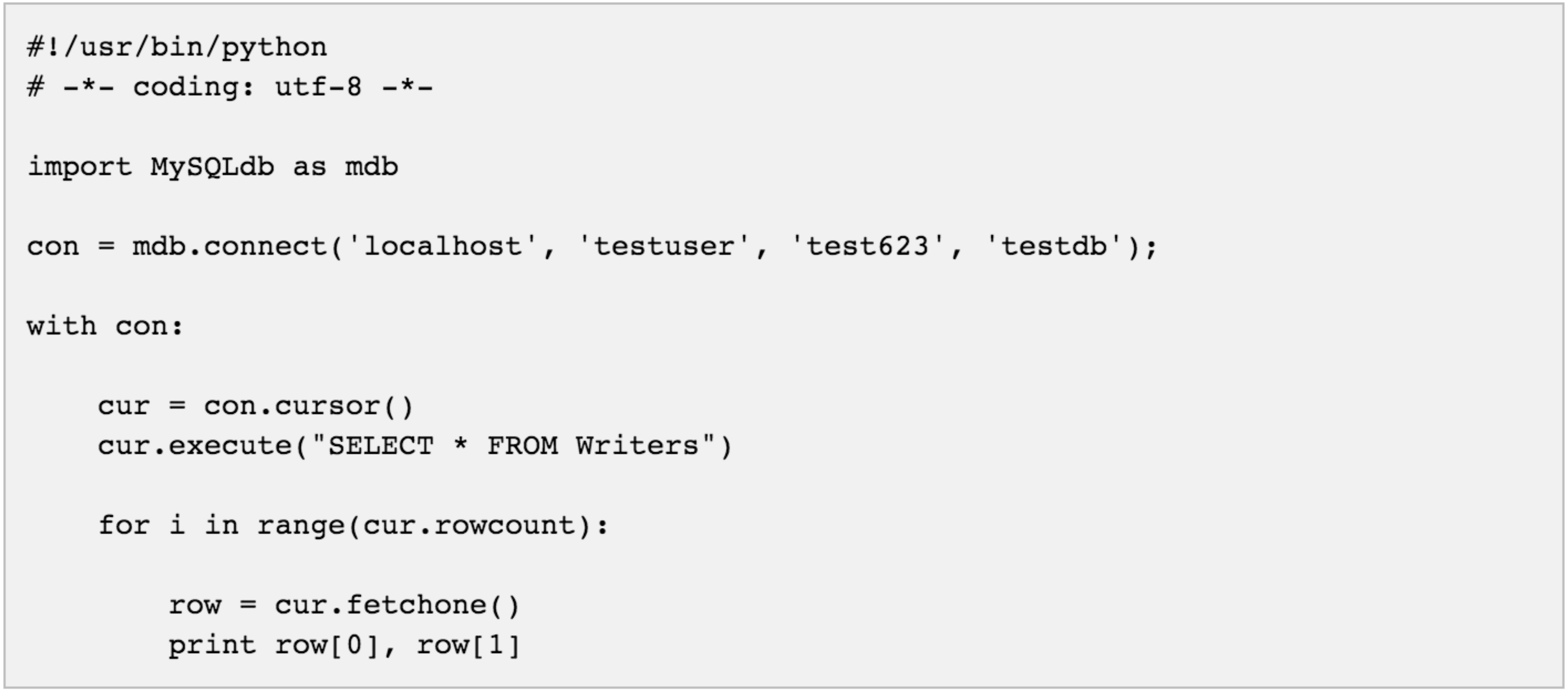


```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb')

with con:

    cur = con.cursor(mdb.cursors.DictCursor)
    cur.execute("SELECT * FROM Writers LIMIT 4")

    rows = cur.fetchall()

    for row in rows:
        print row["Id"], row["Name"]
```

3.2 FETCH ONE

Using a mysqDB (python 2.7x) or mysqlclient (python 3.x)

- In your Python script:
    1. Fetching row by row:
        1. After execution get the number of results (*cur.rowcount*)
        2. In a FOR loop: Use *fetchone()* to get one row at a time.



```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb');

with con:

    cur = con.cursor()
    cur.execute("SELECT * FROM Writers")

    for i in range(cur.rowcount):

        row = cur.fetchone()
        print row[0], row[1]
```

4. ADDING USER INPUT

Using a mysqDB (python 2.7x) or mysqlclient (python 3.x)

- In your Python script:
  1. Working with user input: with regular string manipulation

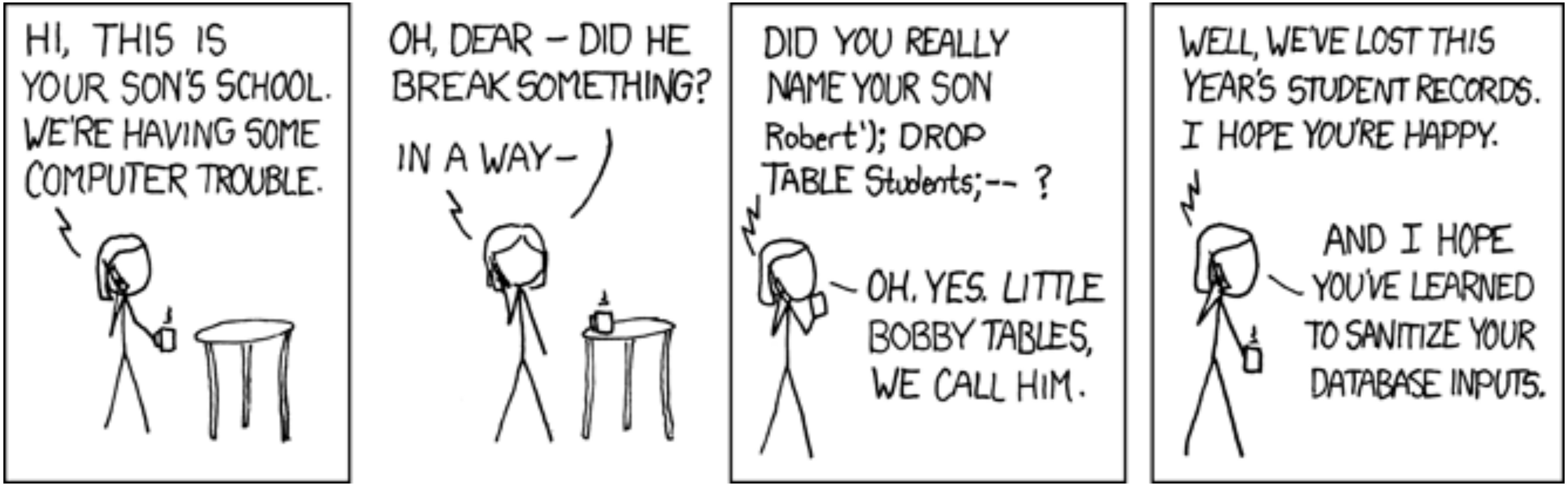
```
>>> student_name = raw_input(“Enter a student name”)
>>> query=“SELECT * from Students WHERE FirstName = %s” % (student_name)
>>> Cur.execute(query)
```

LITTLE BOBBY TABLES

```
>>> student_name = raw_input(“Robert’; DROP TABLE Students; --”)

>>> query=“SELECT * from Students WHERE FirstName = ‘%s’ ” %
 (student_name)

>>> Cur.execute(query)
```

LITTLE BOBBY TABLES

```
>>> student_name = raw_input(“Robert’; DROP TABLE Students; --”)

>>> query=“SELECT * from Students WHERE FirstName = ‘%s’ ” %
 (student_name)

>>> Cur.execute(query)
```


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

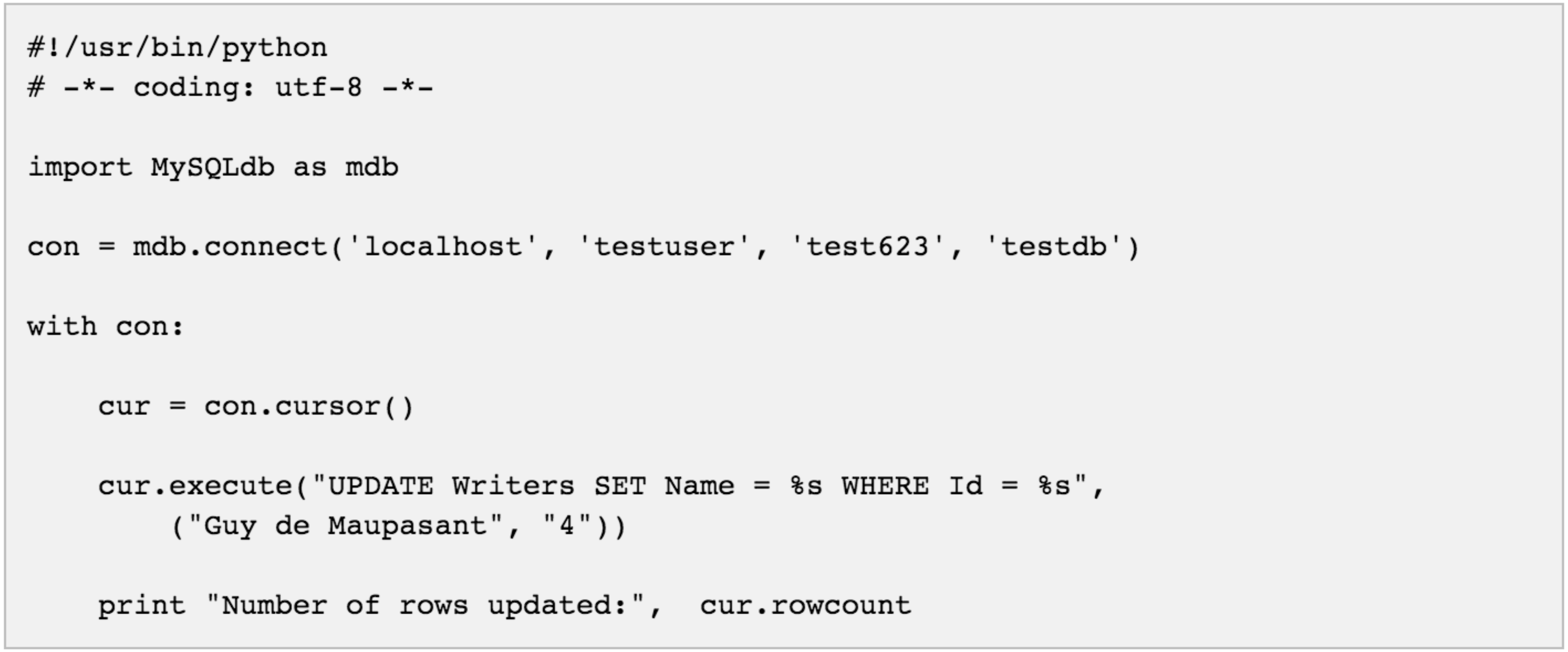
5. PREPARED STATEMENT

Using a mysqDB (python 2.7x) or mysqlclient (python 3.x)

- In your Python script:

  1. Using a "**Prepared Statement**" to:

     - Prevents the reparsing of SQL statements
     - Used for statements executed more than once

