INLS 560
Programming for Information Professionals

Text Processing

UNC
SCHOOL OF INFORMATION
AND LIBRARY SCIENCE

Joan Boone
jpboone@email.unc.edu

Part 1: Overview

Part 2: Structured data: XML and JSON

Part 3: Parsing JSON data with Python

Text Processing

Many applications involve some form of text processing

- Data and text mining
- Natural language processing
- Indexing
- Metadata generation
- Data interchange
- Re-purposing content, e.g., data visualization, to improve understanding and interpretation of data

With the proliferation of big data and open data, these applications become increasingly important.

Text data takes many forms

Unstructured text

- Similar to the e-book text files found on the Project Gutenberg site
- Published content, data generated from speech, tweets
- Processing often requires Natural Language Processing (NLP) tools that work with human language data to categorize words, classify text and analyze sentence structure and meaning

Tabular data (semi-structured)

- Typically organized in rows and columns
- Examples: spreadsheets, CSV files, log data

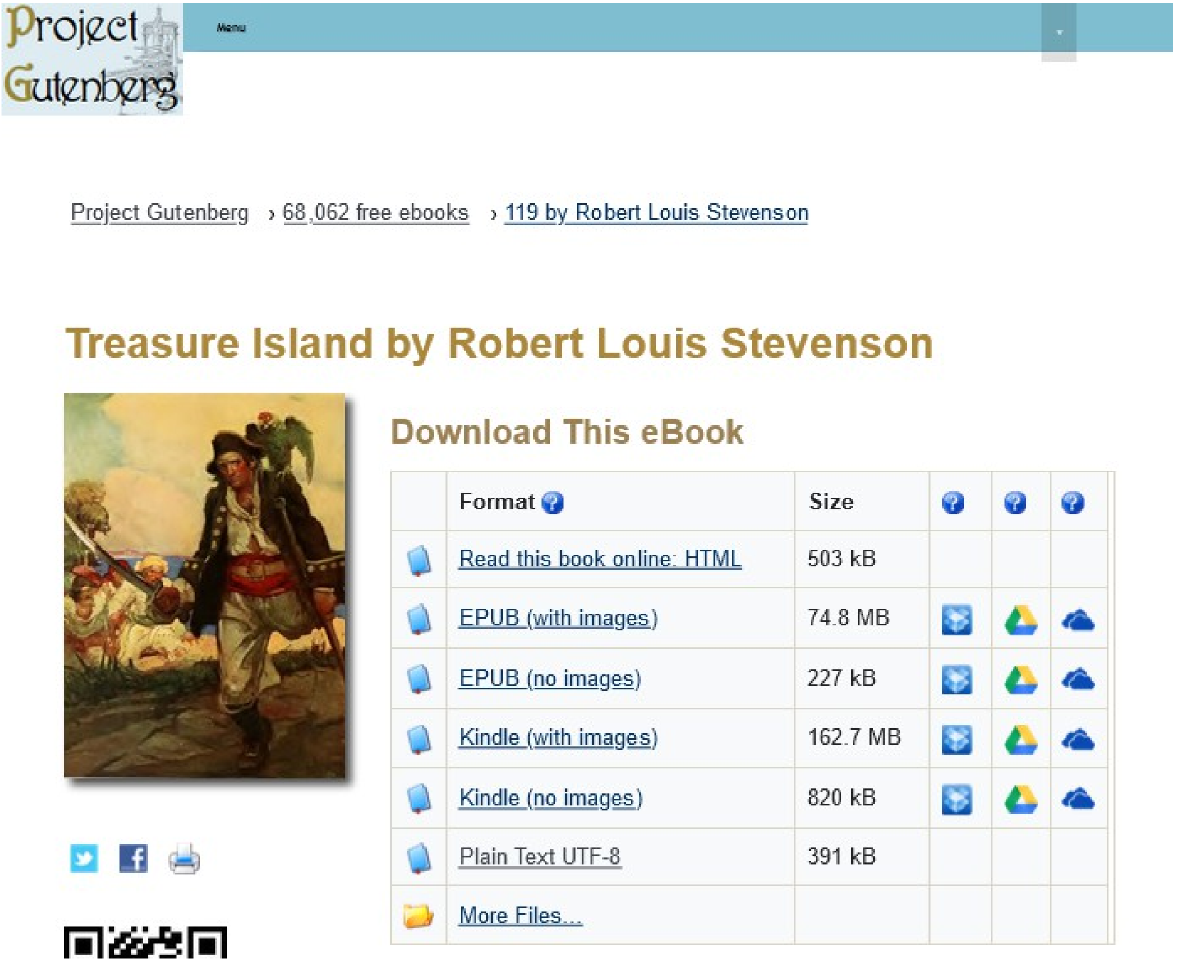
Structured data

- Organized in a specific format that describes and defines data
- Examples: JSON and XML data formats

