GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

LECTURE 2: DATA COLLECTION AND STORAGE

LAST TIME:

- INTRO TO DATA SCIENCE
- COMPUTER SETUP AND DATA WORKFLOW

QUESTIONS?

I. RELATIONAL DATABASES
II. NOSQL DATABASES
III. JSON, APIS, AND SCRAPING

EXERCISES:
MYSQL TUTORIAL
BEAUTIFULSOUP TUTORIAL

I. RELATIONAL DATABASES

Databases are a structured data source optimized for efficient retrieval and persistent storage.

Databases are a structured data source optimized for efficient retrieval and persistent storage.

structured: we have to pre-define organization strategy

retrieval: the ability to read data out

storage: the ability to write data and save it

Relational databases are traditionally organized in the following manner:

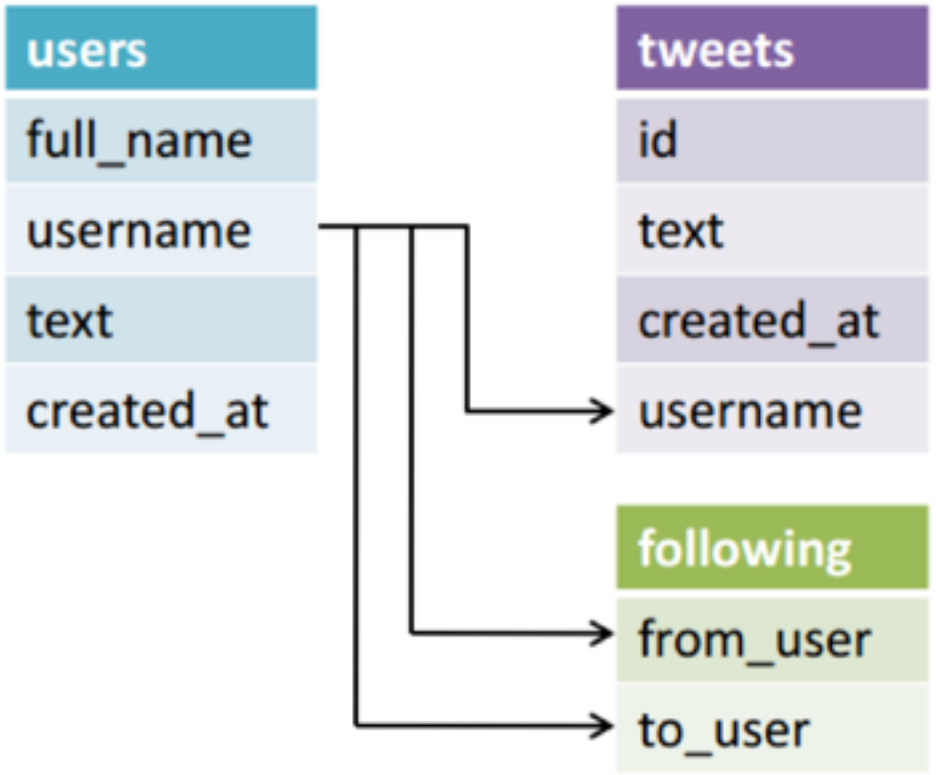
A database has tables which represent individual entities or objects

Tables have predefined schema – rules that tell it what the data will look like

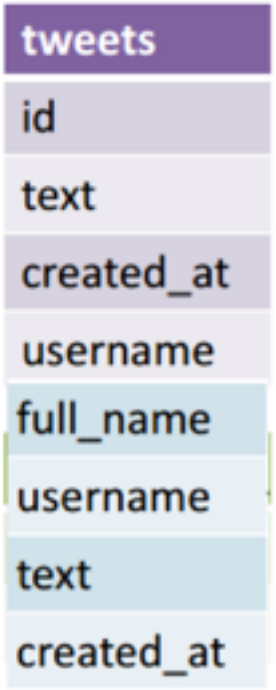
Each table should have a primary key column – a unique identifier for that row

Each table should have a primary key column – a unique identifier for that row

Additionally each table can have a foreign key column – an id that links this to another table.