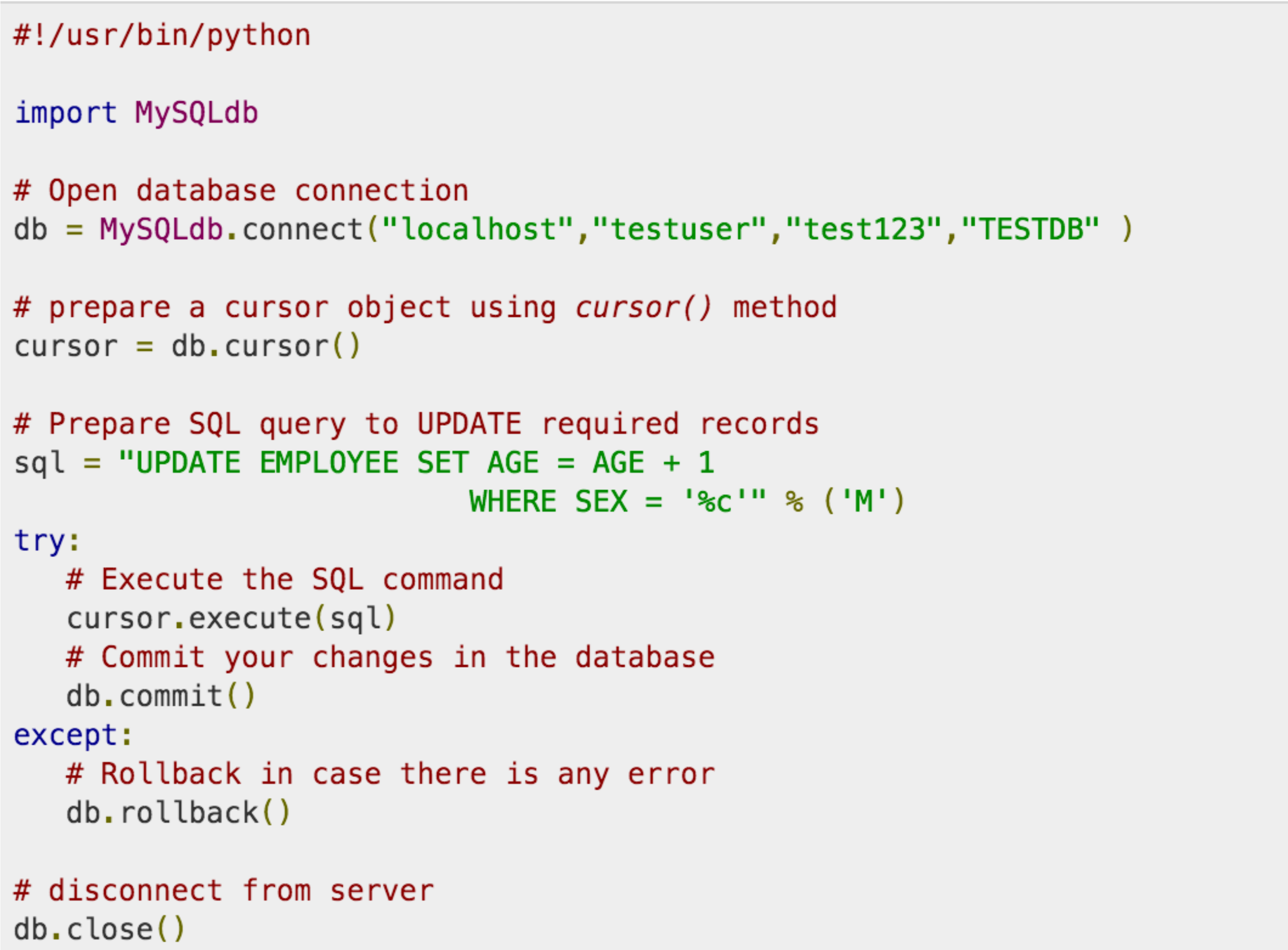


```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb')

with con:

    cur = con.cursor()

    cur.execute("UPDATE Writers SET Name = %s WHERE Id = %s",
        ("Guy de Maupasant", "4"))

    print "Number of rows updated:",  cur.rowcount
```

6.C/U/D OPERATIONS

Performing C U D operations:

- Commit() if everything went well
- Rollback() if there is something wrong

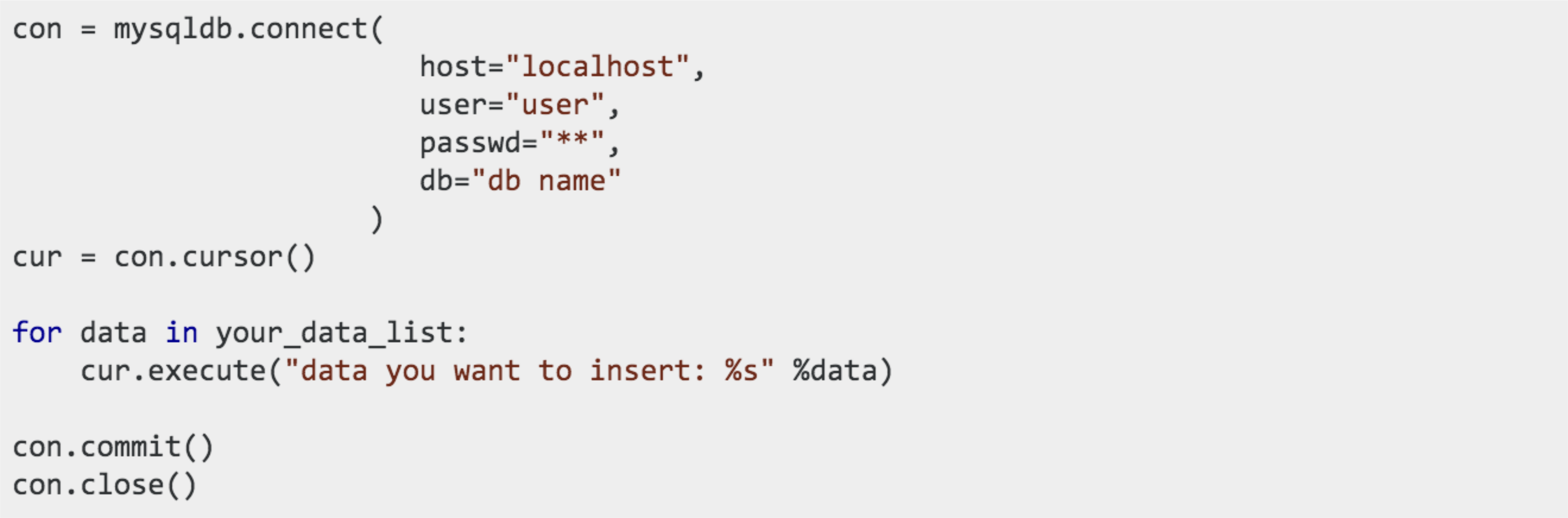


```
#!/usr/bin/python

import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to UPDATE required records
sql = "UPDATE EMPLOYEE SET AGE = AGE + 1
                          WHERE SEX = '%c'" % ('M')
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()

# disconnect from server
db.close()
```

6.1. BATCHED C/U/D OPS.

Performing C U D operations:

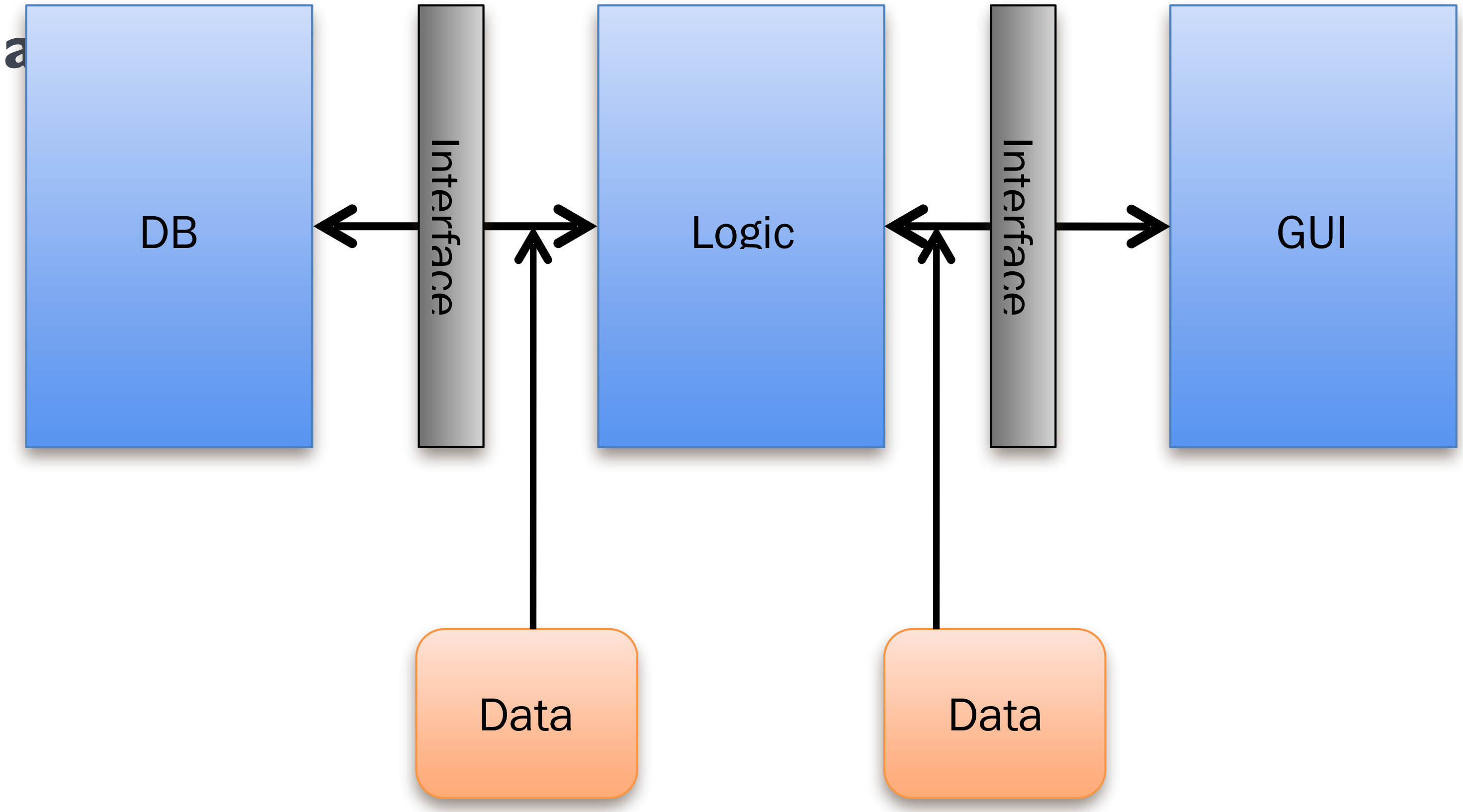
- Using **Batch** CUD operations to boost performance:
  - If it not fast enough, auto-commit might be **ON.**
  - **Add “SET autocommit 0;” to your SQL transaction.**



```
con = mysqldb.connect(
                        host="localhost",
                        user="user",
                        passwd="**",
                        db="db name"
                     )
cur = con.cursor()

for data in your_data_list:
    cur.execute("data you want to insert: %s" %data)

con.commit()
con.close()
```

DB PROGRAMING: GUIDELINES

1. **Use efficient SQL statements**:
   - “SELECT * FROM Students” vs “ SELECT `FirstName`,`LastName` FROM Students”
2. **Secure your code**
   - Prepared statements
   - Input sanitation.
   - Define MySQL users correctly
3. **Separate the DB layer from the UI la**