Unstructured Text

Project Gutenberg

collection of free e-books

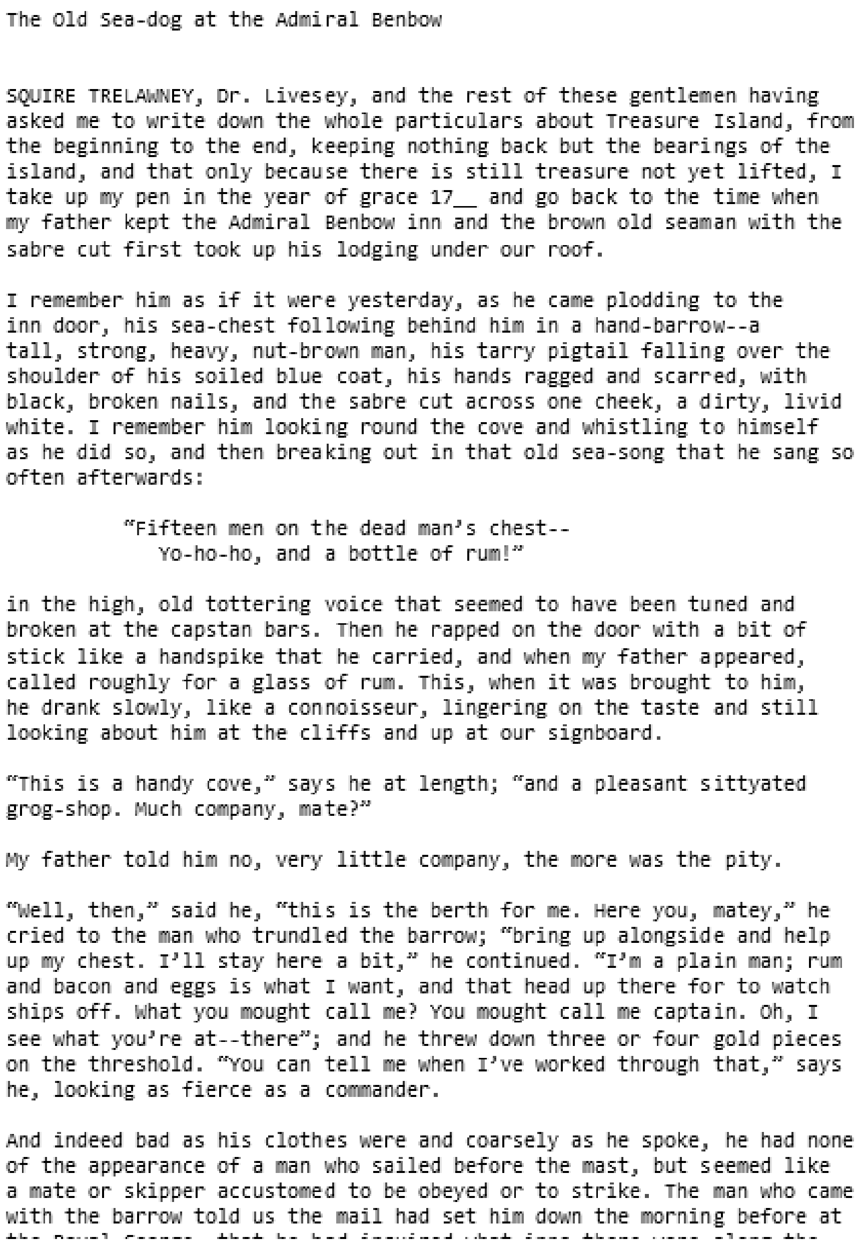


Project Gutenberg › 68,062 free ebooks › 119 by Robert Louis Stevenson

Treasure Island by Robert Louis Stevenson

Download This eBook

	Format	Size			
	Read this book online: HTML	503 kB			
	EPUB (with images)	74.8 MB			
	EPUB (no images)	227 kB			
	Kindle (with images)	162.7 MB			
	Kindle (no images)	820 kB			
	Plain Text UTF-8	391 kB			
	More Files...				



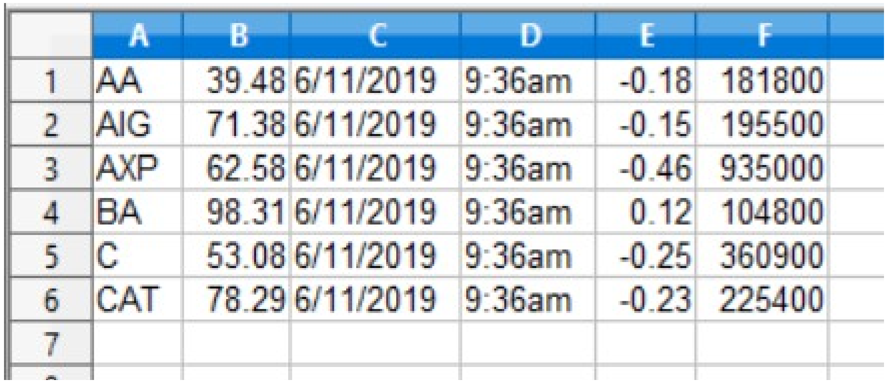
```
The Old Sea-dog at the Admiral Benbow


SQUIRE TRELAWNEY, Dr. Livesey, and the rest of these gentlemen having
asked me to write down the whole particulars about Treasure Island, from
the beginning to the end, keeping nothing back but the bearings of the
island, and that only because there is still treasure not yet lifted, I
take up my pen in the year of grace 17__ and go back to the time when
my father kept the Admiral Benbow inn and the brown old seaman with the
sabre cut first took up his lodging under our roof.

I remember him as if it were yesterday, as he came plodding to the
inn door, his sea-chest following behind him in a hand-barrow--a
tall, strong, heavy, nut-brown man, his tarry pigtail falling over the
shoulder of his soiled blue coat, his hands ragged and scarred, with
black, broken nails, and the sabre cut across one cheek, a dirty, livid
white. I remember him looking round the cove and whistling to himself
as he did so, and then breaking out in that old sea-song that he sang so
often afterwards:

          “Fifteen men on the dead man’s chest--
             Yo-ho-ho, and a bottle of rum!”

in the high, old tottering voice that seemed to have been tuned and
broken at the capstan bars. Then he rapped on the door with a bit of
stick like a handspike that he carried, and when my father appeared,
called roughly for a glass of rum. This, when it was brought to him,
he drank slowly, like a connoisseur, lingering on the taste and still
looking about him at the cliffs and up at our signboard.

“This is a handy cove,” says he at length; “and a pleasant sittyated
grog-shop. Much company, mate?”

My father told him no, very little company, the more was the pity.

“Well, then,” said he, “this is the berth for me. Here you, matey,” he
cried to the man who trundled the barrow; “bring up alongside and help
up my chest. I’ll stay here a bit,” he continued. “I’m a plain man; rum
and bacon and eggs is what I want, and that head up there for to watch
ships off. What you mought call me? You mought call me captain. Oh, I
see what you’re at--there”; and he threw down three or four gold pieces
on the threshold. “You can tell me when I’ve worked through that,” says
he, looking as fierce as a commander.

And indeed bad as his clothes were and coarsely as he spoke, he had none
of the appearance of a man who sailed before the mast, but seemed like
a mate or skipper accustomed to be obeyed or to strike. The man who came
with the barrow told us the mail had set him down the morning before at
```

Processing Tabular Data (semi-structured)

CSV view (stocks.csv)

```
"AA",39.48,"6/11/2019","9:36am",-0.18,181800
"AIG",71.38,"6/11/2019","9:36am",-0.15,195500
"AXP",62.58,"6/11/2019","9:36am",-0.46,935000
"BA",98.31,"6/11/2019","9:36am",+0.12,104800
"C",53.08,"6/11/2019","9:36am",-0.25,360900
"CAT",78.29,"6/11/2019","9:36am",-0.23,225400
```

Spreadsheet view



	A	B	C	D	E	F
1	AA	39.48	6/11/2019	9:36am	-0.18	181800
2	AIG	71.38	6/11/2019	9:36am	-0.15	195500
3	AXP	62.58	6/11/2019	9:36am	-0.46	935000
4	BA	98.31	6/11/2019	9:36am	0.12	104800
5	C	53.08	6/11/2019	9:36am	-0.25	360900
6	CAT	78.29	6/11/2019	9:36am	-0.23	225400
7						

