

# [301] CSV Files

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**

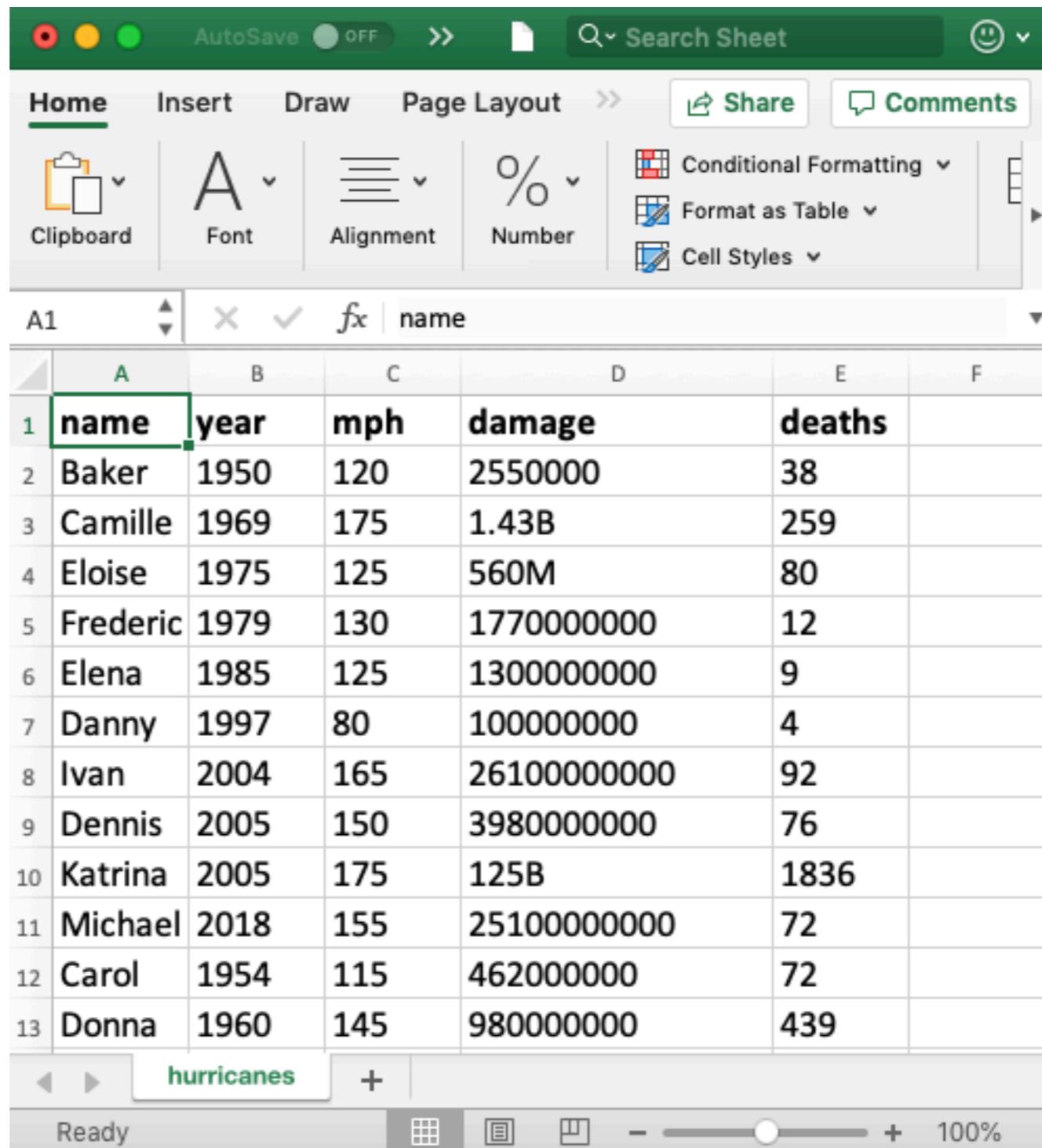
CSVs

Reading a CSV to a list of lists

Coding examples

# Spreadsheets (e.g., Excel)

Spreadsheets are tables of cells, organized by rows and columns



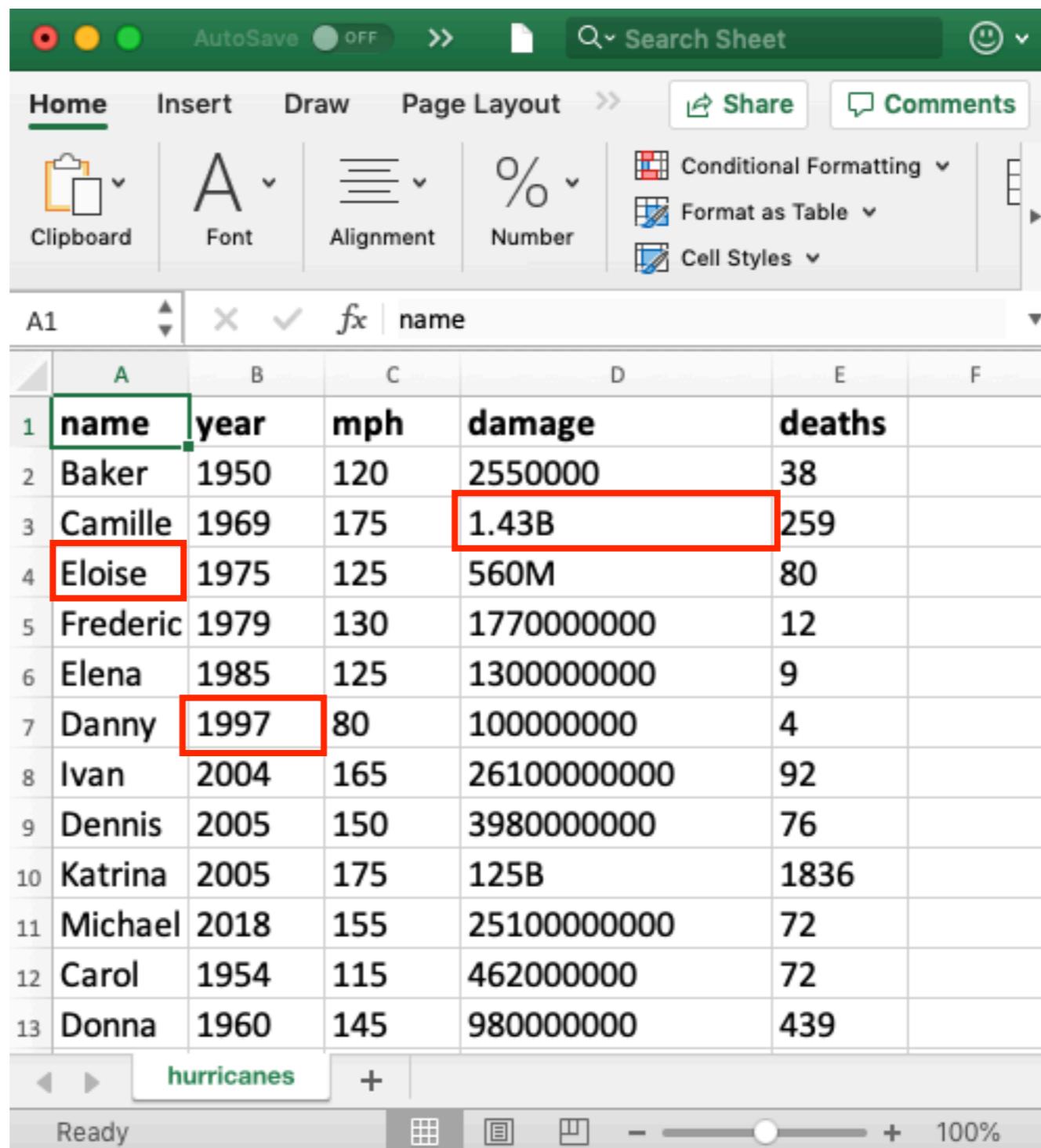
A screenshot of a Microsoft Excel spreadsheet titled "Search Sheet". The ribbon at the top shows tabs for Home, Insert, Draw, and Page Layout, with Home selected. Below the ribbon are toolbars for Clipboard, Font, Alignment, and Number. The main area displays a table of 13 rows and 6 columns. The columns are labeled A through F. Row 1 contains the column headers: name, year, mph, damage, and deaths. Rows 2 through 13 contain data for various hurricanes. The "name" column is highlighted with a green border. The "damage" column contains large numerical values representing billions or millions of dollars. The "deaths" column contains smaller numerical values representing the number of fatalities. The bottom of the screen shows the status bar with "Ready" and zoom controls.

