Lecture #2a: Data Engineering

S-109A Introduction to Data Science

Pavlos Protopapas and Kevin Rader

Outline

Part A: focus on data and relational storage

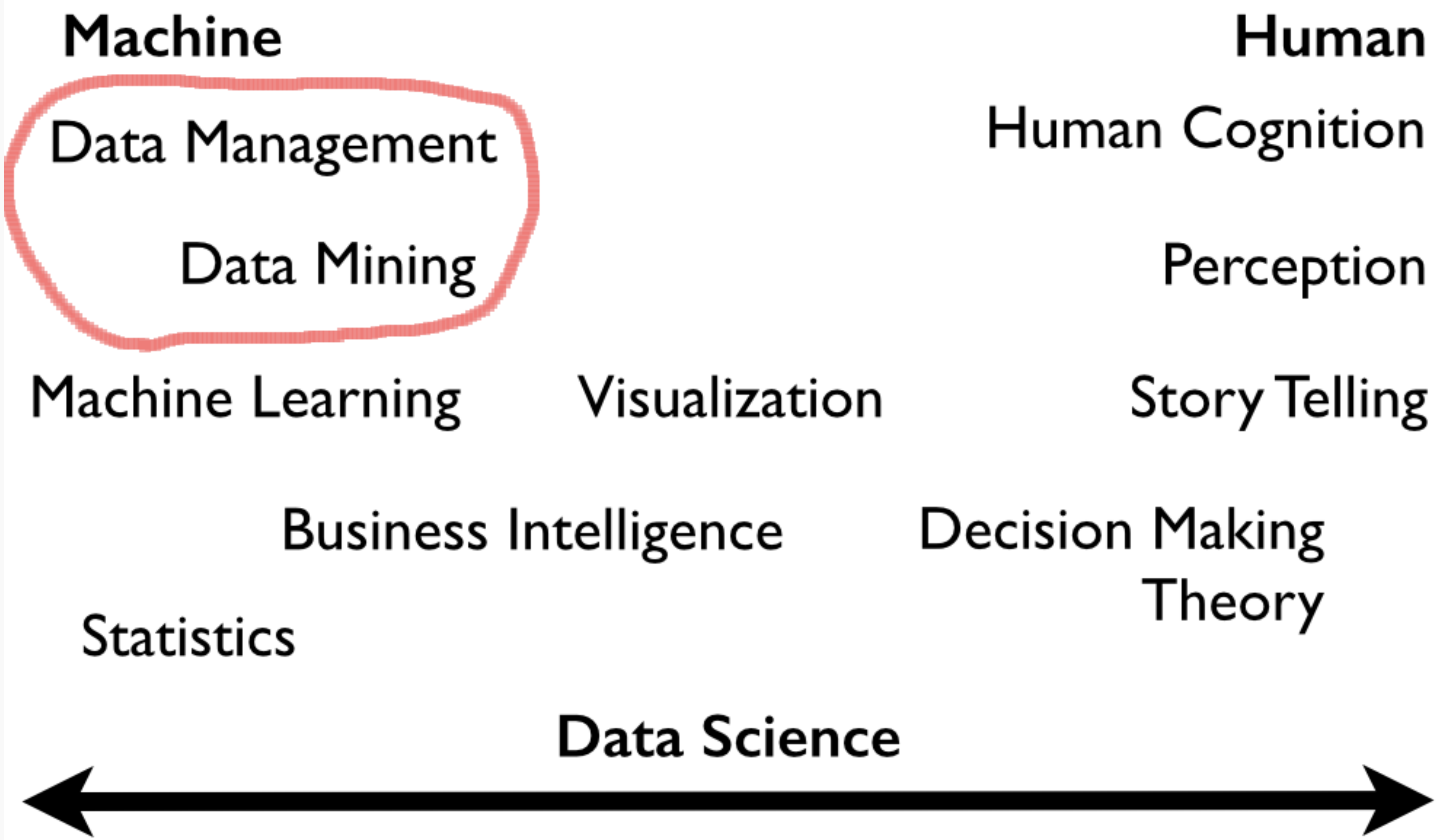
- How do we engineer features from the web?
- What is a relational Database?
- What Grammar of Data does it follow?
- How is this grammar implemented in Pandas?

Part B: focus on the visualization part of EDA.

In reality, both go together

It took about three years before the BellKor's Pragmatic Chaos team managed to win the prize … The winning algorithm was … so complex that it was never implemented by Netflix.[1]

[1] https://hbr.org/2012/10/big-data-hype-and-reality



Machine

Human

Data Management

Human Cognition

Data Mining

Perception

Machine Learning

Visualization

Story Telling

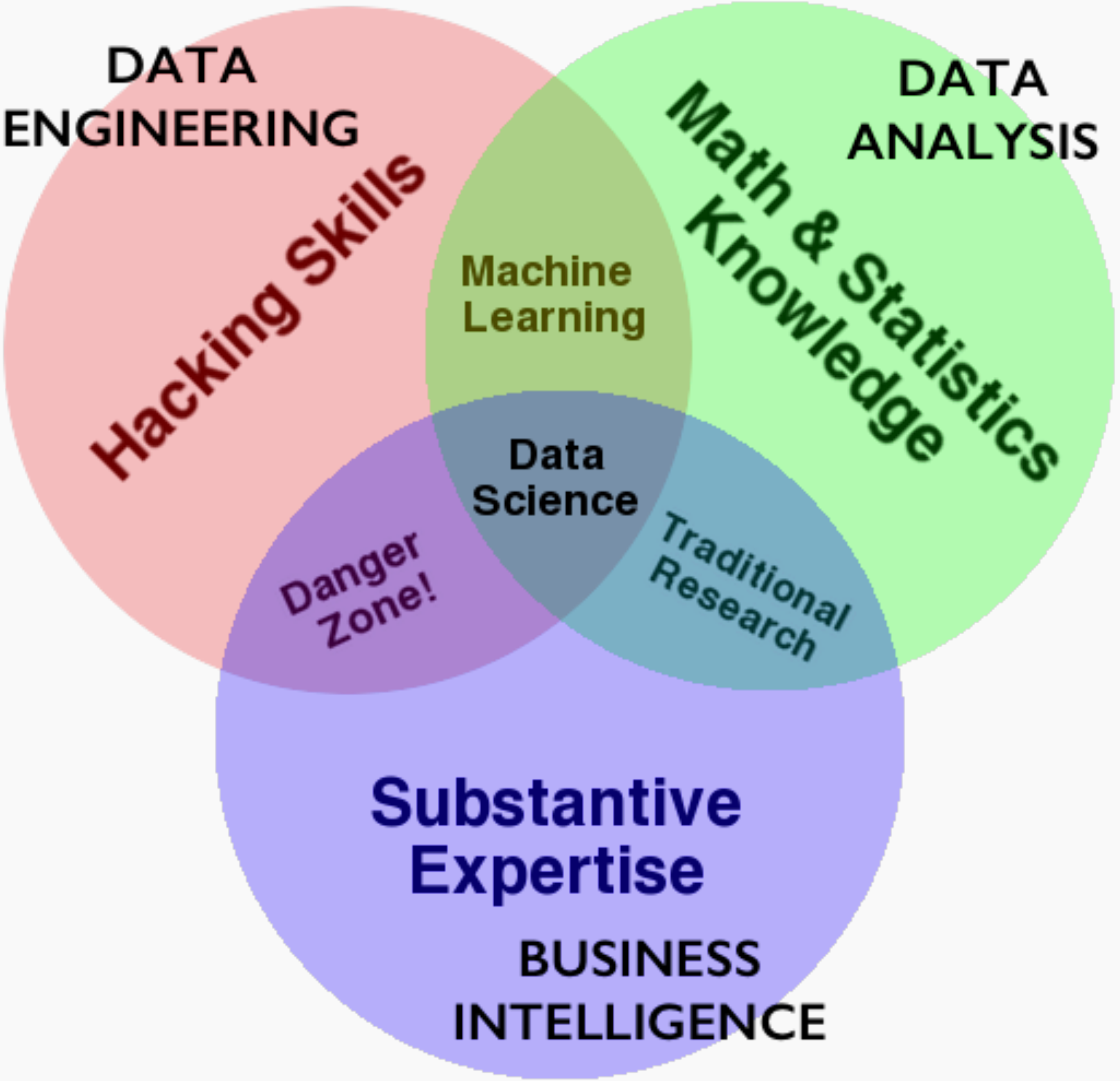
Business Intelligence

Decision Making Theory

Statistics

Data Science

Inspired by Daniel Keim, "Visual Analytics: Definition, Process, and Challenges"

