GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

LECTURE 2: INTRO TO PYTHON AND DATA COLLECTION

LAST TIME:

- INTRO TO DATA SCIENCE
- COMPUTER SETUP AND DATA WORKFLOW

QUESTIONS?

AGENDA

I. PYTHON DATA STRUCTURES
II. JSON, APIS, AND SCRAPING

EXERCISES:
PYTHON DATA STRUCTURES WALKTHROUGH + COMMENT ALONG
BUILDING A NYTIMES API MODULE

REVIEW: DATA SCIENCE DEFINITION

“Data Science is the process of extracting meaning from data through data mining, statistics, and machine learning techniques.

Data scientists validate hypotheses, generate models, and extrapolate conclusions in order to communicate the meaning of data.”

I. PYTHON DATA STRUCTURES

High variety of languages used in practice:

Statistics: Python, R, Matlab, Julia
Applications: Python, Ruby, Scala, Java
Querying: SQL, Hive, Pig

STRENGTHS AND WEAKNESSES

Python in nature is not a statistical language, though used in a variety of ways: web applications, server maintenance, reading and writing text files:

```
web development https://www.djangoproject.com/
systems admin http://docs.fabfile.org/en/1.6/
(etc) https://github.com/languages/Python
```

Python evolved alongside Bioinformatics and Data Analysis, introducing stats and machine learning packages:
numpy, scipy, statsmodels, pandas, sk-learn, NLTK

STRENGTHS AND WEAKNESSES

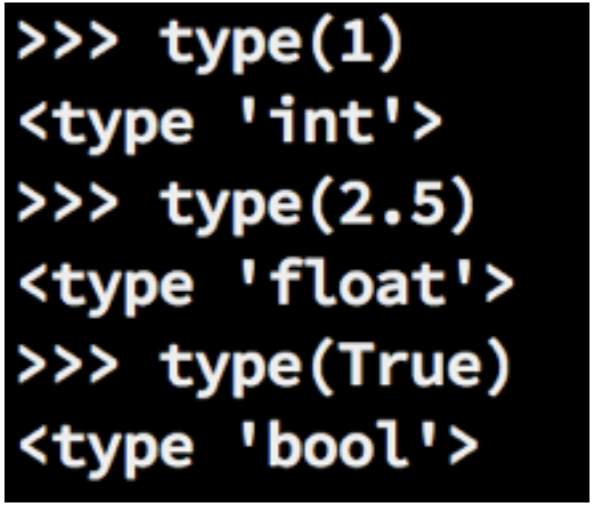
ADVANTAGES

- VERY FAST, COMPARATIVELY
- USEFUL ACROSS PLATFORMS
- EASY TO INTEGRATE
- COMMON OOP ARCHITECTURE
- GREAT DOC SUPPORT

DISADVANTAGES

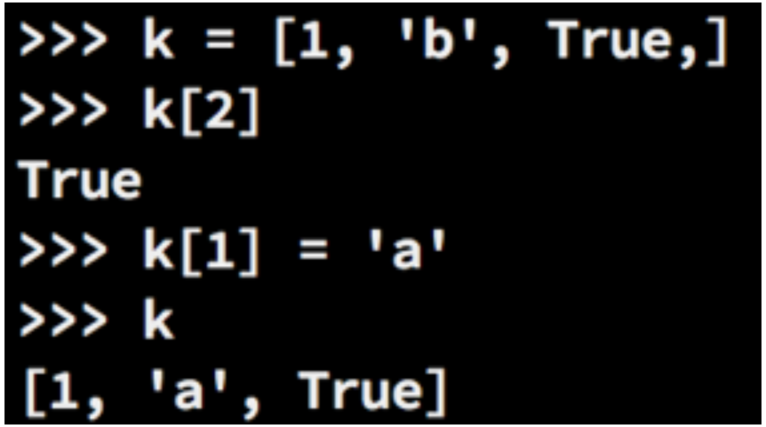
- NATURAL DISPLAY LESS READABLE
- LESS FRIENDLY
- LACK OF PARALLEL PROCESSING

The most basic data structure is the None type. This is the equivalent of NULL in other languages.
There are three basic numeric types: int, float, and boolean.