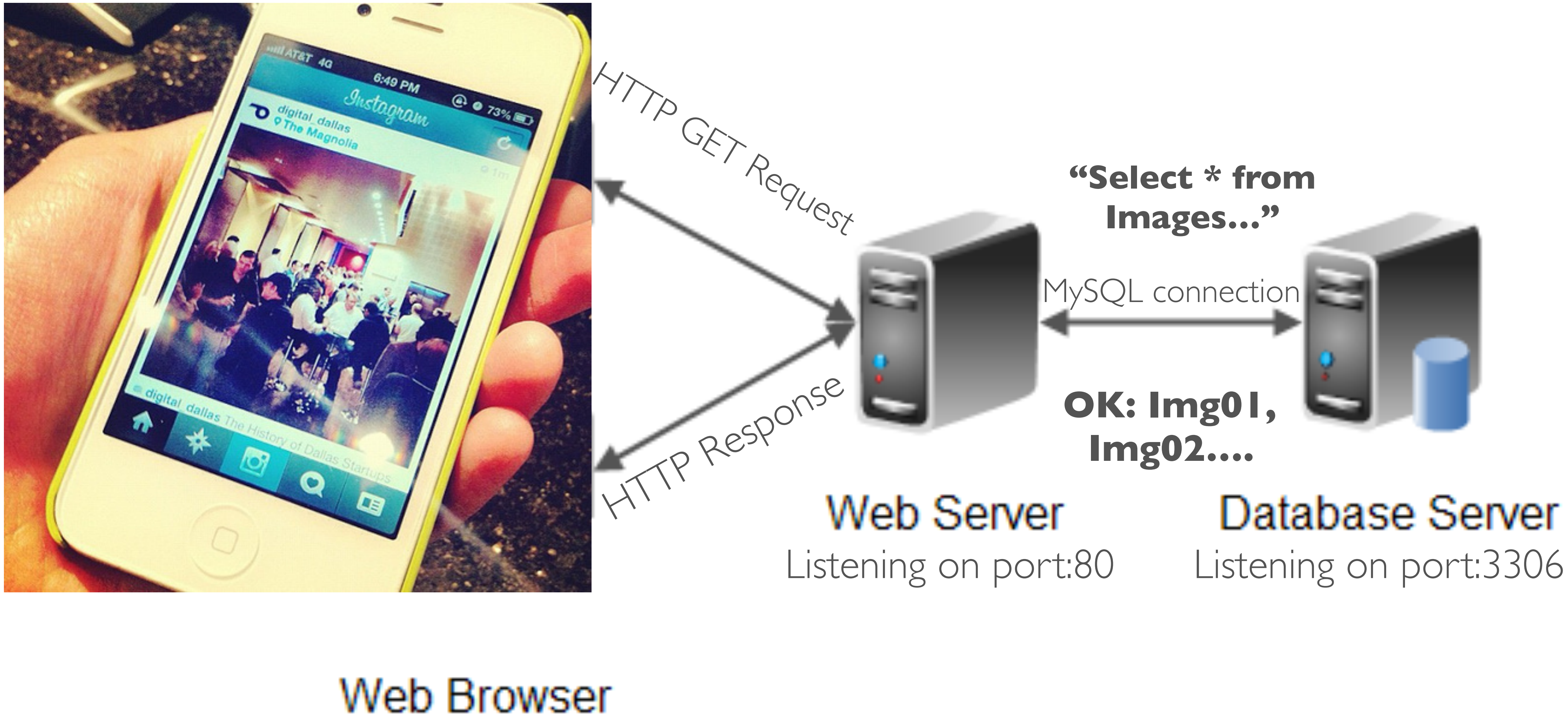


AGENDA FOR TODAY

- The final project
- Advanced Mysql
- Database programming
- **Recap: DB servers in the web**
- **Web programming architecture**
- HTTP on a need-to-know basis.
- How to use web APIs

DATABASE ARCHITECTURE ON THE WEB (NETWORK)

- Web browser and web server are communicating via the **HTTP protocol**.
- Web servers (and MySQL clients) are communicating via the **MySQL protocol** (TCP)



HTTP GET Request

HTTP Response

"Select * from Images..."

MySQL connection

OK: Img01, Img02....

Web Server

Listening on port:80

Database Server

Listening on port:3306

Web Browser

WEB PROGRAMMING: DEFINITIONS

- **Web Server:**

  A computer program that accepts HTTP requests and return HTTP responses with optional data content.

  A computer that runs a computer program as described above.

  Most common platforms: **Apache**, **IIS (Microsoft), Enginex**

- **Web Client (browser):**

  A software application for retrieving, presenting, and traversing information resources on the World Wide Web

  Usually parses HTML (HyperText Markup Language) , CSS and JavaScript and present it to the user as a **web page**. (More details on the next recitation).

  Most common browser: **Firefox, Google Chrome, Ms Internet Explorer, Safari**

- **Web API (Application Programming Interface):**

  A publicly exposed endpoint to a defined request-response message system, (typically expressed in JSON or XML)

WEB PROGRAMMING: DEFINITIONS

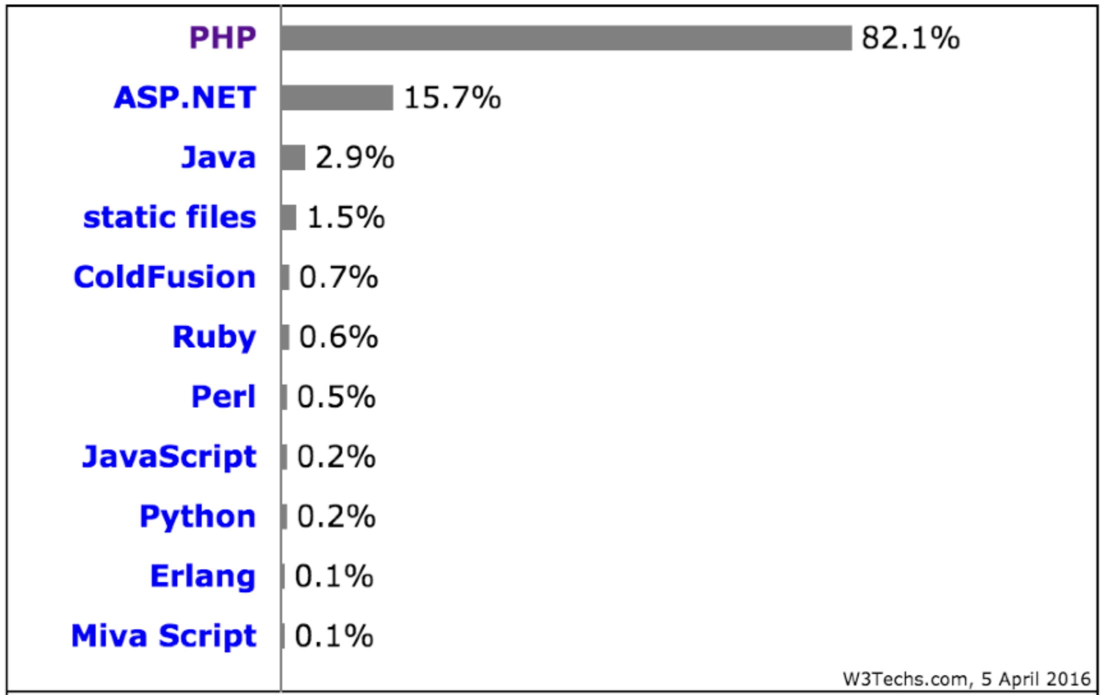
Web Server programming language:

A server-side programming language for executing code that reads HTTP requests and generates HTTP responses.

Designed for the web architecture:

- Multiple clients accessing a web server on the same
- Content is dynamic

Most programming languages can handle HTTP requests (e.g., C, C++, Python, Java etc.)



W3Techs.com, 5 April 2016

Percentages of websites using various server-side programming languages
Note: a website may use more than one server-side programming language

AGENDA FOR TODAY

- The final project
- Advanced Mysql
- Database programming
- Recap: DB servers in the web
- Web programming architecture
- **HTTP on a need-to-know basis.**
- How to use web APIs

INTRO TO HTTP

HTTP (Hyper Text Transfer Protocol)

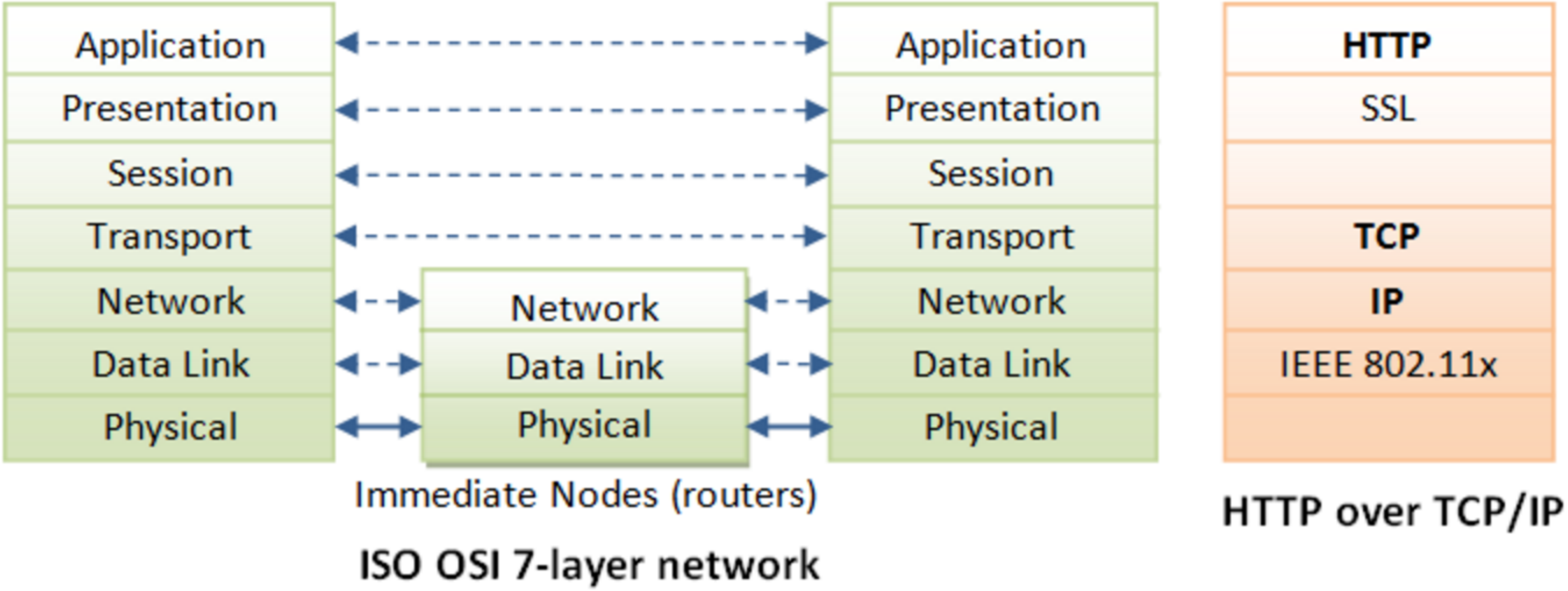
An **application layer** protocol

Hyper Text: A text displayed on a computer display or other electronic devices with references (hyperlinks) to other text which the reader can immediately access, or where text can be revealed progressively at multiple levels of detail

Based on **Client Requests** of **Resources (URI)** and **Server Response**

Resources to be accessed by HTTP are identified using Uniform Resource Identifiers (URIs).

Can be referring to web pages, media (image/video) or other data objects.