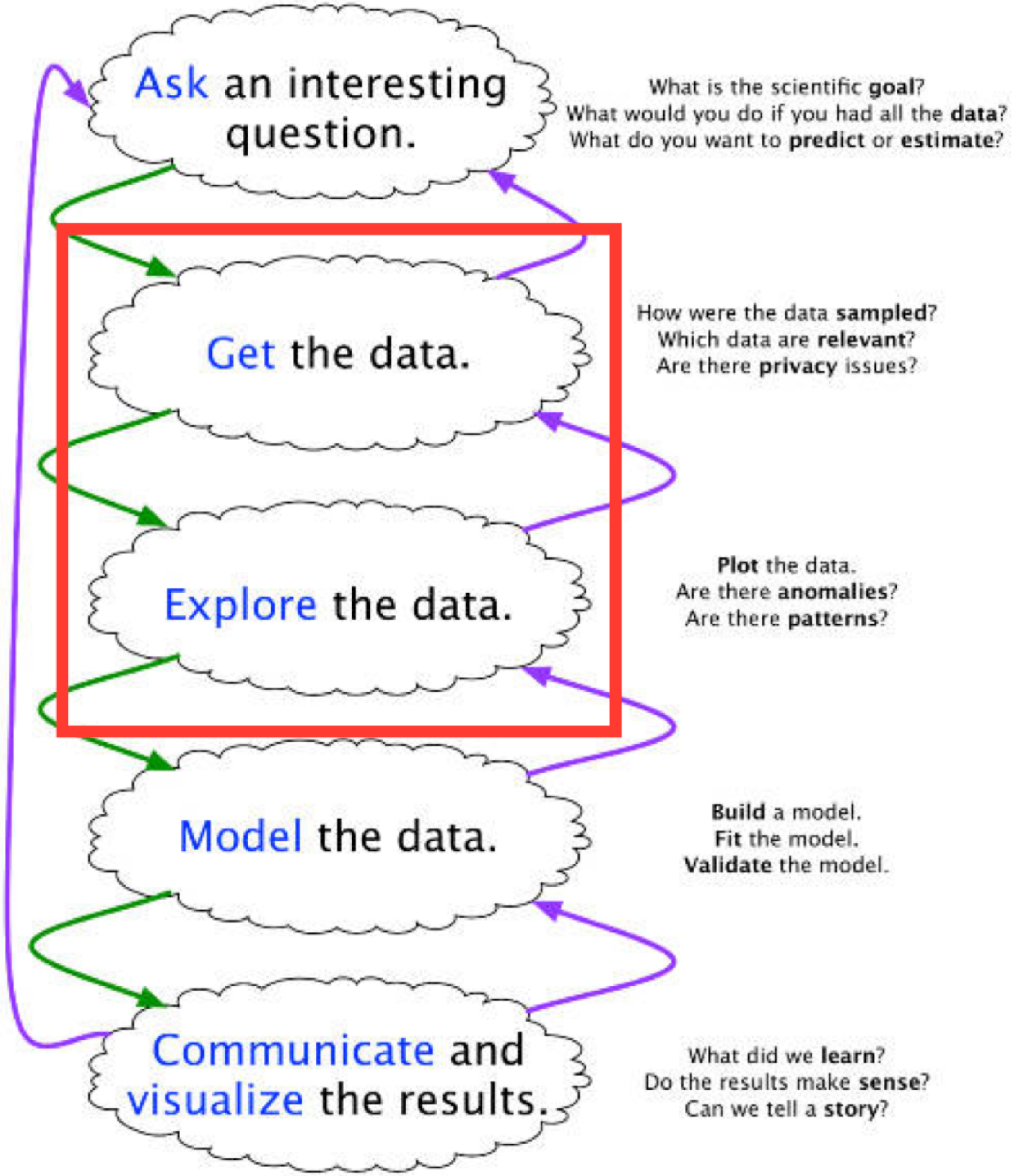

DATA ENGINEERING
Hacking Skills
Machine Learning
DATA ANALYSIS
Math & Statistics Knowledge
Data Science
Danger Zone!
Traditional Research
Substantive Expertise
BUSINESS INTELLIGENCE

Data Engineering

- **data**: scraping, API, feature engineering, all part of EDA
- **compute**: code, python, R, julia, spark, hadoop
- **storage/database**: pandas, SQL, NoSQL, HBase, disk, memory
- **devops**: AWS, docker, mesos, repeatability
- **product**: database, web, API, viz, UI, story

Different at different scales....

The Data Science Process



Ask an interesting question.

What is the scientific **goal**?
What would you do if you had all the **data**?
What do you want to **predict** or **estimate**?

Get the data.

How were the data **sampled**?
Which data are **relevant**?
Are there **privacy** issues?

Explore the data.

Plot the data.
Are there **anomalies**?
Are there **patterns**?

Model the data.

Build a model.
Fit the model.
Validate the model.

Communicate and visualize the results.

What did we **learn**?
Do the results make **sense**?
Can we tell a **story**?

Joe Blitzstein and Hanspeter Pfister, created for the Harvard data science course http://cs109.org/.

The basic EDA workflow[1]

1. **Build** a DataFrame from the data (ideally, put all data in this object)
2. **Clean** the DataFrame. It should have the following properties:
   - Each row describes a single object
   - Each column describes a property of that object
   - Columns are numeric whenever appropriate
   - Columns contain atomic properties that cannot be further decomposed
3. **Explore global properties**. Use histograms, scatter plots, and aggregation functions to summarize the data.
4. **Explore group properties**. Use groupby, queries, and small multiples to compare subsets of the data.

[1] enunciated in this form by Chris Beaumont, the first Head TF of cs109

Relational Database

Relational Database

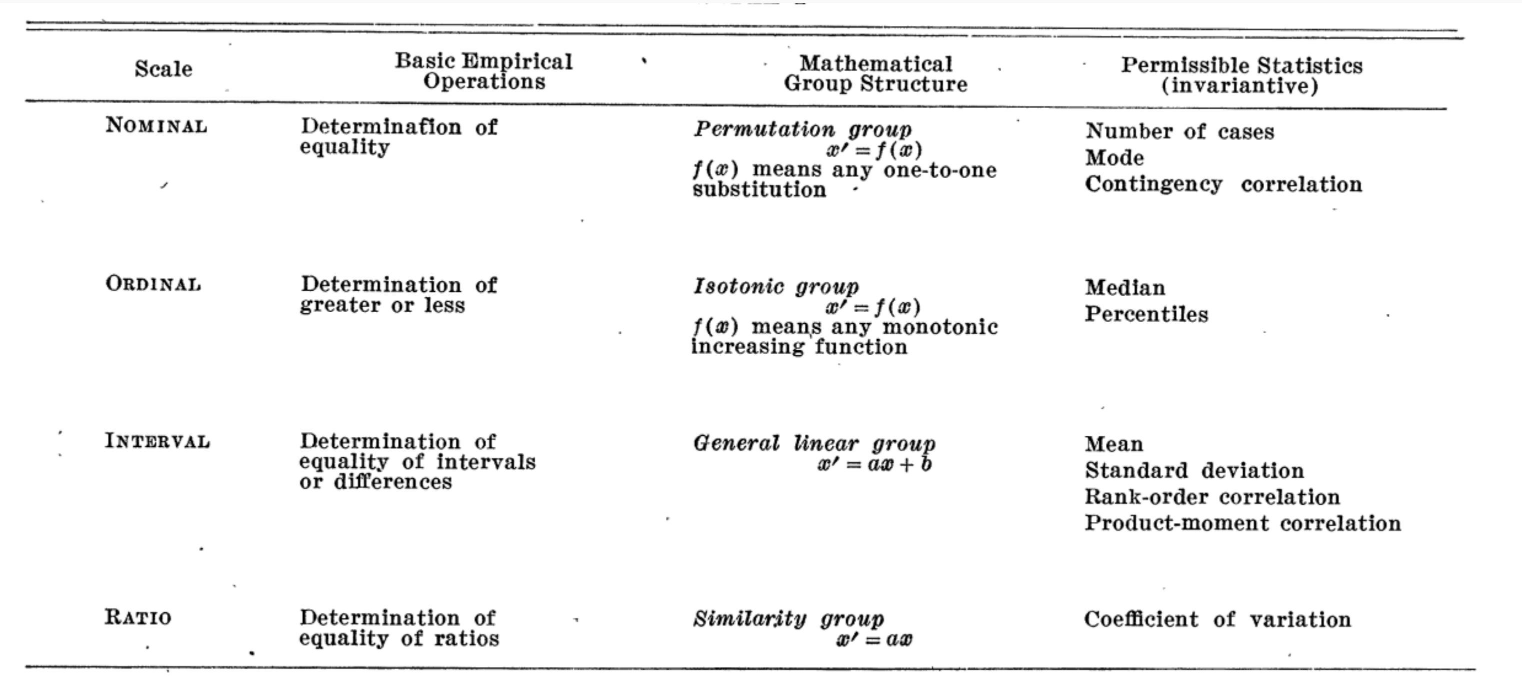
- Don't say_: seek 20 bytes onto disk and pick up from there. The next row is 50 bytes hence
- Say: select data from a set. I don't care where it is, just get the row to me.
- It’s just the table Kevin talked about last time …

