[220 / 319] Tabular Data

Meena Syamkumar
Andy Kuemmel

Learning Objectives Today

CSV format

- purpose
- syntax
- comparison to spreadsheet

Reading CSV files

- without header
- with header
- type casting

Chapter 16 of Sweigart, to (and including) "Reading Data from Reader Objects in a for Loop"

https://automatetheboringstuff.com/2e/chapter16/

Today's Outline

Spreadsheets

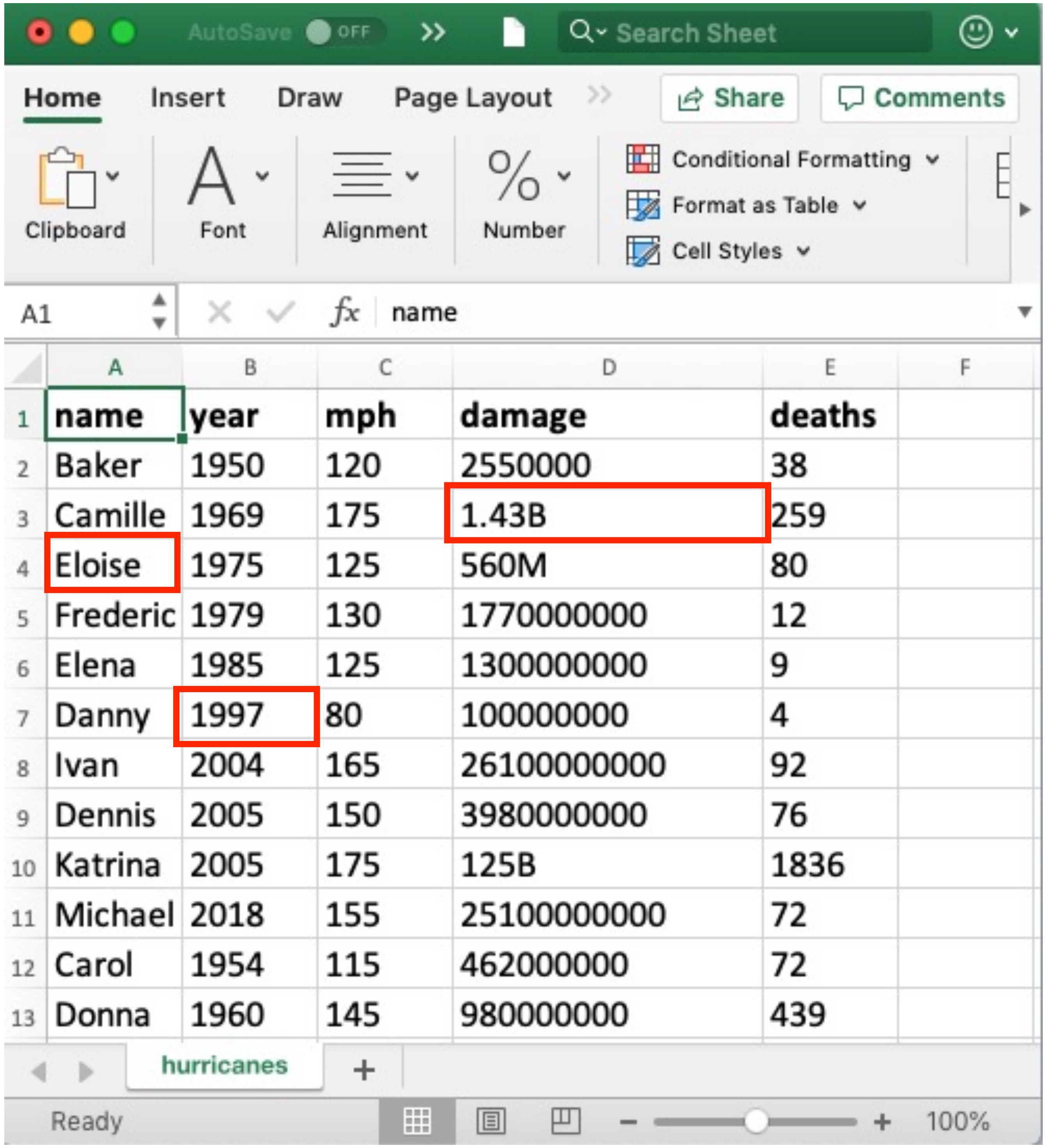
CSVs

Reading a CSV to a list of lists

Coding examples

Spreadsheets (e.g., Excel)

