

# [30 | ] Tabular Data

Tyler Caraza-Harter

# Learning Objectives Today

## CSV format

- purpose
- syntax
- comparison to spreadsheet

## Reading CSV files

- without header
- with header
- type casting

Chapter 14 of Sweigart, to (and including)  
“Reading Data from Reader Objects in a for Loop”

# Today's Outline

**Spreadsheets**

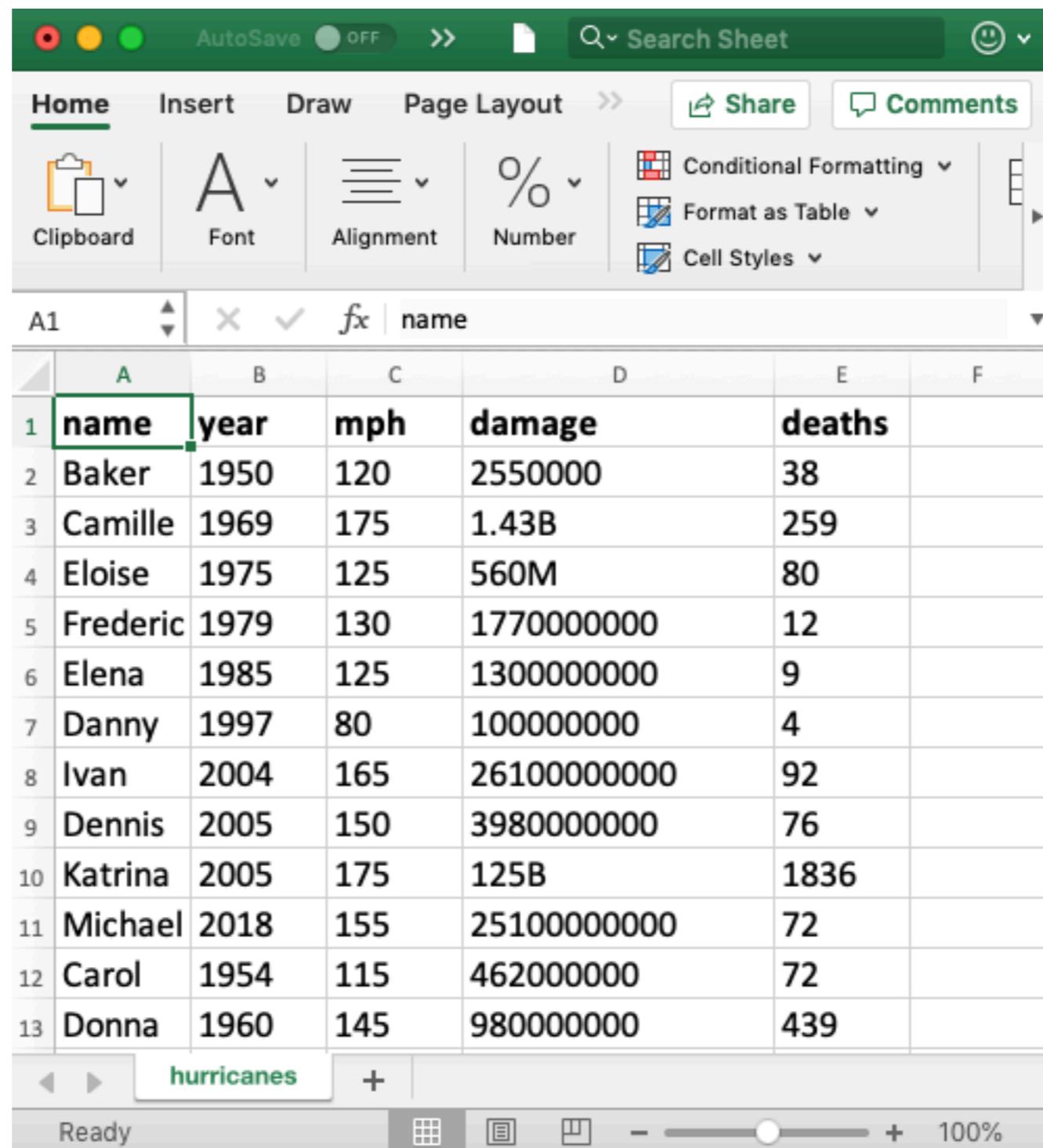
**CSVs**

**Reading a CSV to a list of lists**

**Coding examples**

# Spreadsheets (e.g., Excel)

Spreadsheets are tables of cells, organized by rows and columns



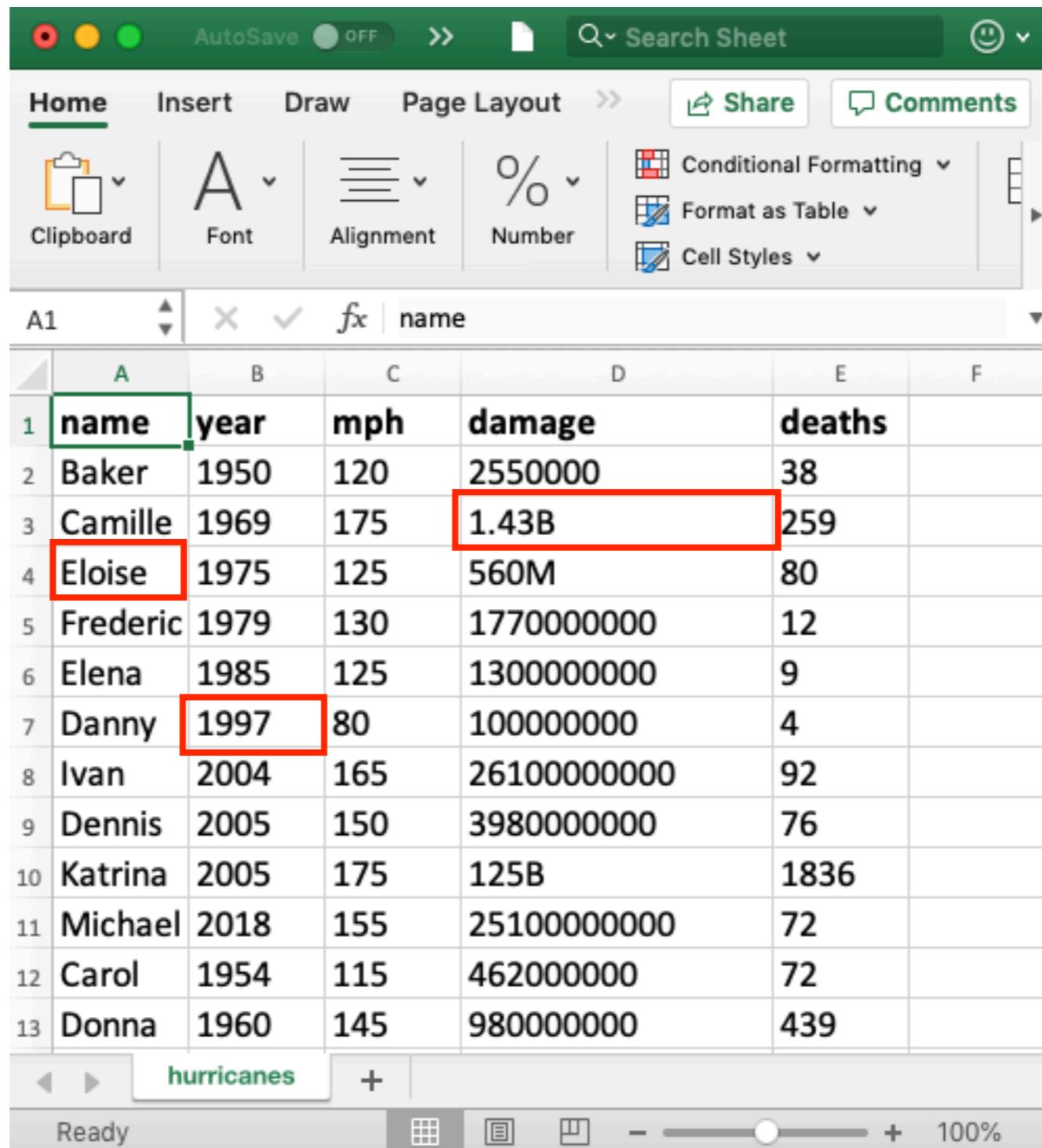
A screenshot of a Microsoft Excel spreadsheet titled "Search Sheet". The ribbon at the top shows tabs for Home, Insert, Draw, and Page Layout, with Home selected. Below the ribbon are toolbars for Clipboard, Font, Alignment, and Number, along with buttons for Share, Comments, Conditional Formatting, Format as Table, and Cell Styles. The formula bar shows "A1" and "name". The main area contains a table with 13 rows and 6 columns. The columns are labeled "name", "year", "mph", "damage", and "deaths". The first row is a header. The data includes entries for Baker (1950), Camille (1969), Eloise (1975), Frederic (1979), Elena (1985), Danny (1997), Ivan (2004), Dennis (2005), Katrina (2005), Michael (2018), Carol (1954), and Donna (1960). The "damage" column contains large values like 2550000 and 1770000000, while the "deaths" column contains smaller values like 38 and 1836. The last row is empty. The bottom of the screen shows the tab "hurricanes" is selected, along with other tabs and icons.