Relational Database

- A collection of tables related to each other through common data values.
- Rows represent attributes of something
- Everything in a column is values of *one* attributes
- A cell is expected to be atomic
- Tables are related to each other if they have columns called keys which represent the same values

Scales of Measurement[1]

- Quantitative (Interval and Ratio)
- Ordinal
- Nominal



Scale	Basic Empirical Operations	Mathematical Group Structure	Permissible Statistics (invariantive)
NOMINAL	Determinafion of equality	*Permutation group* $x' = f(x)$ $f(x)$ means any one-to-one substitution	Number of cases Mode Contingency correlation
ORDINAL	Determination of greater or less	*Isotonic group* $x' = f(x)$ $f(x)$ means any monotonic increasing function	Median Percentiles
INTERVAL	Determination of equality of intervals or differences	*General linear group* $x' = ax + b$	Mean Standard deviation Rank-order correlation Product-moment correlation
RATIO	Determination of equality of ratios	*Similarity group* $x' = ax$	Coefficient of variation

[1] S. S. Stevens, Science, New Series, Vol. 103, No. 2684 (Jun. 7, 1946), pp. 677-680

Grammar of Data

Grammar of Data

Been there for a while (SQL, Pandas), formalized in dplyr[4]:

- provide simple verbs for simple things. These are func8ons corresponding tO common data manipulation tasks.
- second idea is that backend does not matter. Here we constrain ourselves to Pandas.
- multiple backends implemented in Pandas, Spark, Impala, Pig, dplyr, ibis, blaze

[4] Hadley Wickham: https://cran.rstudio.com/web/packages/dplyr/vigne0es/introducton.html

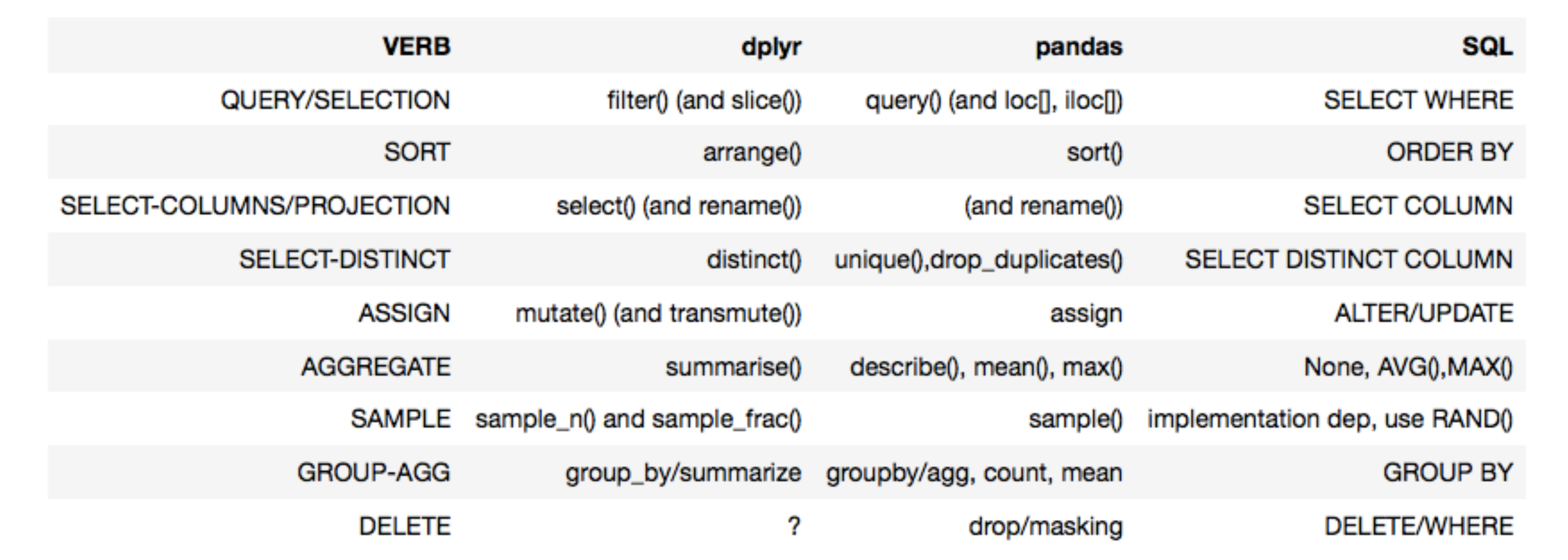
Grammar of Data

Why bother?

- learn how to do core data manipula2ons, no matter what the system
- relational databases cri2cal for non-memory fits. Big installed base.
- one off questions: google, stack-overflow, http://chrisalbon.com

Grammar of Data

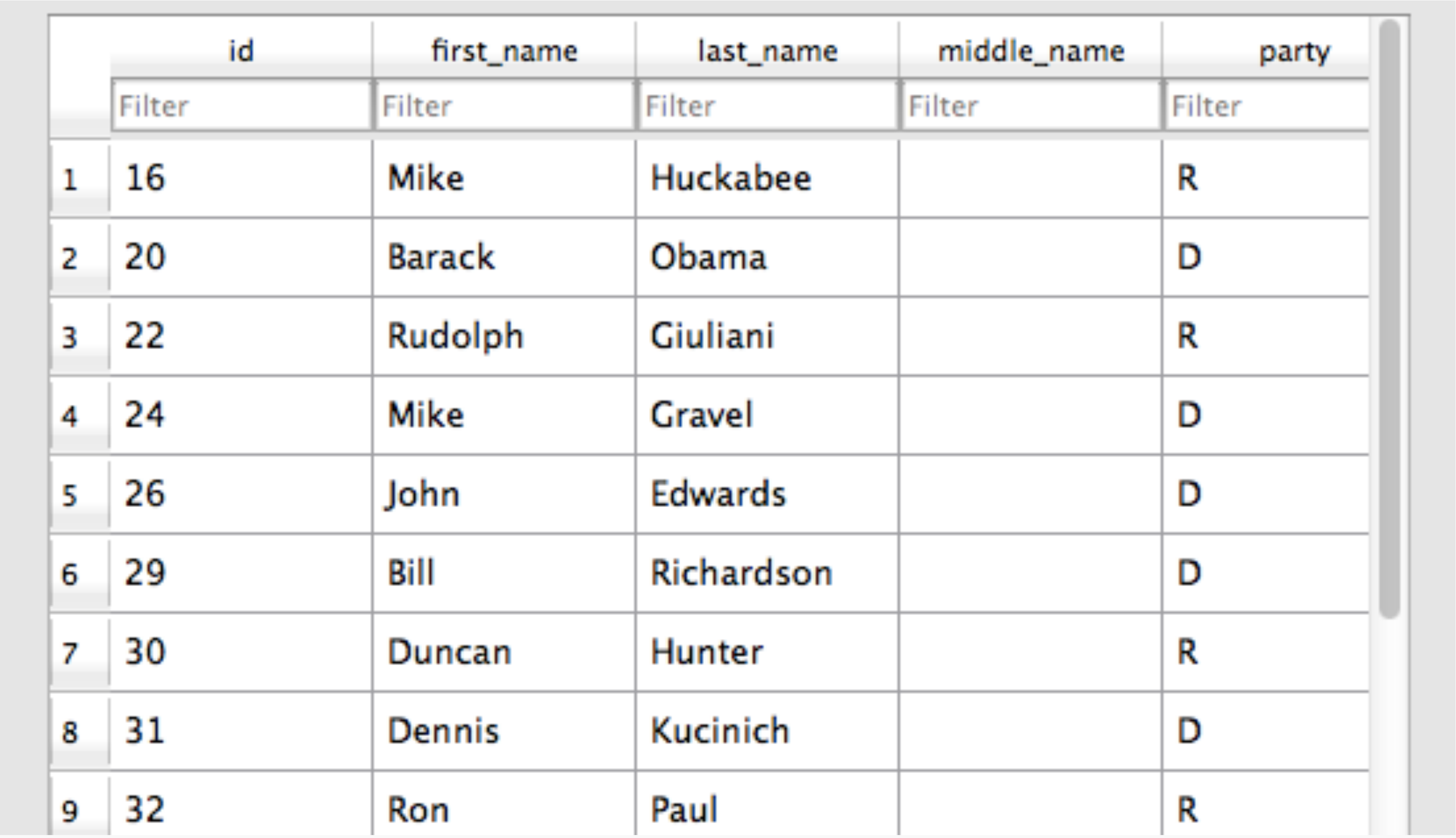
For cleaning and for transforma1on:



VERB	dplyr	pandas	SQL
QUERY/SELECTION	filter() (and slice())	query() (and loc[], iloc[])	SELECT WHERE
SORT	arrange()	sort()	ORDER BY
SELECT-COLUMNS/PROJECTION	select() (and rename())	(and rename())	SELECT COLUMN
SELECT-DISTINCT	distinct()	unique(),drop_duplicates()	SELECT DISTINCT COLUMN
ASSIGN	mutate() (and transmute())	assign	ALTER/UPDATE
AGGREGATE	summarise()	describe(), mean(), max()	None, AVG(),MAX()
SAMPLE	sample_n() and sample_frac()	sample()	implementation dep, use RAND()
GROUP-AGG	group_by/summarize	groupby/agg, count, mean	GROUP BY
DELETE	?	drop/masking	DELETE/WHERE

Grammar of Data

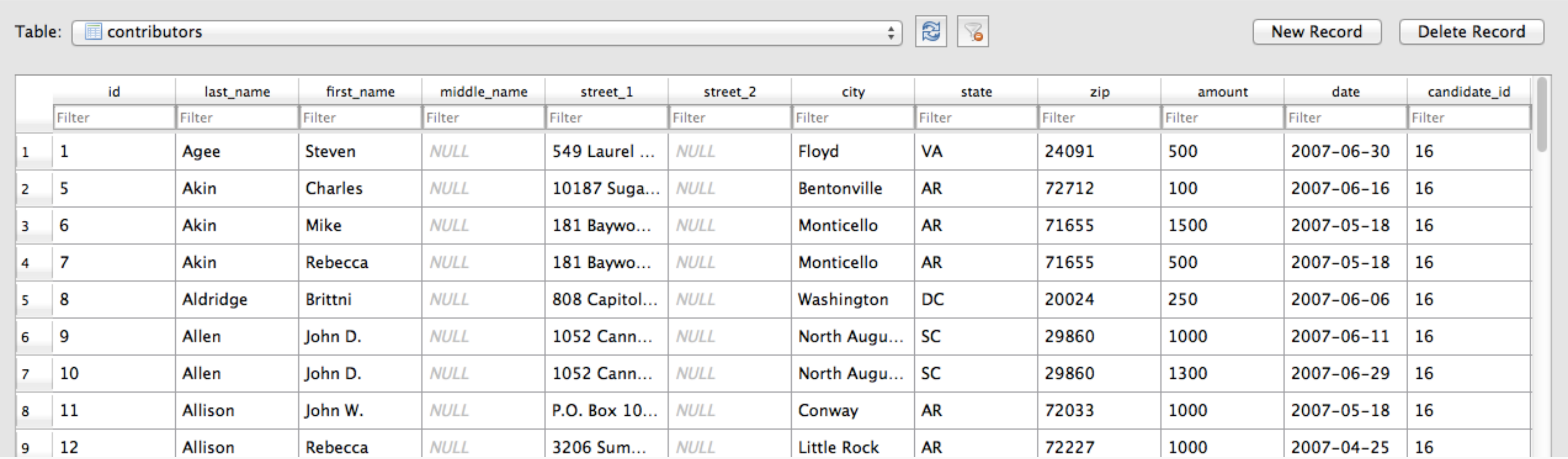
Example: Candidates



	id	first_name	last_name	middle_name	party
	Filter	Filter	Filter	Filter	Filter
1	16	Mike	Huckabee		R
2	20	Barack	Obama		D
3	22	Rudolph	Giuliani		R
4	24	Mike	Gravel		D
5	26	John	Edwards		D
6	29	Bill	Richardson		D
7	30	Duncan	Hunter		R
8	31	Dennis	Kucinich		D
9	32	Ron	Paul		R

Grammar of Data

Contributors



	id	last_name	first_name	middle_name	street_1	street_2	city	state	zip	amount	date	candidate_id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Agee	Steven	NULL	549 Laurel ...	NULL	Floyd	VA	24091	500	2007-06-30	16
2	5	Akin	Charles	NULL	10187 Suga...	NULL	Bentonville	AR	72712	100	2007-06-16	16
3	6	Akin	Mike	NULL	181 Baywo...	NULL	Monticello	AR	71655	1500	2007-05-18	16
4	7	Akin	Rebecca	NULL	181 Baywo...	NULL	Monticello	AR	71655	500	2007-05-18	16
5	8	Aldridge	Brittni	NULL	808 Capitol...	NULL	Washington	DC	20024	250	2007-06-06	16
6	9	Allen	John D.	NULL	1052 Cann...	NULL	North Augu...	SC	29860	1000	2007-06-11	16
7	10	Allen	John D.	NULL	1052 Cann...	NULL	North Augu...	SC	29860	1300	2007-06-29	16
8	11	Allison	John W.	NULL	P.O. Box 10...	NULL	Conway	AR	72033	1000	2007-05-18	16
9	12	Allison	Rebecca	NULL	3206 Sum...	NULL	Little Rock	AR	72227	1000	2007-04-25	16