	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

cells



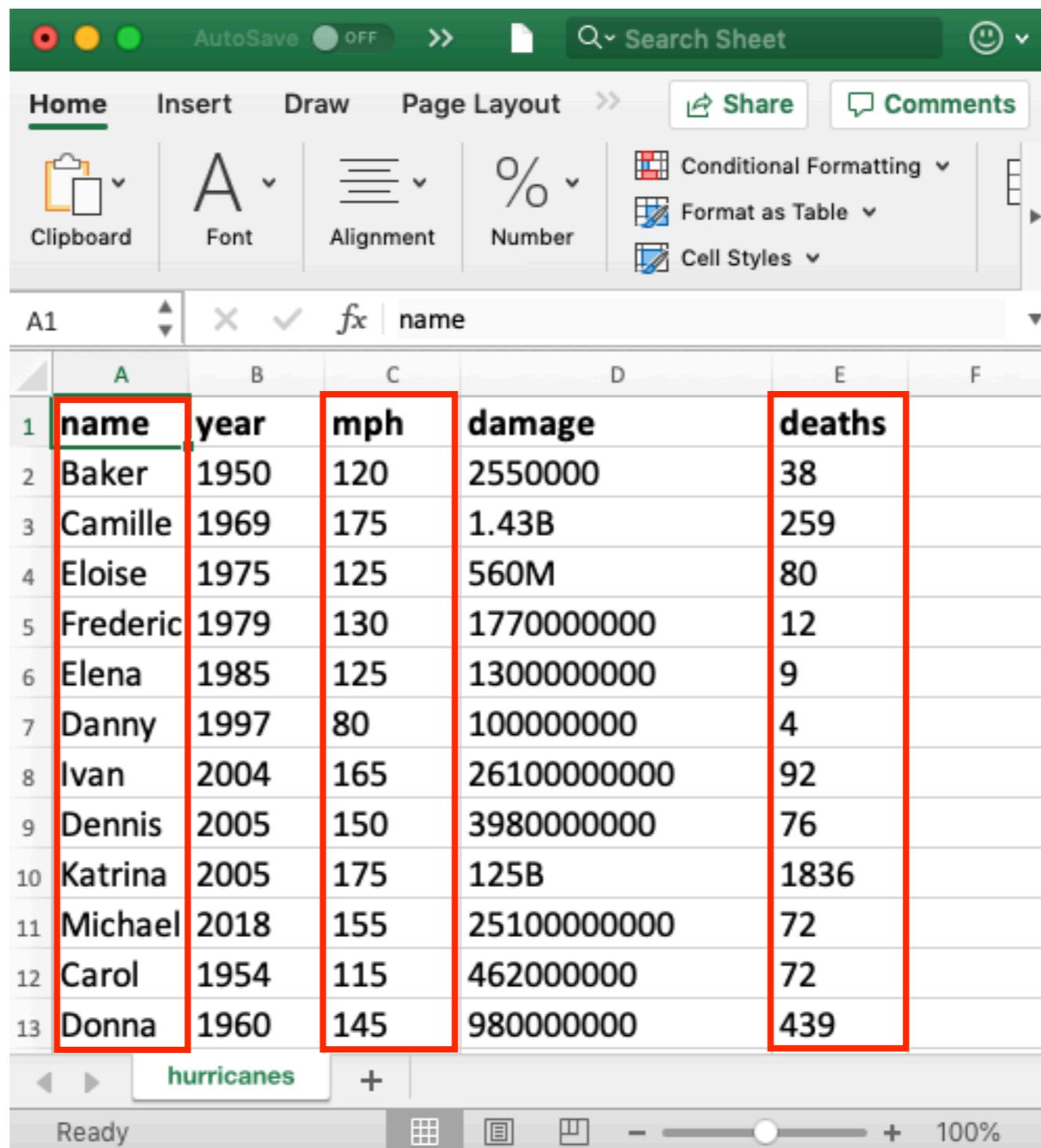
A screenshot of a Microsoft Excel spreadsheet titled "hurricanes". The table has 13 rows and 6 columns. The columns are labeled "name", "year", "mph", "damage", "deaths", and an empty column F. The "name" column is bolded. Row 1 contains the header labels. Rows 2 through 13 contain data for various hurricanes. The "name" column for row 4 (Eloise) is highlighted with a red box. The "damage" column for row 3 (Camille) is highlighted with a red box. The "year" column for row 7 (Danny) is highlighted with a red box. The "deaths" column for row 10 (Katrina) is highlighted with a red box. The "name" column for row 13 (Donna) is highlighted with a green box. The "damage" column for row 13 (Donna) is also highlighted with a green box. The "name" column for row 1 (Baker) is also highlighted with a green box.

	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

columns



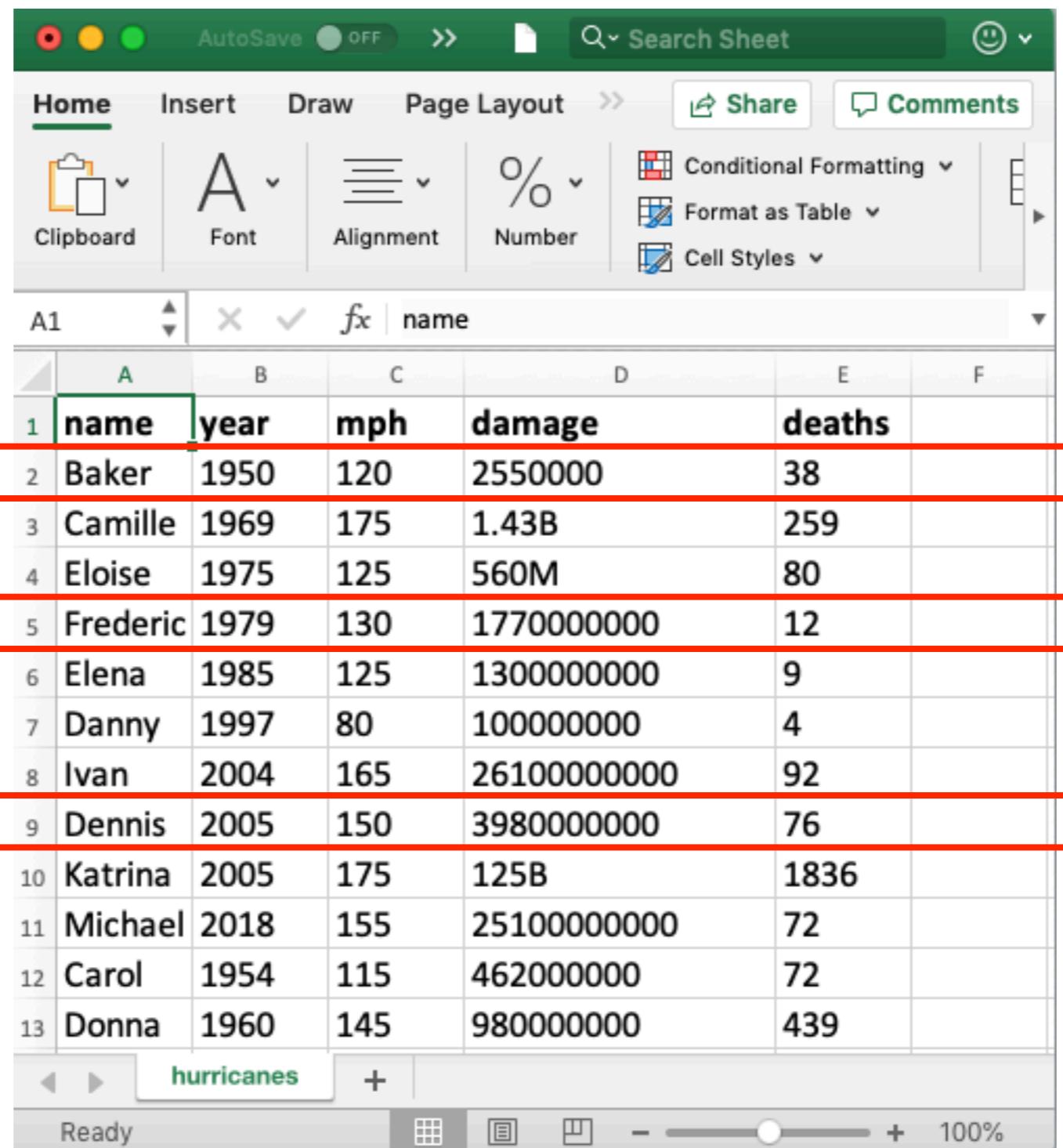
The screenshot shows a Microsoft Excel spreadsheet titled "Search Sheet". The ribbon menu is visible at the top, with "Home" selected. The table consists of 13 rows and 6 columns, labeled A through F. The columns are: name, year, mph, damage, and deaths. The first row contains column headers. The "name" 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