	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	
7	Danny	1997	80	100000000	4	
8	Ivan	2004	165	26100000000	92	
9	Dennis	2005	150	3980000000	76	
10	Katrina	2005	175	125B	1836	
11	Michael	2018	155	25100000000	72	
12	Carol	1954	115	462000000	72	
13	Donna	1960	145	980000000	439	

# Spreadsheets (e.g., Excel)

Spreadsheets are tables of cells, organized by rows and columns

cells



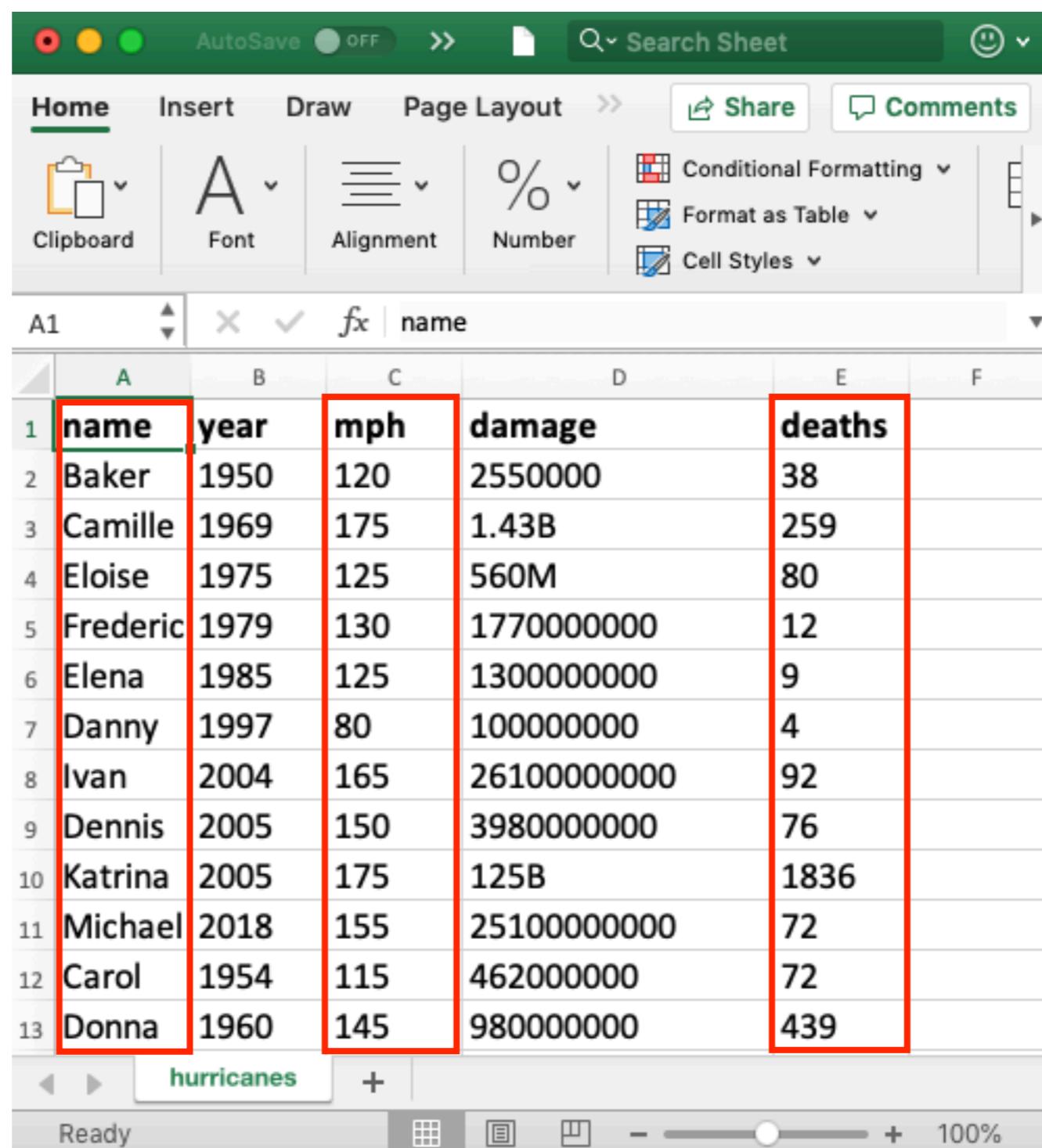
The screenshot shows a Microsoft Excel spreadsheet titled "Search Sheet". The ribbon menu is visible at the top, with "Home" selected. The formula bar shows "A1" and "name". The main area contains a table with 13 rows and 6 columns. The columns are labeled "name", "year", "mph", "damage", and "deaths". The first row is a header. The "name" column has a green border around the first cell ("name"). The "deaths" column has a red border around the second cell ("38"). The "name" column has a red border around the 4th cell ("Eloise"). The "damage" column has a red border around the 3rd cell ("1.43B"). The "year" column has a red border around the 7th cell ("1997"). The "damage" column has a red border around the 13th cell ("439"). The table is currently sorted by the "name" column. The status bar at the bottom shows "Ready" and "100%".

	A	B	C	D	E	F
1	<b>name</b>	<b>year</b>	<b>mph</b>	<b>damage</b>	<b>deaths</b>	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	
7	Danny	1997	80	100000000	4	
8	Ivan	2004	165	26100000000	92	
9	Dennis	2005	150	3980000000	76	
10	Katrina	2005	175	125B	1836	
11	Michael	2018	155	25100000000	72	
12	Carol	1954	115	462000000	72	
13	Donna	1960	145	980000000	439	

# Spreadsheets (e.g., Excel)

Spreadsheets are tables of cells, organized by rows and columns

columns



The screenshot shows a Microsoft Excel spreadsheet titled "Search Sheet". The ribbon menu is visible at the top, with "Home" selected. The table consists of 13 rows and 6 columns, labeled A through F. The columns are: name, year, mph, damage, and deaths. The first row contains column headers. The "name" column is highlighted with a green border, while the other four columns are highlighted with red borders. The data in the columns is as follows:

	name	year	mph	damage	deaths
1	Baker	1950	120	2550000	38
2	Camille	1969	175	1.43B	259
3	Eloise	1975	125	560M	80
4	Frederic	1979	130	1770000000	12
5	Elena	1985	125	1300000000	9
6	Danny	1997	80	100000000	4
7	Ivan	2004	165	26100000000	92
8	Dennis	2005	150	3980000000	76
9	Katrina	2005	175	125B	1836
10	Michael	2018	155	25100000000	72
11	Carol	1954	115	462000000	72
12	Donna	1960	145	980000000	439

# Spreadsheets (e.g., Excel)

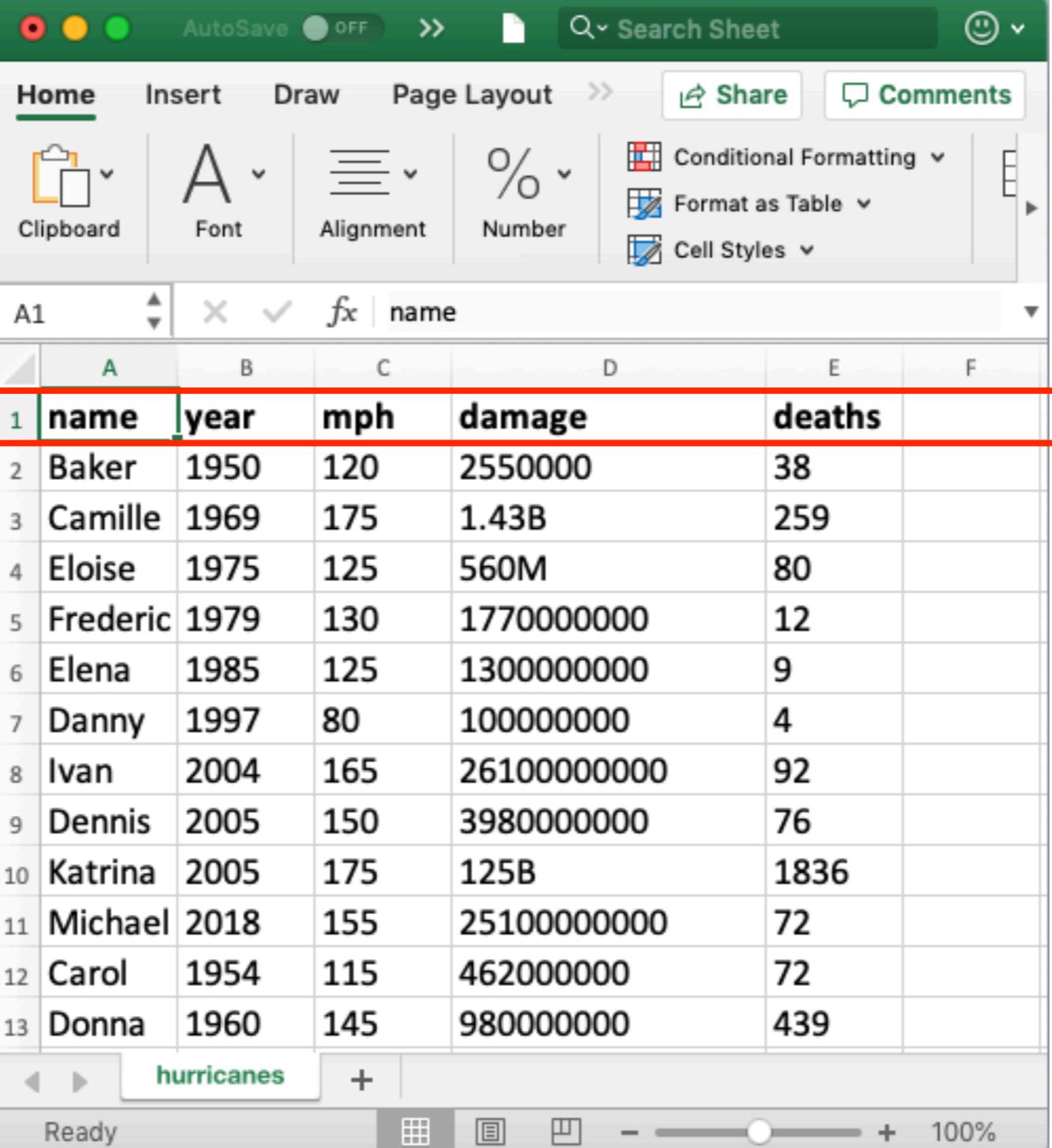
Spreadsheets are tables of cells, organized by rows and columns

**rows**

	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	
7	Danny	1997	80	100000000	4	
8	Ivan	2004	165	26100000000	92	
9	Dennis	2005	150	3980000000	76	
10	Katrina	2005	175	125B	1836	
11	Michael	2018	155	25100000000	72	
12	Carol	1954	115	462000000	72	
13	Donna	1960	145	980000000	439	

# Spreadsheets (e.g., Excel)

Spreadsheets are tables of cells, organized by rows and columns



The screenshot shows a Microsoft Excel spreadsheet titled "Search Sheet". The ribbon menu is visible at the top, with "Home" selected. The formula bar shows "A1" and "name". The main area contains a table of data with the first row highlighted in red, labeled "header". The table has columns for index (1), name, year, mph, damage, and deaths. The data shows various hurricanes from 1950 to 2018, with increasing intensity and damage over time.