Grammar of Data

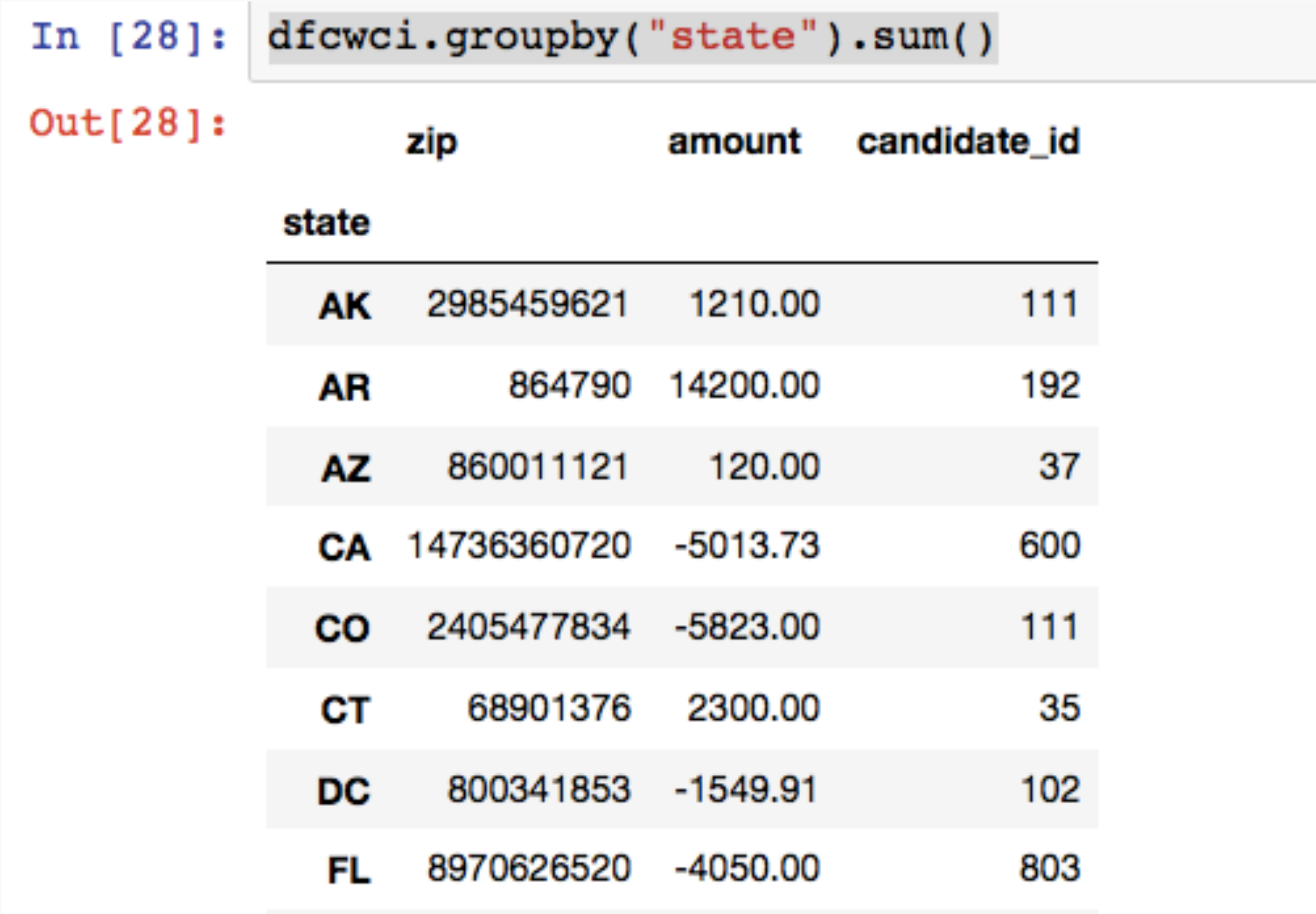
Operations:

- QUERY: dfcwci[(dfcwci.state=='VA') & (dfcwci.amount < 400)]
- SORT: dfcwci.sort_values(by="amount", ascending=False)
- SELECT-COLUMNS: dfcwci[['first_name', 'amount']]
- SELECT-DISTINCT: dfcwci[['last_name','first_name']].drop_duplicates()
- ASSIGN: dfcwci['name']=dfcwci['last_name']+", "+dfcwci['first_name']
- ASSIGN(in-place): dfcwci.loc[dfcwci.state=='VA', 'name']="junk”
- AGGREGATE: dfcwci.amount.max(), dfcwci.describe() DELETE: del dfcwci['name'] (DROPCOLUMN)

Grammar of Data

Split-Apply-Combine:

- GROUP-AGG
- splitting the data into groups based on some criteria
- applying a function to each group independently
- combining the results into a data structure



```
In [28]: dfcwci.groupby("state").sum()
```

Out[28]:

	zip	amount	candidate_id
state			
AK	2985459621	1210.00	111
AR	864790	14200.00	192
AZ	860011121	120.00	37
CA	14736360720	-5013.73	600
CO	2405477834	-5823.00	111
CT	68901376	2300.00	35
DC	800341853	-1549.91	102
FL	8970626520	-4050.00	803

Grammar of Data

RELATIONSHIPS (in addition to rubric)

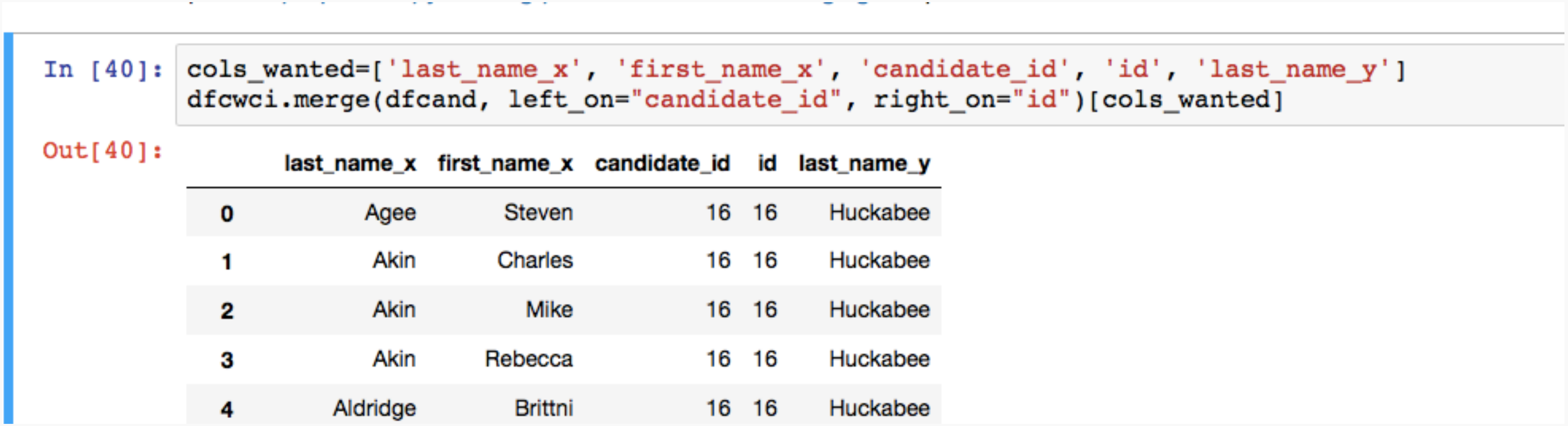
- we usually need to combine data from multiple sources
- different systems have different ways, most copy SQL (pandas)
- sub-select:

```
obamaid=dfcand.query("last_name=='Obama'")['id'].values[0]
obamacontrib=dfcwci.query("candidate_id==%i" % obamaid)
```

Grammar of Data

JOINS:

- combine tables on a common key-value
- 90% of the time, EXPLICIT INNER JOIN



```
In [40]: cols_wanted=['last_name_x', 'first_name_x', 'candidate_id', 'id', 'last_name_y']
         dfcwci.merge(dfcand, left_on="candidate_id", right_on="id")[cols_wanted]
```

Out[40]:

	last_name_x	first_name_x	candidate_id	id	last_name_y
0	Agee	Steven	16	16	Huckabee
1	Akin	Charles	16	16	Huckabee
2	Akin	Mike	16	16	Huckabee
3	Akin	Rebecca	16	16	Huckabee
4	Aldridge	Brittni	16	16	Huckabee

Web Servers

- A server is a long running process (also called daemon) which listens on a pre-specified port
- and responds to a request, which is sent using a protocol called HTTP
- A browser must first we must parse the url. Everything after a # is a fragment. Untill then it’s the DNS name or ip address, followed by the URL.

Web Servers

Example:

Our notebooks also talk to a local web server on our machines: `http://localhost:8888/Documents/cs109/BLA.ipynb#something`

- protocol is http, hostname is localhost, port is 8888
- url is /Documents/cs109/BLA.ipynb
- url fragment is #something

Request is sent to localhost on port 8888. It says:

```
Request: GET /request-URI HTTP/version
```

Example with Response: Google

```
GET / HTTP/1.0
Host: www.google.com

HTTP/1.0 200 OK
Date: Mon, 14 Nov 2016 04:49:02 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is ..."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: NID=90=gb5q7b0...; expires=Tue, 16-May-2017 04:49:02 GMT;
path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding

<!doctype html><html itemscope=""
itemtype="http://schema.org/WebPage" lang="en">
<head><meta content="Search the world's information,
```

HTTP Status Codes[1]

- 200 OK:

Means that the server did whatever the client wanted it to, and all is well.

- 201 Created:

The request has been fulfilled and resulted in a new resource being created. The newly created resource can be referenced by the URI(s) returned in the enpty of the response, with the most specific URI for the resource given by a Location header field.

- 400: Bad request

The request sent by the client didn't have the correct syntax.

- 401: Unauthorized

Means that the client is not allowed to access the resource. This may change if the client retries with an authorization header.

- 403: Forbidden

The client is not allowed to access the resource and authorizaton will not help.