ISO OSI 7-layer network

HTTP over TCP/IP

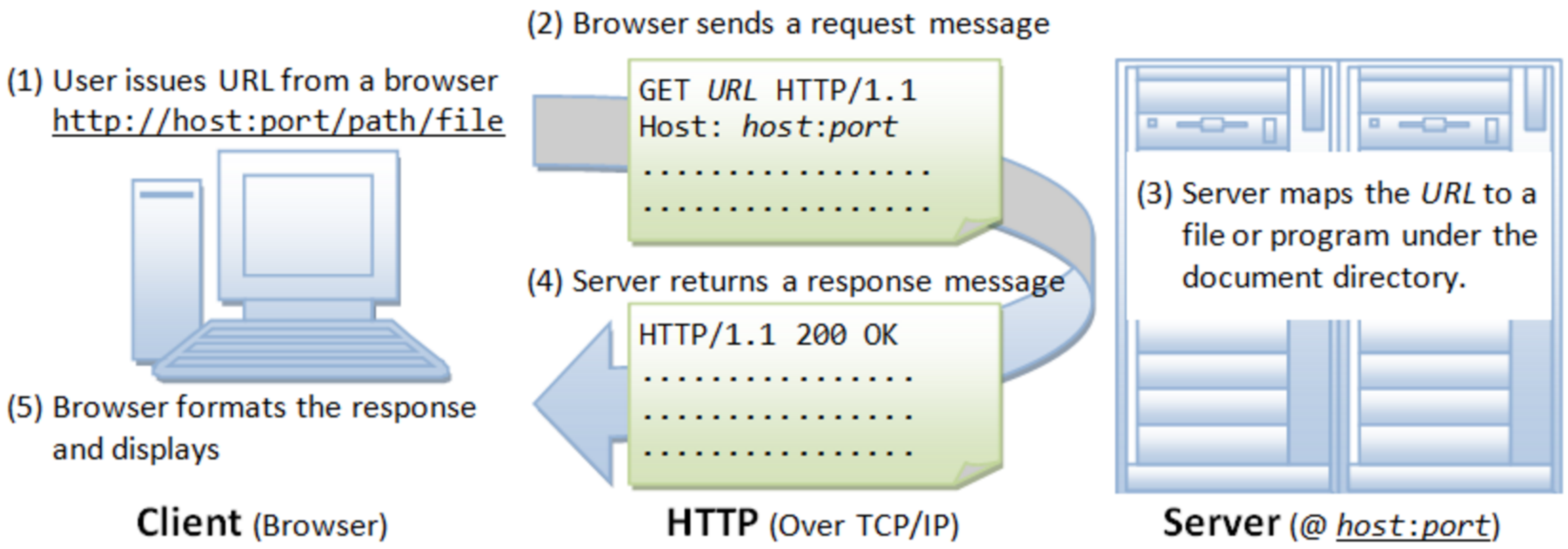
INTRO TO HTTP

HTTP Session

An HTTP client initiates a request by establishing a Transmission Control Protocol (TCP) connection to a particular port on a server (typically port 80,)

An HTTP server listening on that port waits for a client's request message.

Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own.



(1) User issues URL from a browser
http://host:port/path/file

(2) Browser sends a request message

```
GET URL HTTP/1.1
Host: host:port
.................
.................
```

(3) Server maps the *URL* to a file or program under the document directory.

(4) Server returns a response message

```
HTTP/1.1 200 OK
................
................
................
```

(5) Browser formats the response and displays

INTRO TO HTTP

HTTP Requests

Most common client requests are **HTTP GET** and **HTTP POST**

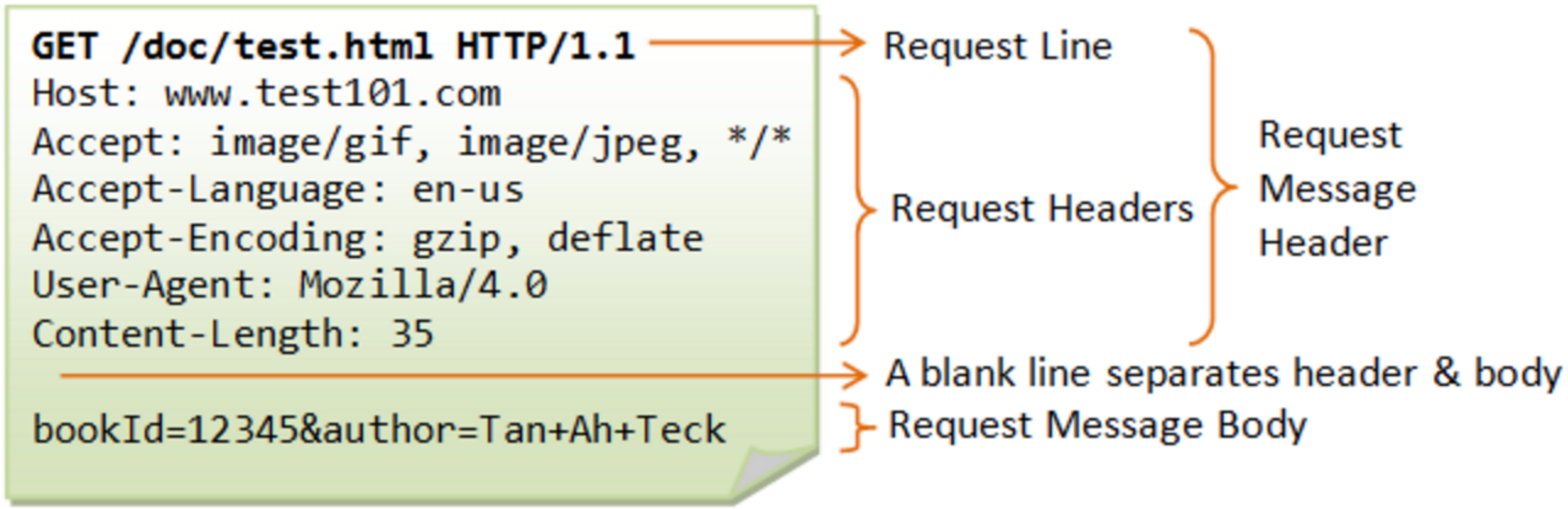
HTTP GET can transfer parameters within the URL

Example: **https://www.google.co.il/?q=database+systems**

HTTP POST is used to post data up to the web server

HTTP Request headers

Used to pass information to the web server such as language, supported encoding, User-Agent, etc.



```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

INTRO TO HTTP

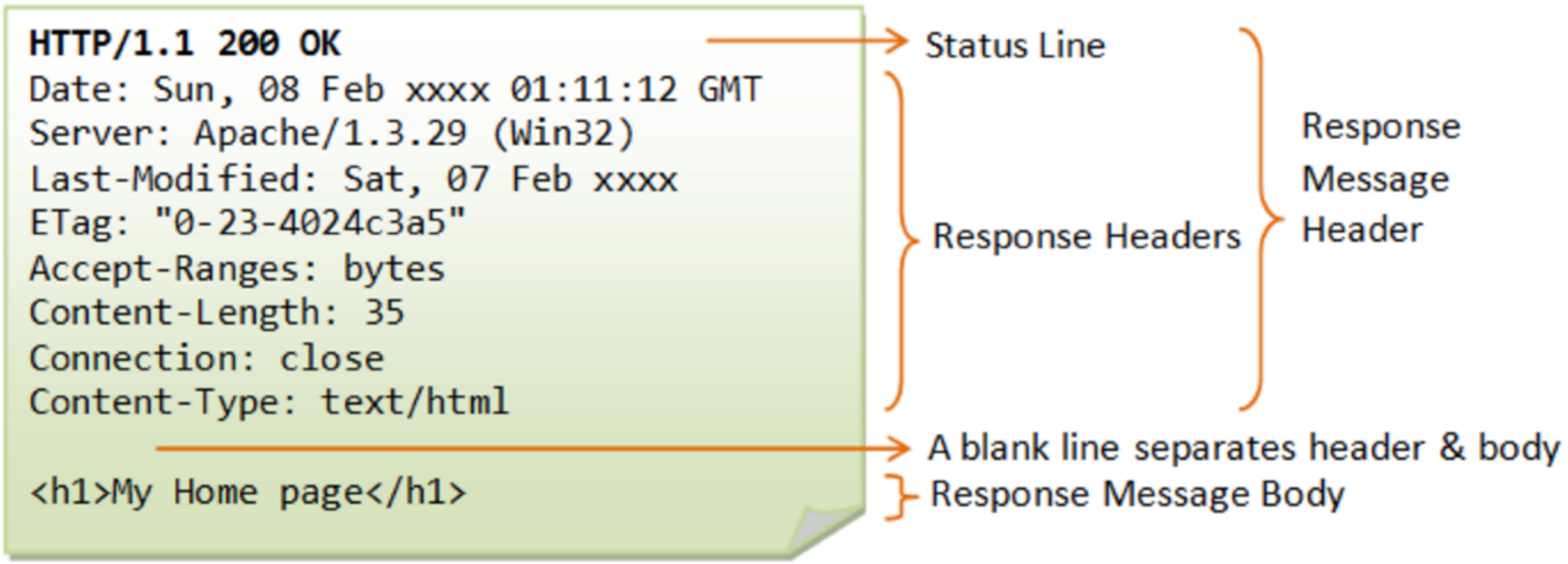
HTTP Response

The first line is called the status line, followed by optional response header(s).

The status line has the following syntax:

- HTTP-version status-code reason-phrase
- HTTP-version: The HTTP version used in this session. Either HTTP/1.0 and HTTP/1.1.
- status-code: a 3-digit number generated by the server to reflect the outcome of the request.
- reason-phrase: gives a short explanation to the status code.

Common status code and reason phrase are "200 OK", "404 Not Found", "403 Forbidden", "500 Internal Server Error".



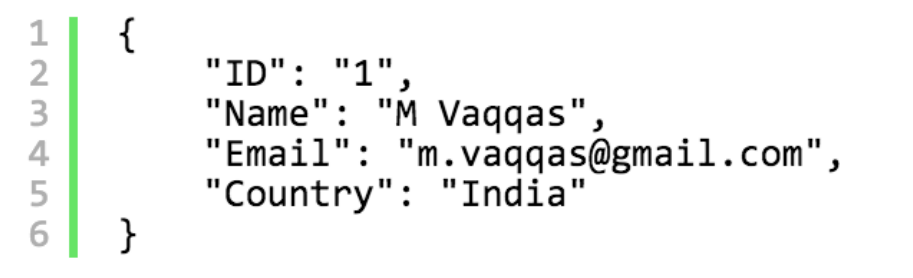
```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

AGENDA FOR TODAY

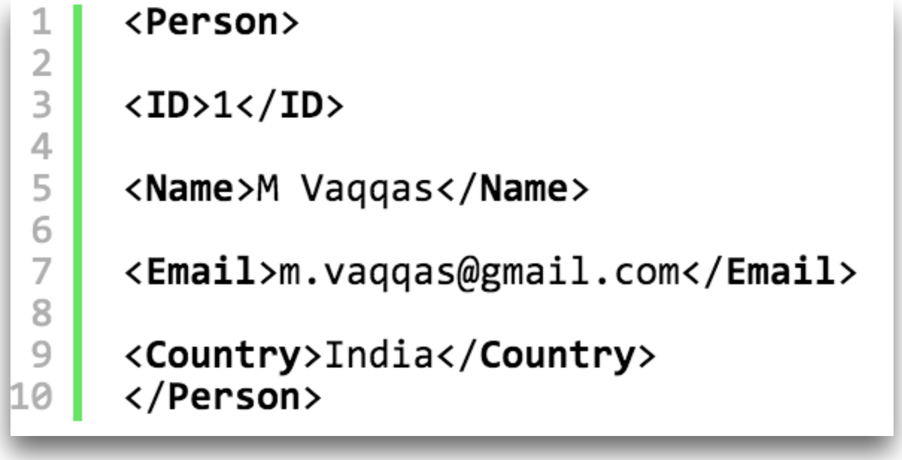
- The final project
- Advanced Mysql
- Database programming
- Recap: DB servers in the web
- Web programming architecture
- HTTP on a need-to-know basis.
- **How to use web APIs**

WEB SERVICES

- **A web service is like a website but is *__structured__*.**
- **It is for programs, not for humans.**
- **RESTful: REpresentational State Transter (ful)**
- **REST APIs have the following characteristics:**
  - **Representations:** which are the objects like in OOP
  - **Messages:** the client and the servers are sending messages to each other
  - **Stateless:** Like the internet. REST is stateless.
  - **Links between resources:** Same as in URI and URLs.
- **The response message will be in JSON or XML**