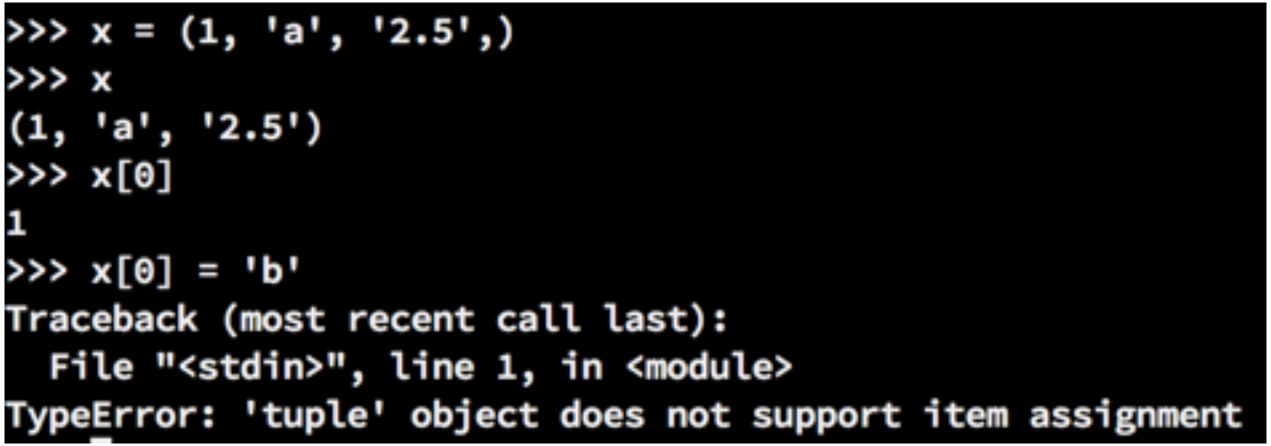
```
>>> type(1)
<type 'int'>
>>> type(2.5)
<type 'float'>
>>> type(True)
<type 'bool'>
```

The next basic data type is the Python list.
A list is an ordered collection of elements, and these elements can be of arbitrary type.
Lists are mutable, meaning they can be changed in-place.



```
>>> k = [1, 'b', True,]
>>> k[2]
True
>>> k[1] = 'a'
>>> k
[1, 'a', True]
```

Likewise, tuples are immutable arrays of arbitrary elements.



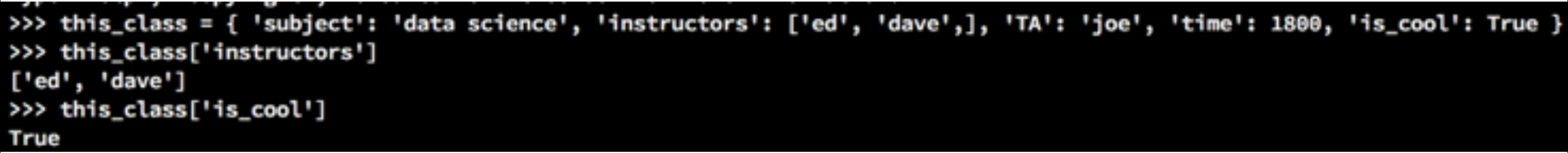
```
>>> x = (1, 'a', '2.5',)
>>> x
(1, 'a', '2.5')
>>> x[0]
1
>>> x[0] = 'b'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

The string type in Python represents an immutable ordered array of characters.

Strings support slicing and indexing operations like arrays, and have many other string-specific functions as well.

String processing is one area where Python excels.

Associative arrays (or hash tables) are implemented in Python as the dictionary type.