- 404: Not found

Seen this one before? :) It means that the server has not heard of the resource and has no further clues as to what the client should do about it. In other words: dead link.

- 500: Internal server error

Something went wrong inside the server.

- 501: Not implemented

The request method is not supported by the server

[1] from http://www.garshol.priv.no/download/text/http-tut.htm)

Web Servers

Requests:

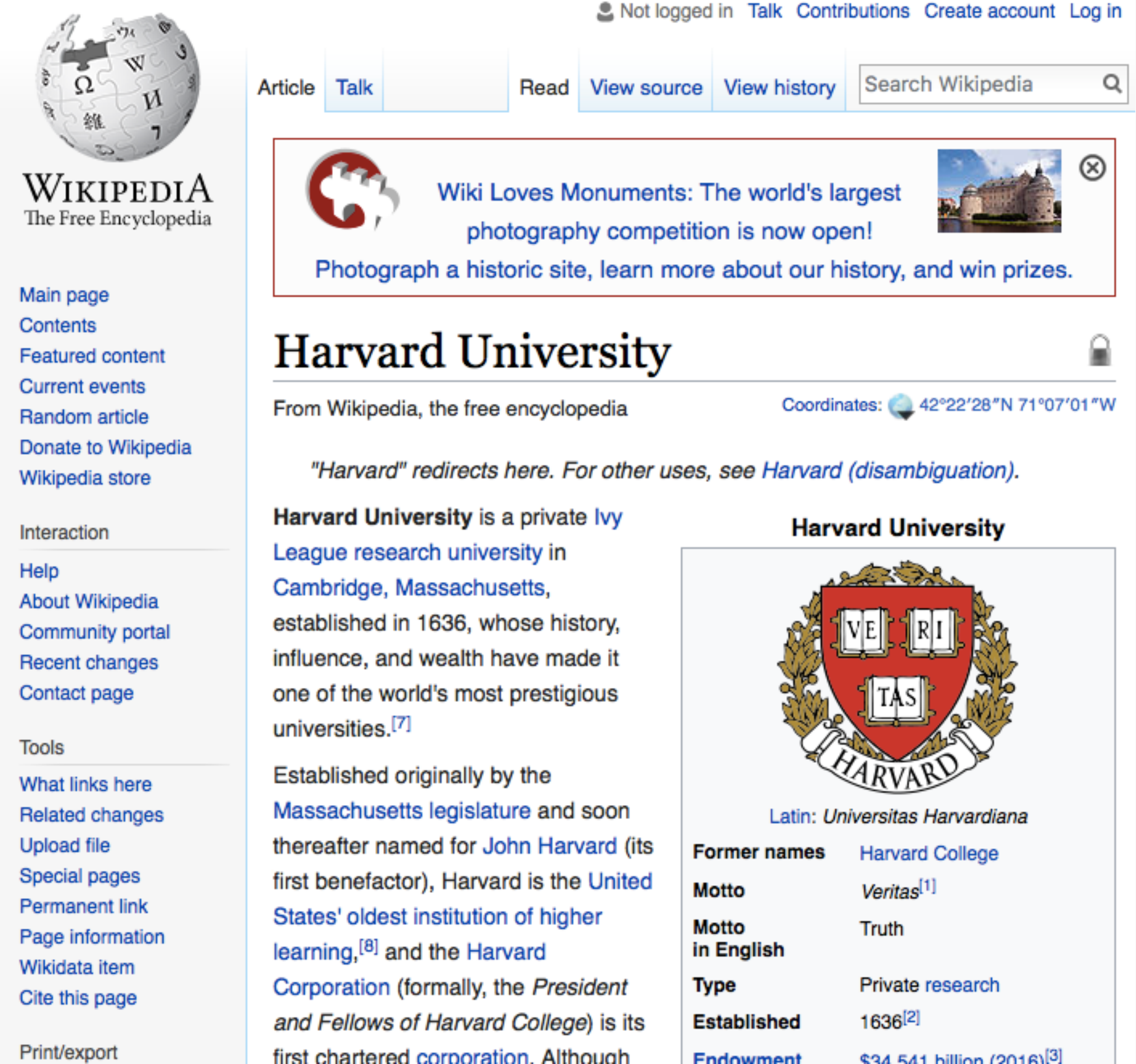
- great module built into python for http requests

```
req=requests.get("https://en.wikipedia.org/wiki/Harvard_University")
```

<Response [200]>

page = req.text

```
'<!DOCTYPE html>\n<html class="client-nojs" lang="en" dir="ltr">\n<head>\n<meta
charset="UTF-8"/>\n<title>Harvard University -
Wikipedia</title>\n<script>document.documentElement.className=document.documentElem
nt.className.replace( /(^|\\s)client-nojs(\\s|$)/,"$1client-js$2"
);</script>\n<script>(window.RLQ=window.RLQ||[]).push(function(){mw.config.set({
"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNamespaceNumber"
:0,"wgPageName":"Harvard_University","wgTitle":"Harva...'
```



WIKIPEDIA
The Free Encyclopedia

Wiki Loves Monuments: The world's largest photography competition is now open!
Photograph a historic site, learn more about our history, and win prizes.

Harvard University

From Wikipedia, the free encyclopedia

Coordinates: 42°22′28″N 71°07′01″W

"Harvard" redirects here. For other uses, see Harvard (disambiguation).

Harvard University is a private Ivy League research university in Cambridge, Massachusetts, established in 1636, whose history, influence, and wealth have made it one of the world's most prestigious universities.[7]

Established originally by the Massachusetts legislature and soon thereafter named for John Harvard (its first benefactor), Harvard is the United States' oldest institution of higher learning,[8] and the Harvard Corporation (formally, the *President and Fellows of Harvard College*) is its first chartered corporation. Although

Harvard University

Latin: *Universitas Harvardiana*

Former names	Harvard College
Motto	*Veritas*[1]
Motto in English	Truth
Type	Private research
Established	1636[2]
Endowment	$34.541 billion (2016)[3]

Python data scraping

Python data scraping

- Why scrape the web?
- vast source of information, combine with other data sets
- companies have not provided APIs
- automate tasks
- keep up with sites
- fun!

Python data scraping

copyrights and permission:

- be careful and polite
- give credit
- care about media law
- don't be evil (no spam, overloading sites, etc.)

Python data scraping

Robots.txt

- specified by web site owner
- gives instructions to web robots (aka your script)
- is located at the top-level directory of the web server
- e.g.: http://google.com/robots.txt

HTML

- angle brackets
- should be in pairs, eg `<p>Hello</p>`
- maybe in implicit bears, such as `<br/>`

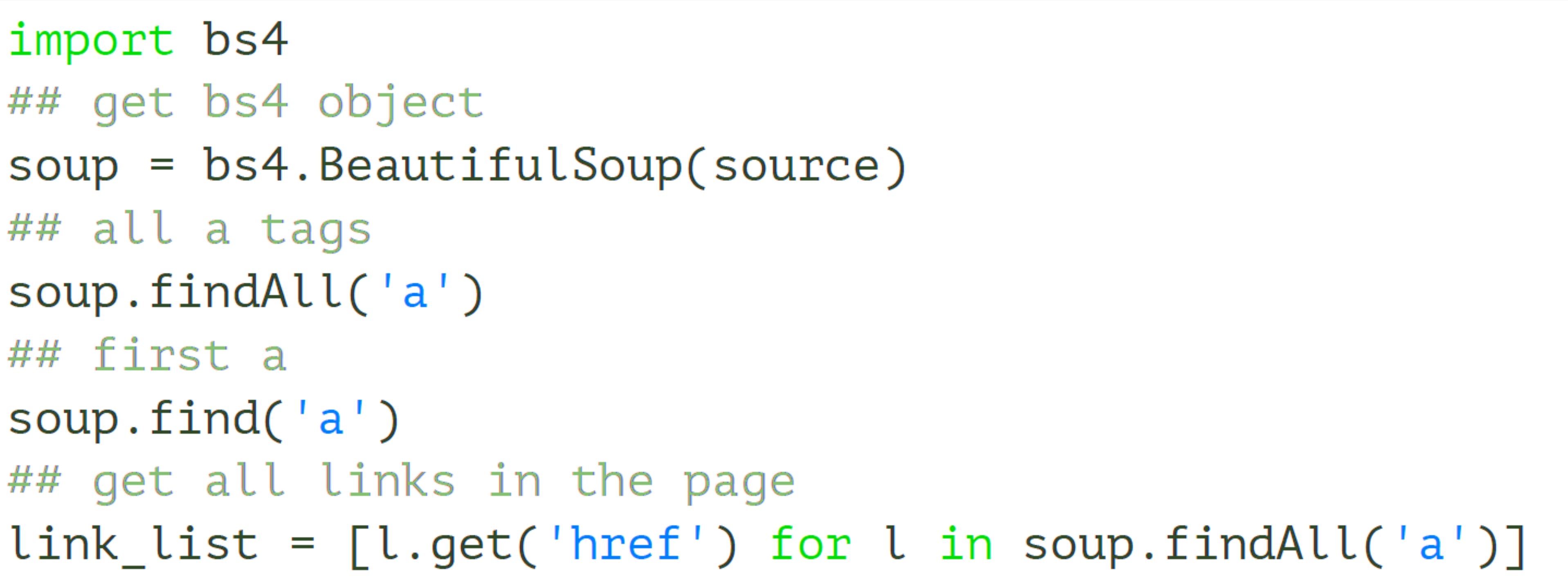
```
<!DOCTYPE html>
<html>
    <head>
        <title>Title</title>
    </head>
    <body>
        <h1>Body Title</h1>
        <p >Body Content</p>
    </body>
</html>
```