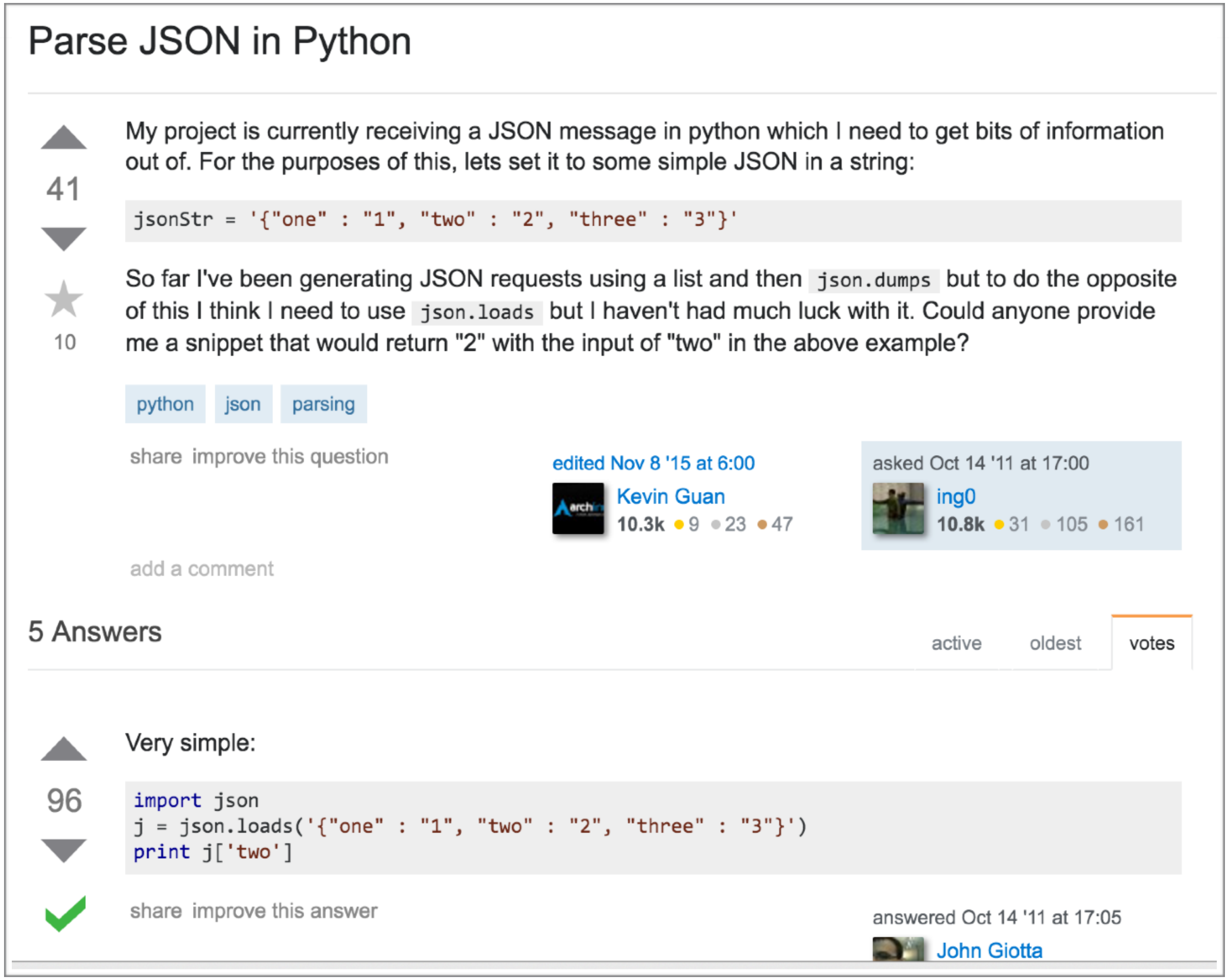
```
{
    "ID": "1",
    "Name": "M Vaqqas",
    "Email": "m.vaqqas@gmail.com",
    "Country": "India"
}
```



```
<Person>

<ID>1</ID>

<Name>M Vaqqas</Name>

<Email>m.vaqqas@gmail.com</Email>

<Country>India</Country>
</Person>
```

STACKEXCHANGE API

- **Q&A platform , one of its known instances is stack overflow**



Parse JSON in Python

41

10

My project is currently receiving a JSON message in python which I need to get bits of information out of. For the purposes of this, lets set it to some simple JSON in a string:

```
jsonStr = '{"one" : "1", "two" : "2", "three" : "3"}'
```

So far I've been generating JSON requests using a list and then `json.dumps` but to do the opposite of this I think I need to use `json.loads` but I haven't had much luck with it. Could anyone provide me a snippet that would return "2" with the input of "two" in the above example?

python json parsing

share improve this question

edited Nov 8 '15 at 6:00
Kevin Guan
10.3k ● 9 ● 23 ● 47

asked Oct 14 '11 at 17:00
ing0
10.8k ● 31 ● 105 ● 161

add a comment

5 Answers

active oldest votes

96

✓

Very simple:

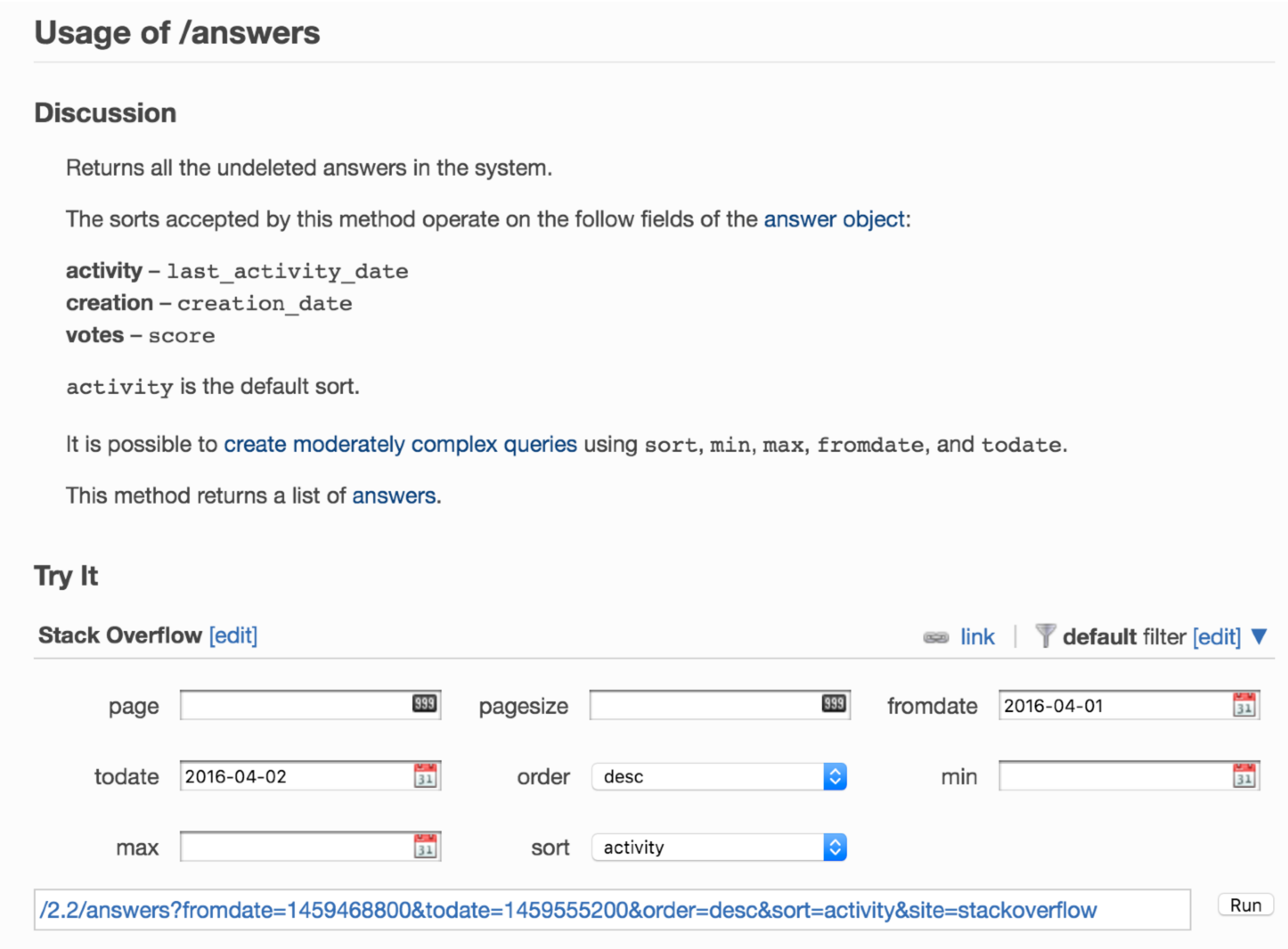
```
import json
j = json.loads('{"one" : "1", "two" : "2", "three" : "3"}')
print j['two']
```

share improve this answer

answered Oct 14 '11 at 17:05
John Giotta

STACKEXCHANGE API

- **Stack exchange API example :**



Usage of /answers

Discussion

Returns all the undeleted answers in the system.

The sorts accepted by this method operate on the follow fields of the answer object:

activity – `last_activity_date`
creation – `creation_date`
votes – `score`

`activity` is the default sort.

It is possible to create moderately complex queries using `sort`, `min`, `max`, `fromdate`, and `todate`.

This method returns a list of answers.

Try It

Stack Overflow [edit]

link | **default** filter [edit] ▼

page

pagesize

fromdate 2016-04-01

todate 2016-04-02

order desc

min

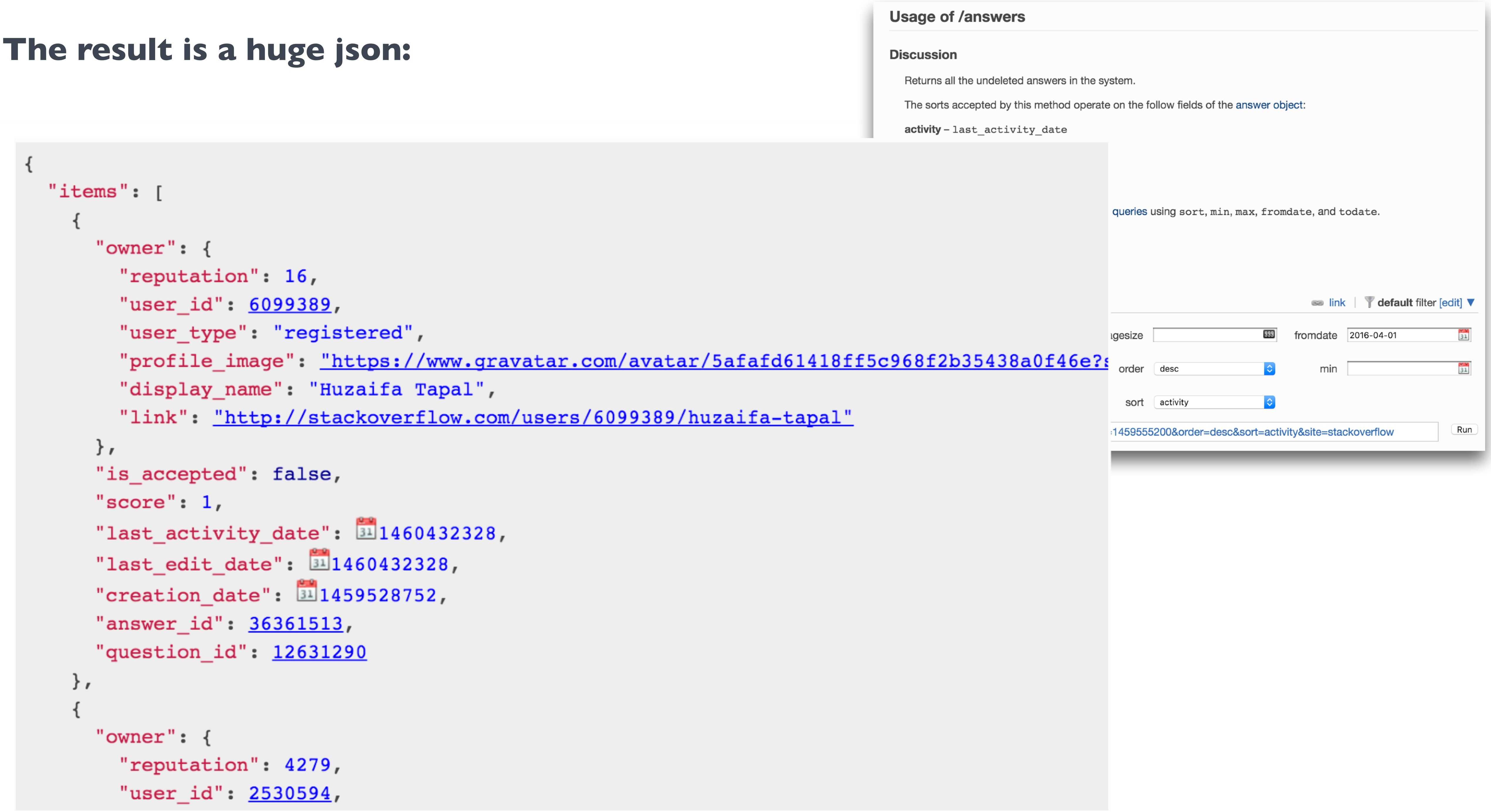
max

sort activity

/2.2/answers?fromdate=1459468800&todate=1459555200&order=desc&sort=activity&site=stackoverflow

STACKEXCHANGE API

- **The result is a huge json:**



Usage of /answers

Discussion

Returns all the undeleted answers in the system.