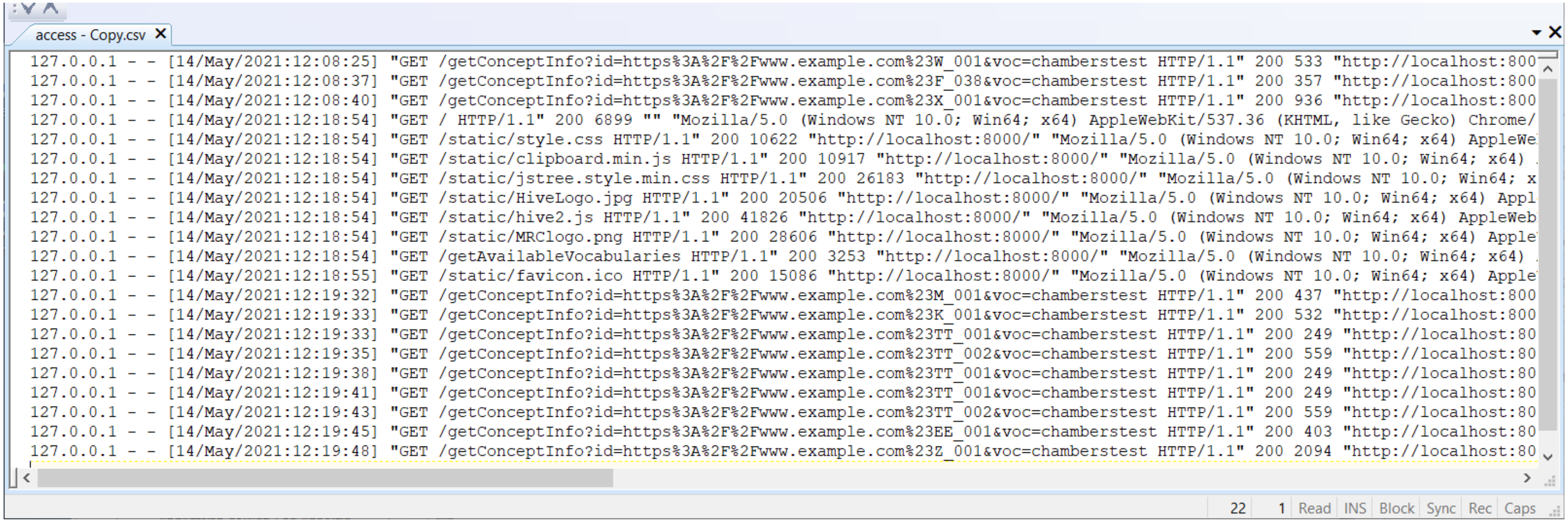
```python
stockfile = open('stocks.csv', 'r')
for line in stockfile:
    line = line.strip()
    column = line.split(',')
    print(column[0], "closed at ", column[1],
          "with", column[4], "change")
stockfile.close()
```

Output

```
"AA" closed at  39.48 with -0.18 change
"AIG" closed at  71.38 with -0.15 change
"AXP" closed at  62.58 with -0.46 change
"BA" closed at  98.31 with +0.12 change
"C" closed at  53.08 with -0.25 change
"CAT" closed at  78.29 with -0.23 change
```