1	name	year	mph	damage	deaths
2	Baker	1950	120	2550000	38
3	Camille	1969	175	1.43B	259
4	Eloise	1975	125	560M	80
5	Frederic	1979	130	1770000000	12
6	Elena	1985	125	1300000000	9
7	Danny	1997	80	100000000	4
8	Ivan	2004	165	26100000000	92
9	Dennis	2005	150	3980000000	76
10	Katrina	2005	175	125B	1836
11	Michael	2018	155	25100000000	72
12	Carol	1954	115	462000000	72
13	Donna	1960	145	980000000	439

# Spreadsheets (e.g., Excel)

Spreadsheets often allow different **data types**

	A	B	C	D	E	F
1	<b>name</b>	<b>year</b>	<b>mph</b>	<b>damage</b>	<b>deaths</b>	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	
7	Danny	1997	80	100000000	4	
8	Ivan	2004	165	26100000000	92	
9	Dennis	2005	150	3980000000	76	
10	Katrina	2005	175	125B	1836	
11	Michael	2018	155	25100000000	72	
12	Carol	1954	115	462000000	72	
13	Donna	1960	145	980000000	439	

# Spreadsheets (e.g., Excel)

Spreadsheets often allow different **fonts**

**bold** → **name**

**regular** → **Eloise**

	A	B	C	D	E	F
1	<b>name</b>	<b>year</b>	<b>mph</b>	<b>damage</b>	<b>deaths</b>	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	
7	Danny	1997	80	100000000	4	
8	Ivan	2004	165	26100000000	92	
9	Dennis	2005	150	3980000000	76	
10	Katrina	2005	175	125B	1836	
11	Michael	2018	155	25100000000	72	
12	Carol	1954	115	462000000	72	
13	Donna	1960	145	980000000	439	

# Spreadsheets (e.g., Excel)

Spreadsheets often support **multiple sheets**

A screenshot of Microsoft Excel showing a table of hurricane data. The table has columns labeled 'name', 'year', 'mph', 'damage', and 'deaths'. The first row contains the column headers. The data rows are numbered 2 through 13. A red box highlights the '+' button in the bottom-left corner of the worksheet tab bar, which is used to add new sheets. The tab bar also shows the name 'hurricanes' and other sheet tabs.

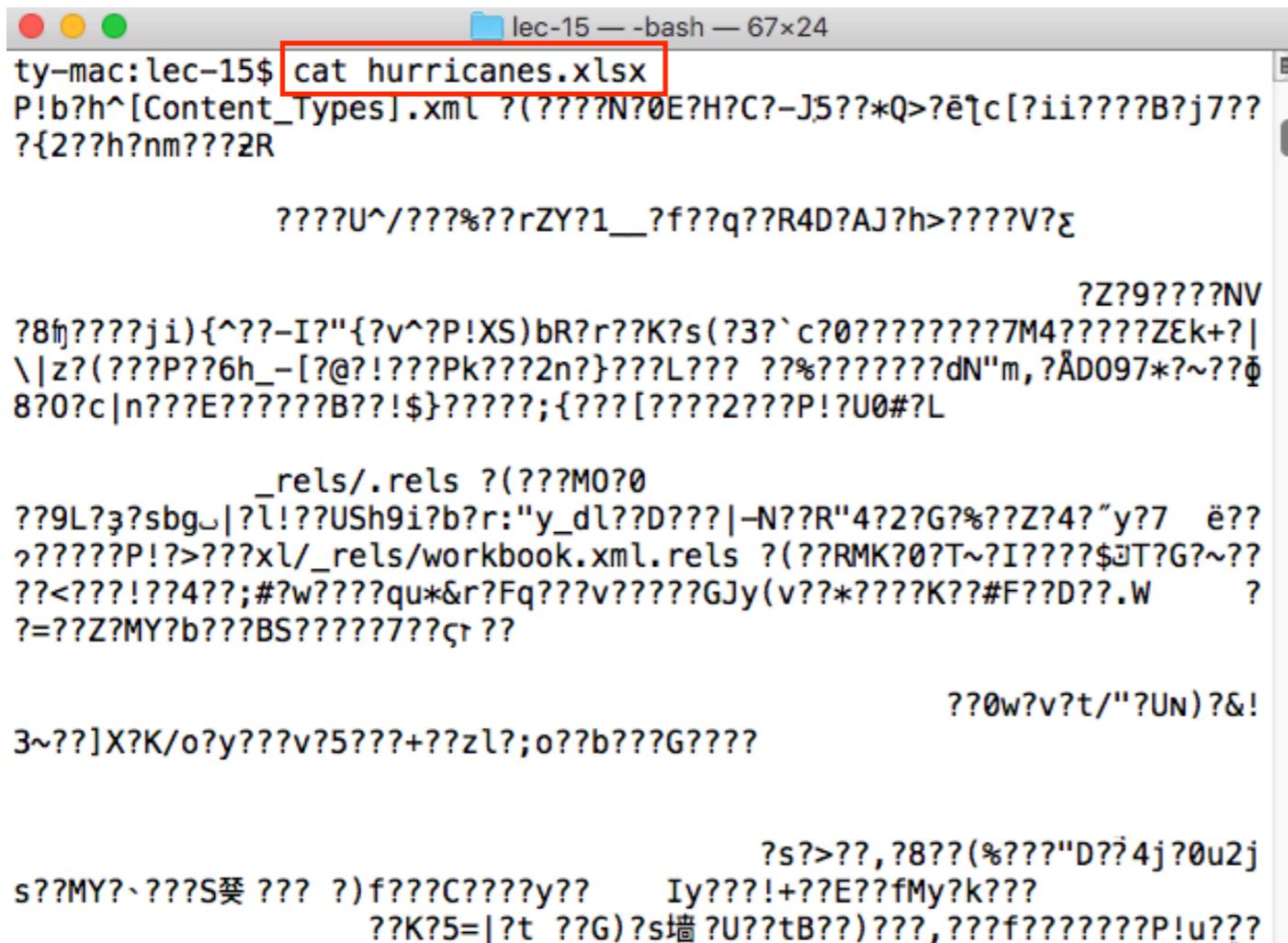
	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	
7	Danny	1997	80	100000000	4	
8	Ivan	2004	165	26100000000	92	
9	Dennis	2005	150	3980000000	76	
10	Katrina	2005	175	125B	1836	
11	Michael	2018	155	25100000000	72	
12	Carol	1954	115	462000000	72	
13	Donna	1960	145	980000000	439	

more tables of data

# Excel Files

Extension: .xlsx

Format: **binary** → just 0's and 1's, not human-readable characters.  
Need special software...



A terminal window titled "lec-15 — -bash — 67x24" displays the command "ty-mac:lec-15\$ cat hurricanes.xlsx". The output consists of a long string of binary data represented by various characters, including question marks, underscores, and special symbols like "Ã", "€", and "£". The text is mostly illegible but shows the internal structure of the Excel file.

```
ty-mac:lec-15$ cat hurricanes.xlsx
P!b?h^[Content_Types].xml ?(????N?0E?H?C?-J5??*Q>?€?c[?i?i????B?j7??
?{2??h?nm???R
????U^/????%??rZY?1__?f??q??R4D?AJ?h>????V?€
?Z?9????NV
?8?j????ji){^??-I?"{?v^?P!XS)bR?r??K?s(?3?`c?0?????????M4?????Z?k+?|?
\|z?(??P??6h_-[@?!???Pk???2n?}???L??? ??%??????dN"m,?ÅD097*?~??Ø
8?0?c|n???E??????B??!$}?????;{???[?????2???P!?U0#?L
_rels/.rels ?(???M0?0
??9L?3?sbg_!|?l!??USh9i?b?r:"y_dl?D???|-N??R"4?2?G?%??Z?4?"y?7  €??
??????P!?>???xl/_rels/workbook.xml.rels ?(???RMK?0?T~?I?????$ØT?G?~??
??<???!??4??;#?w????qu*&r?Fq???v?????GJy(v??*????K??#F??D???.W      ?
?=??Z?MY?b???BS??????ç? ???
??0w?v?t/"?UN)?&
3~??]X?K/o?y???v?5???+??zl?;o??b???G?????
?s?>??,?8??(%???"D??4j?0u2j
s??MY?~??S? ??)f???C????y?? Iy??!+??E??fMy?k??
??K?5=|?t ??G)?s?U??tB??)???,??f???????P!u???
```

Writing code to read data from  
Excel files is tricky, unless you  
use special modules