users
full_name
username
text
created_at
tweets
id
text
created_at
username
following
from_user
to_user

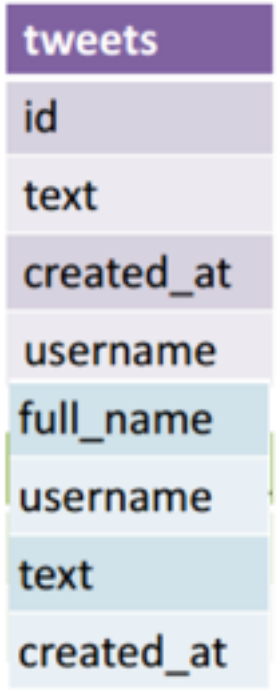
We could have had a table structure as follows:
Why is this different?



We could have had a table structure as follows:
Why is this different?

We would repeat the user information in each row.

This is called denormalization.



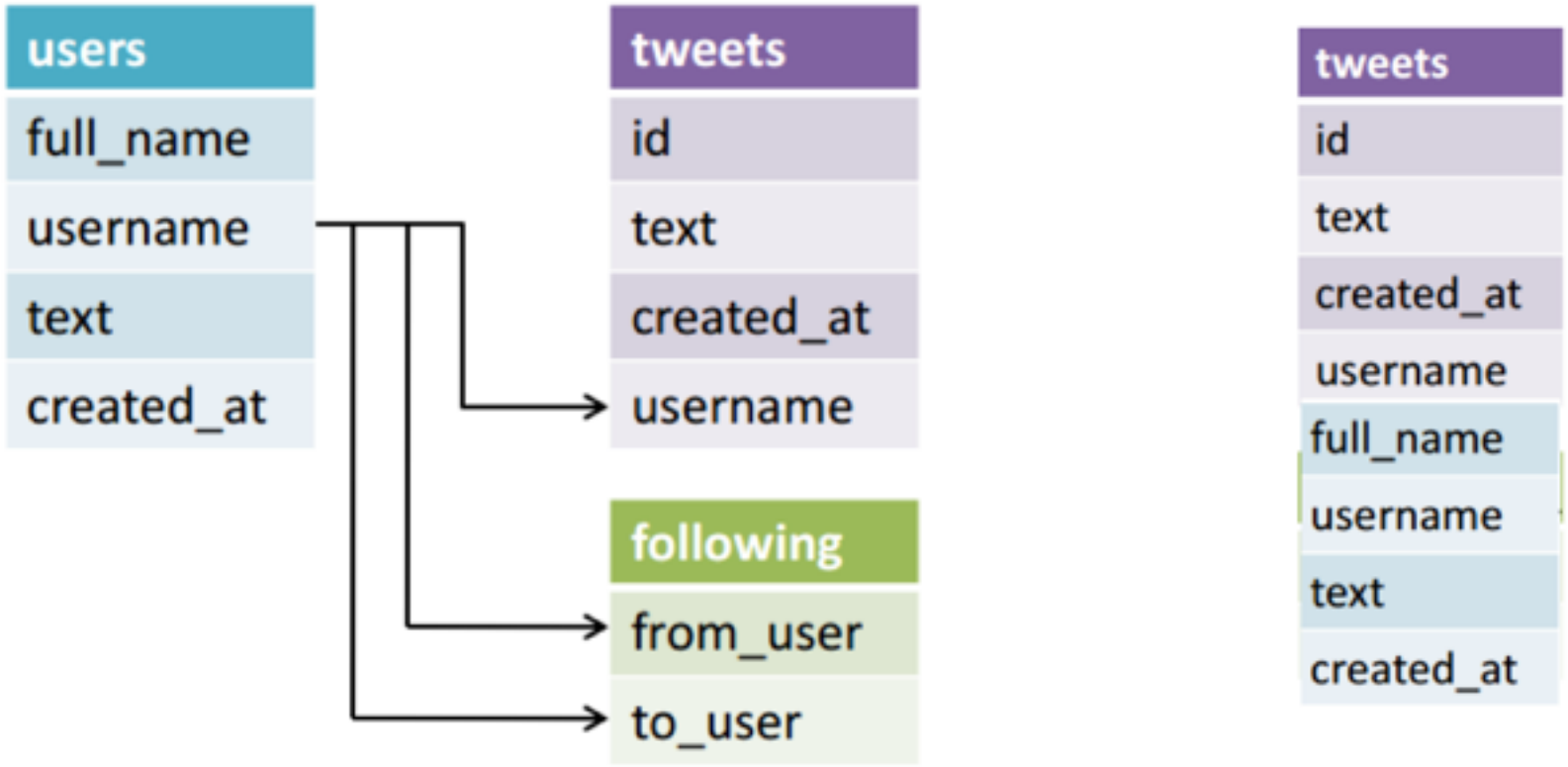
Normalized Data: Many tables to reduce redundant or repeated data in a table