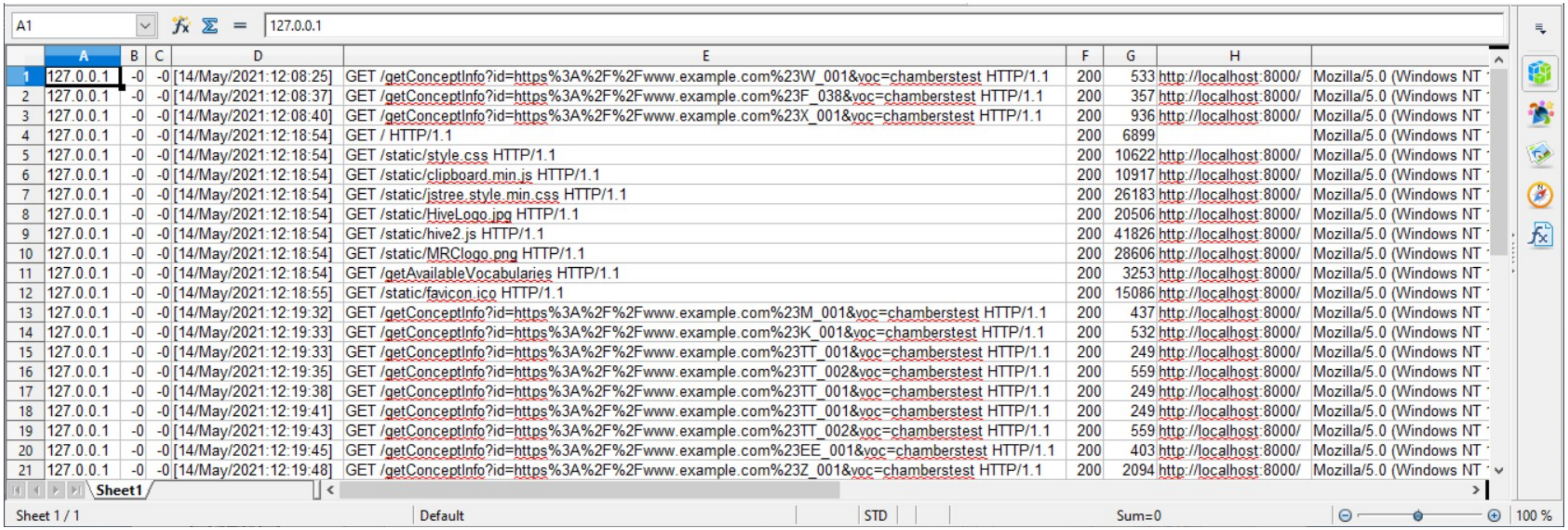
Web Access Logs are Tabular Data

Web access.log



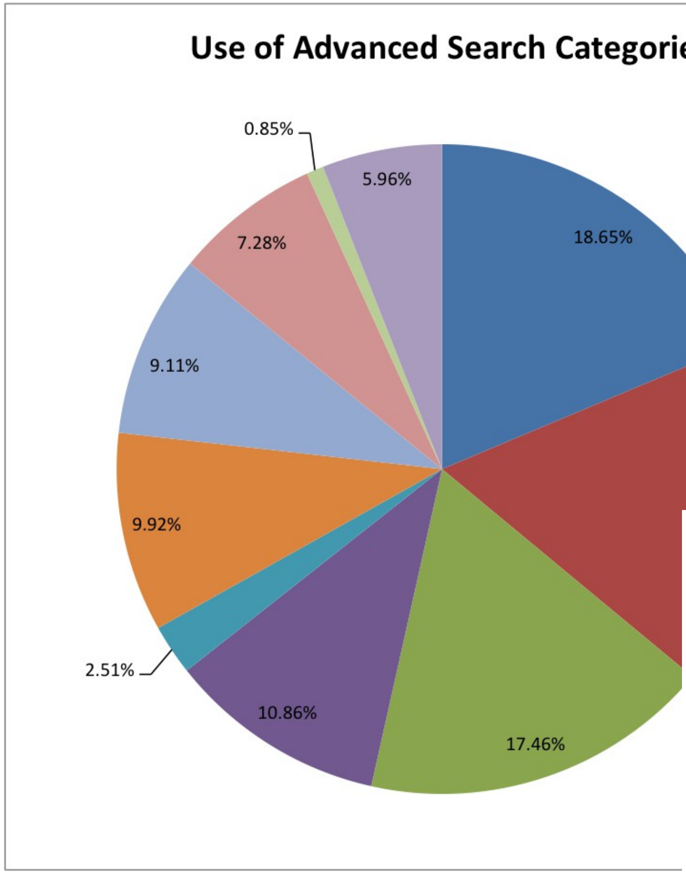
```
127.0.0.1 - - [14/May/2021:12:08:25] "GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23W_001&voc=chamberstest HTTP/1.1" 200 533 "http://localhost:800
127.0.0.1 - - [14/May/2021:12:08:37] "GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23F_038&voc=chamberstest HTTP/1.1" 200 357 "http://localhost:800
127.0.0.1 - - [14/May/2021:12:08:40] "GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23X_001&voc=chamberstest HTTP/1.1" 200 936 "http://localhost:800
127.0.0.1 - - [14/May/2021:12:18:54] "GET / HTTP/1.1" 200 6899 "" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
127.0.0.1 - - [14/May/2021:12:18:54] "GET /static/style.css HTTP/1.1" 200 10622 "http://localhost:8000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWe
127.0.0.1 - - [14/May/2021:12:18:54] "GET /static/clipboard.min.js HTTP/1.1" 200 10917 "http://localhost:8000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) 
127.0.0.1 - - [14/May/2021:12:18:54] "GET /static/jstree.style.min.css HTTP/1.1" 200 26183 "http://localhost:8000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x
127.0.0.1 - - [14/May/2021:12:18:54] "GET /static/HiveLogo.jpg HTTP/1.1" 200 20506 "http://localhost:8000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl
127.0.0.1 - - [14/May/2021:12:18:54] "GET /static/hive2.js HTTP/1.1" 200 41826 "http://localhost:8000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWeb
127.0.0.1 - - [14/May/2021:12:18:54] "GET /static/MRClogo.png HTTP/1.1" 200 28606 "http://localhost:8000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Apple
127.0.0.1 - - [14/May/2021:12:18:54] "GET /getAvailableVocabularies HTTP/1.1" 200 3253 "http://localhost:8000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) 
127.0.0.1 - - [14/May/2021:12:18:55] "GET /static/favicon.ico HTTP/1.1" 200 15086 "http://localhost:8000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Apple
127.0.0.1 - - [14/May/2021:12:19:32] "GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23M_001&voc=chamberstest HTTP/1.1" 200 437 "http://localhost:800
127.0.0.1 - - [14/May/2021:12:19:33] "GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23K_001&voc=chamberstest HTTP/1.1" 200 532 "http://localhost:800
127.0.0.1 - - [14/May/2021:12:19:33] "GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23TT_001&voc=chamberstest HTTP/1.1" 200 249 "http://localhost:80
127.0.0.1 - - [14/May/2021:12:19:35] "GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23TT_002&voc=chamberstest HTTP/1.1" 200 559 "http://localhost:80
127.0.0.1 - - [14/May/2021:12:19:38] "GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23TT_001&voc=chamberstest HTTP/1.1" 200 249 "http://localhost:80
127.0.0.1 - - [14/May/2021:12:19:41] "GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23TT_001&voc=chamberstest HTTP/1.1" 200 249 "http://localhost:80
127.0.0.1 - - [14/May/2021:12:19:43] "GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23TT_002&voc=chamberstest HTTP/1.1" 200 559 "http://localhost:80
127.0.0.1 - - [14/May/2021:12:19:45] "GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23EE_001&voc=chamberstest HTTP/1.1" 200 403 "http://localhost:80
127.0.0.1 - - [14/May/2021:12:19:48] "GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23Z_001&voc=chamberstest HTTP/1.1" 200 2094 "http://localhost:80
```