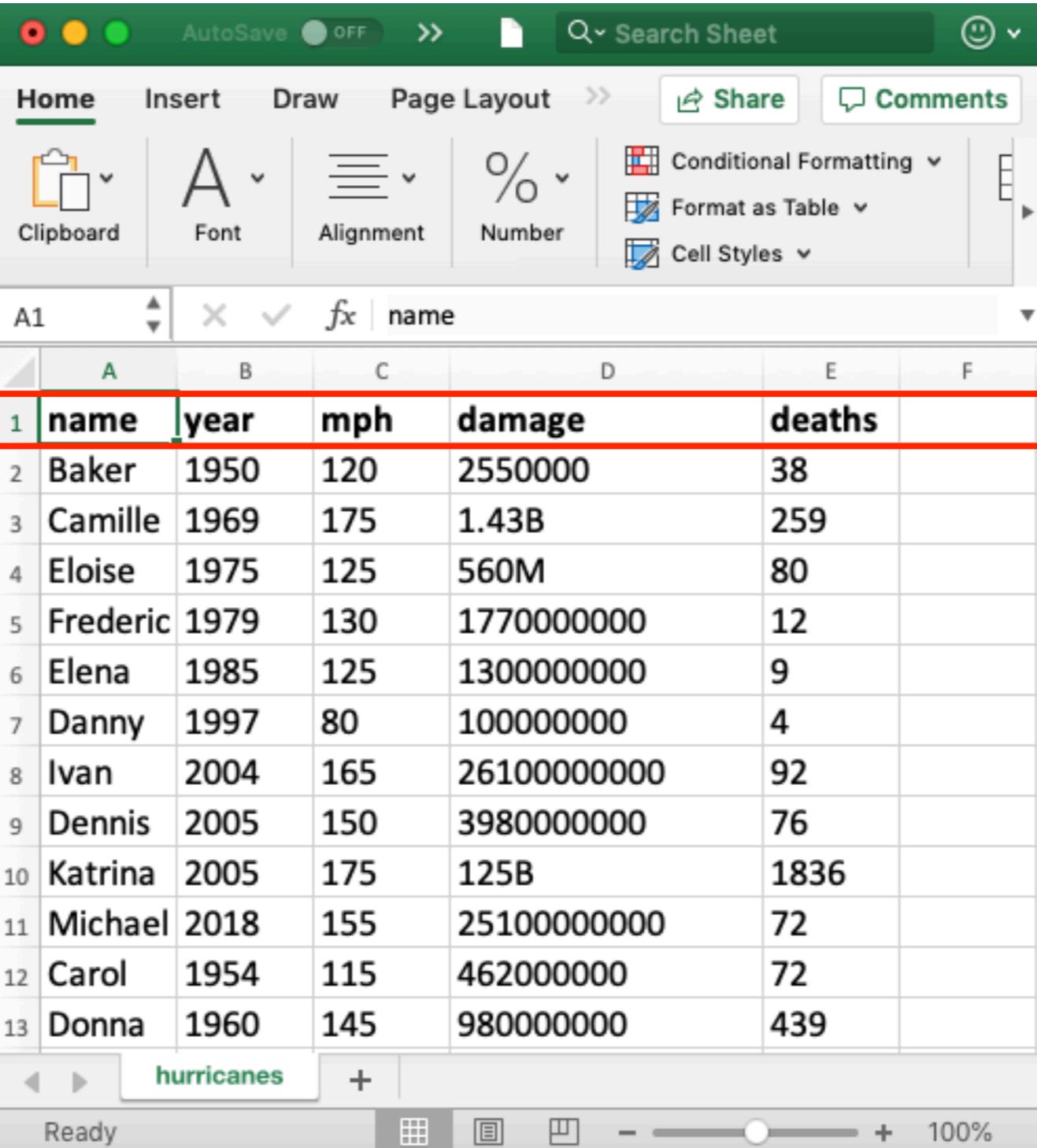


A screenshot of a Microsoft Excel spreadsheet titled "Search Sheet". The ribbon menu is visible at the top, showing "Home" as the active tab. The formula bar shows "A1" and "name". The main area contains a table of data with the following columns: name, year, mph, damage, and deaths. The first row is a header, and the subsequent 13 rows are data entries. The rows are highlighted with red boxes. The table has a green border. The bottom of the screen shows the status bar with "Ready" and various 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



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. The column headers are "name", "year", "mph", "damage", and "deaths". The data rows are numbered 1 through 13. The table is set against a grid background.

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)

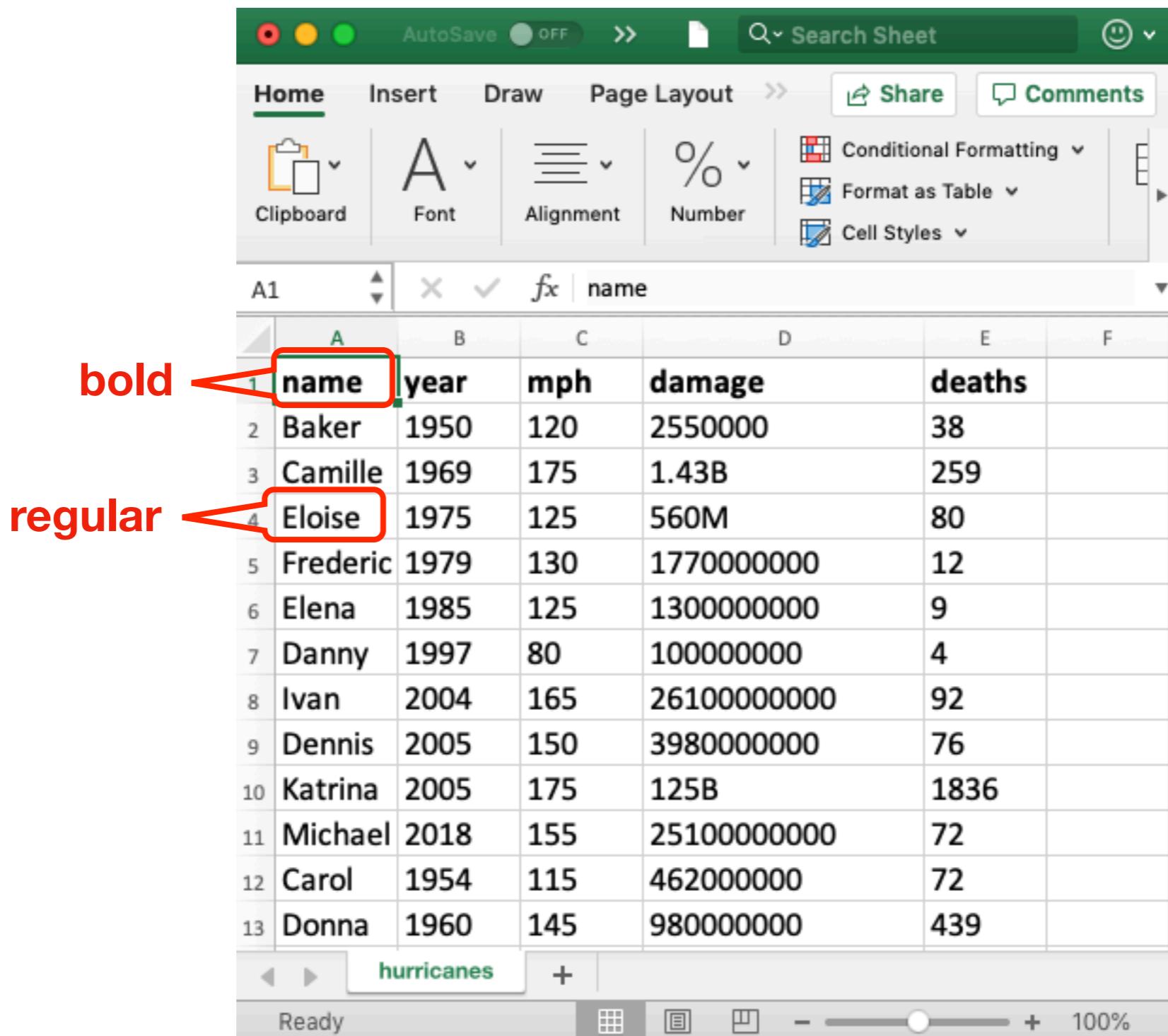
Spreadsheets often allow different **data types**

text      numbers

	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**



The screenshot shows a Microsoft Excel spreadsheet titled "Search Sheet". The ribbon menu is visible at the top, with "Home" selected. The font dropdown on the ribbon is set to bold. The table below contains 13 rows of data about hurricanes. The first row, labeled "name", is bolded, while the second row, "Baker", is in regular font. The columns are labeled "name", "year", "mph", "damage", and "deaths". The "damage" column contains large numerical values, some with commas and some with underscores.

	A	B	C	D	E	F
1	<b>name</b>	year	mph	<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 for name, year, mph, damage, and deaths. The first row is a header. A red box highlights the '+' button in the bottom-left corner of the sheet tab bar, which is labeled 'hurricanes'. A red arrow points from this button to the text 'more tables of data'.