The sorts accepted by this method operate on the follow fields of the answer object:

activity – `last_activity_date`

queries using `sort`, `min`, `max`, `fromdate`, and `todate`.

fromdate: 2016-04-01

order: desc

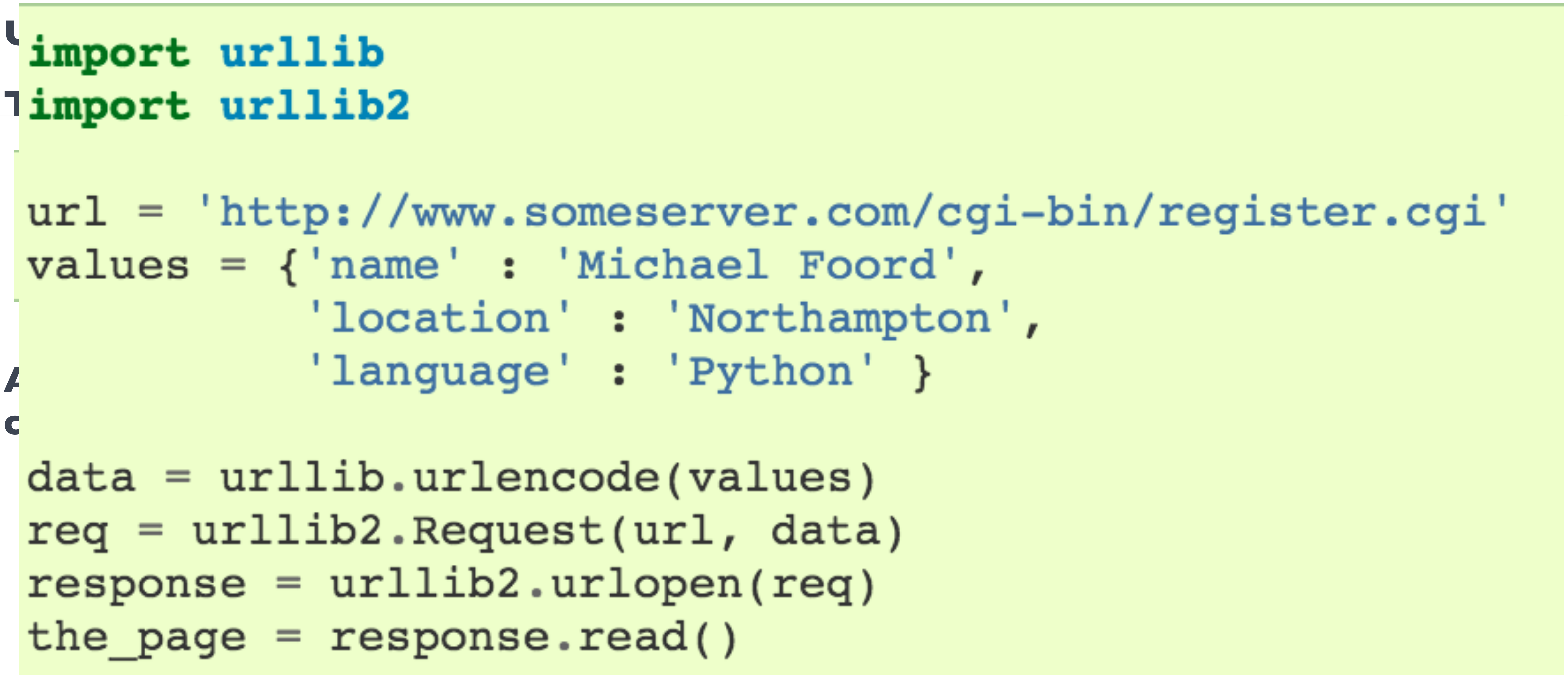
sort: activity

:1459555200&order=desc&sort=activity&site=stackoverflow

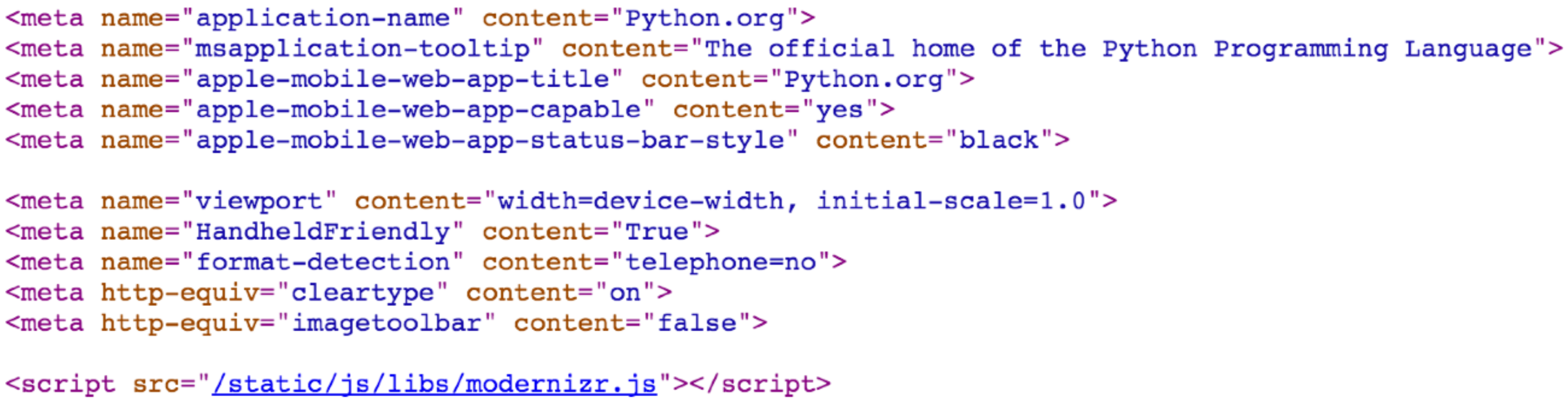
```
{
  "items": [
    {
      "owner": {
        "reputation": 16,
        "user_id": 6099389,
        "user_type": "registered",
        "profile_image": "https://www.gravatar.com/avatar/5afafd61418ff5c968f2b35438a0f46e?s
        "display_name": "Huzaifa Tapal",
        "link": "http://stackoverflow.com/users/6099389/huzaifa-tapal"
      },
      "is_accepted": false,
      "score": 1,
      "last_activity_date": 1460432328,
      "last_edit_date": 1460432328,
      "creation_date": 1459528752,
      "answer_id": 36361513,
      "question_id": 12631290
    },
    {
      "owner": {
        "reputation": 4279,
        "user_id": 2530594,
```

USING PYTHON FOR WEB API

- U
- T
- A
  c



```
import urllib
import urllib2

url = 'http://www.someserver.com/cgi-bin/register.cgi'
values = {'name' : 'Michael Foord',
          'location' : 'Northampton',
          'language' : 'Python' }

data = urllib.urlencode(values)
req = urllib2.Request(url, data)
response = urllib2.urlopen(req)
the_page = response.read()
```



```
<meta name="application-name" content="Python.org">
<meta name="msapplication-tooltip" content="The official home of the Python Programming Language">
<meta name="apple-mobile-web-app-title" content="Python.org">
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-status-bar-style" content="black">

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="HandheldFriendly" content="True">
<meta name="format-detection" content="telephone=no">
<meta http-equiv="cleartype" content="on">
<meta http-equiv="imagetoolbar" content="false">

<script src="/static/js/libs/modernizr.js"></script>
```

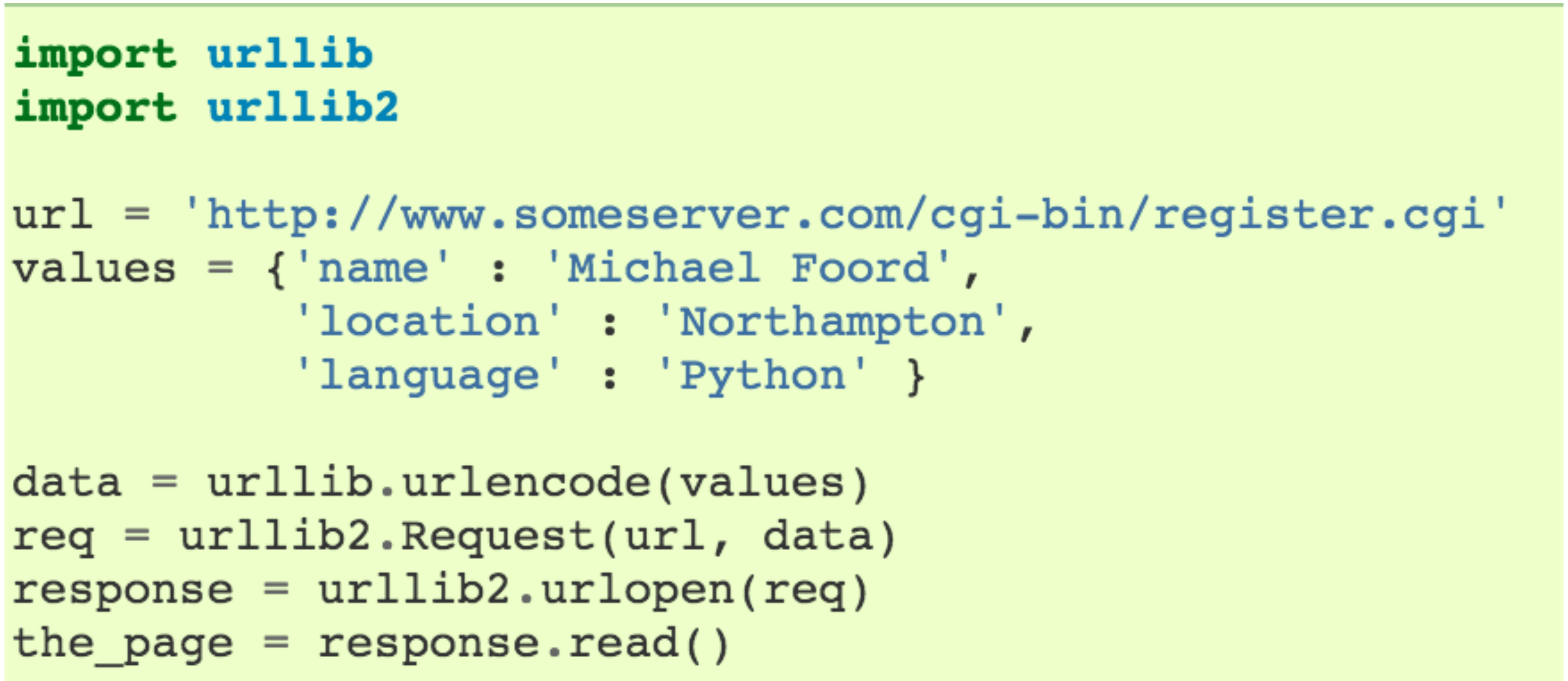
USING PYTHON FOR WEB API

- Using a “request” object, you can generate a post request:

•Create a dictionaries with variables and values

•Create a new Request object and load it with the URL and the dict.