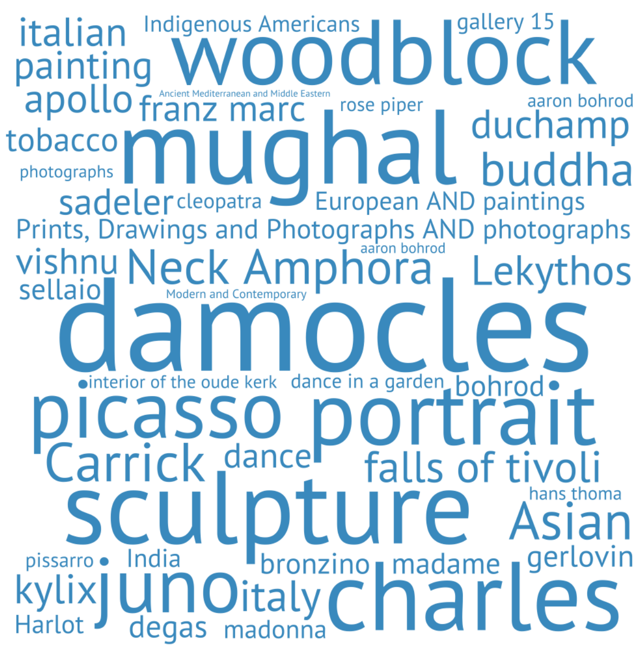
access.log in CSV format



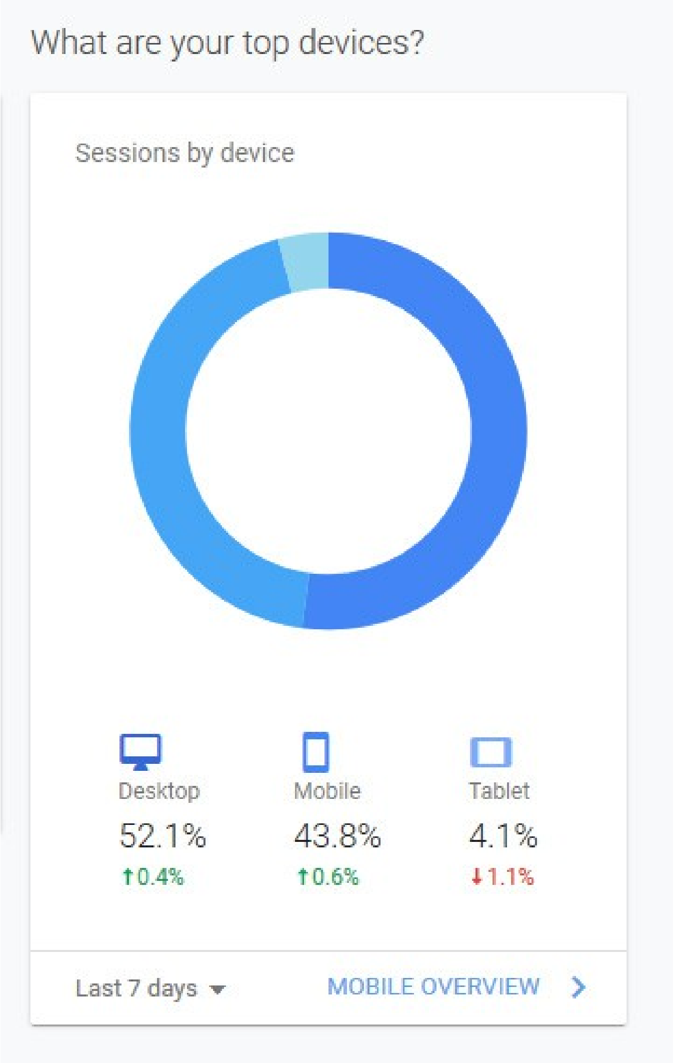
	A	B	C	D	E	F	G	H	
1	127.0.0.1	-0	-0	[14/May/2021:12:08:25]	GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23W_001&voc=chamberstest HTTP/1.1	200	533	http://localhost:8000/	Mozilla/5.0 (Windows NT
2	127.0.0.1	-0	-0	[14/May/2021:12:08:37]	GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23F_038&voc=chamberstest HTTP/1.1	200	357	http://localhost:8000/	Mozilla/5.0 (Windows NT
3	127.0.0.1	-0	-0	[14/May/2021:12:08:40]	GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23X_001&voc=chamberstest HTTP/1.1	200	936	http://localhost:8000/	Mozilla/5.0 (Windows NT
4	127.0.0.1	-0	-0	[14/May/2021:12:18:54]	GET / HTTP/1.1	200	6899		Mozilla/5.0 (Windows NT
5	127.0.0.1	-0	-0	[14/May/2021:12:18:54]	GET /static/style.css HTTP/1.1	200	10622	http://localhost:8000/	Mozilla/5.0 (Windows NT
6	127.0.0.1	-0	-0	[14/May/2021:12:18:54]	GET /static/clipboard.min.js HTTP/1.1	200	10917	http://localhost:8000/	Mozilla/5.0 (Windows NT
7	127.0.0.1	-0	-0	[14/May/2021:12:18:54]	GET /static/jstree.style.min.css HTTP/1.1	200	26183	http://localhost:8000/	Mozilla/5.0 (Windows NT
8	127.0.0.1	-0	-0	[14/May/2021:12:18:54]	GET /static/HiveLogo.jpg HTTP/1.1	200	20506	http://localhost:8000/	Mozilla/5.0 (Windows NT
9	127.0.0.1	-0	-0	[14/May/2021:12:18:54]	GET /static/hive2.js HTTP/1.1	200	41826	http://localhost:8000/	Mozilla/5.0 (Windows NT
10	127.0.0.1	-0	-0	[14/May/2021:12:18:54]	GET /static/MRClogo.png HTTP/1.1	200	28606	http://localhost:8000/	Mozilla/5.0 (Windows NT
11	127.0.0.1	-0	-0	[14/May/2021:12:18:54]	GET /getAvailableVocabularies HTTP/1.1	200	3253	http://localhost:8000/	Mozilla/5.0 (Windows NT
12	127.0.0.1	-0	-0	[14/May/2021:12:18:55]	GET /static/favicon.ico HTTP/1.1	200	15086	http://localhost:8000/	Mozilla/5.0 (Windows NT
13	127.0.0.1	-0	-0	[14/May/2021:12:19:32]	GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23M_001&voc=chamberstest HTTP/1.1	200	437	http://localhost:8000/	Mozilla/5.0 (Windows NT
14	127.0.0.1	-0	-0	[14/May/2021:12:19:33]	GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23K_001&voc=chamberstest HTTP/1.1	200	532	http://localhost:8000/	Mozilla/5.0 (Windows NT
15	127.0.0.1	-0	-0	[14/May/2021:12:19:33]	GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23TT_001&voc=chamberstest HTTP/1.1	200	249	http://localhost:8000/	Mozilla/5.0 (Windows NT
16	127.0.0.1	-0	-0	[14/May/2021:12:19:35]	GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23TT_002&voc=chamberstest HTTP/1.1	200	559	http://localhost:8000/	Mozilla/5.0 (Windows NT
17	127.0.0.1	-0	-0	[14/May/2021:12:19:38]	GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23TT_001&voc=chamberstest HTTP/1.1	200	249	http://localhost:8000/	Mozilla/5.0 (Windows NT
18	127.0.0.1	-0	-0	[14/May/2021:12:19:41]	GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23TT_001&voc=chamberstest HTTP/1.1	200	249	http://localhost:8000/	Mozilla/5.0 (Windows NT
19	127.0.0.1	-0	-0	[14/May/2021:12:19:43]	GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23TT_002&voc=chamberstest HTTP/1.1	200	559	http://localhost:8000/	Mozilla/5.0 (Windows NT
20	127.0.0.1	-0	-0	[14/May/2021:12:19:45]	GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23EE_001&voc=chamberstest HTTP/1.1	200	403	http://localhost:8000/	Mozilla/5.0 (Windows NT
21	127.0.0.1	-0	-0	[14/May/2021:12:19:48]	GET /getConceptInfo?id=https%3A%2F%2Fwww.example.com%23Z_001&voc=chamberstest HTTP/1.1	200	2094	http://localhost:8000/	Mozilla/5.0 (Windows NT

Analysis and Visualization of Web Logs





Searching for Art Records: A Log Analysis of the Ackland Art Museum's Collection Search System
by Meredith Hale


What are your top devices?