Denormalized Data: Wide data with fields often repeated but removes the need to join together multiple tables

Normalized Data: Many tables to reduce redundant or repeated data in a table

Denormalized Data: Wide data with fields often repeated but removes the need to join together multiple tables

This is a trade off of speed vs storage.

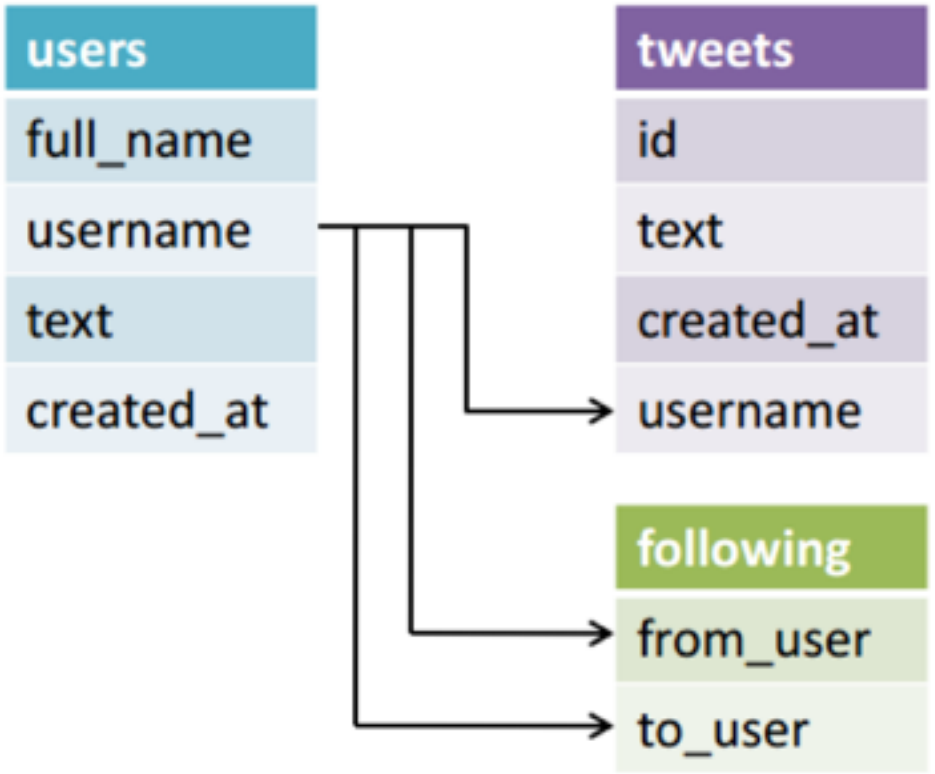

users
full_name
username
text
created_at
tweets
id
text
created_at
username
following
from_user
to_user
tweets
id
text
created_at
username
full_name
username
text
created_at

Q: How do we commonly evaluate databases?

read-speed vs write-speed
space considerations
(and many, many other criteria)

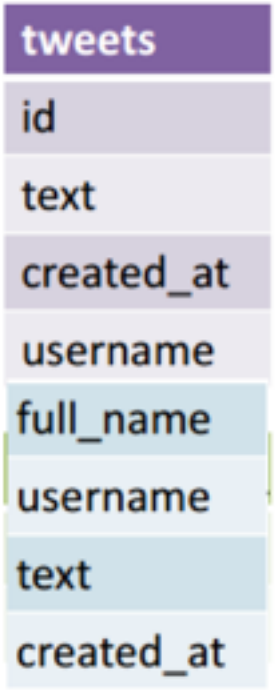
Q: Why are normalized tables (possibly) slower to read?