# Today's Outline

Spreadsheets

CSVs

Reading a CSV to a list of lists

Coding examples

# CSVs

CSV is a simple data format that stands for  
**Comma-Separated Values**

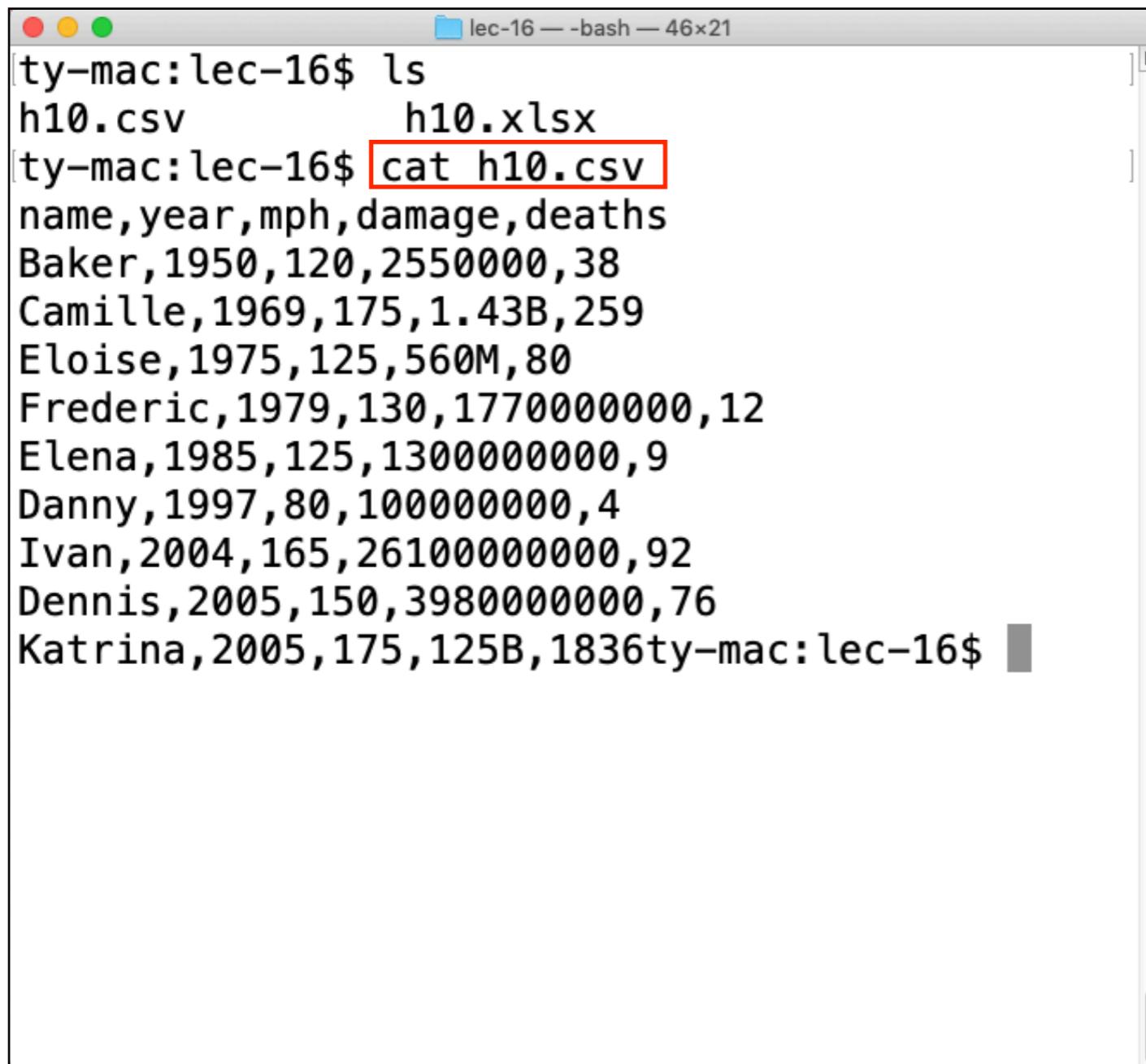
CSVs are like simple spreadsheets

- organize cells of data into rows and columns
  - only one sheet per file
  - only holds strings
  - no way to specify font, borders, cell size, etc
- you'll do lots of type casting/conversion!
- 

# CSV Files

Extension: .csv

Format: plain text → just open in any editor (notepad, textedit, idle, etc) and you'll be able to read it



A screenshot of a Mac OS X terminal window titled "lec-16 — bash — 46x21". The window shows the command line interface with the following text:

```
ty-mac:lec-16$ ls
h10.csv      h10.xlsx
ty-mac:lec-16$ cat h10.csv
name,year,mph,damage,deaths
Baker,1950,120,2550000,38
Camille,1969,175,1.43B,259
Eloise,1975,125,560M,80
Frederic,1979,130,1770000000,12
Elena,1985,125,1300000000,9
Danny,1997,80,100000000,4
Ivan,2004,165,2610000000,92
Dennis,2005,150,3980000000,76
Katrina,2005,175,125B,1836ty-mac:lec-16$
```

Writing code that understands  
CSV files is easy

# Basic Syntax

## Table

Name	Date	Time	Status	Latitude	Longitude	WindSpeed	Ocean
HEIDI	19671019	1200	TD	20.5N	54.0W	25	Atlantic
OLAF	19850822	0	TD	12.9N	102.2W	25	Pacific
TINA	19920917	1200	TD	10.4N	98.5W	25	Pacific
EMMY	19760820	1200	TD	14.0N	48.0W	20	Atlantic

## Corresponding CSV

Name,Date,Time,Status,Latitude,Longitude,WindSpeed,Ocean
HEIDI,19671019,1200,TD,20.5N,54.0W,25,Atlantic
OLAF,19850822,0,TD,12.9N,102.2W,25,Pacific
TINA,19920917,1200,TD,10.4N,98.5W,25,Pacific
EMMY,19760820,1200,TD,14.0N,48.0W,20,Atlantic

Each row is a line of the file

# Basic Syntax

## Table

Name	Date	Time	Status	Latitude	Longitude	WindSpeed	Ocean
HEIDI	19671019	1200	TD	20.5N	54.0W	25	Atlantic
OLAF	19850822	0	TD	12.9N	102.2W	25	Pacific
TINA	19920917	1200	TD	10.4N	98.5W	25	Pacific
EMMY	19760820	1200	TD	14.0N	48.0W	20	Atlantic

## Corresponding CSV

Name,Date,Time,Status,Latitude,Longitude,WindSpeed,Ocean  
HEIDI,19671019,1200,TD,20.5N,54.0W,25,Atlantic  
OLAF,19850822,0,TD,12.9N,102.2W,25,Pacific  
TINA,19920917,1200,TD,10.4N,98.5W,25,Pacific  
EMMY,19760820,1200,TD,14.0N,48.0W,20,Atlantic

Each row is a line of the file

# Basic Syntax

## Table

Name	Date	Time	Status	Latitude	Longitude	WindSpeed	Ocean
HEIDI	19671019	1200	TD	20.5N	54.0W	25	Atlantic
OLAF	19850822	0	TD	12.9N	102.2W	25	Pacific
TINA	19920917	1200	TD	10.4N	98.5W	25	Pacific
EMMY	19760820	1200	TD	14.0N	48.0W	20	Atlantic

## Corresponding CSV

Name,Date,Time,Status,Latitude,Longitude,WindSpeed,Ocean  
HEIDI,19671019,1200,TD,20.5N,54.0W,25,Atlantic  
OLAF,19850822,0,TD,12.9N,102.2W,25,Pacific  
TINA,19920917,1200,TD,10.4N,98.5W,25,Pacific  
EMMY,19760820,1200,TD,14.0N,48.0W,20,Atlantic

Cells...

# Basic Syntax

## Table

Name	Date	Time	Status	Latitude	Longitude	WindSpeed	Ocean
HEIDI	19671019	1200	TD	20.5N	54.0W	25	Atlantic
OLAF	19850822	0	TD	12.9N	102.2W	25	Pacific
TINA	19920917	1200	TD	10.4N	98.5W	25	Pacific
EMMY	19760820	1200	TD	14.0N	48.0W	20	Atlantic

## Corresponding CSV

Name,Date,Time,Status,Latitude,Longitude,WindSpeed,Ocean  
HEIDI,19671019,1200,TD,20.5N,54.0W,25,Atlantic  
OLAF,19850822,0,TD,12.9N,102.2W,25,Pacific  
TINA,19920917,1200,TD,10.4N,98.5W,25,Pacific  
EMMY,19760820,1200,TD,14.0N,48.0W,20,Atlantic

... are separated by commas

# Basic Syntax

Table

Name	Date	Time	Status	Latitude	Longitude	WindSpeed	Ocean
HEIDI	19671019	1200	TD	20.5N	54.0W	25	Atlantic
OLAF	19850822	0	TD	12.9N	102.2W	25	Pacific
TINA	19920917	1200	TD	10.4N	98.5W	25	Pacific
EMMY	19760820	1200	TD	14.0N	48.0W	20	Atlantic

C We call characters that act as separators “**delimiters**”

Newlines delimit rows

The comma is a delimiter between cells in a row

EMMY,19760820,1200,TD,14.0N,48.0W,20,Atlantic

... are separated by commas

# Advanced Syntax

We won't go into details here, but there are some complexities

Motivation for more complicated syntax

- *what if a cell contains a newline?*
- *what if we want a comma inside a cell?*
- *what if a cell contains a quote?*
- *what if we want to use different delimiters between rows/cells?*

usually better to use a general CSV module than roll your own

# Today's Outline

Spreadsheets

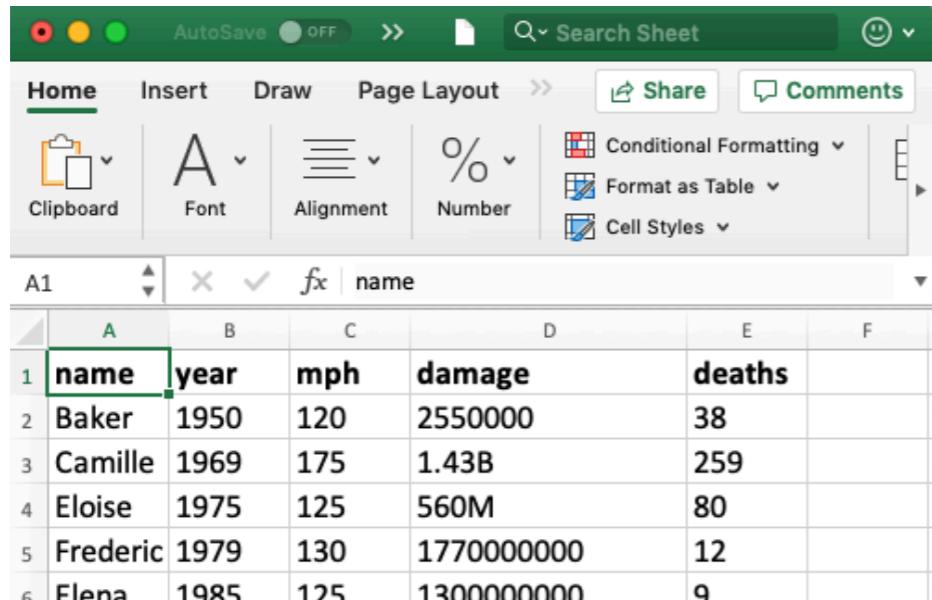
CSVs

Reading a CSV to a list of lists

Coding examples

# Data Management

## I. spreadsheet in Excel



A screenshot of the Microsoft Excel application interface. The ribbon at the top shows tabs for Home, Insert, Draw, Page Layout, Share, and Comments. The Home tab is selected. Below the ribbon, there are toolbars for Clipboard, Font, Alignment, and Number. The main area shows a table with data from row 1 to 6. Row 1 contains column headers: name, year, mph, damage, and deaths. Rows 2 through 6 contain data for Baker, Camille, Eloise, Frederic, and Elena respectively. The 'damage' column for Frederic contains a value of 17700000000, which is displayed with commas as thousands separators.