Sessions by device

Desktop	Mobile	Tablet
52.1%	43.8%	4.1%
↑0.4%	↑0.6%	↓1.1%

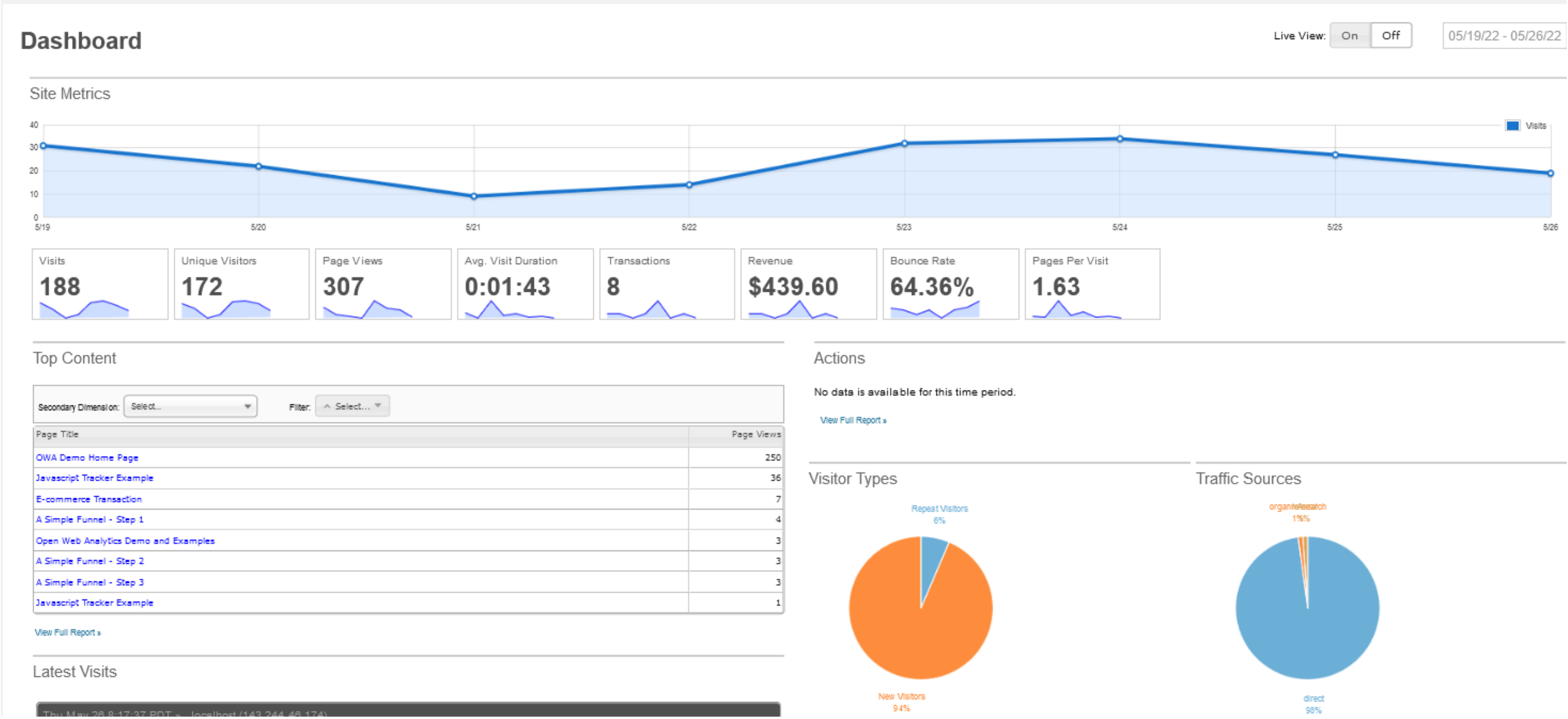
Last 7 days ▾

MOBILE OVERVIEW >

Google Analytics
for a website

Web Analytics:
Application of Web Log Analysis

Open Web Analytics



Dashboard

Live View: On Off

05/19/22 - 05/26/22

Site Metrics

Visits	Unique Visitors	Page Views	Avg. Visit Duration	Transactions	Revenue	Bounce Rate	Pages Per Visit
188	172	307	0:01:43	8	$439.60	64.36%	1.63

Top Content

Secondary Dimension: Select... Filter: Select...

Page Title	Page Views
OWA Demo Home Page	250
Javascript Tracker Example	36
E-commerce Transaction	7
A Simple Funnel - Step 1	4
Open Web Analytics Demo and Examples	3
A Simple Funnel - Step 2	3
A Simple Funnel - Step 3	3
Javascript Tracker Example	1

View Full Report »

Latest Visits

Thu May 26 8:17:37 PDT » localhost (143.244.48.174)

Actions

No data is available for this time period.

View Full Report »

Visitor Types

Traffic Sources

Structured Data
Standardized Formats: XML and JSON

XML

```xml
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

JSON

```json
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

Python Support for Text Processing

Many built-in and third party libraries

- NLTK for natural language processing
- Sci-kit for machine learning
- lxml for processing XML and HTML
- Beautiful Soup, Scrapy.org for screen-scraping
- NumPy, pandas for scientific computing and data analysis