A: We'll have to get data from multiple tables to answer some questions


users
full_name
username
text
created_at
tweets
id
text
created_at
username
following
from_user
to_user

Q: Why are denormalized tables (possibly) slower to write?

A: We'll have to write more information on each write


tweets
id
text
created_at
username
full_name
username
text
created_at

SQL is a query language to load, retrieve, and update data in relational databases

Most commonly known SQL-like Databases include:
Oracle
MySQL/MariaDB
PostgreSQL

SELECT: Allows you to retrieve information from a table

Syntax:

```
SELECT col1, col2
FROM table WHERE [some condition]
```

Example:

```
SELECT poll_title, poll_date FROM polls WHERE
romney_pct > obama_pct
```

GROUP BY: Allows you to aggregate information.

Syntax:

```
SELECT col1, AVG(col2)
FROM table GROUP BY col1
```

Example:

```
SELECT poll_date, AVG(obama_pct)
FROM polls GROUP BY poll_date
```

GROUP BY: Allows you to aggregate information.

Syntax:

```
SELECT col1, AVG(col2)
FROM table GROUP BY col1
```

Example:

```
SELECT poll_date, AVG(obama_pct)
FROM polls GROUP BY poll_date
```

GROUP BY: Allows you to aggregate information.

Syntax:

```
SELECT col1, AVG(col2)
FROM table GROUP BY col1
```

There are usually a few common built-in operations:

SUM, AVG, MIN, MAX, COUNT

JOIN: Allows you to combine multiple tables

Syntax:

```
SELECT t1.c1, t1.c2, t2.c2
FROM t1
      INNER JOIN t2 ON t1.c1 = t2.c2
```

JOIN: Allows you to combine multiple tables

Syntax:

```
SELECT t1.c1, t1.c2, t2.c2
FROM t1
     INNER JOIN t2 ON t1.c1 = t2.c2
```

INSERT: Allows you to add data to tables

Syntax:

```
INSERT INTO table1 (col1, col2)
VALUES (...)

INSERT INTO classroom (first_name, last_name)
VALUES (‘John’, ‘Doe’)
```

LAB: MYSQL QUERYING

III. NO-SQL DATABASES

NoSQL databases are a new trend in databases

NoSQL databases are a new trend in databases

The name NoSQL refers to the lack of a relational structure between stored objects

Most importantly they attempt to minimize the need for JOIN operations, or solve other data needs

Memcached, Redis

Apache HBase

Cassandra

MongoDB

Hadoop

III. JSON, APIS, AND SCRAPING, OH MY

Mongo's document structure is highly based off of JSON.

JSON (JavaScript Object Notation) is a borrowed JavaScript structure turned into a string that can be passed between applications.