```
>>> this_class = { 'subject': 'data science', 'instructors': ['ed', 'dave',], 'TA': 'joe', 'time': 1800, 'is_cool': True }
>>> this_class['instructors']
['ed', 'dave']
>>> this_class['is_cool']
True
```

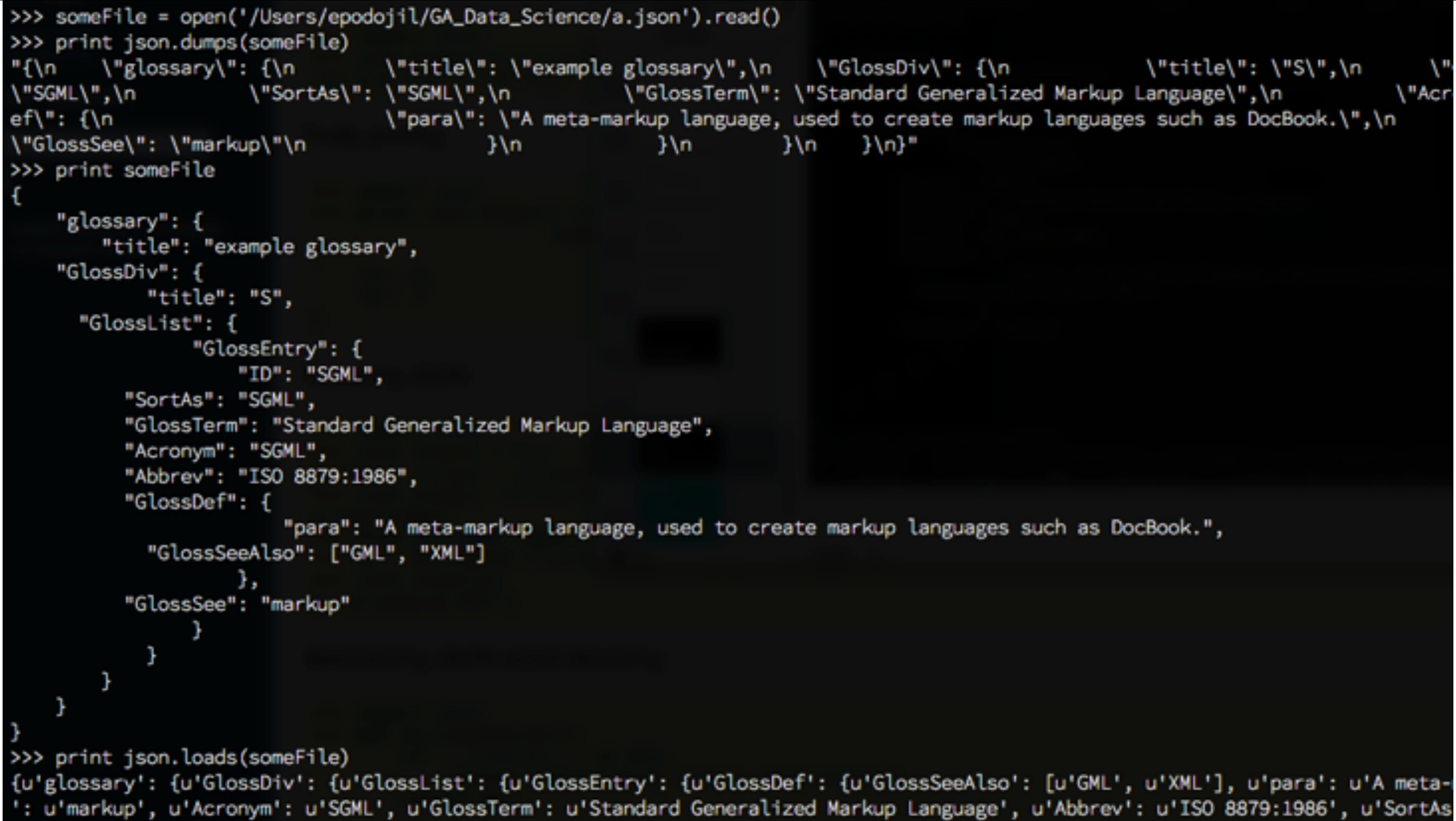
III. JSON, APIS, AND SCRAPING, OH MY