Spreadsheets are tables of cells, organized by rows and columns



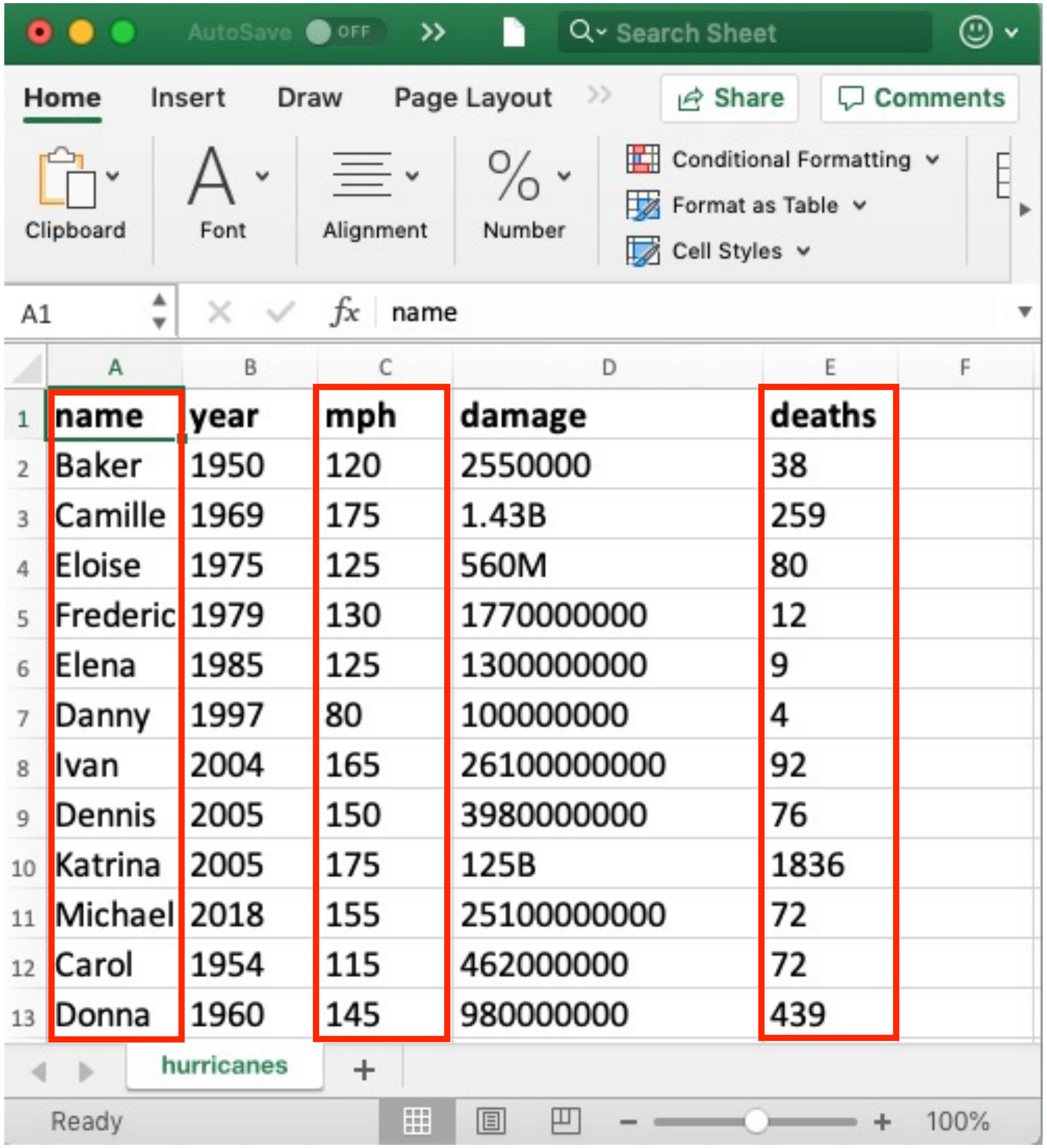
	A	B	C	D	E	F
1	**name**	**year**	**mph**	**damage**	**deaths**	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	
7	Danny	1997	80	100000000	4	
8	Ivan	2004	165	26100000000	92	
9	Dennis	2005	150	3980000000	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	

cells

Spreadsheets (e.g., Excel)