•Execute the request via urlopen

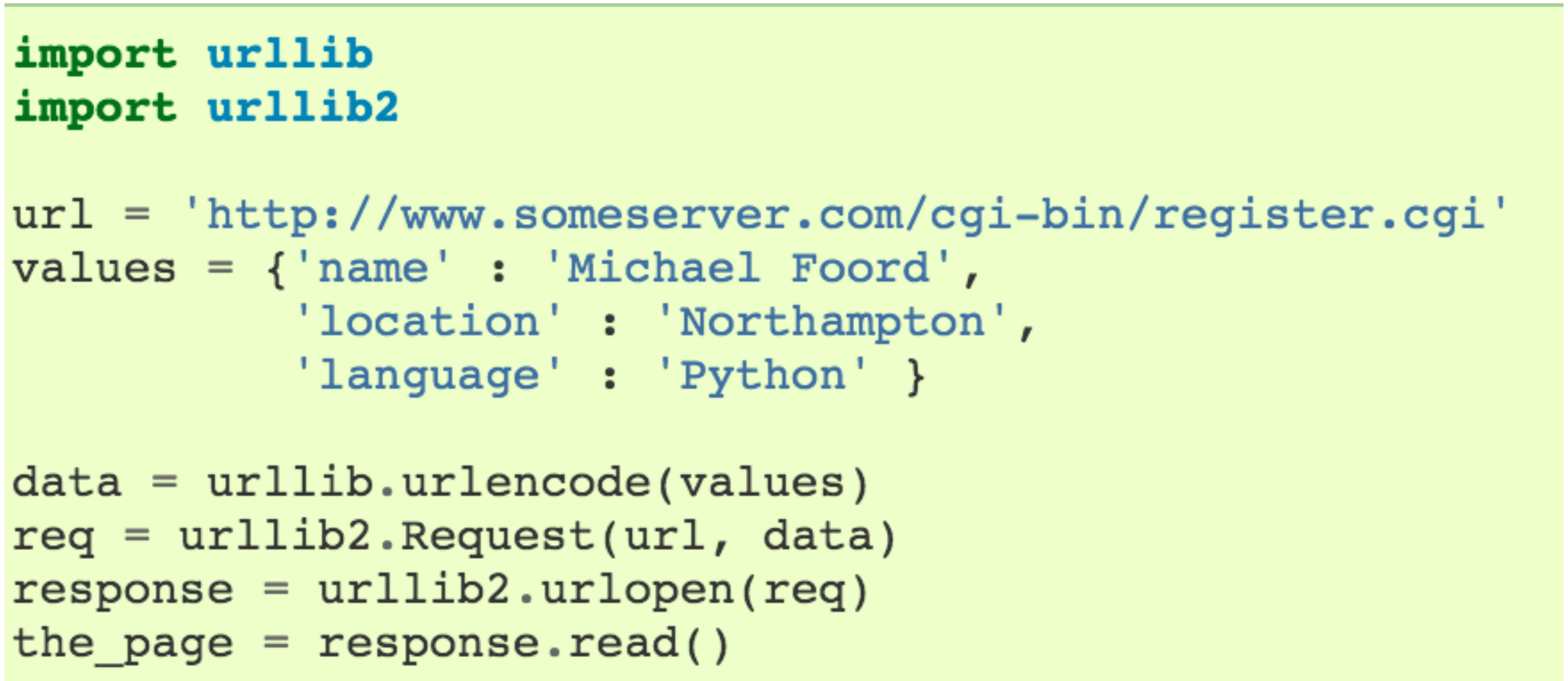


```
import urllib
import urllib2

url = 'http://www.someserver.com/cgi-bin/register.cgi'
values = {'name' : 'Michael Foord',
          'location' : 'Northampton',
          'language' : 'Python' }

data = urllib.urlencode(values)
req = urllib2.Request(url, data)
response = urllib2.urlopen(req)
the_page = response.read()
```

USING PYTHON FOR WEB API

Using a “request” object, you can generate a post request:

- Create a dictionaries with variables and values
- Create a new Request object and load it with the URL and the dict.
- Execute the request via urlopen

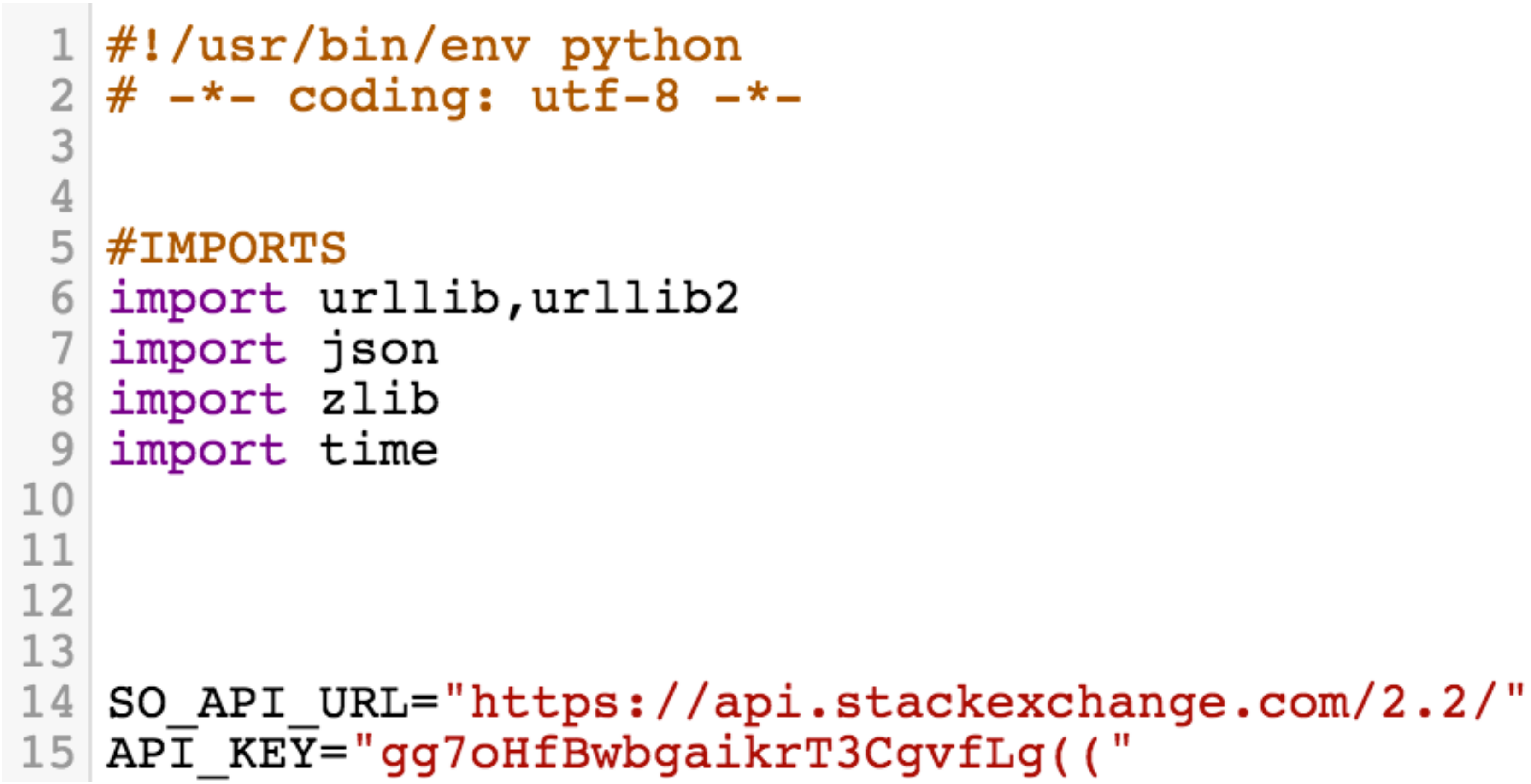


```
import urllib
import urllib2

url = 'http://www.someserver.com/cgi-bin/register.cgi'
values = {'name' : 'Michael Foord',
          'location' : 'Northampton',
          'language' : 'Python' }

data = urllib.urlencode(values)
req = urllib2.Request(url, data)
response = urllib2.urlopen(req)
the_page = response.read()
```

USING PYTHON FOR STACK EXCHANGE API

SETUP

- We will need to import libraries for HTTP handling, JSON handling and Zlib compression handling.
- Using the stack exchange API key we get more quota.

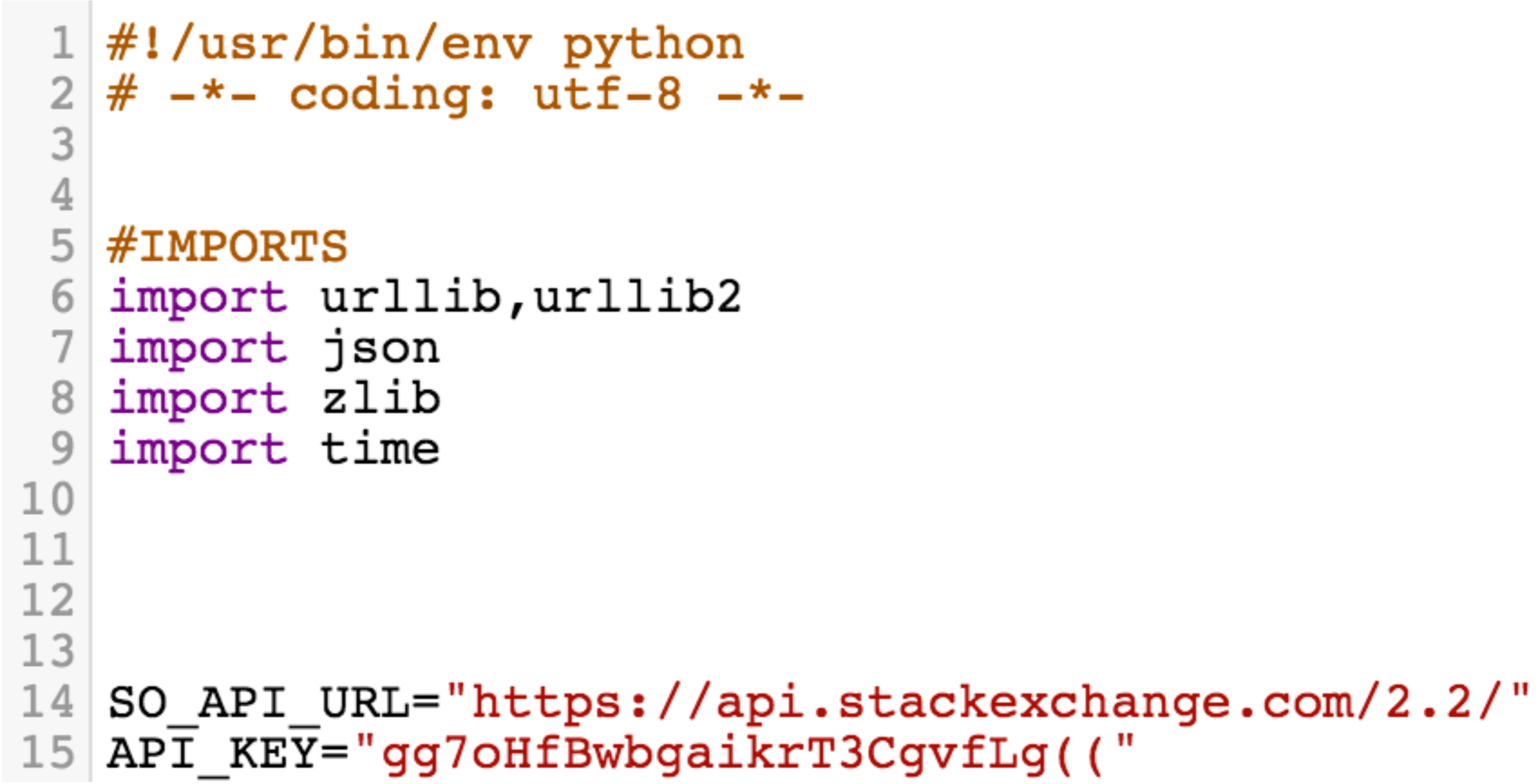


```
#!/usr/bin/env python
# -*- coding: utf-8 -*-


#IMPORTS
import urllib,urllib2
import json
import zlib
import time




SO_API_URL="https://api.stackexchange.com/2.2/"
API_KEY="gg7oHfBwbgaikrT3CgvfLg(("
```

USING PYTHON FOR STACK EXCHANGE API

SETUP

- We will need to import libraries for HTTP handling, JSON handling and Zlib compression handling.
- Using the stack exchange API key we get more quota.