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

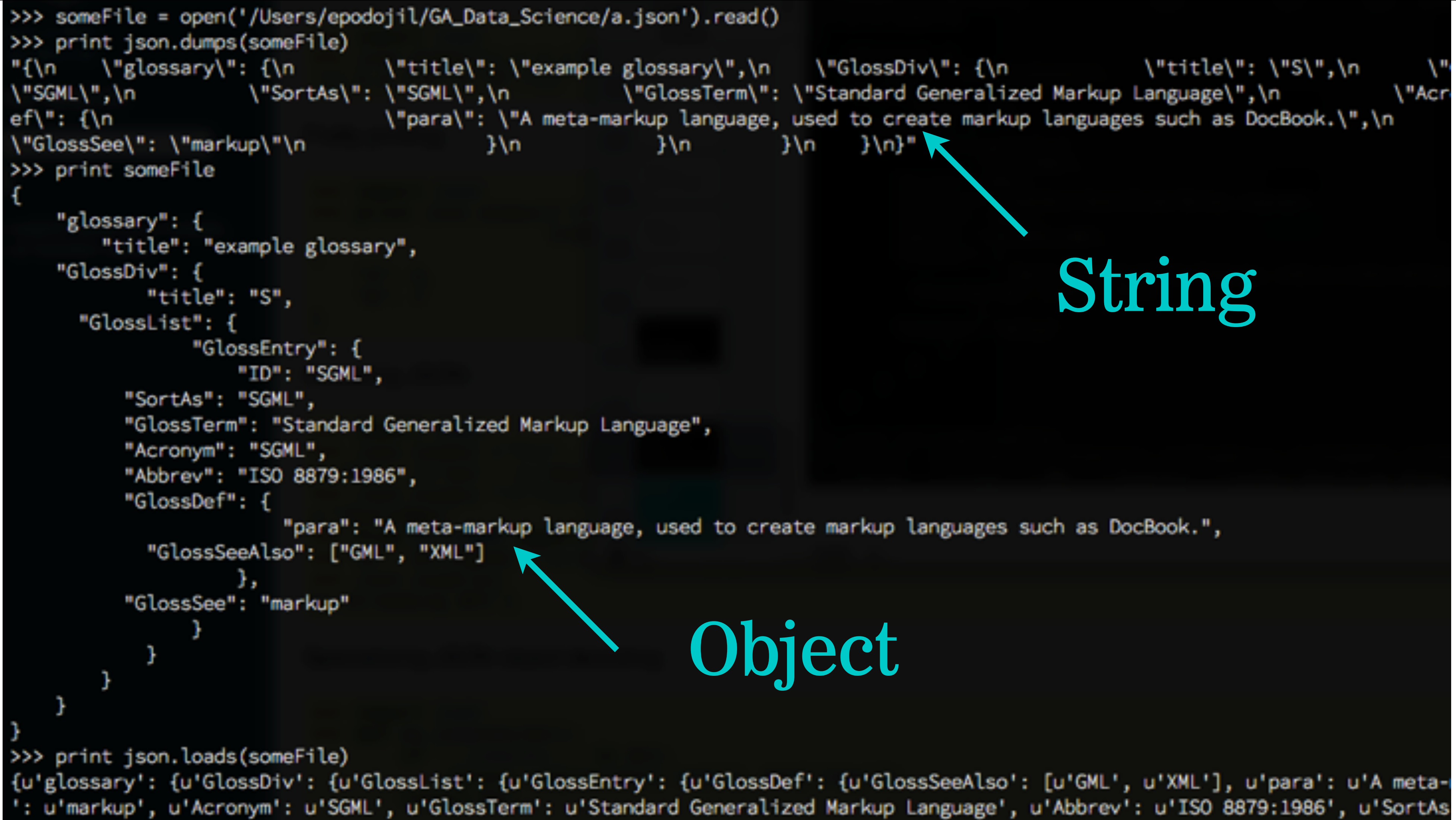
JSON

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

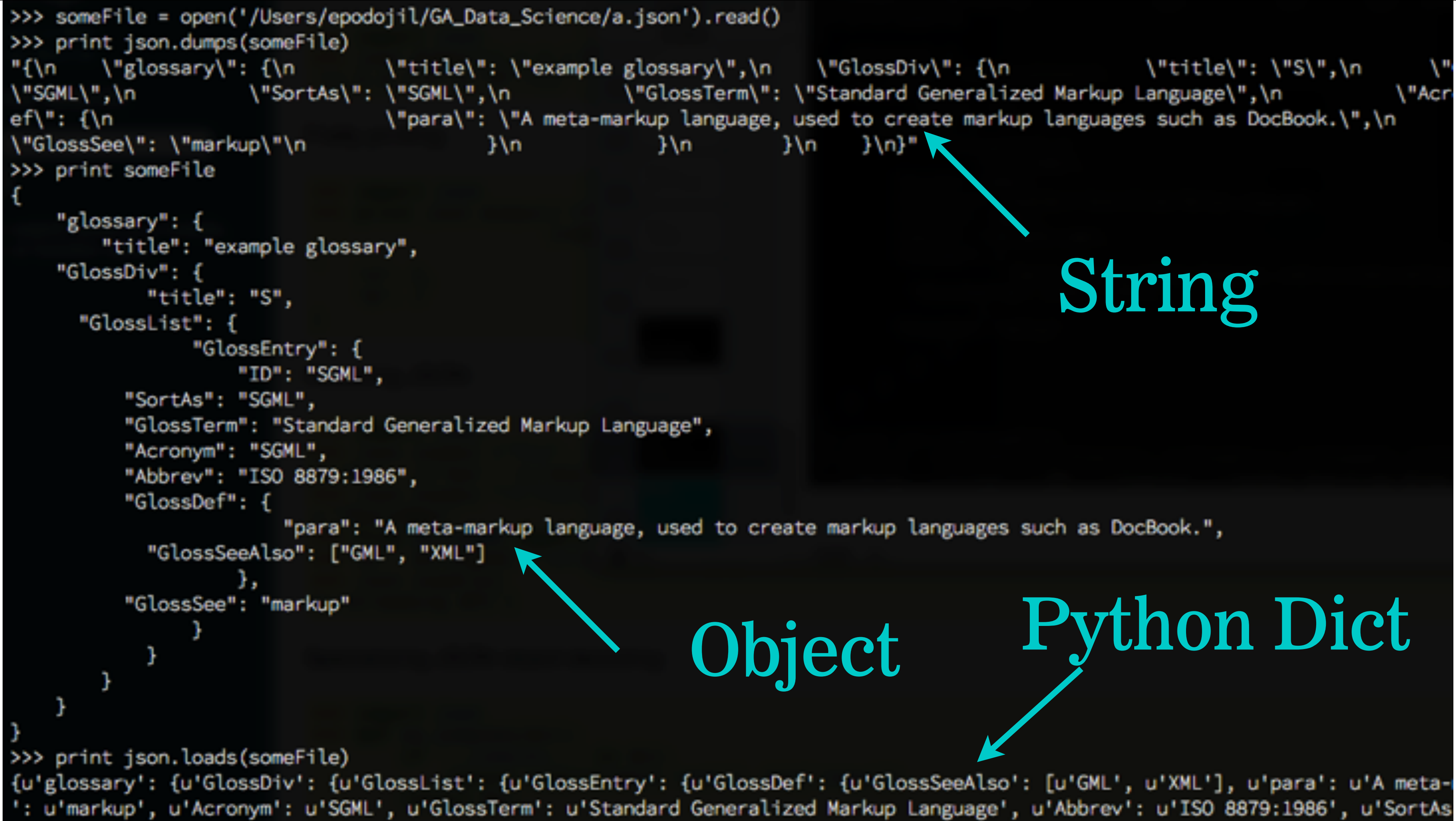
JSON is passed through applications as a string, and converted into native objects per their language.