Spreadsheets are tables of cells, organized by rows and columns

columns

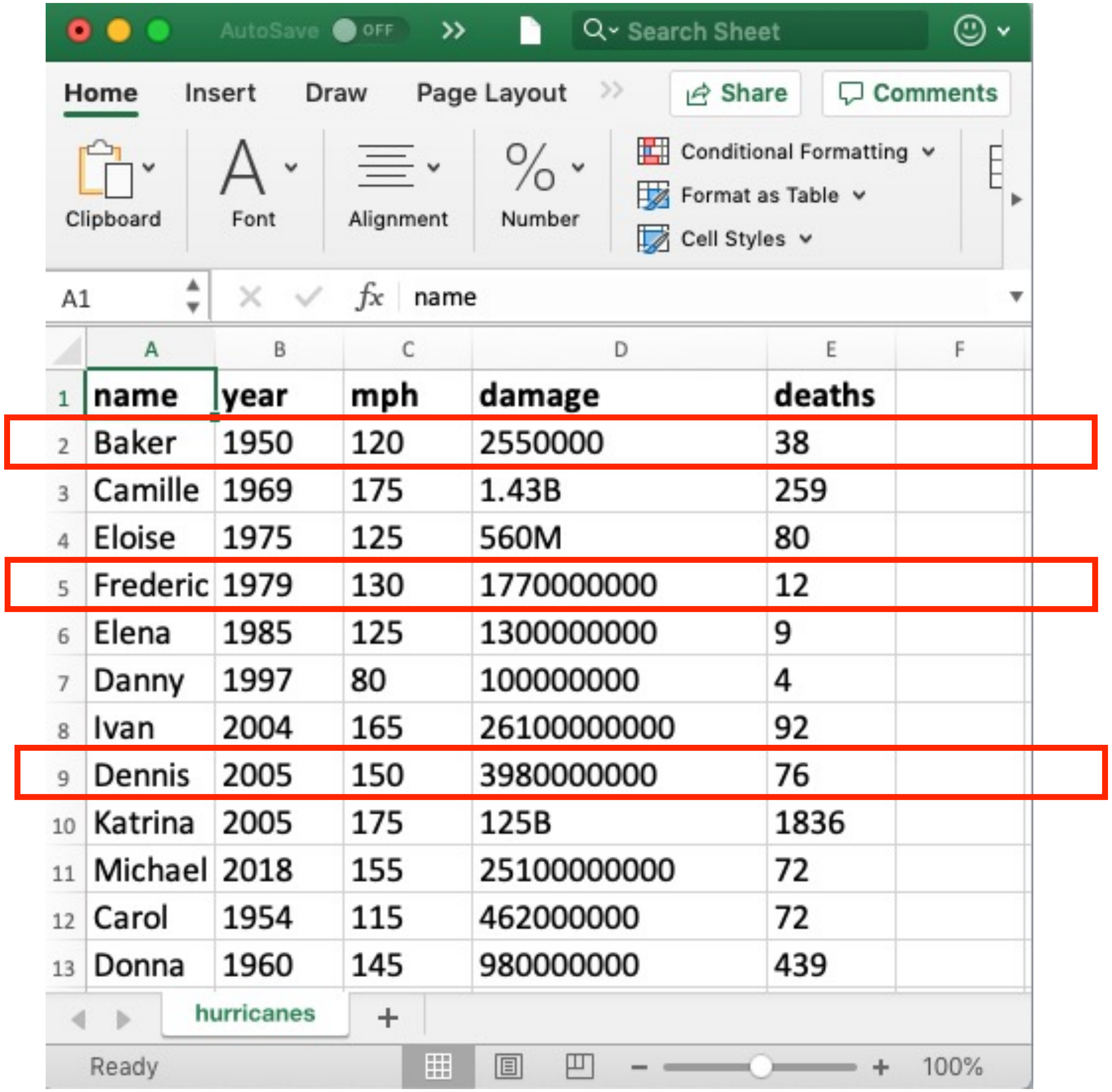


	A	B	C	D	E	F
1	**name**	**year**	**mph**	**damage**	**deaths**	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	
7	Danny	1997	80	100000000	4	
8	Ivan	2004	165	26100000000	92	
9	Dennis	2005	150	3980000000	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