	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	17700000000	12	
6	Elena	1985	125	1300000000	9	



Save As  
.CSV

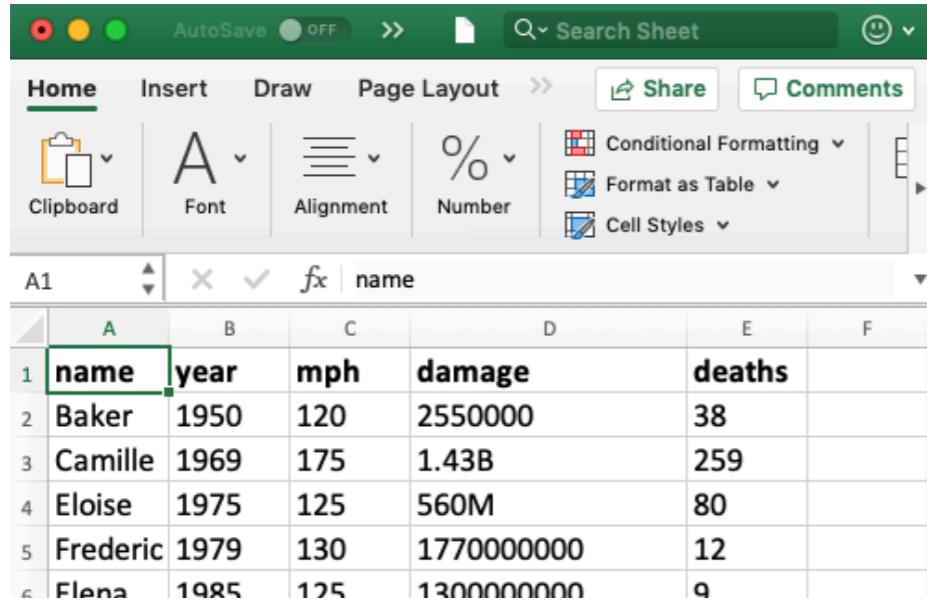
## 2. CSV file saved somewhere

```
name,year,mph,damage,deaths
Baker,1950,120,2550000,38
Camille,1969,175,1.43B,259
Eloise,1975,125,560M,80
Frederic,1979,130,17700000000,12
```

# Data Management

## 3. Python Program

### I. spreadsheet in Excel



	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	

Save As  
.CSV

### 2. CSV file saved somewhere

```
name,year,mph,damage,deaths
Baker,1950,120,2550000,38
Camille,1969,175,1.43B,259
Eloise,1975,125,560M,80
Frederic,1979,130,1770000000,12
```

list of lists

[

```
[ "name", "year", ... ],  
[ "Baker", "1950", ... ],
```

...

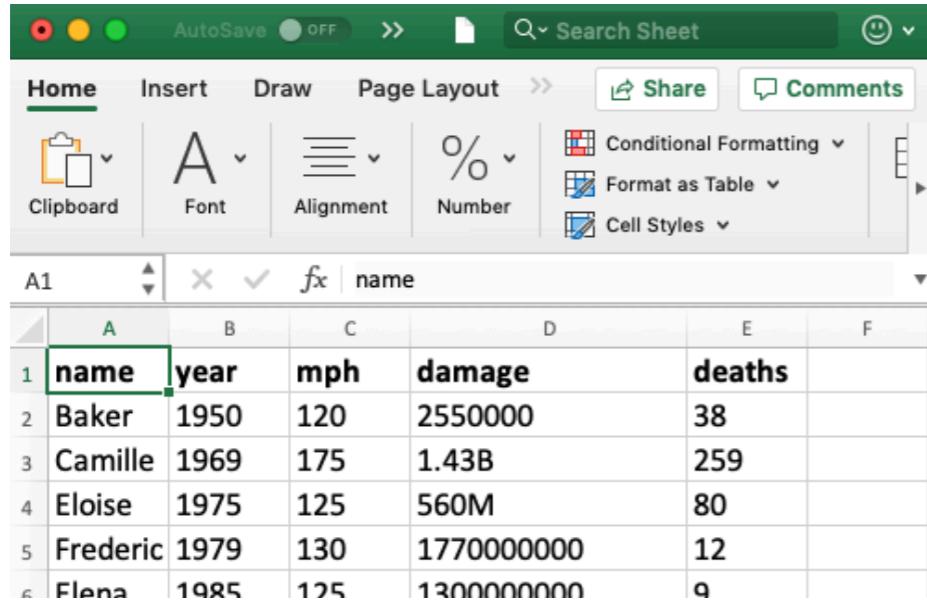
]

Parsing Code

we'll come  
back to this

# Data Management

## I. spreadsheet in Excel



	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Frena	1985	125	1300000000	9	

Save As  
.CSV

## 2. CSV file saved somewhere

```
name,year,mph,damage,deaths
Baker,1950,120,2550000,38
Camille,1969,175,1.43B,259
Eloise,1975,125,560M,80
Frederic,1979,130,1770000000,12
```

## 3. Python Program

Analysis Code

`rows[1][0] → ???`

[

`[ "name", "year", ... ],  
[ "Baker", "1950", ... ],`

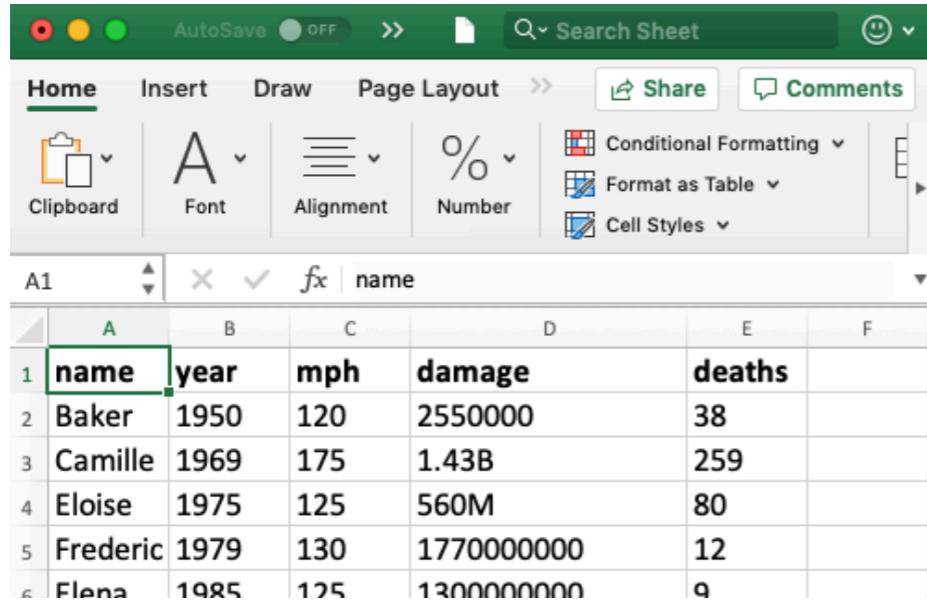
...

]

Parsing Code

# Data Management

## I. spreadsheet in Excel



	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	

Save As  
.CSV

## 2. CSV file saved somewhere

```
name,year,mph,damage,deaths
Baker,1950,120,2550000,38
Camille,1969,175,1.43B,259
Eloise,1975,125,560M,80
Frederic,1979,130,1770000000,12
```

## 3. Python Program

Analysis Code

`rows[1][0] → "Baker"`

[

`[ "name", "year", ... ],  
[ "Baker", "1950", ... ],`

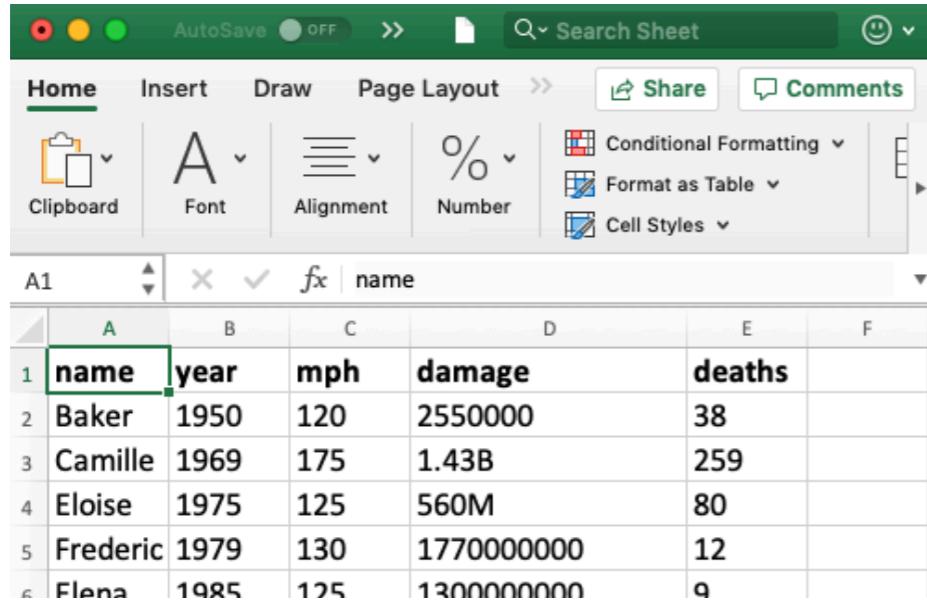
...