JSON is passed through applications as a string, and converted into native objects per their language.

```
>>> someFile = open('/Users/epodojil/GA_Data_Science/a.json').read()
>>> print json.dumps(someFile)
"{\n    \"glossary\": {\n        \"title\": \"example glossary\",\n    \"GlossDiv\": {\n            \"title\": \"S\",\n      \"
\"SGML\",\n          \"SortAs\": \"SGML\",\n          \"GlossTerm\": \"Standard Generalized Markup Language\",\n          \"Acr
ef\": {\n                        \"para\": \"A meta-markup language, used to create markup languages such as DocBook.\",\n
\"GlossSee\": \"markup\"\n                }\n            }\n        }\n    }\n}"
>>> print someFile
{
    "glossary": {
        "title": "example glossary",
    "GlossDiv": {
            "title": "S",
      "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
                    },
          "GlossSee": "markup"
                }
            }
        }
    }
}
>>> print json.loads(someFile)
{u'glossary': {u'GlossDiv': {u'GlossList': {u'GlossEntry': {u'GlossDef': {u'GlossSeeAlso': [u'GML', u'XML'], u'para': u'A meta-
': u'markup', u'Acronym': u'SGML', u'GlossTerm': u'Standard Generalized Markup Language', u'Abbrev': u'ISO 8879:1986', u'SortAs
```

```
>>> someFile = open('/Users/epodojil/GA_Data_Science/a.json').read()
>>> print json.dumps(someFile)
"{\n    \"glossary\": {\n        \"title\": \"example glossary\",\n    \"GlossDiv\": {\n            \"title\": \"S\",\n      \"
\"SGML\",\n          \"SortAs\": \"SGML\",\n          \"GlossTerm\": \"Standard Generalized Markup Language\",\n          \"Acr
ef\": {\n                        \"para\": \"A meta-markup language, used to create markup languages such as DocBook.\",\n
\"GlossSee\": \"markup\"\n                }\n            }\n        }\n    }\n}"
>>> print someFile
{
    "glossary": {
        "title": "example glossary",
    "GlossDiv": {
            "title": "S",
      "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
                    },
          "GlossSee": "markup"
                }
            }
        }
    }
}
>>> print json.loads(someFile)
{u'glossary': {u'GlossDiv': {u'GlossList': {u'GlossEntry': {u'GlossDef': {u'GlossSeeAlso': [u'GML', u'XML'], u'para': u'A meta-
': u'markup', u'Acronym': u'SGML', u'GlossTerm': u'Standard Generalized Markup Language', u'Abbrev': u'ISO 8879:1986', u'SortAs
```

String

Object

```
>>> someFile = open('/Users/epodojil/GA_Data_Science/a.json').read()
>>> print json.dumps(someFile)
"{\n    \"glossary\": {\n        \"title\": \"example glossary\",\n    \"GlossDiv\": {\n            \"title\": \"S\",\n      \"
\"SGML\",\n          \"SortAs\": \"SGML\",\n          \"GlossTerm\": \"Standard Generalized Markup Language\",\n          \"Acr
ef\": {\n                        \"para\": \"A meta-markup language, used to create markup languages such as DocBook.\",\n
\"GlossSee\": \"markup\"\n                }\n            }\n        }\n    }\n}"
>>> print someFile
{
    "glossary": {
        "title": "example glossary",
    "GlossDiv": {
            "title": "S",
      "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
                    },
          "GlossSee": "markup"
                }
            }
        }
    }
}
>>> print json.loads(someFile)
{u'glossary': {u'GlossDiv': {u'GlossList': {u'GlossEntry': {u'GlossDef': {u'GlossSeeAlso': [u'GML', u'XML'], u'para': u'A meta-
': u'markup', u'Acronym': u'SGML', u'GlossTerm': u'Standard Generalized Markup Language', u'Abbrev': u'ISO 8879:1986', u'SortAs
```