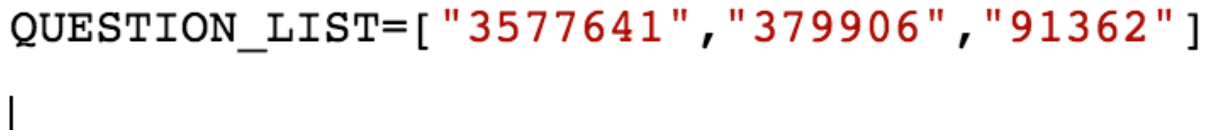


```
#!/usr/bin/env python
# -*- coding: utf-8 -*-


#IMPORTS
import urllib,urllib2
import json
import zlib
import time




SO_API_URL="https://api.stackexchange.com/2.2/"
API_KEY="gg7oHfBwbgaikrT3CgvfLg(("
```

USING PYTHON FOR STACK EXCHANGE API

We want to get answers to questions by their question ID.

- Assume this is the question ID list :



```
QUESTION_LIST=["3577641","379906","91362"]
```

- The basic method for retrieving:

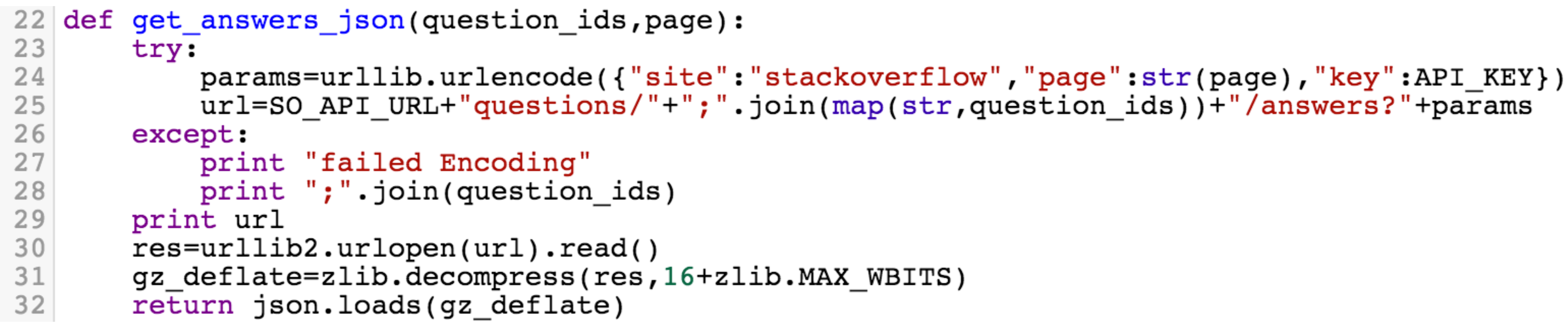
1. Preparing list of ***url encoded*** parameters (line 24)

2.compiling the URL (line 25)

3. Executing the request (line 30)

4.decompressing the results (31)

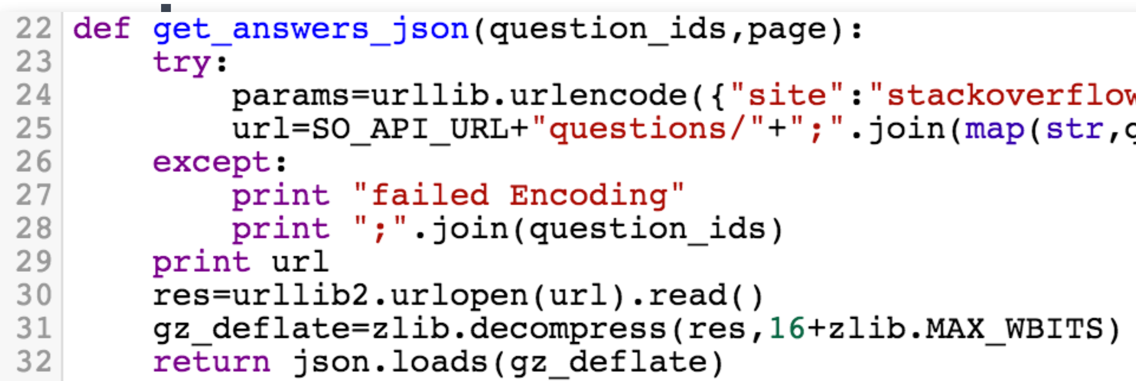
5.Parsing the Json into a dictionary and return it (line 32)



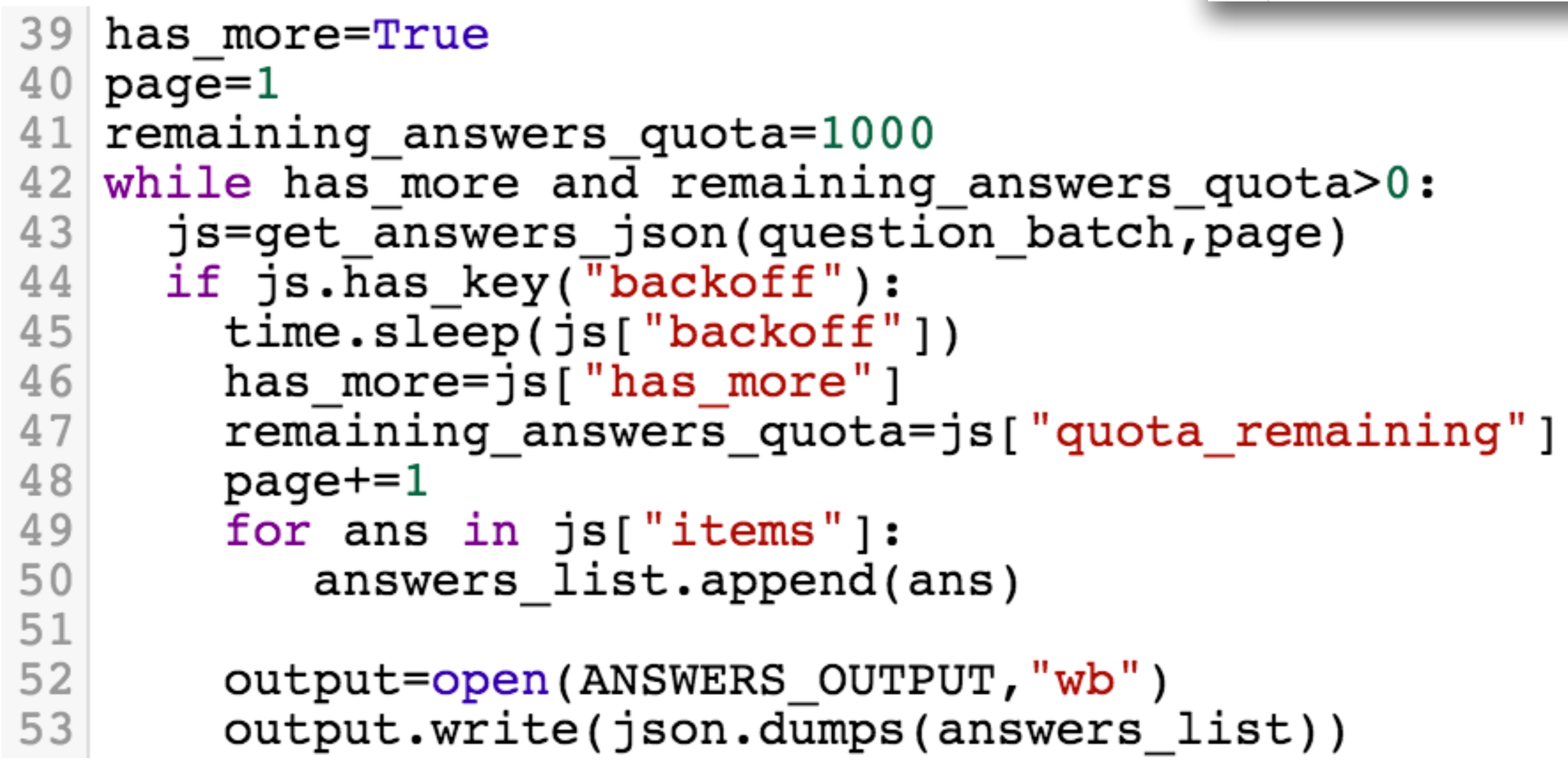
```
22 def get_answers_json(question_ids,page):
23     try:
24         params=urllib.urlencode({"site":"stackoverflow","page":str(page),"key":API_KEY})
25         url=SO_API_URL+"questions/"+";".join(map(str,question_ids))+"/answers?"+params
26     except:
27         print "failed Encoding"
28         print ";".join(question_ids)
29     print url
30     res=urllib2.urlopen(url).read()
31     gz_deflate=zlib.decompress(res,16+zlib.MAX_WBITS)
32     return json.loads(gz_deflate)
```

USING PYTHON FOR STACK EXCHANGE API

- **Still it is not so simple as stack exchange are not פרייארים:**
  - ★**Requests quota is limited**
  - ★**“Backoff”: If you don’t wait the backoff, you are b**
  - ★**They don't send all results at once (“hasMore”)**
  - ★**No more than 100 questions IDs can be sent at on**



```
def get_answers_json(question_ids,page):
    try:
        params=urllib.urlencode({"site":"stackoverflow
        url=SO_API_URL+"questions/"+";".join(map(str,q
    except:
        print "failed Encoding"
        print ";".join(question_ids)
    print url
    res=urllib2.urlopen(url).read()
    gz_deflate=zlib.decompress(res,16+zlib.MAX_WBITS)
    return json.loads(gz_deflate)
```



```
has_more=True
page=1
remaining_answers_quota=1000
while has_more and remaining_answers_quota>0:
  js=get_answers_json(question_batch,page)
  if js.has_key("backoff"):
    time.sleep(js["backoff"])
    has_more=js["has_more"]
    remaining_answers_quota=js["quota_remaining"]
    page+=1
    for ans in js["items"]:
       answers_list.append(ans)

    output=open(ANSWERS_OUTPUT,"wb")
    output.write(json.dumps(answers_list))
```

ON THE NEXT LECTURE

- The very basics of web programming:
  - Installing Xampp (Apache, MySQL,PHP)
  - Introduction PHP and server side scripting
  - Introduction to HTML, CSS and JavaScript programming