]

Parsing Code

# Data Management

## I. spreadsheet in Excel



	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Frena	1985	125	1300000000	9	

Save As  
.CSV

## 2. CSV file saved somewhere

```
name,year,mph,damage,deaths
Baker,1950,120,2550000,38
Camille,1969,175,1.43B,259
Eloise,1975,125,560M,80
Frederic,1979,130,1770000000,12
```

## 3. Python Program

Analysis Code

rows[3][1] → ???

list of lists

[ ... ]

[ "name", "year", ... ],  
[ "Baker", "1950", ... ],

...

Parsing Code

# Data Management

## I. spreadsheet in Excel

	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	

Save As  
.CSV

## 2. CSV file saved somewhere

```
name,year,mph,damage,deaths
Baker,1950,120,2550000,38
Camille,1969,175,1.43B,259
Eloise,1975,125,560M,80
Frederic,1979,130,1770000000,12
```

## 3. Python Program

Analysis Code

`rows[3][1] → "1975"`

list of lists

[ ... ]

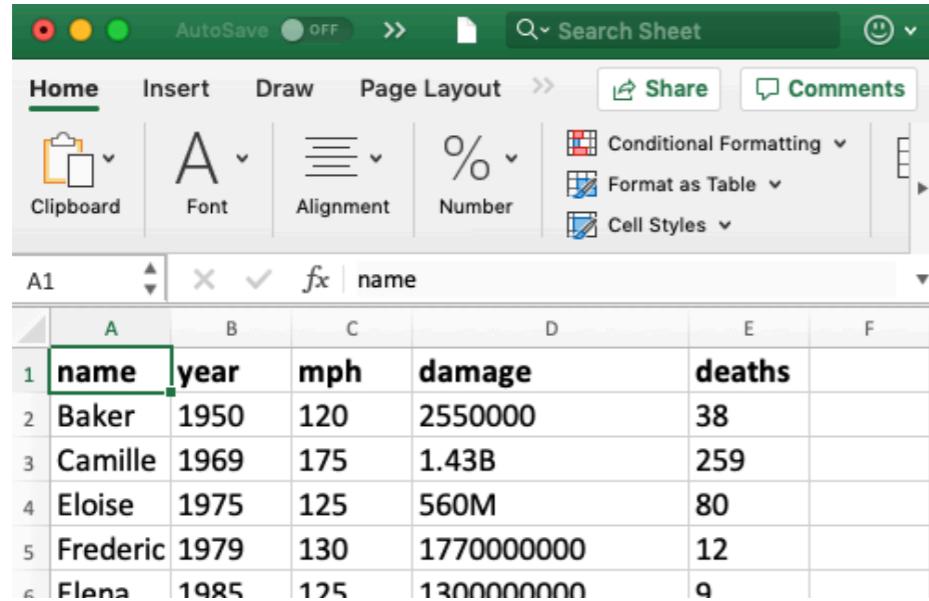
[ "name", "year", ... ],  
[ "Baker", "1950", ... ],

...

Parsing Code

# Data Management

## I. spreadsheet in Excel



	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	

Save As  
.CSV

## 2. CSV file saved somewhere

```
name,year,mph,damage,deaths
Baker,1950,120,2550000,38
Camille,1969,175,1.43B,259
Eloise,1975,125,560M,80
Frederic,1979,130,1770000000,12
```

## 3. Python Program

Analysis Code

`rows[1][-1] → ???`

list of lists

[ ... ]

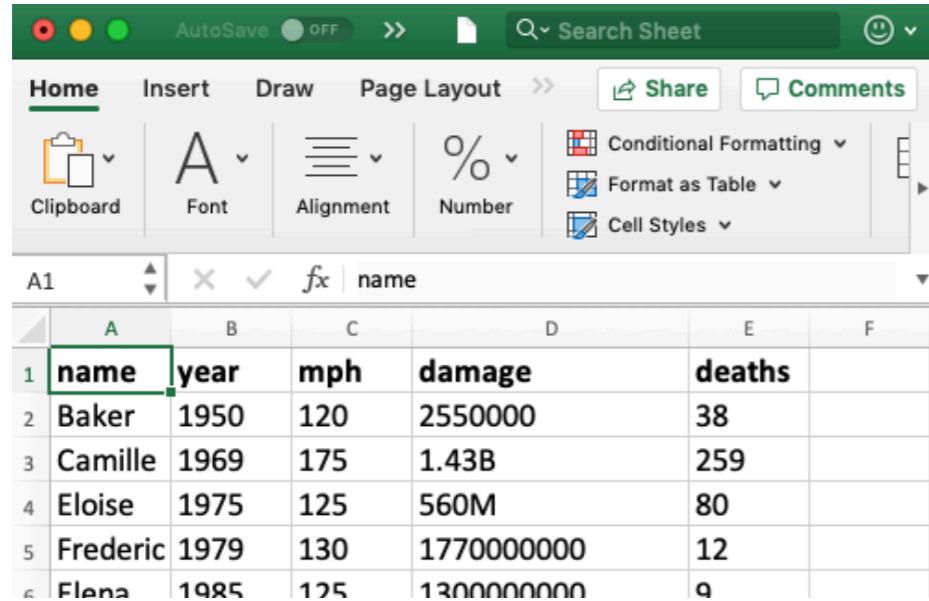
[ "name", "year", ... ],  
[ "Baker", "1950", ... ],

...

Parsing Code

# Data Management

## I. spreadsheet in Excel



	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Frena	1985	125	1300000000	9	

Save As  
.CSV

## 2. CSV file saved somewhere

```
name,year,mph,damage,deaths
Baker,1950,120,2550000,38
Camille,1969,175,1.43B,259
Eloise,1975,125,560M,80
Frederic,1979,130,1770000000,12
```

## 3. Python Program

Analysis Code  
`rows[1][-1] → "38"`

[

[ "name", "year", ... ],  
[ "Baker", "1950", ... ],

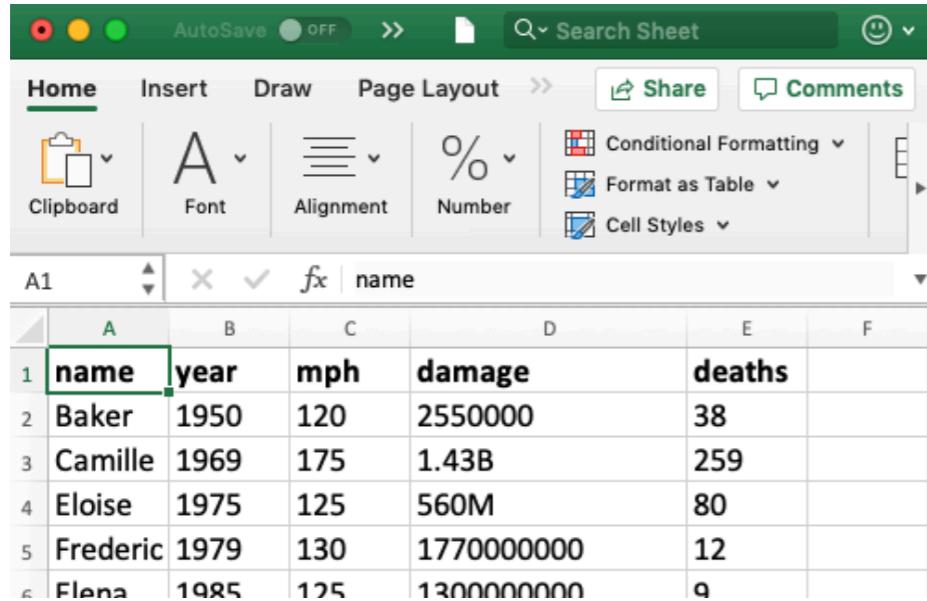
...

]

Parsing Code

# Data Management

## I. spreadsheet in Excel



	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	

Save As  
.CSV

## 2. CSV file saved somewhere

```
name,year,mph,damage,deaths
Baker,1950,120,2550000,38
Camille,1969,175,1.43B,259
Eloise,1975,125,560M,80
Frederic,1979,130,1770000000,12
```

## 3. Python Program

Analysis Code

`rows[0][-2] → ???`

list of lists

[ ... ]

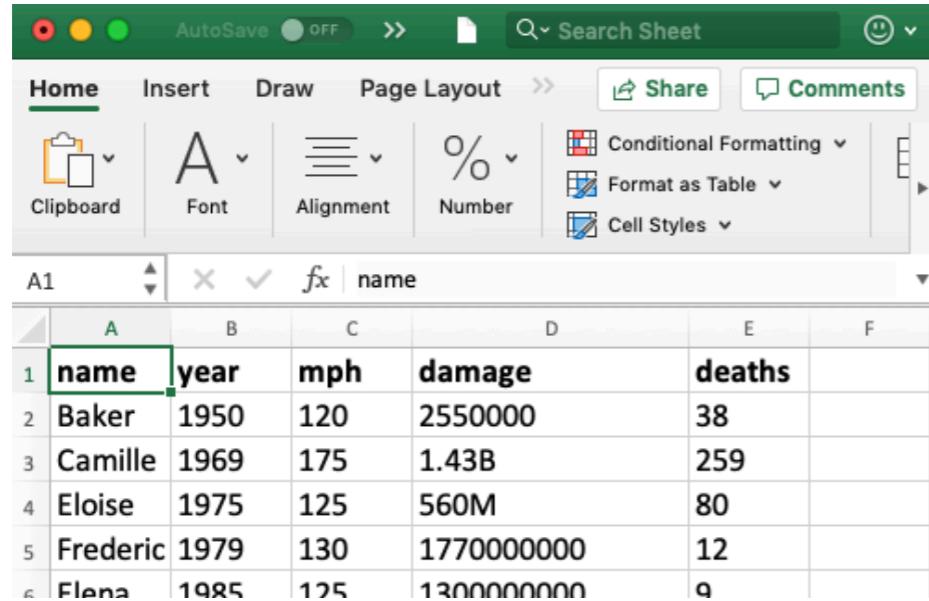
[ "name", "year", ... ],  
[ "Baker", "1950", ... ],

...