String

Object

Python Dict

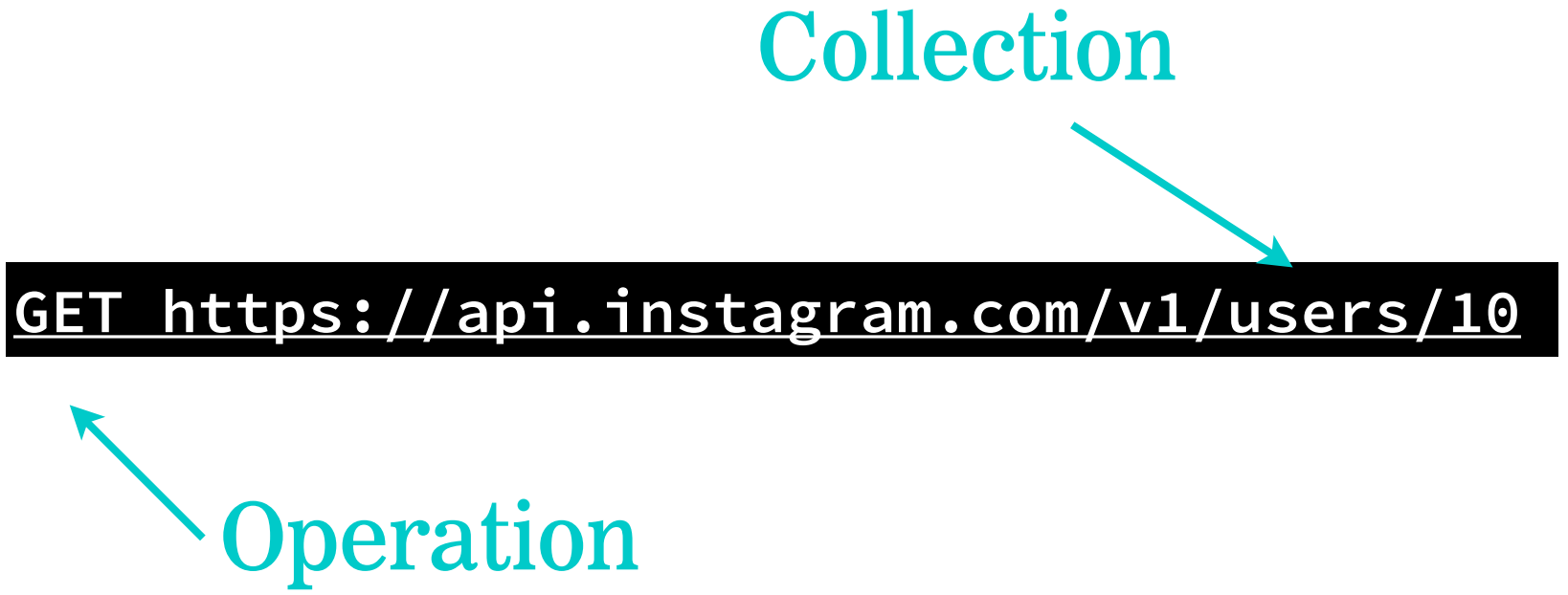
APIs (Application Programming Interface) allow people to interact with the structures of an application to get, put, delete, or update data.

APIs (Application Programming Interface) allow people to interact with the structures of an application to get, put, delete, or update data.

Best practices for APIs are to use RESTful principles.

RESTful APIs include:

- The Base URL and collection.
- An interactive media type (usually JSON)
- Operations (GET, PUT, POST, DELETE)
- Driven by Hypertext (http requests)


Collection
GET https://api.instagram.com/v1/users/10
Operation

```
GET https://api.instagram.com/v1/users/
search/?q=andy
```

Querystring/Search

RESTful APIs can always be accessed using cURL requests: hence why hypertext access is a requirement!

Most have language libraries to make it easier to access through the language of your choice.

http://www.pythonapi.com/

The least organized way to grab data is by using web scraping tools such as Beautiful Soup, Scrapy, or Nokogiri

Advantages:

- Granularity in accessibility of data
- High value of control

The least organized way to grab data is by using web scraping tools such as Beautiful Soup, Scrapy, or Nokogiri