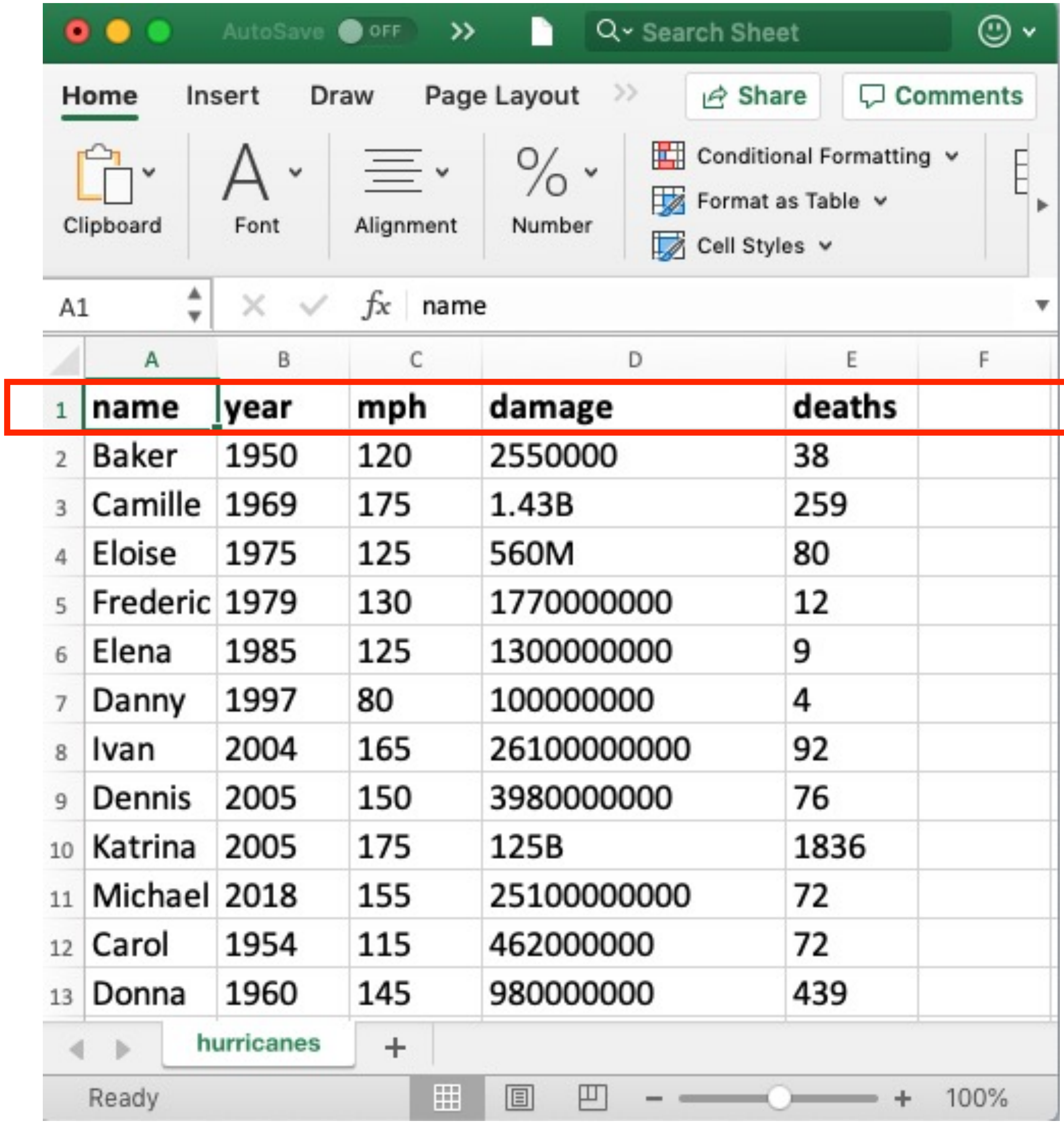
rows



	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



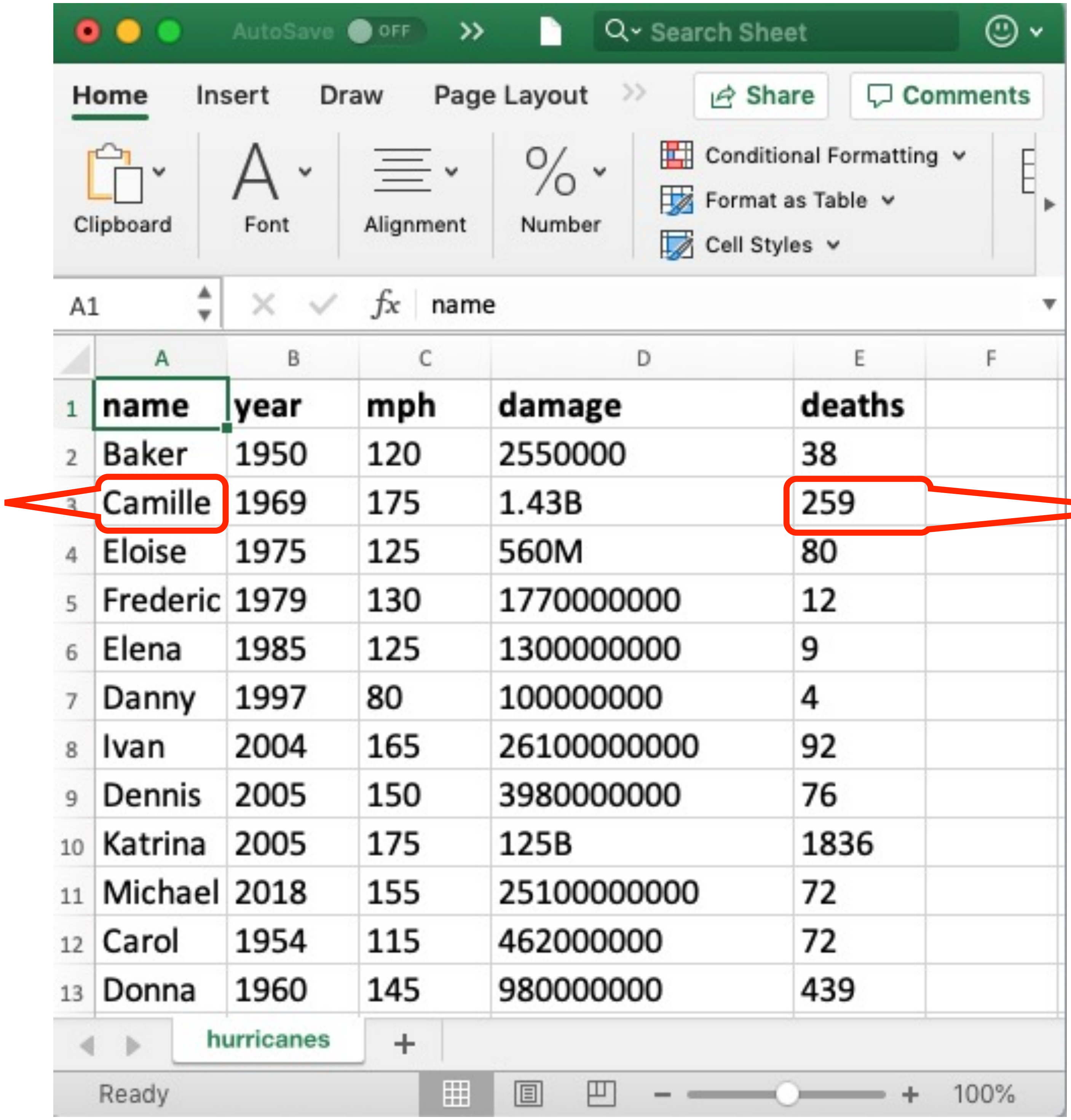
A1: name

header

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	26100000000	92
Dennis	2005	150	3980000000	76
Katrina	2005	175	125B	1836
Michael	2018	155	25100000000	72
Carol	1954	115	462000000	72
Donna	1960	145	980000000	439

Spreadsheets (e.g., Excel)