	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	

Ready      + 100%

more tables of data

# Excel Files

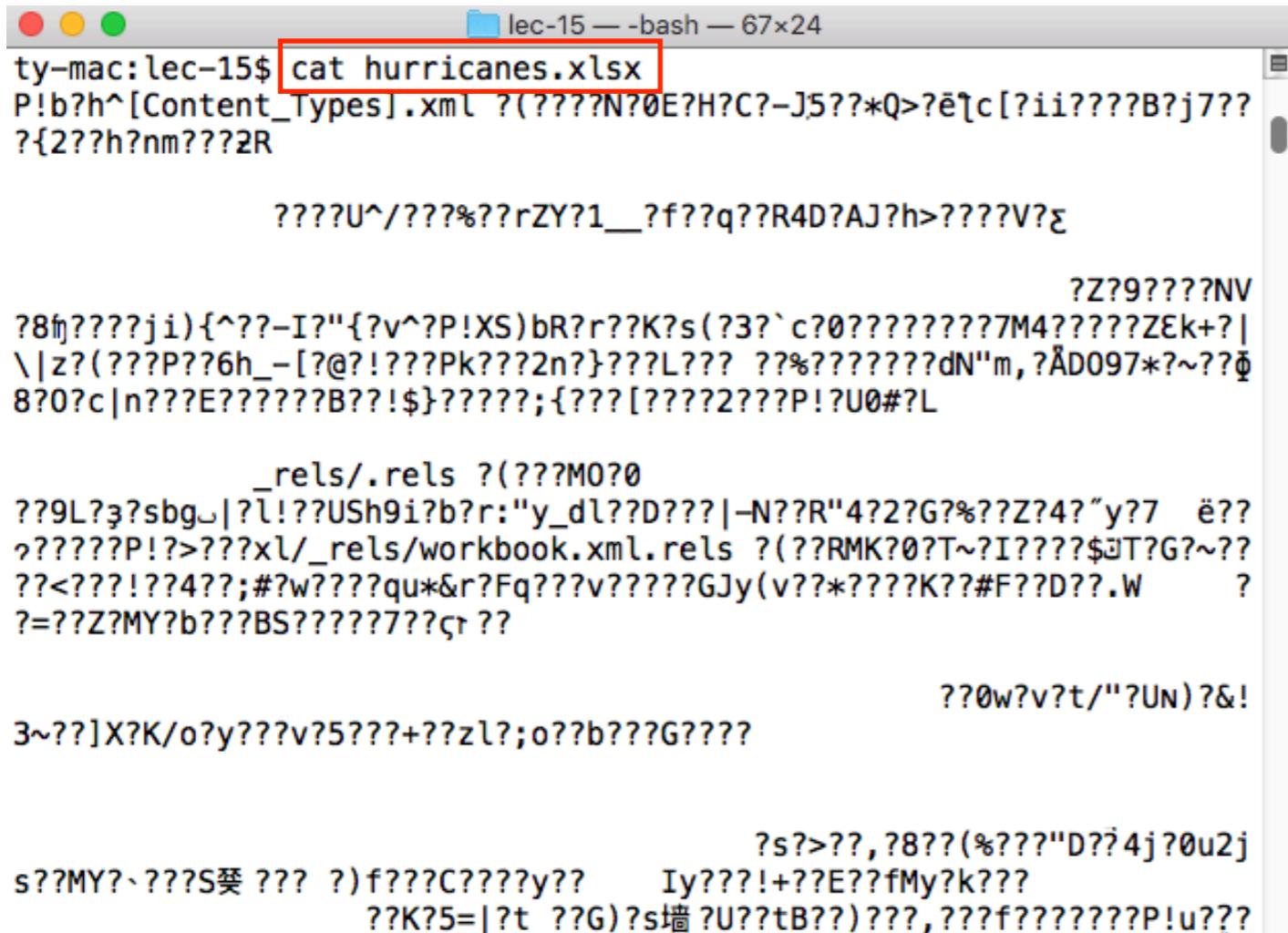
Extension: .xlsx

Format: binary

# Excel Files

Extension: .xlsx

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



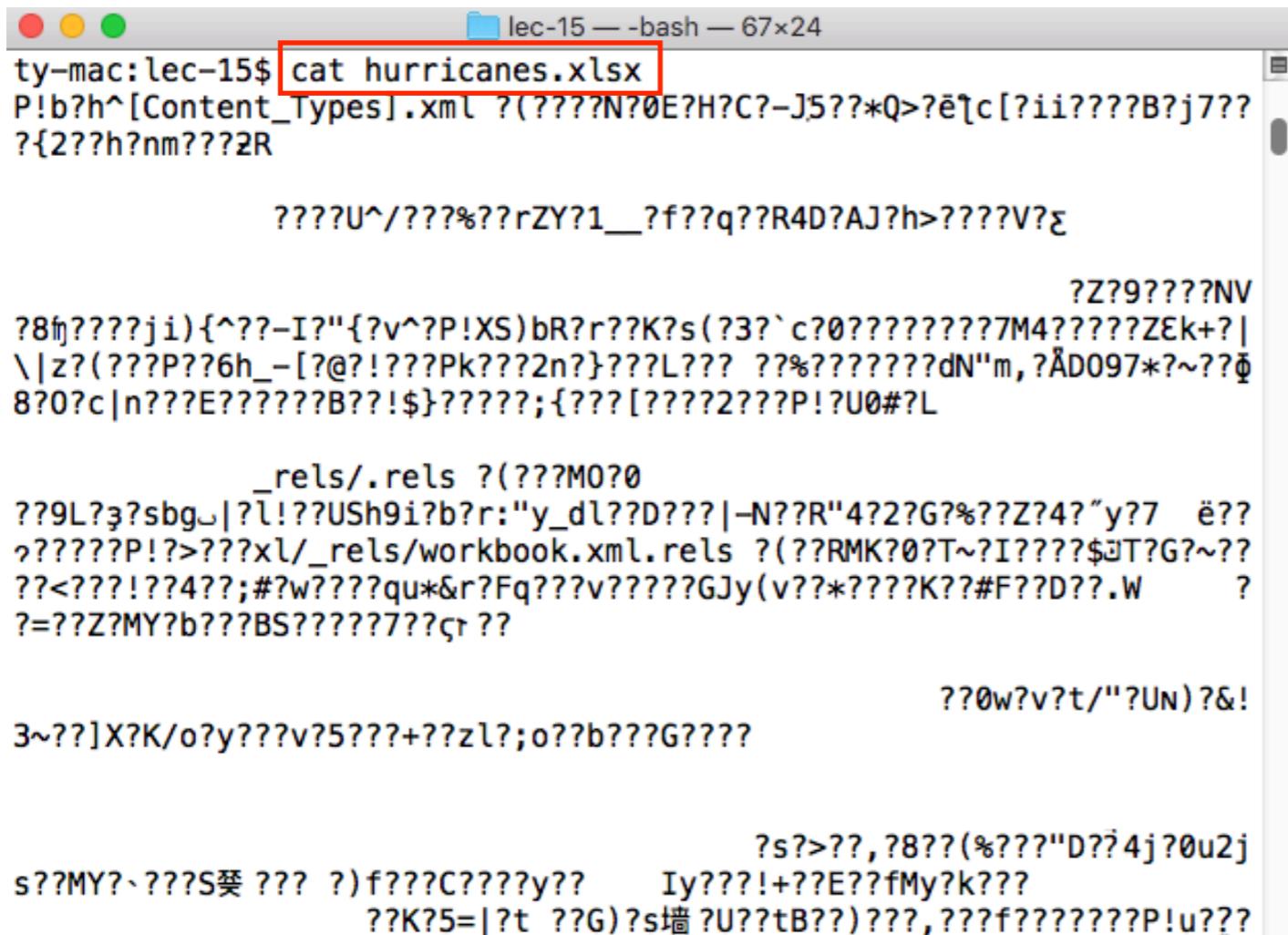
```
ty-mac:lec-15$ cat hurricanes.xlsx
P!b?h^[Content_Types].xml ?(????N?0E?H?C?-J5??*Q>?é]c[?ii????B?j7??
?{2??h?nm???R
????U^/????%??rZY?1__?f??q??R4D?AJ?h>????V?ξ
?Z?9????NV
?8ñ????ji){^??-I?"{?v^?P!XS)bR?r??K?s(?3?`c?0?????????M4?????ZEk+?|
\|z?(??P??6h_-[@?!???Pk???2n?}??L??? ??%??????dN"m,?ÅD097*?~??Φ
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  é??
??????P!?>???xl/_rels/workbook.xml.rels ?(???RMK?0?T~?I?????$JT?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葵 ??? ?)f???C????y?? Iy??!!+??E??fMy?k???
??K?5=|?t ??G)?s墙 ?U??tB??)???,??f???????P!u???
```