Advantages:
- Granularity in accessibility of data
- High value of control

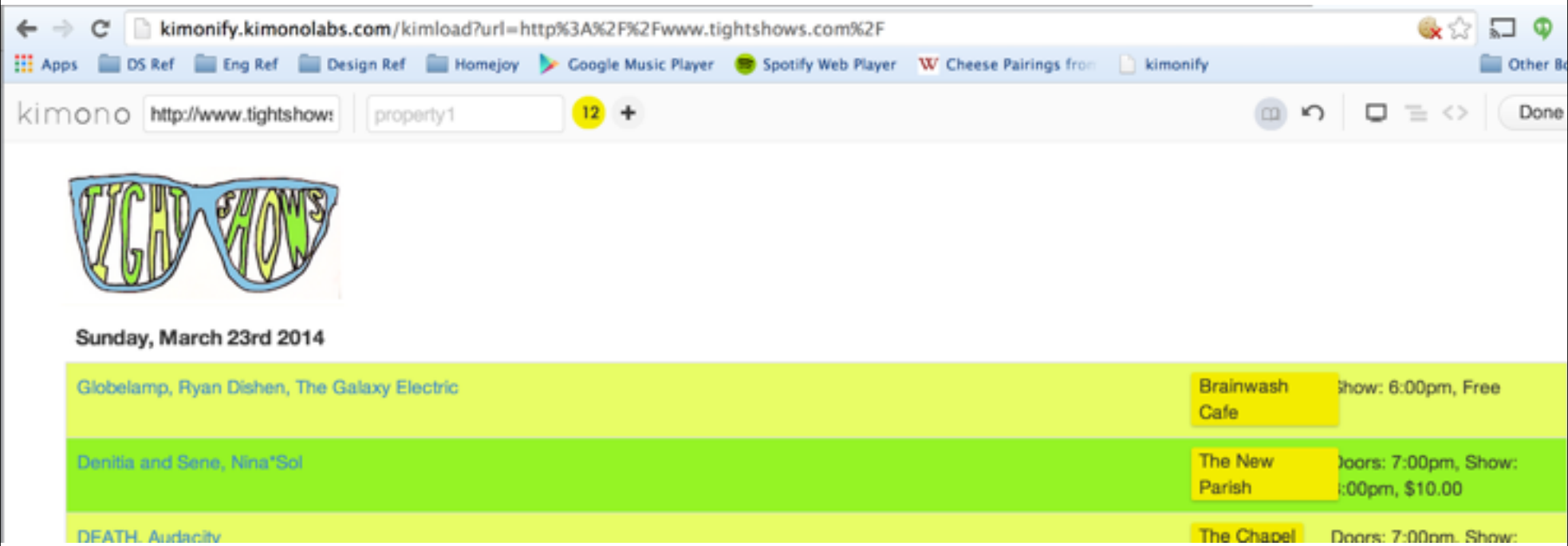
Disadvantages:
- Webpages change—very easy to break
- Requires an intense amount of work to keep functional

```
from bs4 import BeautifulSoup
import urllib2

req  = urllib2.Request('http://www.tightshows.com')
data = urllib2.urlopen(req).read()

soup = BeautifulSoup(data)

for link in soup.find_all('a'):
    if link.get('href')[0:7] == '/venues':
        print(link.get('href'))
```

```
from bs4 import BeautifulSoup
import urllib2

req  = urllib2.Request('http://www.tightshows.com')
data = urllib2.urlopen(req).read()

soup = BeautifulSoup(data)

for link in soup.find_all('a'):
    if link.get('href')[0:7] == '/venues':
        print(link.get('href'))
```

no ids on links

only way to find venues


kimonify.kimonolabs.com/kimload?url=http%3A%2F%2Fwww.tightshows.com%2F
Apps
DS Ref
Eng Ref
Design Ref
Homejoy
Google Music Player
Spotify Web Player
Cheese Pairings fro
kimonify
Other Bo
kimono
http://www.tightshow
property1
12
Done
TIGHT SHOWS
Sunday, March 23rd 2014
Globelamp, Ryan Dishen, The Galaxy Electric
Brainwash Cafe
how: 6:00pm, Free
Denitia and Sene, Nina*Sol
The New Parish
oors: 7:00pm, Show:
:00pm, $10.00
DEATH, Audacity
The Chapel
Doors: 7:00pm, Show:


1 collection
JSON CSV RSS
Download JSON
Select all text
Sunday, March 23rd 2014
Globelamp, Ryan Dishen, The Galaxy Electric
Danitia and Siena, Nina'Sol
DEATH, Audacity
Astronautalis, Playdough, Transit, Low Country Kingdom
{
"collection1": [
{
"venue": {
"text": "Brainwash Cafe",
"href": "http://www.tightshows.com/venues/5_brainwash"
}
},
{
"venue": {
"text": "The New Parish",
"href": "http://www.tightshows.com/venues/6_new_parish"
}
},
{

LAB: DATA COLLECTION

DISCUSSION

1. We've now gone over how to start off the data workflow through data collection and organization—what's next in our steps?

NEXT CLASS SUBJECT: DATA MANIPULATION