Common text processing techniques for structured data

- Regular expressions
- XML parsing
- JSON parsing

Part 1: Overview

Part 2: Structured data: XML and JSON

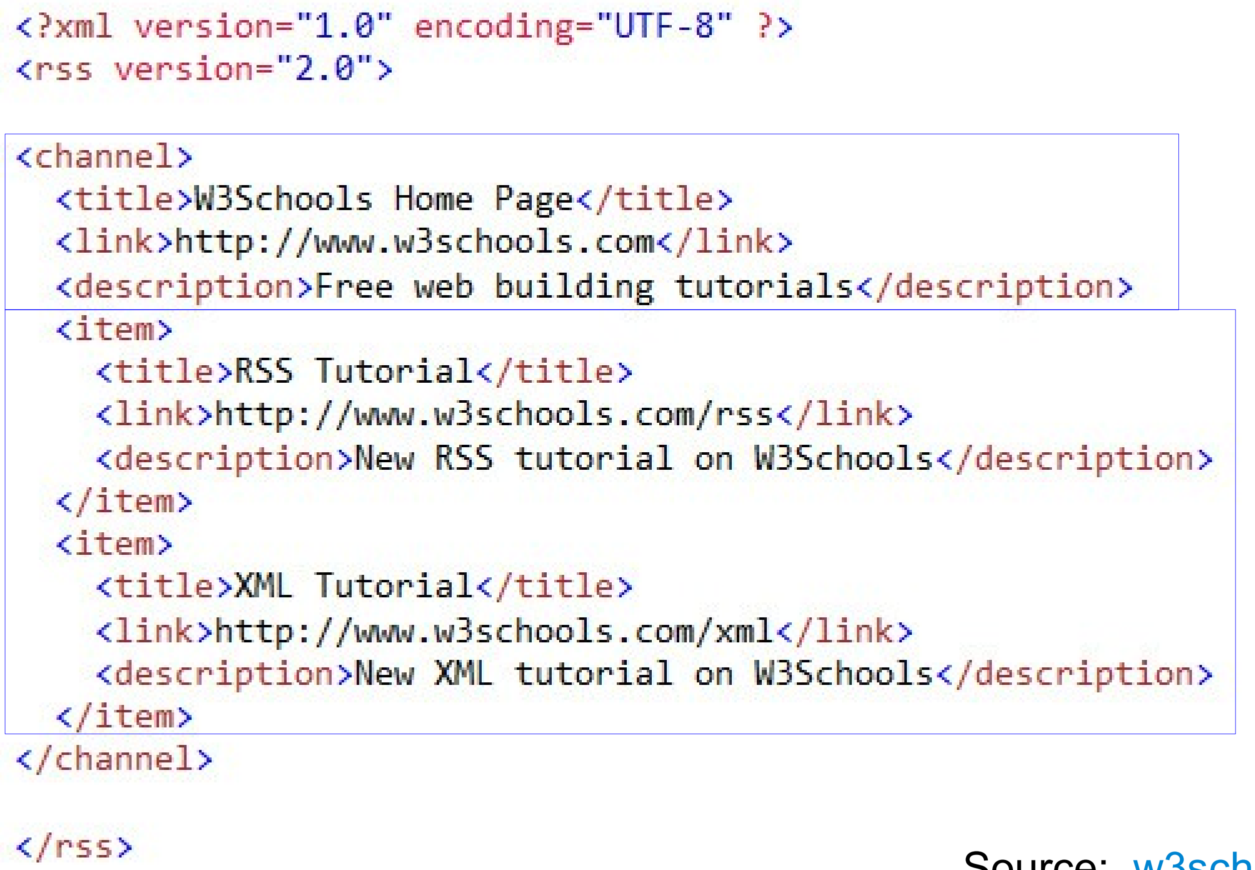
Part 3: Parsing JSON data with Python

XML Data Format

- eXtensible Markup Language (XML) is a set of rules for encoding documents in machine-readable form
- Some popular uses
  - Data interchange: sharing information in a standardized and descriptive format, often among heterogeneous applications
  - Publication, re-purposing: database content can be exported as XML and then converted to HTML for inclusion in websites
  - Content syndication: websites that frequently update their content (news websites or blogs) often provide an XML feed that other programs can use
- Parsing XML data is a common task for many kinds of applications

XML Example: RSS Feeds

- RSS (Really Simple Syndication) allows easy syndication of website content
- Useful for websites that are updated frequently, e.g., news sites, blogs, calendars. Examples: Wired, ESPN, NPR
- Written in XML. No official standard, but there is a specification (RSS 2.0) that defines the syntax rules.



```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">

<channel>
  <title>W3Schools Home Page</title>
  <link>http://www.w3schools.com</link>
  <description>Free web building tutorials</description>
  <item>
    <title>RSS Tutorial</title>
    <link>http://www.w3schools.com/rss</link>
    <description>New RSS tutorial on W3Schools</description>
  </item>
  <item>
    <title>XML Tutorial</title>
    <link>http://www.w3schools.com/xml</link>
    <description>New XML tutorial on W3Schools</description>
  </item>
</channel>

</rss>
```

`<channel>` element describes the RSS feed and has 3 required child elements

`<item>` elements define articles in the RSS feed and have 3 required child elements: **`<title>`**, **`<link>`**, and **`<description>`**

Source: w3schools XML RSS

JSON Data Format

JavaScript Object Notation (JSON) is a standard text format for representing structured data.

Similarities with XML

- Human/machine-readable and self-describing
- Hierarchical data format
- Language-independent (although the syntax is derived from that used by JavaScript to create objects)
- Parsers are available with many programming languages
- Used for data interchange, e.g., sending data from a server to a client based on a request

Some benefits of JSON over XML

- Lightweight, less verbose, simpler syntax
- Maps more directly to data structures of programming languages, e.g., JavaScript and Python

Why Python + JSON

- The proliferation of data, especially open data, creates opportunities for analysis, and for the extraction of information and insights from this data
- Much of this data is available in JSON format
- Python is an excellent programming language for analyzing structured data in many formats, including JSON
- Python can also be used to re-purpose data so that it is easier to understand, and to derive insights and trends. For example, rendering content in a more meaningful way on a web page, or visualizing patterns in charts
- But first, you need to parse the data to extract the information you want...

JSON Data Format

```
{"employees":[
     {"firstName":"John", "lastName":"Doe"},
     {"firstName":"Anna", "lastName":"Smith"},
     {"firstName":"Peter", "lastName":"Jones"}
   ]
}
```

JSON is built on two structures:

- A collection of name/value pairs (like a Python dictionary) delimited by **{ }**
- An ordered list of values (like a Python list) delimited by **[ ]**

Syntax is important!

- JSON requires double quotes to be used around strings and property names. Single quotes are not valid.
- Validation is important – a single misplaced comma or colon, or a mis-matched bracket, will make the JSON text impossible to parse.

JSONLint is a useful tool for validating and formatting JSON

w3schools: Python JSON

Basic Lists and Dictionaries in Python

`word_list`

```
['my',
'father',
'kept',
'the',
'admiral',
'benbow',
'inn',
'and',
'the',
'brown',
'old',
  ...
]
```

`word_frequency_dictionary`

```
{'man': 232,
'captain': 208,
'silver': 201,
'doctor': 159,
'time': 130,
'good': 123,
'hand': 119,
'long': 113,
'back': 106,
'cried': 103,
'hands': 102,
'sir': 101,
  ...
}
```