# Excel Files

Extension: .xlsx

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



```
ty-mac:lec-15$ cat hurricanes.xlsx
P!b?h^[Content_Types].xml ?(????N?0E?H?C?-J5??*Q>?é]c[?ii????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*?~?? 
8?0?c|n???E??????B??!$}?????;{???[?????2???P!?U0#?L
_rels/.rels ?(???M0?0
??9L?3?sbg_!|?l!??USh9i?b?r:"y_dl?D???|-N??R"4?2?G?%??Z?4?"y?7  ??
??????P!?>???xl/_rels/workbook.xml.rels ?(???RMK?0?T~?I?????$ T?G?~??
??<???!??4??;#?w????qu*&r?Fq???v?????GJy(v??*????K??#F??D???.W      ?
?=??Z?MY?b???BS??????ct ???
??0w?v?t/"?UN)?&
3~??]X?K/o?y???v?5???+??zl?;o??b???G?????
?s?>??,?8??(%???"D??4j?0u2j
s??MY?、??S葵 ??? ?)f???C????y?? Iy??!+??E??fMy?k???
??K?5=|?t ??G)?s墙 ?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**

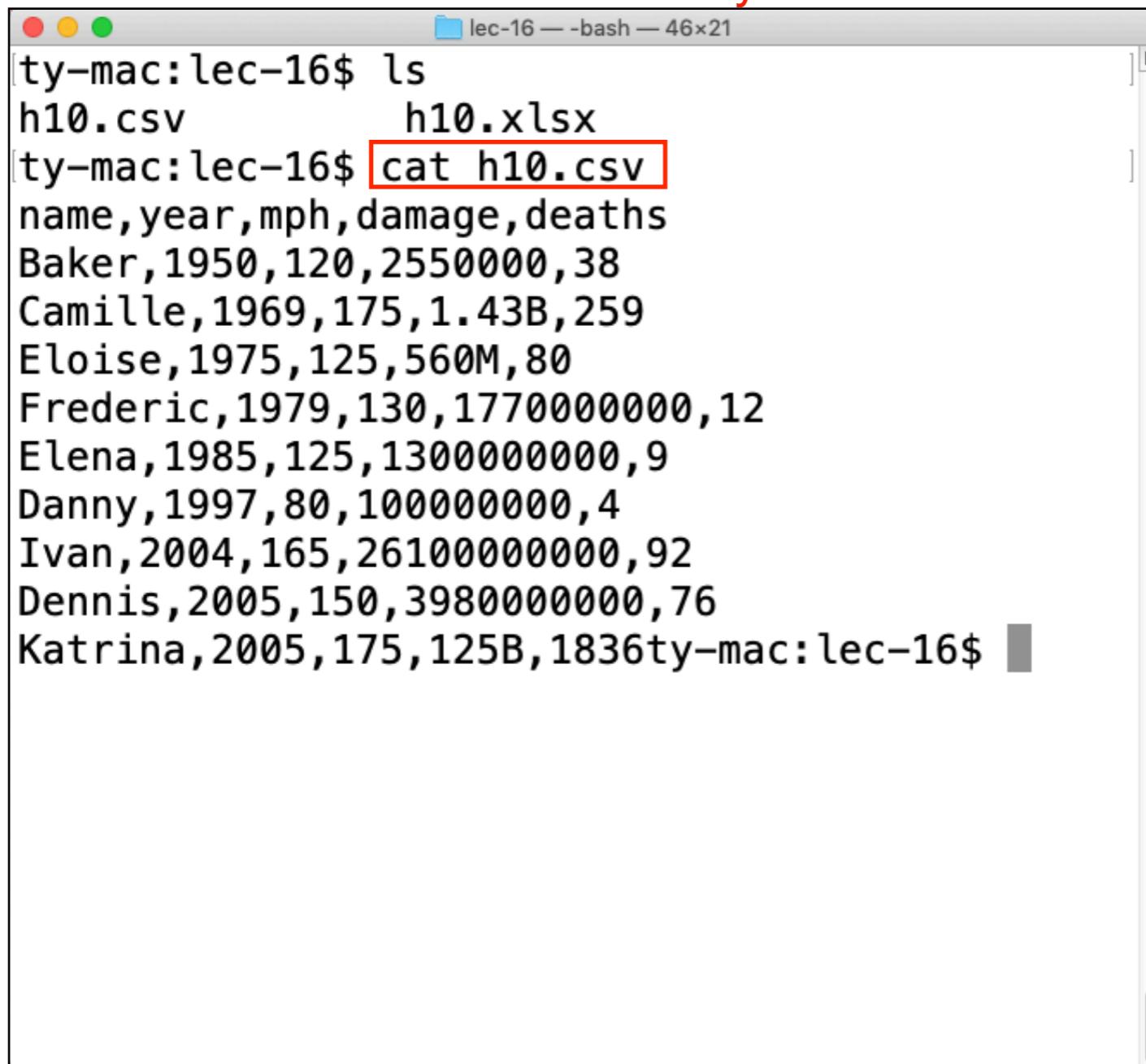
CSVs are like simple spreadsheets

- 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

# 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 "ls" followed by two files: "h10.csv" and "h10.xlsx". Below this, the command "cat h10.csv" is run, and its output is displayed. The output is a CSV file containing data about hurricanes. The columns are labeled "name", "year", "mph", "damage", and "deaths". The data rows are: 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, 1836.

```
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,1836
ty-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

# 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

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?

# Today's Outline

Spreadsheets

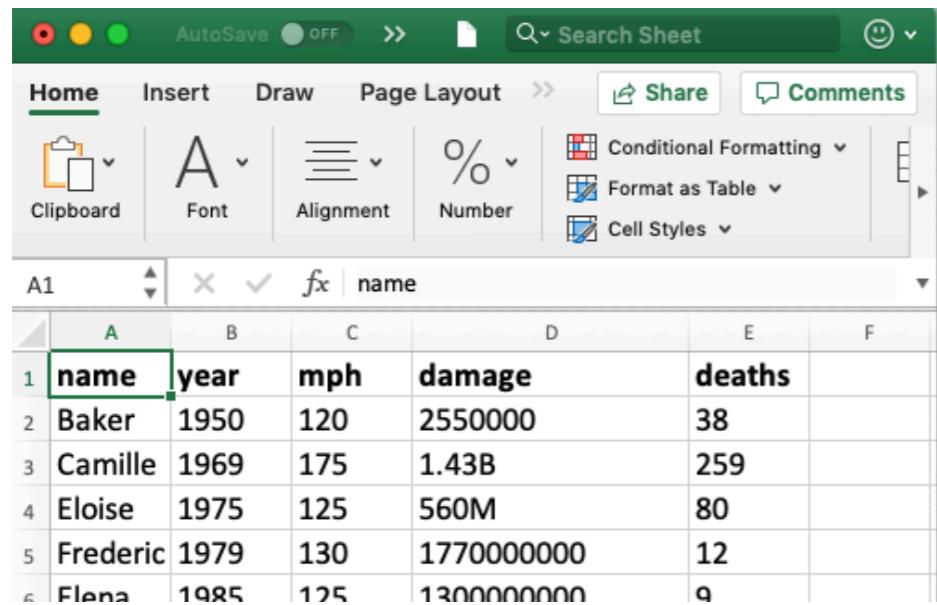
CSVs

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

Coding examples

# Data Management

## 1. spreadsheet in Excel

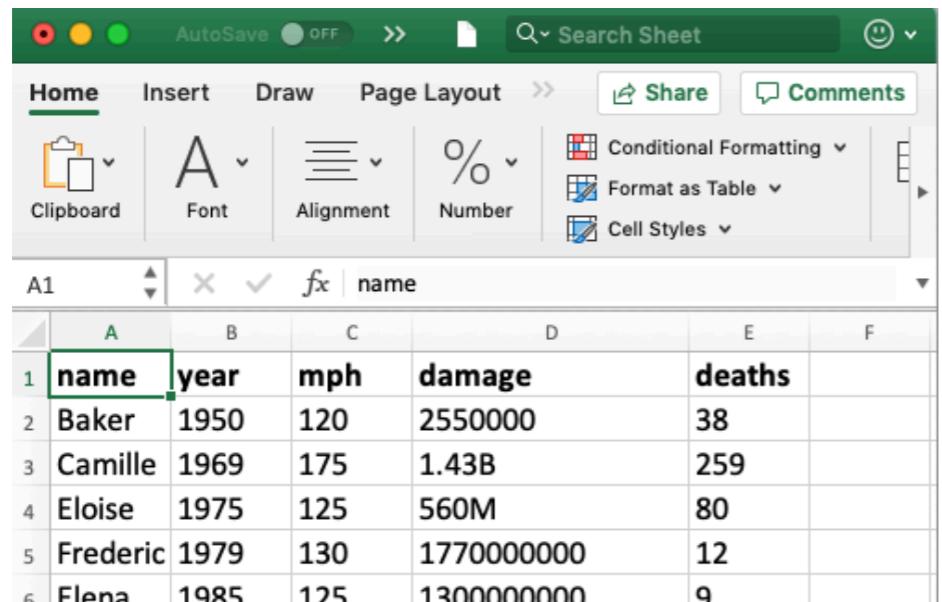


A screenshot of the Microsoft Excel application interface. The ribbon at the top shows tabs for Home, Insert, Draw, Page Layout, Share, and Comments. The Home tab is selected. Below the ribbon, there are toolbars for Clipboard, Font, Alignment, and Number. The main area shows a table with data starting from row 1. Row 1 contains column headers: name, year, mph, damage, and deaths. Rows 2 through 6 contain data for Hurricane Baker, Camille, Eloise, Frederic, and Elena respectively. The 'damage' column for Frederic and Elena contains extremely large values (17700000000 and 13000000000) which are likely artifacts from the original image.