```
>>> 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
```



```
>>> 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
```

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

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

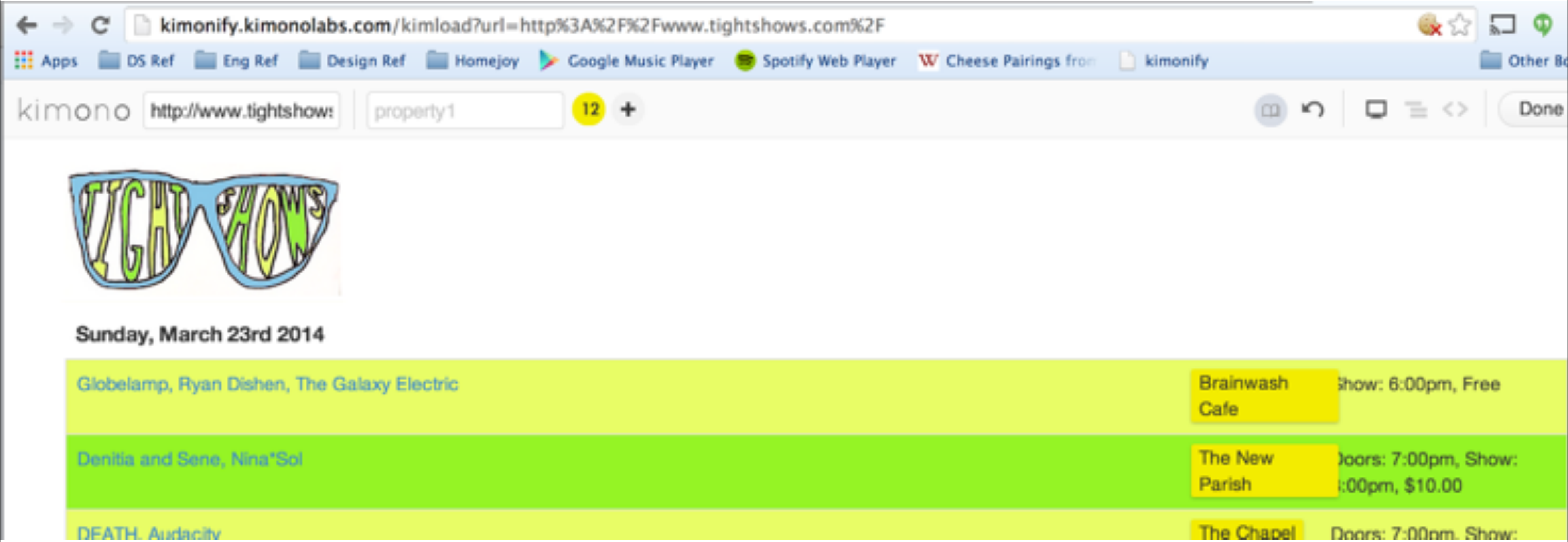
```
from bs4 import BeautifulSoup
import urllib2