Developer Tools

- ctrl/cmd shi- i in chrome
- cmd-option-i in safari
- look for "inspect element"
- locate details of tags

Beautiful Soup

- will normalize dirty html
- basic usage



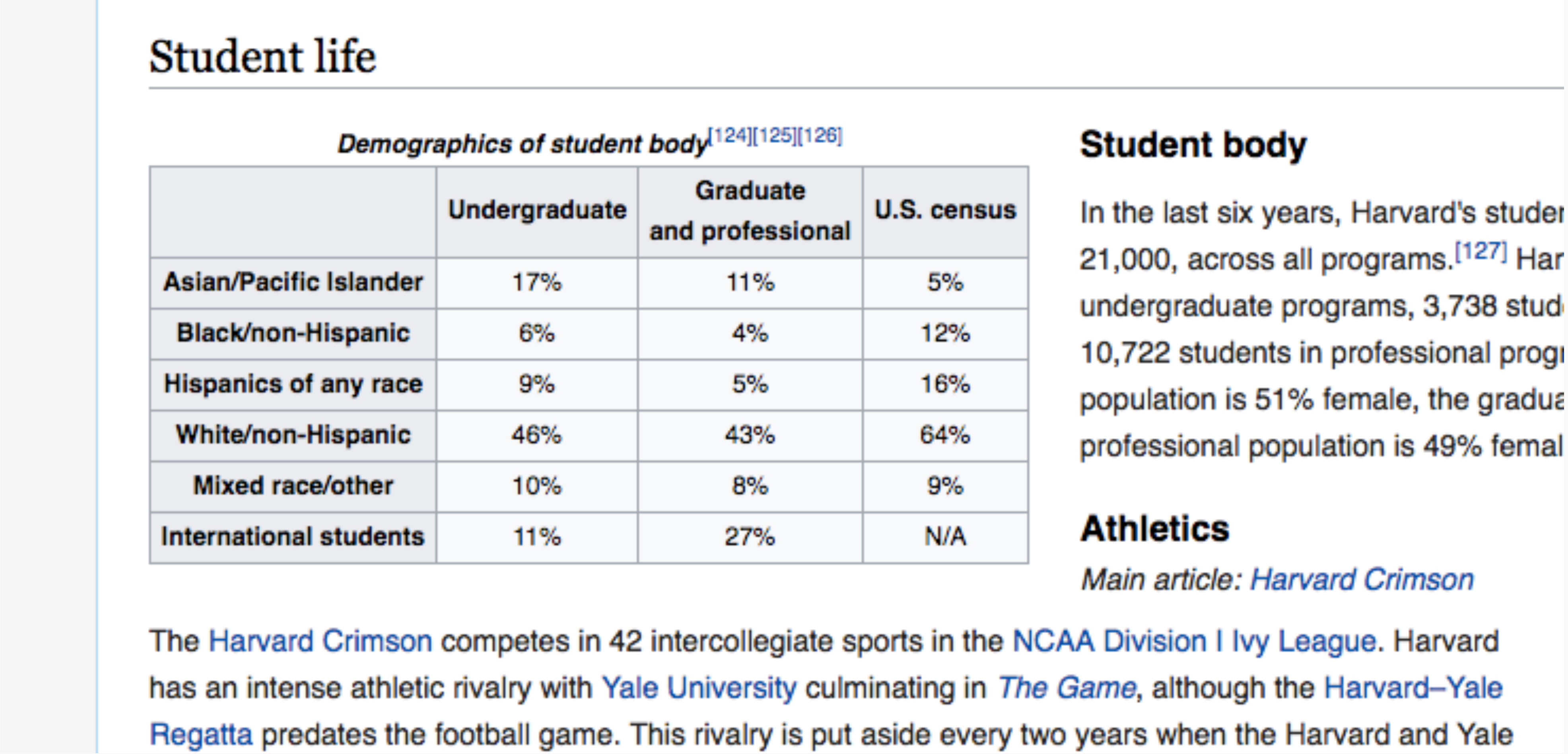
```
import bs4
## get bs4 object
soup = bs4.BeautifulSoup(source)
## all a tags
soup.findAll('a')
## first a
soup.find('a')
## get all links in the page
link_list = [l.get('href') for l in soup.findAll('a')]
```

HTML is a tree

```
tree = bs4.BeautifulSoup(source)

## get html root node
root_node = tree.html
## get head from root using contents
head = root_node.contents[0]
## get body from root
body = root_node.contents[1]
## could directly access body
tree.body
```

Demographics table we want



Student life

Demographics of student body[124][125][126]

	Undergraduate	Graduate and professional	U.S. census
Asian/Pacific Islander	17%	11%	5%
Black/non-Hispanic	6%	4%	12%
Hispanics of any race	9%	5%	16%
White/non-Hispanic	46%	43%	64%
Mixed race/other	10%	8%	9%
International students	11%	27%	N/A

Student body

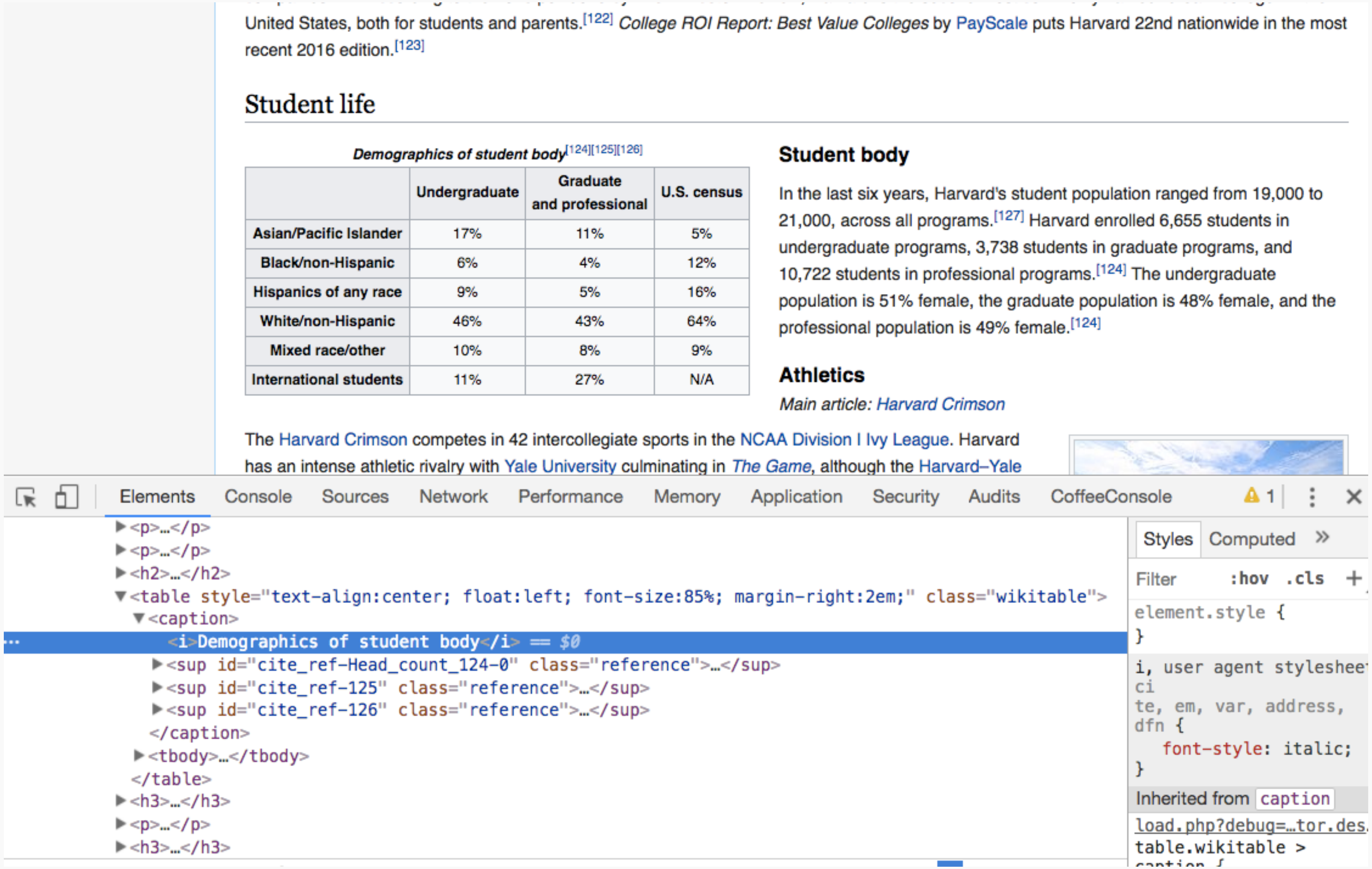
In the last six years, Harvard's studer
21,000, across all programs.[127] Har
undergraduate programs, 3,738 stud
10,722 students in professional progr
population is 51% female, the gradua
professional population is 49% femal