	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	13000000000	9	

# Data Management

## 1. spreadsheet in Excel



The screenshot shows an Excel spreadsheet with the following data:

	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
```

# Data Management

## 3. Python Program

### 1. 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

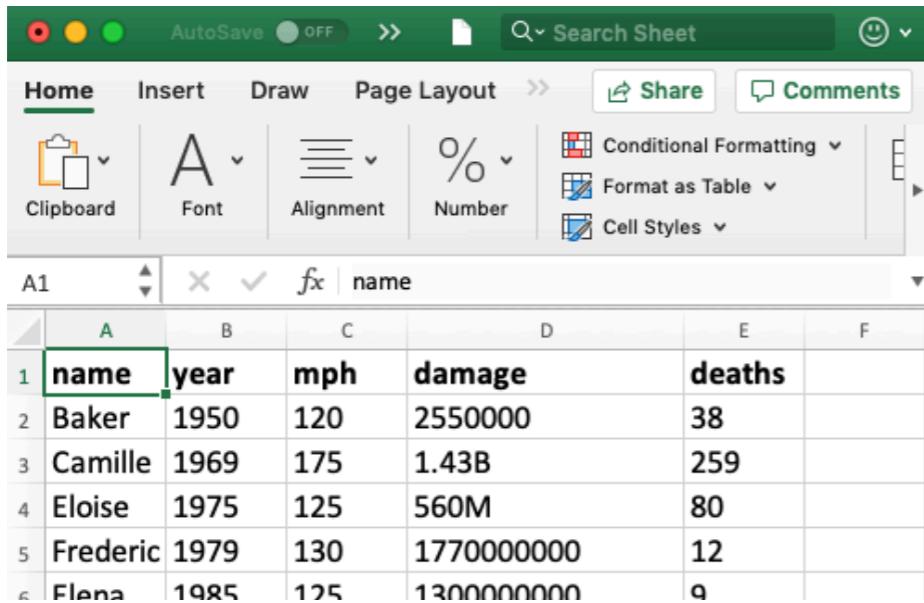
### 2. CSV file saved somewhere

```
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
```

# Data Management

## 3. Python Program

### 1. 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

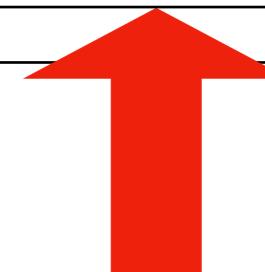
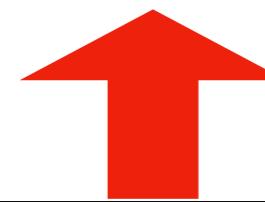
[

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

...

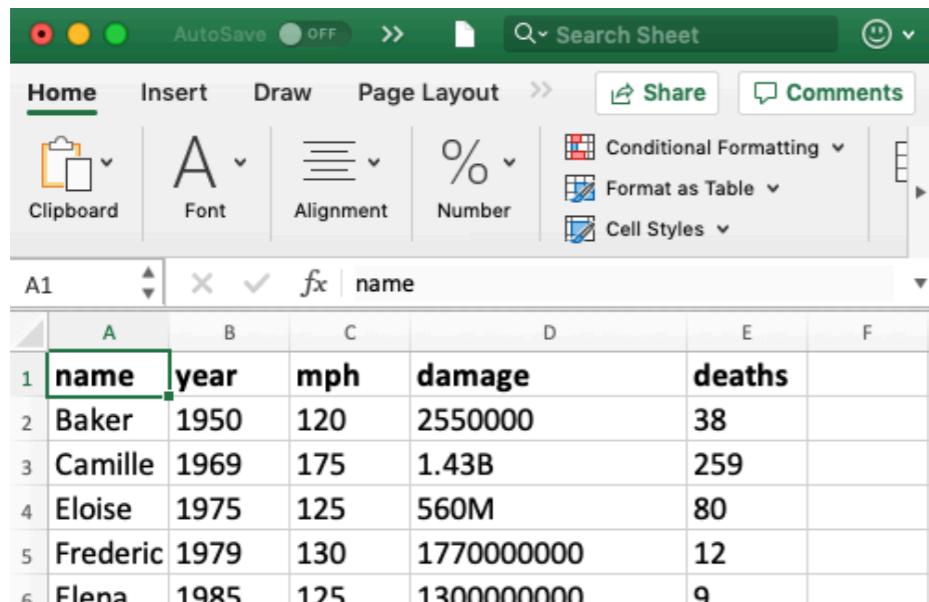
]

Parsing Code



# Data Management

## 1. 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] → ???`

[                    ]

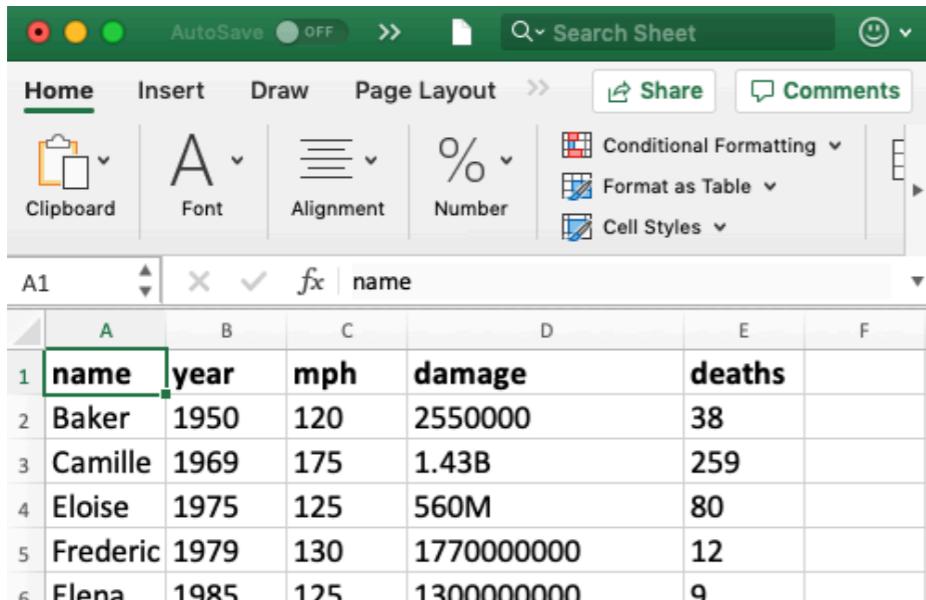
[ "name", "year", ... ],  
[ "Baker", "1950", ... ],

...

Parsing Code

# Data Management

## 1. 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"
```

list of lists

[

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

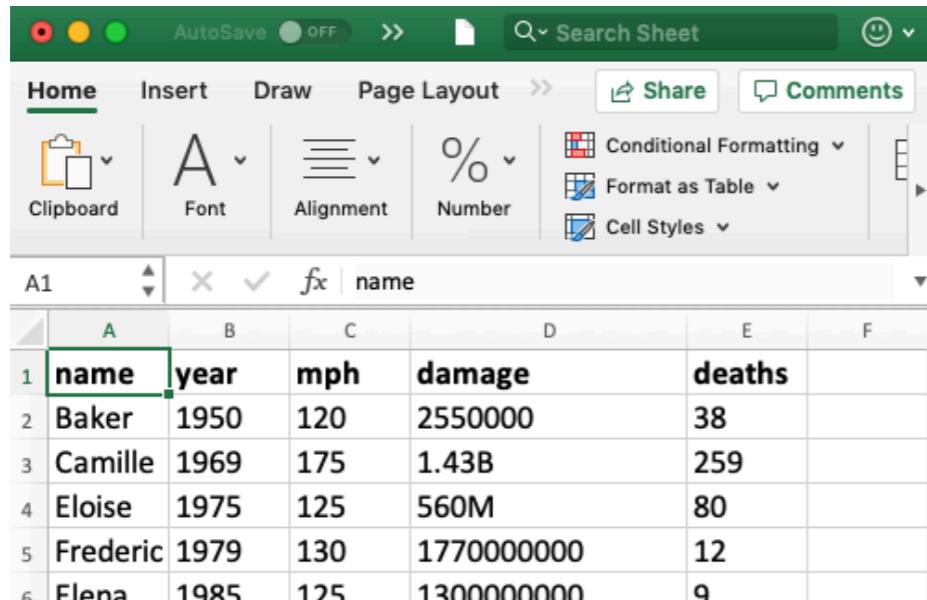
...

]

### Parsing Code

# Data Management

## 1. 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] → ???`

[

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

...