Part 1: Overview

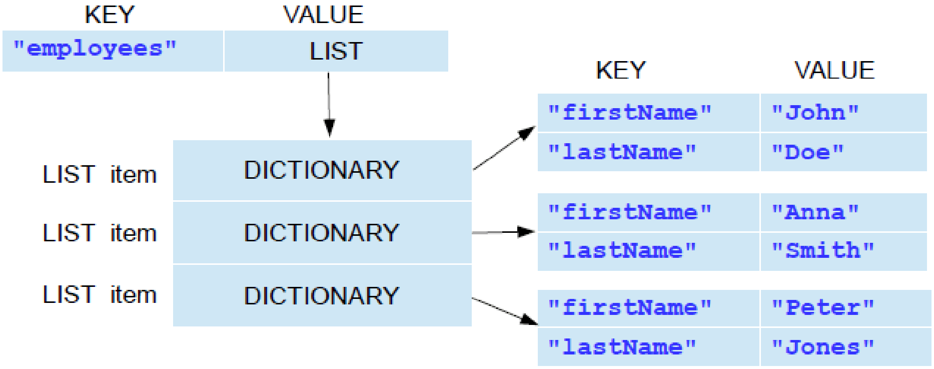
Part 2: Structured data: XML and JSON

Part 3: Parsing JSON data with Python

Parsing Employee Data in JSON Format

employee.json

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
  ]
}
```



KEY	VALUE
"employees"	LIST

LIST item	DICTIONARY
LIST item	DICTIONARY
LIST item	DICTIONARY

KEY	VALUE
"firstName"	"John"
"lastName"	"Doe"
"firstName"	"Anna"
"lastName"	"Smith"
"firstName"	"Peter"
"lastName"	"Jones"

parse_employee_json.py

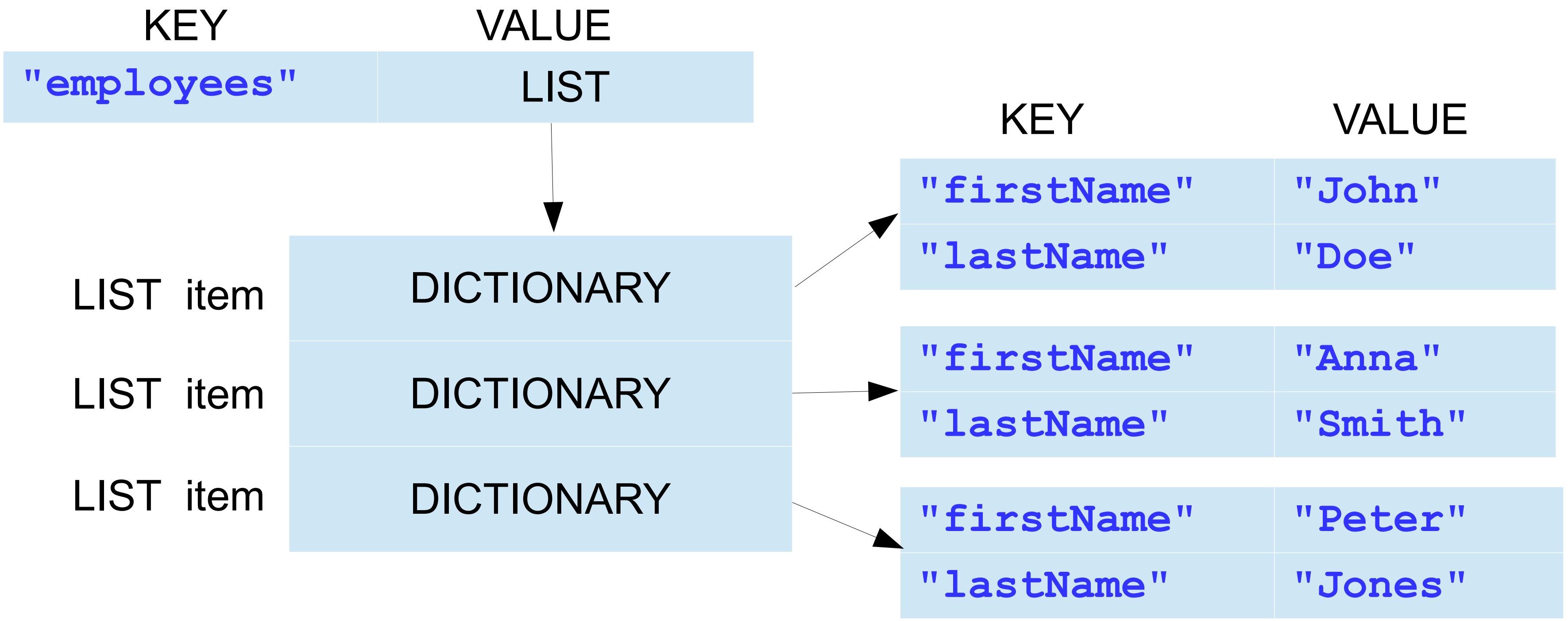
- Read input file
- Parse contents to produce Python dictionary and list
- Loop through list to print employee names

Output

```
John Doe
Anna Smith
Peter Jones
```

Python uses Dictionaries and Lists to represent JSON data

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
  ]
}
```



KEY	VALUE
"employees"	LIST

LIST item	DICTIONARY
LIST item	DICTIONARY
LIST item	DICTIONARY

KEY	VALUE
"firstName"	"John"
"lastName"	"Doe"

"firstName"	"Anna"
"lastName"	"Smith"

"firstName"	"Peter"
"lastName"	"Jones"

Parsing Employee JSON Data

```
import json
…

input_file = open('employee.json', 'r')
employee_info = input_file.read()
input_file.close()

# Parse the json data
employee_dictionary = json.loads(employee_info)

# Get the list of dictionaries
employee_list = employee_dictionary['employees']

# Loop through each dictionary, extract names
for employee in employee_list:
    firstName = employee['firstName']
    lastName = employee['lastName']
    print(firstName, lastName)
...
```

Import the module
for the JSON parser

Output

```
John Doe
Anna Smith
Peter Jones
```

`loads` converts a string containing JSON text into a Python dictionary

The dictionary has one entry where
key = `'employees'` and
value = a list of items that are dictionaries

Each `employee` is a dictionary with keys for `'firstName'` and `'lastName'`,

w3schools: Python JSON

parse_employee_json.py, employee.json

Jeopardy Data in JSON Format

200,000+ Jeopardy! Questions contains an unordered list of questions where each question is defined by

- 'category' : the question category, e.g. "HISTORY"
- 'value' : $ value of the question as string, e.g. "$200"
- 'question' : text of question
- 'answer' : text of answer
- 'round' : one of "Jeopardy!","Double Jeopardy!","Final Jeopardy!" or "Tiebreaker"
- 'show_number' : string of show number, e.g '4680'
- 'air_date' : the show air date in format YYYY-MM-DD

Jeopardy Data in JSON Format

```
[{
    "category": "HISTORY",
    "air_date": "2004-12-31",
    "question": "'For the last 8 years of his life, Galileo was under house
                 arrest for espousing this man's theory'",
    "value": "$200",
    "answer": "Copernicus",
    "round": "Jeopardy!",
    "show_number": "4680"
}, {
    "category": "ESPN's TOP 10 ALL-TIME ATHLETES",
    "air_date": "2004-12-31",
    "question": "'No. 2: 1912 Olympian; football star at Carlisle Indian School;
                 6 MLB seasons with the Reds, Giants & Braves'",
    "value": "$200",
    "answer": "Jim Thorpe",
    "round": "Jeopardy!",
    "show_number": "4680"
}, {
    "category": "EVERYBODY TALKS ABOUT IT...",
    "air_date": "2004-12-31",
    "question": "'The city of Yuma in this state has a record average of 4,055
                  hours of sunshine each year'",
    "value": "$200",
    "answer": "Arizona",
    "round": "Jeopardy!",
    "show_number": "4680"
},
. . .
]
```

`jeopardy.json` contains 15 Jeopardy questions, a very small subset of the original file.

Exercise: Parse Jeopardy JSON Data

- Read and parse the contents of `jeopardy.json`, like this:

```
jeopardy_questions = input_file.read()
question_list = json.loads(jeopardy_questions)
```

- NOTE: this JSON file is list of dictionaries, unlike the employee JSON file which is a little more complex (a dictionary of lists that contain dictionaries)
- Extract and display the Category, Question, and Answer

```
Category:  HISTORY
Question:  'For the last 8 years of his life, Galileo was under house arrest for espousing
            this man's theory'
Answer:  Copernicus

Category:  ESPN's TOP 10 ALL-TIME ATHLETES
Question:  'No. 2: 1912 Olympian; football star at Carlisle Indian School; 6 MLB seasons
            with the Reds, Giants & Braves'
Answer:  Jim Thorpe
. . .
Category:  EVERYBODY TALKS ABOUT IT...
Question:  'On June 28, 1994 the nat'l weather service began issuing this index that rates
            the intensity of the sun's radiation'
Answer:  the UV index

15 questions in this file
```

Sample Output