Parsing Code

# Data Management

## I. spreadsheet in Excel



	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	

Save As  
.CSV

## 2. CSV file saved somewhere

```
name,year,mph,damage,deaths
Baker,1950,120,2550000,38
Camille,1969,175,1.43B,259
Eloise,1975,125,560M,80
Frederic,1979,130,1770000000,12
```

## 3. Python Program

Analysis Code

`rows[0][-2] → "damage"`

list of lists

[

`[ "name", "year", ... ],  
[ "Baker", "1950", ... ],`

...

]

Parsing Code

# Data Management

## I. spreadsheet in Excel

	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Fiona	1985	125	1300000000	9	

Save As  
.CSV

## 2. CSV file saved somewhere

```
name,year,mph,damage,deaths
Baker,1950,120,2550000,38
Camille,1969,175,1.43B,259
Eloise,1975,125,560M,80
Frederic,1979,130,1770000000,12
```

## 3. Python Program

Analysis Code

`rows[0][-2] → "damage"`

list of lists

[ "name", "year", ... ,  
["Baker", "1950", ... ,

...

Parsing Code

What does this look like?

# Example Copied From Sweigart Ch 14

**Code**

```
import csv
exampleFile = open('example.csv')
exampleReader = csv.reader(exampleFile)
exampleData = list(exampleReader)
```

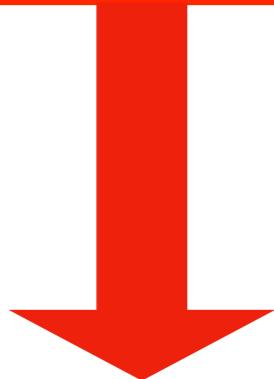
**example.csv**

4/5/2015 13:34,Apples,73
4/5/2015 3:41,Cherries,85
4/6/2015 12:46,Pears,14
4/8/2015 8:59,Oranges,52
4/10/2015 2:07,Apples,152
4/10/2015 18:10,Bananas,23
4/10/2015 2:40,Strawberries,98

# Example Copied From Sweigart Ch 14

**Code**

```
import csv  
exampleFile = open('example.csv')  
exampleReader = csv.reader(exampleFile)  
exampleData = list(exampleReader)  
exampleData
```



**list of  
lists**

```
[['4/5/2015 13:34', 'Apples', '73'], ['4/5/2015 3:41', 'Cherries', '85'],  
 ['4/6/2015 12:46', 'Pears', '14'], ['4/8/2015 8:59', 'Oranges', '52'],  
 ['4/10/2015 2:07', 'Apples', '152'], ['4/10/2015 18:10', 'Bananas', '23'],  
 ['4/10/2015 2:40', 'Strawberries', '98']]
```

# Example Copied From Sweigart Ch 14

```
import csv
exampleFile = open('example.csv')
exampleReader = csv.reader(exampleFile)
exampleData = list(exampleReader)
exampleData
```

**let's generalize this to a function**  
(don't need to know exactly how the code  
works, though we will eventually)

# Example Copied From Sweigart Ch 14

```
import csv  
exampleFile = open('example.csv')  
exampleReader = csv.reader(exampleFile)  
exampleData = list(exampleReader)  
exampleData
```

*output*

**let's generalize this to a function**  
(don't need to know exactly how the code  
works, though we will eventually)

# Example Copied From Sweigart Ch 14

```
def process_csv():
    import csv
    exampleFile = open('example.csv')
    exampleReader = csv.reader(exampleFile)
    exampleData = list(exampleReader)
    exampleData
```

## I. move code to a function

# Example Copied From Sweigart Ch 14

```
import csv

def process_csv():
    import csv
    exampleFile = open('example.csv')
    exampleReader = csv.reader(exampleFile)
    exampleData = list(exampleReader)
    exampleData
```

## 2. move out imports

# Example Copied From Sweigart Ch 14

```
import csv

def process_csv():
    import csv
    exampleFile = open('example.csv')
    exampleReader = csv.reader(exampleFile)
    exampleData = list(exampleReader)
    return exampleData
```

**3. return data to get it out of the function**

# Example Copied From Sweigart Ch 14

```
import csv

def process_csv():
    import csv
    exampleFile = open('example.csv')
    exampleReader = csv.reader(exampleFile)
    exampleData = list(exampleReader)
    return exampleData
```

## 4. generalize input

# Example Copied From Sweigart Ch 14

```
import csv

def process_csv(filename):
    import csv
    exampleFile = open(filename)
    exampleReader = csv.reader(exampleFile)
    exampleData = list(exampleReader)
    return exampleData
```

## 4. generalize input

# Example Copied From Sweigart Ch 14

```
import csv  
  
# copied from https://automatetheboringstuff.com/chapter14/  
def process_csv(filename):  
    import csv  
    exampleFile = open(filename)  
    exampleReader = csv.reader(exampleFile)  
    exampleData = list(exampleReader)  
    return exampleData
```

Reminder!  
cite code  
copied online

**5. cite the code**

# Example Copied From Sweigart Ch 14

```
import csv

# copied from https://automatetheboringstuff.com/chapter14/
def process_csv(filename):
    exampleFile = open(filename, encoding="utf-8")
    exampleReader = csv.reader(exampleFile)
    exampleData = list(exampleReader)
    exampleFile.close()
    return exampleData
```

**keep this handy for copy/paste**

# Today's Outline

Spreadsheets

CSVs

Reading a CSV to a list of lists

Coding examples

# Demo I: Restaurant Location Lookup

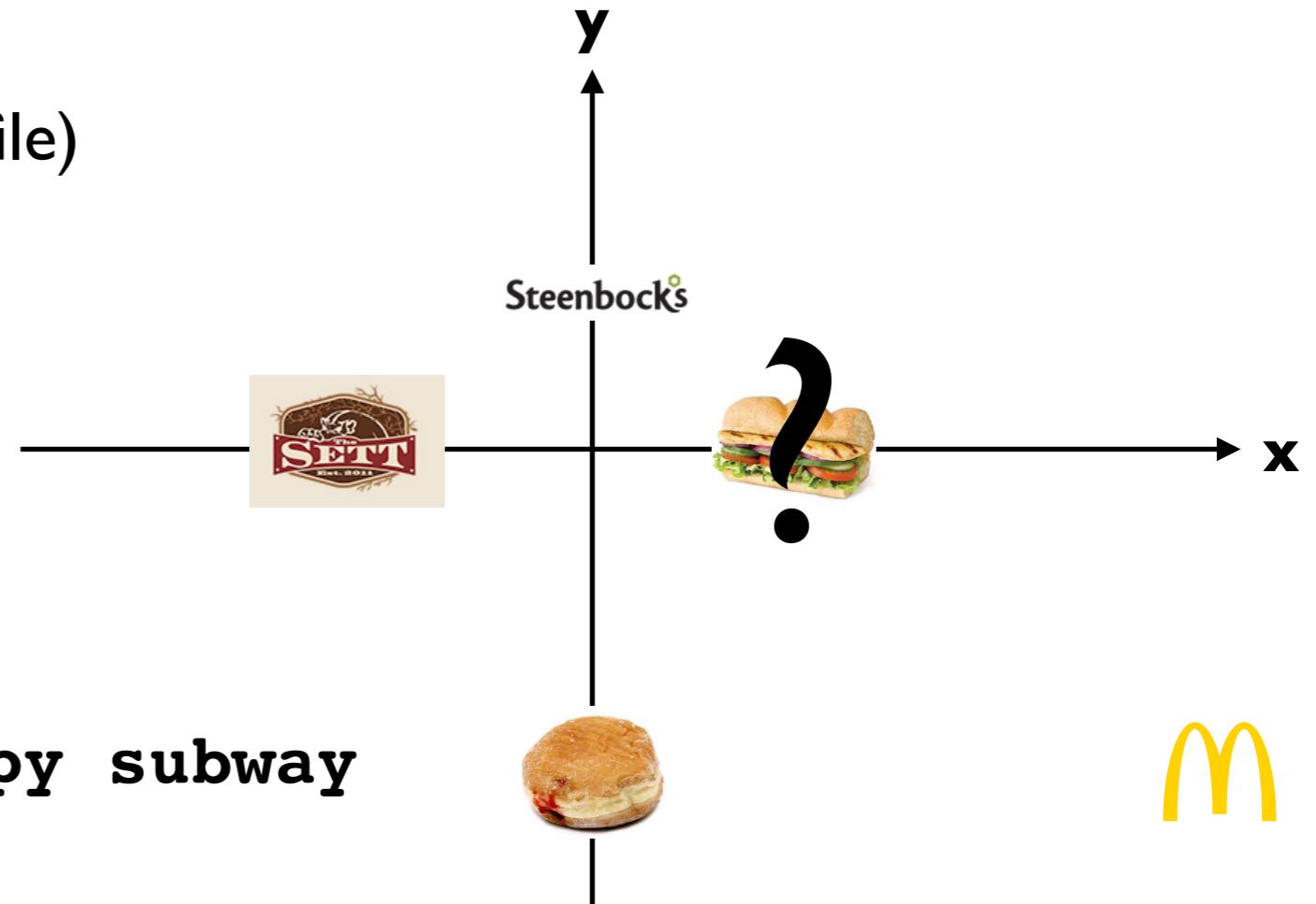
**Goal:** given a restaurant name, give x,y coordinates for it