]

Parsing Code

# Data Management

## 1. 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

## 1. 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", ... ],`

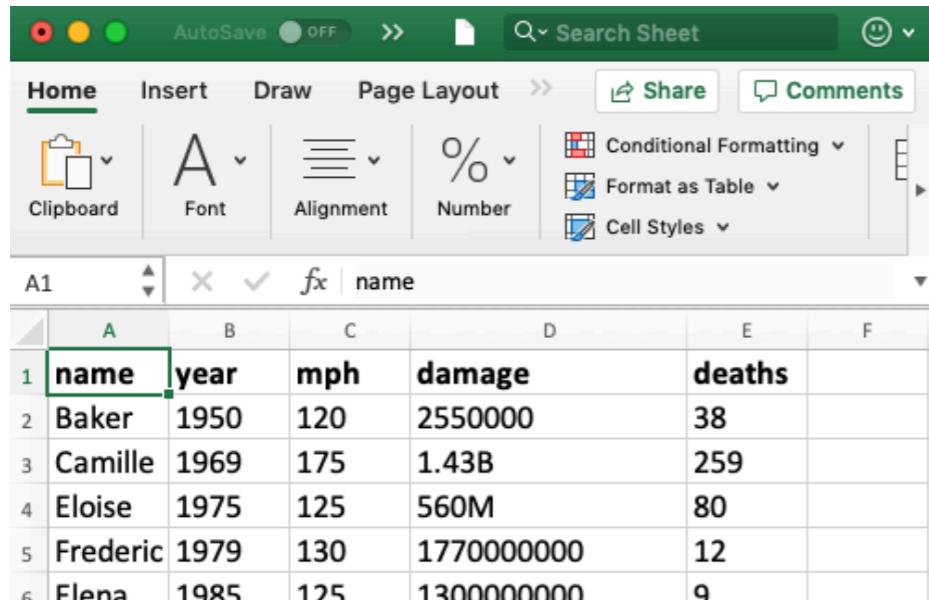
...

]

Parsing Code

# Data Management

## 1. 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"`

list of lists

[

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

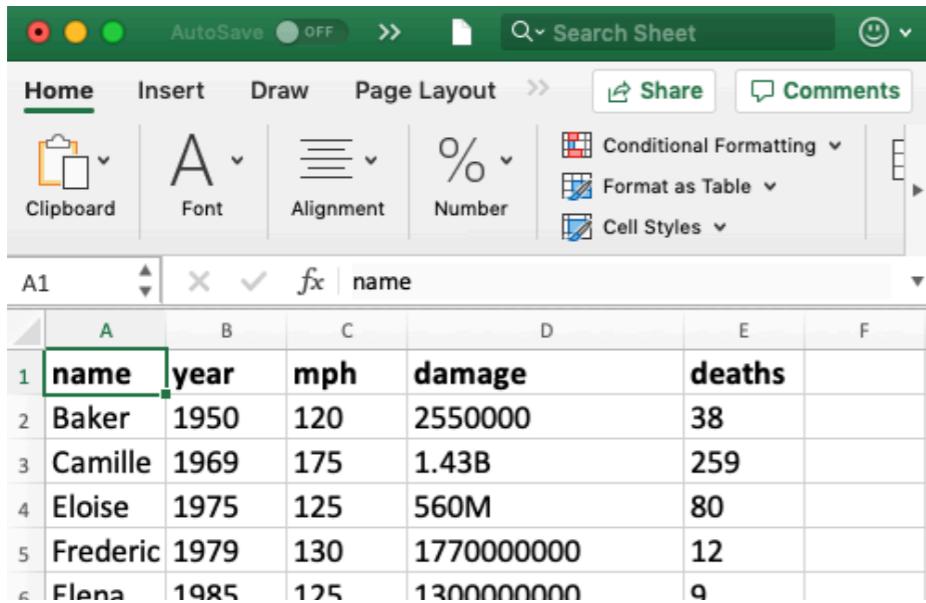
...

]

Parsing Code

# Data Management

## 1. 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[0][-2] → ???`

list of lists

[

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

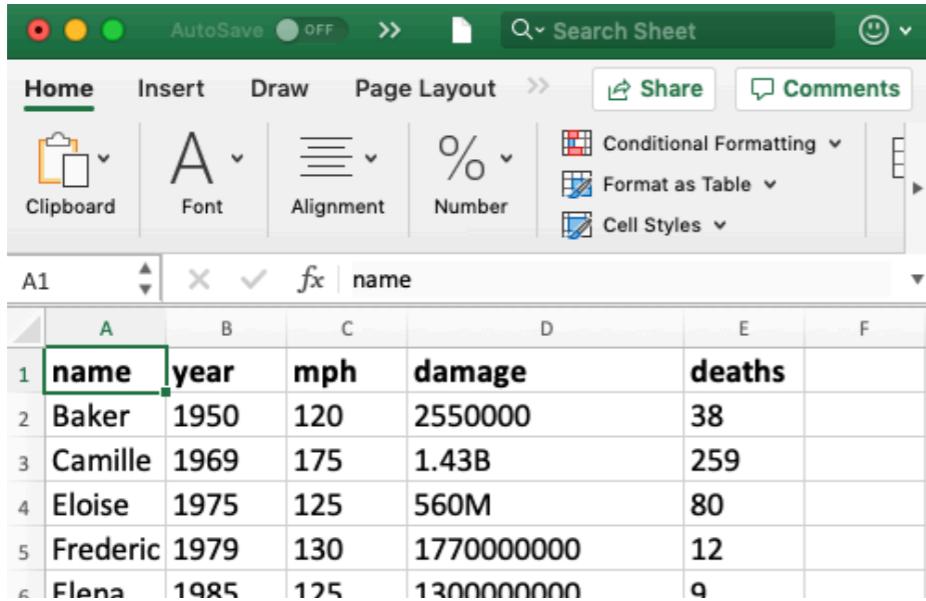
...

]

Parsing Code

# Data Management

## 1. 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

## 1. spreadsheet in Excel

	Name	Date	Time	Status	Latitude	Longitude	WindSpeed	Ocean
1	HEIDI	19671019	1200	TD	20.5N	54.0W	25	Atlantic
2	OLAF	19850822	0	TD	12.9N	102.2W	25	Pacific
3	TINA	19920917	1200	TD	10.4N	98.5W	25	Pacific
4	EMMY	19760820	1200	TD	14.0N	48.0W	20	Atlantic
5	FOUR	20061007	1800	DB	11.3N	159.0W	25	Pacific
6	FLOSSIE	19780904	0	TD	12.0N	39.0W	25	Atlantic
7	GLORIA	19760926	1200	TD	23.0N	58.0W	20	Atlantic
8	MARIA	20050901	1200	TD	18.8N	45.5W	30	Atlantic
9	GILBERT	19880908	1800	TD	12.0N	54.0W	25	Atlantic
10	FELICE	19700912	0	TD	25.5N	77.5W	25	Atlantic
11								

Save As  
.CSV

## 2. CSV file saved somewhere

```
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
```

## 3. Python Program

Analysis Code

`rows[2][-1] → "Pacific"`

list of lists

[

[ "Name", "Date", ... ],  
[ "HEIDI", "19671019", ... ],  
...

]

Parsing Code

What does this look like?

# Example From Sweigart Ch 14

Code

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

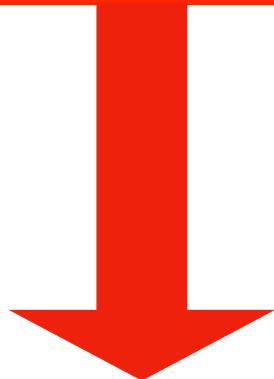
example.csv

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 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 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 From Sweigart Ch 14

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

**We'll eventually learn more about reading files.  
For now, let's copy and paste this to a function  
so we don't need to worry about it.**

# Example From Sweigart Ch 14

```
import csv

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

**We'll eventually learn more about reading files.  
For now, let's copy and paste this to a function  
so we don't need to worry about it.**

# Example From Sweigart Ch 14

```
import csv

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

We'll eventually learn more about reading files.  
For now, let's copy and paste this to a function  
so we don't need to worry about it.

# Example From Sweigart Ch 14

```
import csv

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

We'll eventually learn more about reading files.  
For now, let's copy and paste this to a function  
so we don't need to worry about it.

# Example From Sweigart Ch 14

```
import csv

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

We'll eventually learn more about reading files.  
For now, let's copy and paste this to a function  
so we don't need to worry about it.

# Example 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

We'll eventually learn more about reading files.  
For now, let's copy and paste this to a function  
so we don't need to worry about it.