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

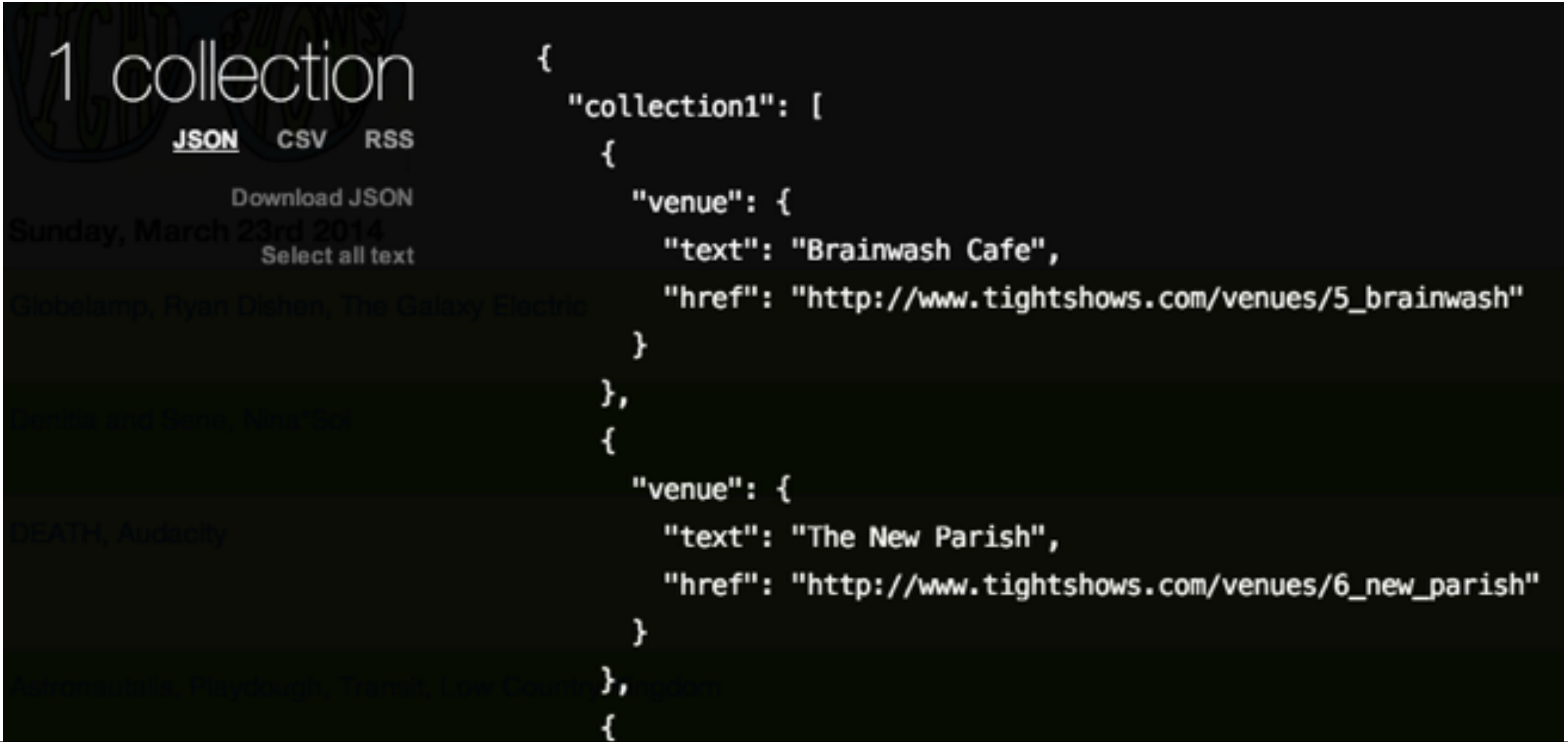
only way to find venues

KIMONOLABS



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:



```
{
  "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: PYTHON AND DATA COLLECTION

NEXT CLASS SUBJECT: NUMPY AND PANDAS