## **Input:**

- Restaurant name (and a CSV file)

## **Output:**

- X,Y coordinates



## **Example:**

```
prompt> python rlookup.py subway
```

```
x=1, y=0
```

```
prompt> python rlookup.py mcdonalds
```

```
x=4, y=-3
```

# Demo 2: Nearest Restaurant Search

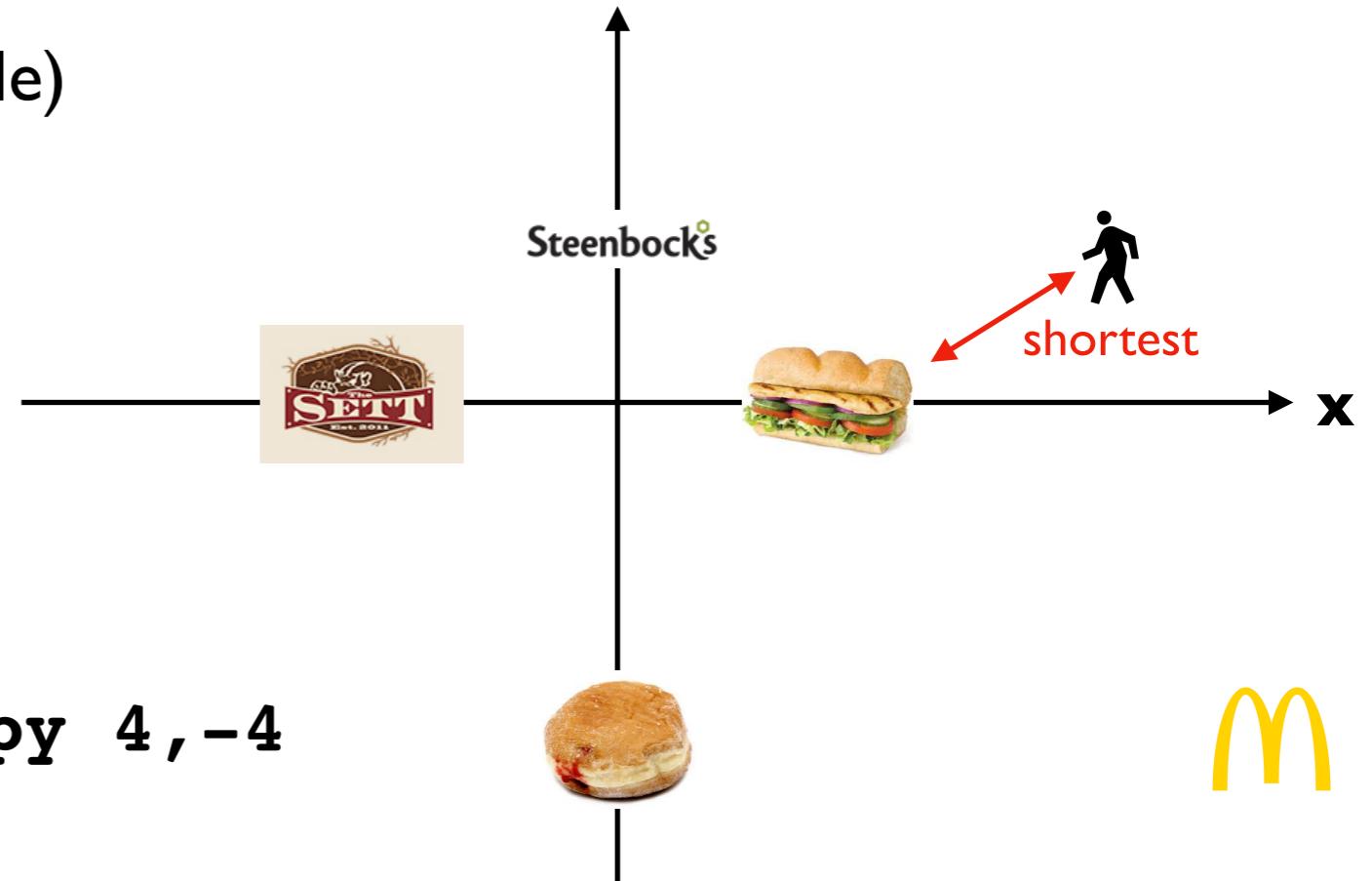
**Goal:** given a location, find the nearest restaurant

## **Input:**

- X,Y coordinates (and a CSV file)

## **Output:**

- nearest restaurant



## **Example:**

```
prompt> python nearest.py 4,-4
```

McDonalds

```
prompt> python nearest.py -2,0
```

The Sett

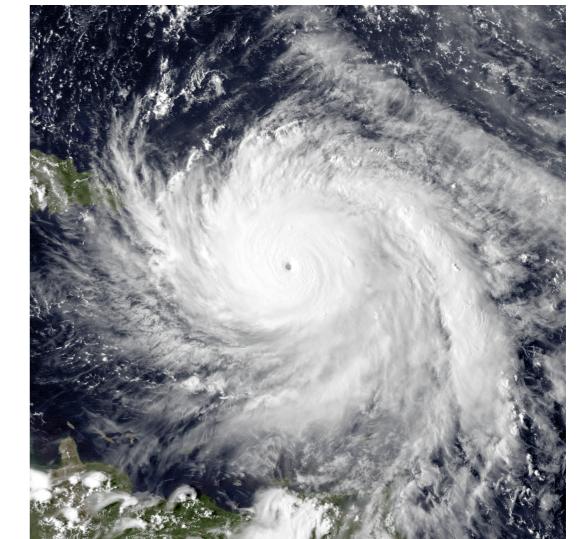


# Demo 3: Hurricane Column Dump

**Goal:** column name, print that data for all hurricanes

## **Input:**

- column name (and a CSV file)



## **Output:**

- data in given column, associated with name

## **Example:**

```
prompt> python dump.py hurricanes.csv year
```

Baker: 1950

Camille: 1969

Eloise: 1975

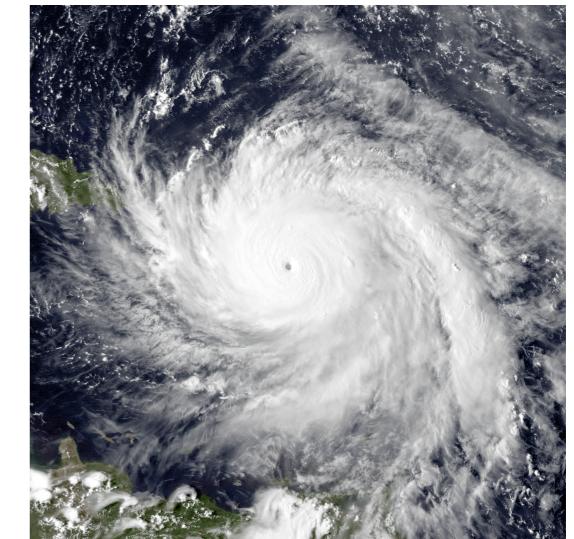
...

# Demo 4: Hurricanes per Year

**Goal:** column name, print that data for all hurricanes

## **Input:**

- none typed (only a CSV file)



## **Output:**

- the number of hurricanes in each year

## **Example:**

```
prompt> python yearly.py
```

```
1967: 23
```

```
1968: 29
```

```
2969: 15
```

```
...
```

# Demo 5: Hurricane Names and Stereotypes

 PNAS

 CrossMark  
click for updates

## Female hurricanes are deadlier than male hurricanes

Kiju Jung<sup>a,1</sup>, Sharon Shavitt<sup>a,b,1</sup>, Madhu Viswanathan<sup>a,c</sup>, and Joseph M. Hilbe<sup>d</sup>

<sup>a</sup>Department of Business Administration and <sup>b</sup>Department of Psychology, Institute of Communications Research, and Survey Research Laboratory, and <sup>c</sup>Women and Gender in Global Perspectives, University of Illinois at Urbana-Champaign, Champaign, IL 61820; and <sup>d</sup>Department of Statistics, T. Denny Sanford School of Social and Family Dynamics, Arizona State University, Tempe, AZ 85287-3701

Edited\* by Susan T. Fiske, Princeton University, Princeton, NJ, and approved May 14, 2014 (received for review February 13, 2014)

**Do people judge hurricane risks in the context of gender-based expectations? We use more than six decades of death rates from US hurricanes to show that feminine-named hurricanes cause significantly more deaths than do masculine-named hurricanes. Laboratory experiments indicate that this is because hurricane names lead to gender-based expectations about severity and this, in turn, guides respondents' preparedness to take protective action. This finding indicates an unfortunate and unintended consequence of the gendered naming of hurricanes, with important implications for policymakers, media practitioners, and the general public concerning hurricane communication and preparedness.**

gender stereotypes | implicit bias | risk perception | natural hazard communication | bounded rationality

violence and destruction (23, 24). We extend these findings to hypothesize that the anticipated severity of a hurricane with a masculine name (Victor) will be greater than that of a hurricane with a feminine name (Victoria). This expectation, in turn, will affect the protective actions that people take. As a result, a hurricane with a feminine vs. masculine name will lead to less protective action and more fatalities.

**Archival Study**

To test this hypothesis, we used archival data on actual fatalities caused by hurricanes in the United States (1950–2012). Ninety-four Atlantic hurricanes made landfall in the United States during this period (25). Nine independent coders who were blind to the hypothesis rated the masculinity vs. femininity of historical hurricane names on two items (1 = very masculine, 11 = very

this analysis is tricky and much debated

what would it take to try to replicate this study?

simple version: classify names and count deaths