# Example From Sweigart Ch 14

```
import csv

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

**keep this handy for copy/paste**

# Today's Outline

Spreadsheets

CSVs

Reading a CSV to a list of lists

**Coding examples**

# Demo 1: Restaurant Location Lookup

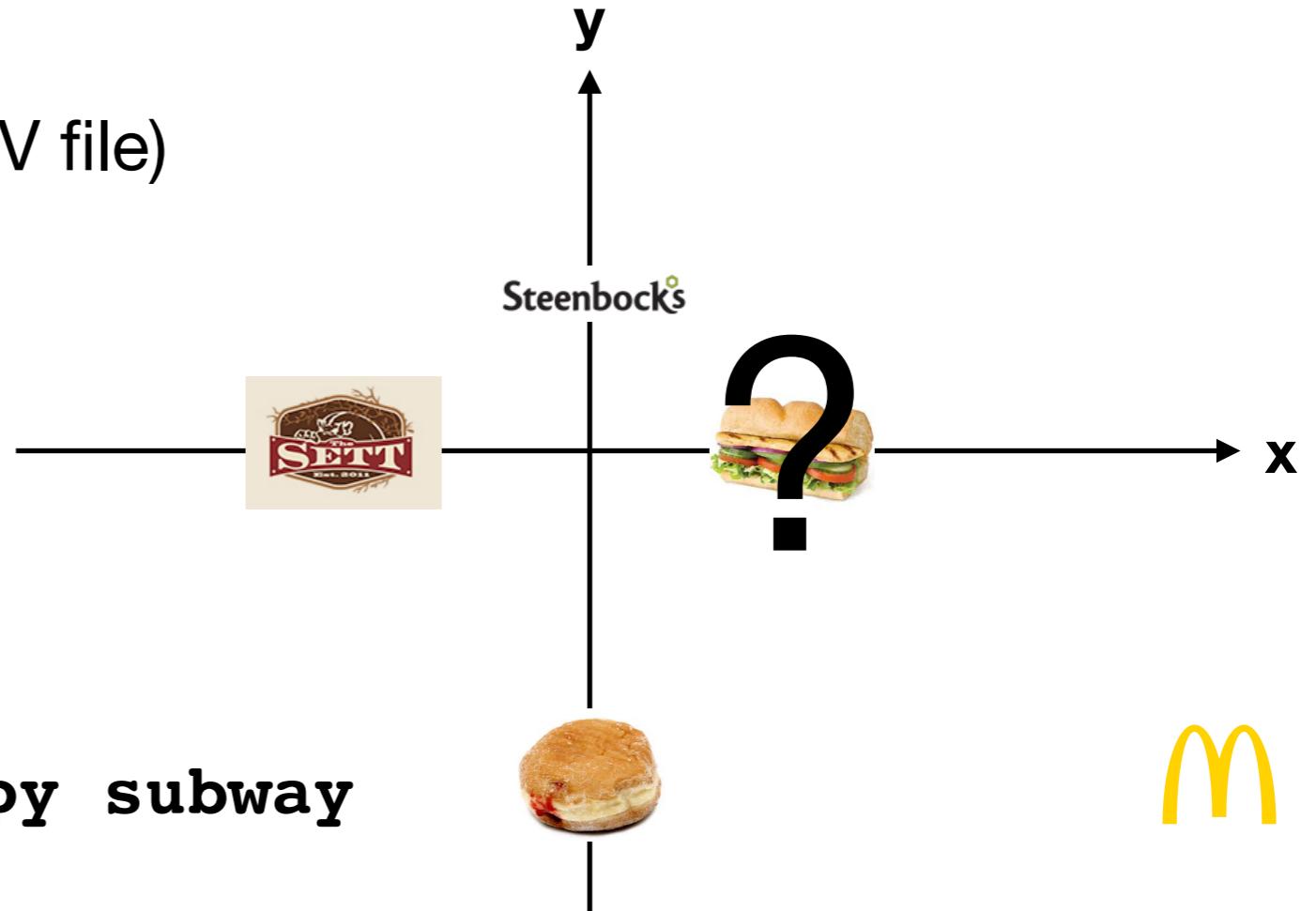
Goal: given a restaurant name, give x,y coordinates for it

## 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

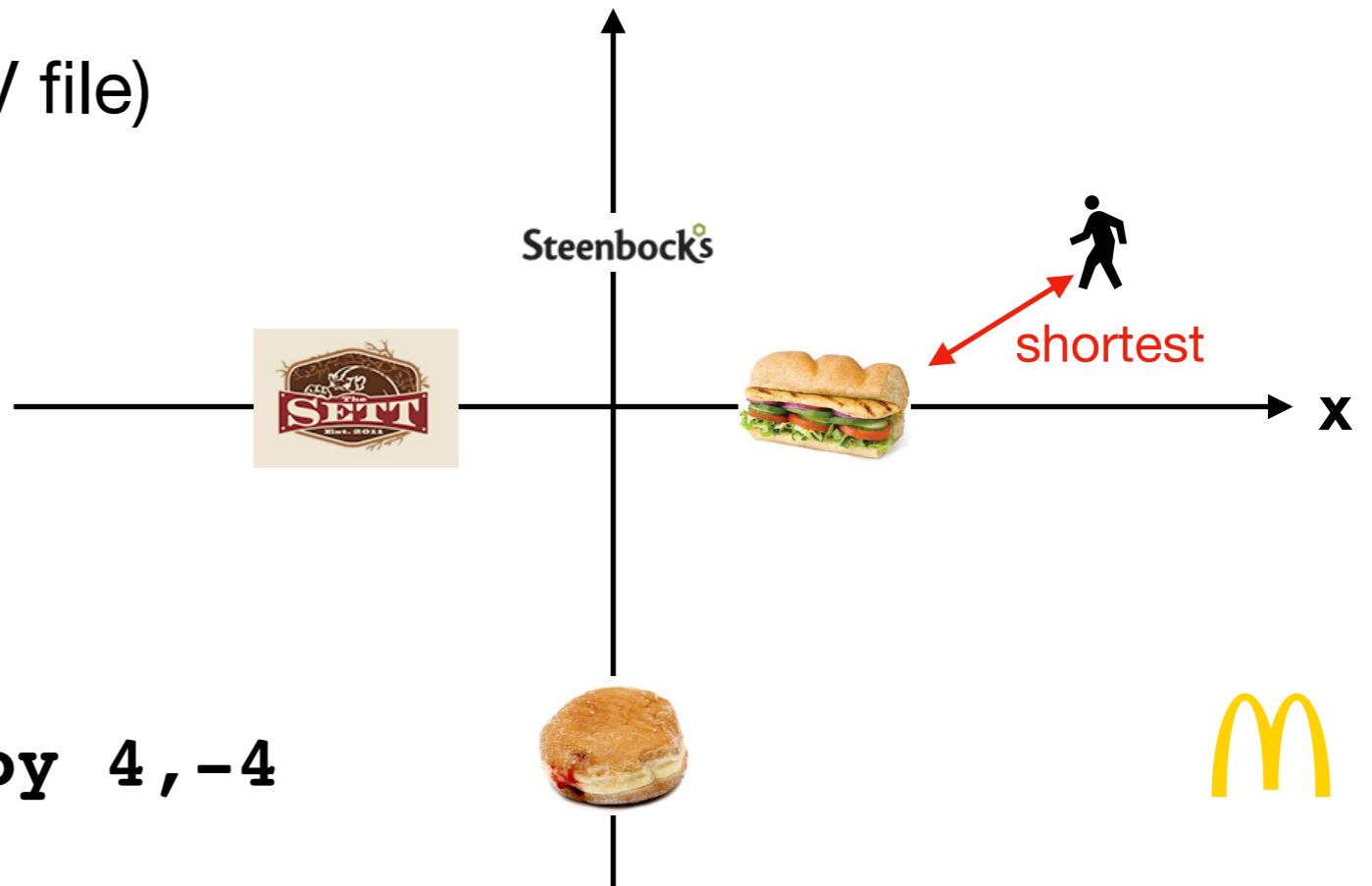
## Example:

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

McDonalds

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

The Sett

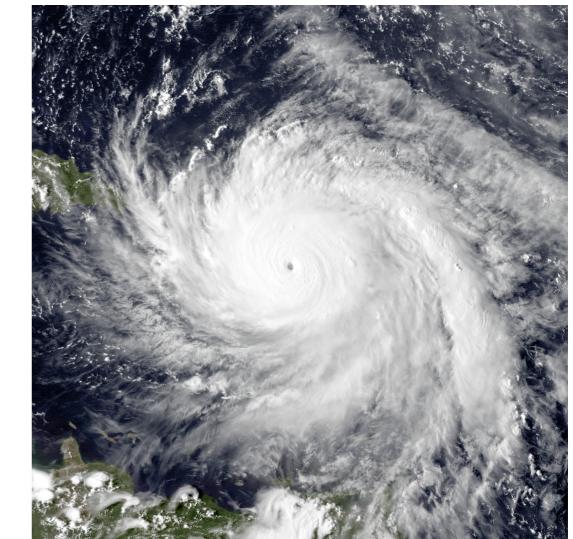


# Demo 3: Hurricane Column Dump

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

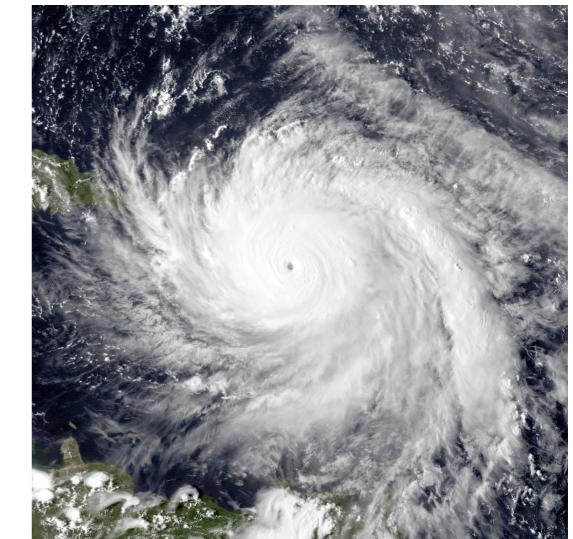
...

# 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



**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

what would it take to try to replicate this study?