Athletics

Main article: Harvard Crimson

The Harvard Crimson competes in 42 intercollegiate sports in the NCAA Division I Ivy League. Harvard has an intense athletic rivalry with Yale University culminating in *The Game*, although the Harvard–Yale Regatta predates the football game. This rivalry is put aside every two years when the Harvard and Yale

Table with sole class wikitable



United States, both for students and parents.[122] *College ROI Report: Best Value Colleges* by PayScale puts Harvard 22nd nationwide in the most recent 2016 edition.[123]

Student life

Demographics of student body[124][125][126]

	Undergraduate	Graduate and professional	U.S. census
Asian/Pacific Islander	17%	11%	5%
Black/non-Hispanic	6%	4%	12%
Hispanics of any race	9%	5%	16%
White/non-Hispanic	46%	43%	64%
Mixed race/other	10%	8%	9%
International students	11%	27%	N/A

Student body

In the last six years, Harvard's student population ranged from 19,000 to 21,000, across all programs.[127] Harvard enrolled 6,655 students in undergraduate programs, 3,738 students in graduate programs, and 10,722 students in professional programs.[124] The undergraduate population is 51% female, the graduate population is 48% female, and the professional population is 49% female.[124]

Athletics

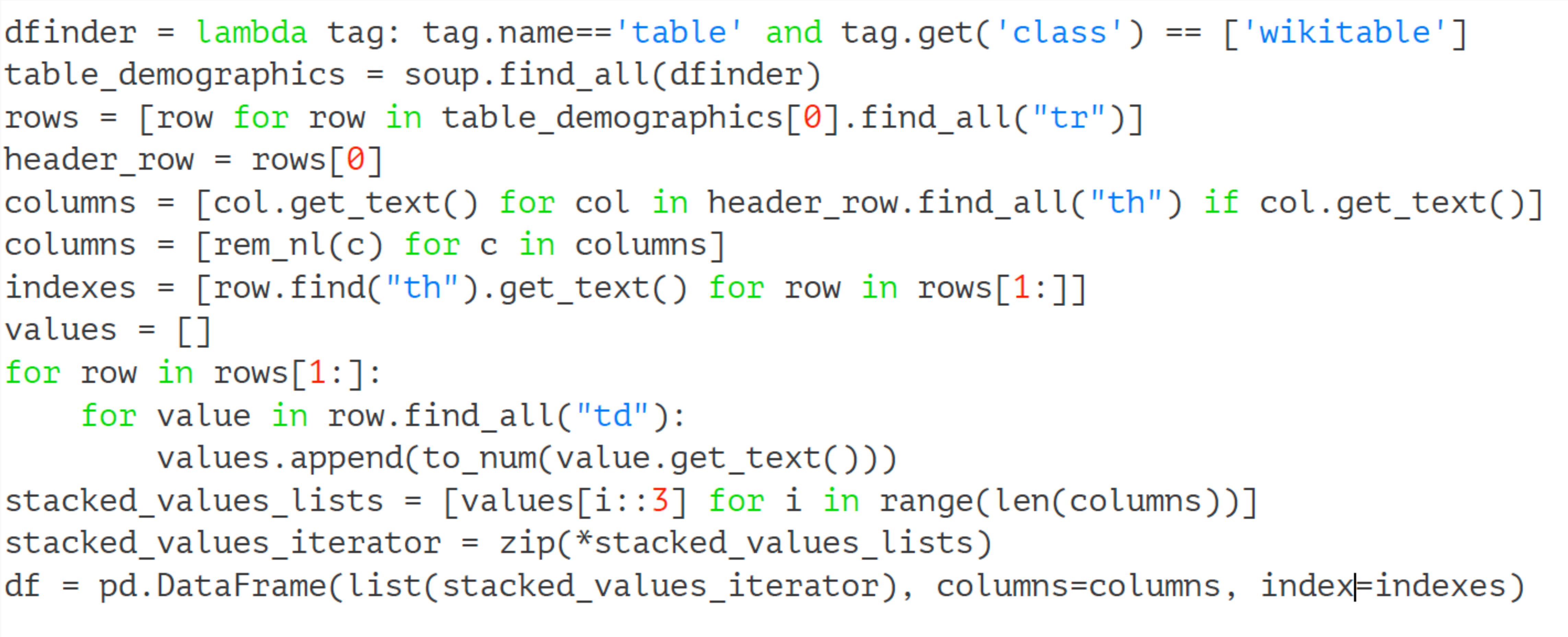
Main article: Harvard Crimson

The Harvard Crimson competes in 42 intercollegiate sports in the NCAA Division I Ivy League. Harvard has an intense athletic rivalry with Yale University culminating in *The Game*, although the Harvard–Yale

```
▶<p>…</p>
▶<p>…</p>
▶<h2>…</h2>
▼<table style="text-align:center; float:left; font-size:85%; margin-right:2em;" class="wikitable">
  ▼<caption>
      <i>Demographics of student body</i> == $0
    ▶<sup id="cite_ref-Head_count_124-0" class="reference">…</sup>
    ▶<sup id="cite_ref-125" class="reference">…</sup>
    ▶<sup id="cite_ref-126" class="reference">…</sup>
    </caption>
  ▶<tbody>…</tbody>
  </table>
▶<h3>…</h3>
▶<p>…</p>
▶<h3>…</h3>
```

```
element.style {
}
i, user agent styleshee
ci
te, em, var, address,
dfn {
   font-style: italic;
}
Inherited from caption
load.php?debug=…tor.des
table.wikitable >
caption {
```

Beautiful Soup Code



```
dfinder = lambda tag: tag.name=='table' and tag.get('class') == ['wikitable']
table_demographics = soup.find_all(dfinder)
rows = [row for row in table_demographics[0].find_all("tr")]
header_row = rows[0]
columns = [col.get_text() for col in header_row.find_all("th") if col.get_text()]
columns = [rem_nl(c) for c in columns]
indexes = [row.find("th").get_text() for row in rows[1:]]
values = []
for row in rows[1:]:
    for value in row.find_all("td"):
        values.append(to_num(value.get_text()))
stacked_values_lists = [values[i::3] for i in range(len(columns))]
stacked_values_iterator = zip(*stacked_values_lists)
df = pd.DataFrame(list(stacked_values_iterator), columns=columns, index=indexes)
```