Spreadsheets often allow different data types



A1 *fx* name

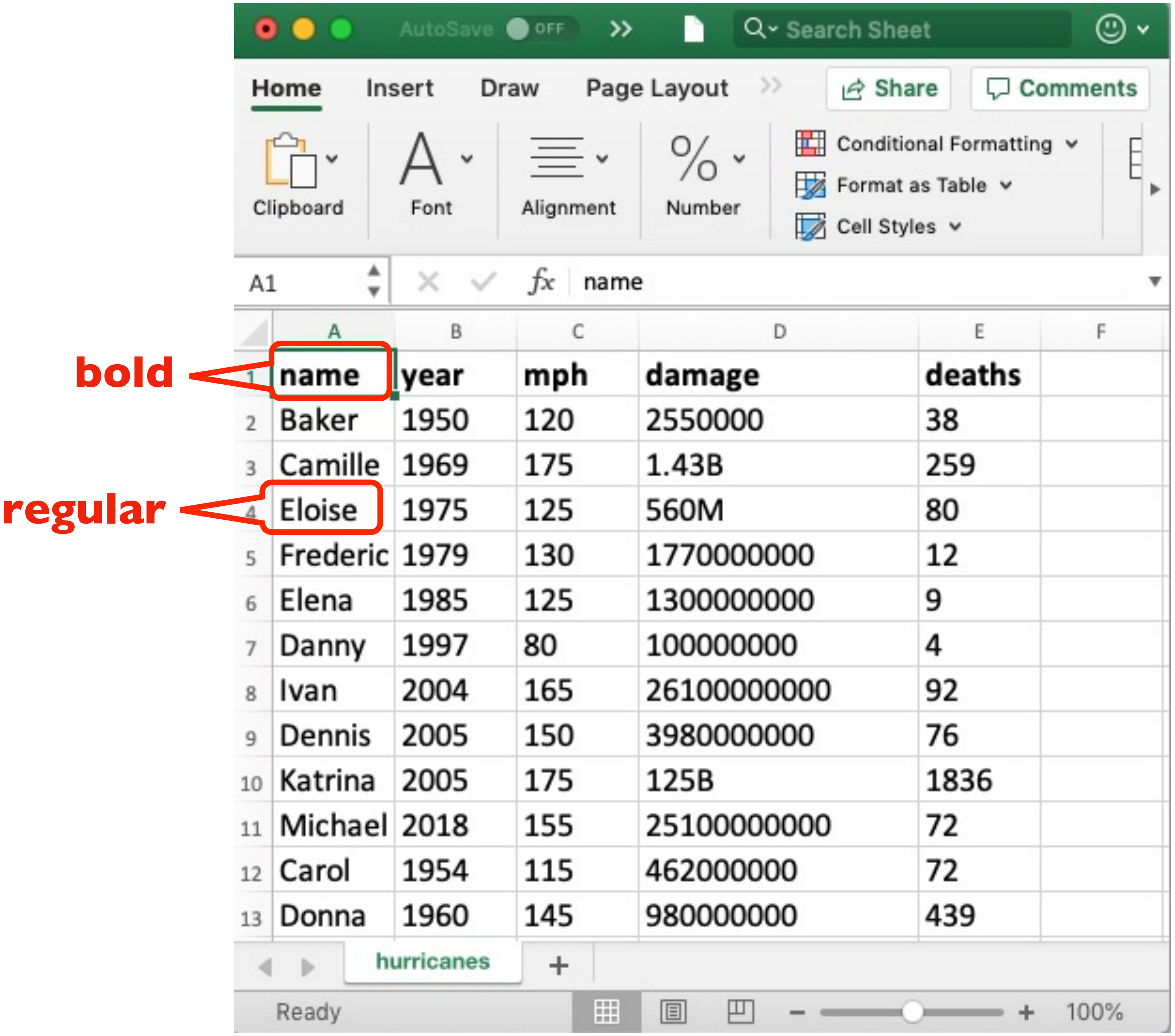
	A	B	C	D	E	F
1	**name**	**year**	**mph**	**damage**	**deaths**	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	
7	Danny	1997	80	100000000	4	
8	Ivan	2004	165	26100000000	92	
9	Dennis	2005	150	3980000000	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	

text

numbers

Spreadsheets (e.g., Excel)

Spreadsheets often allow different fonts

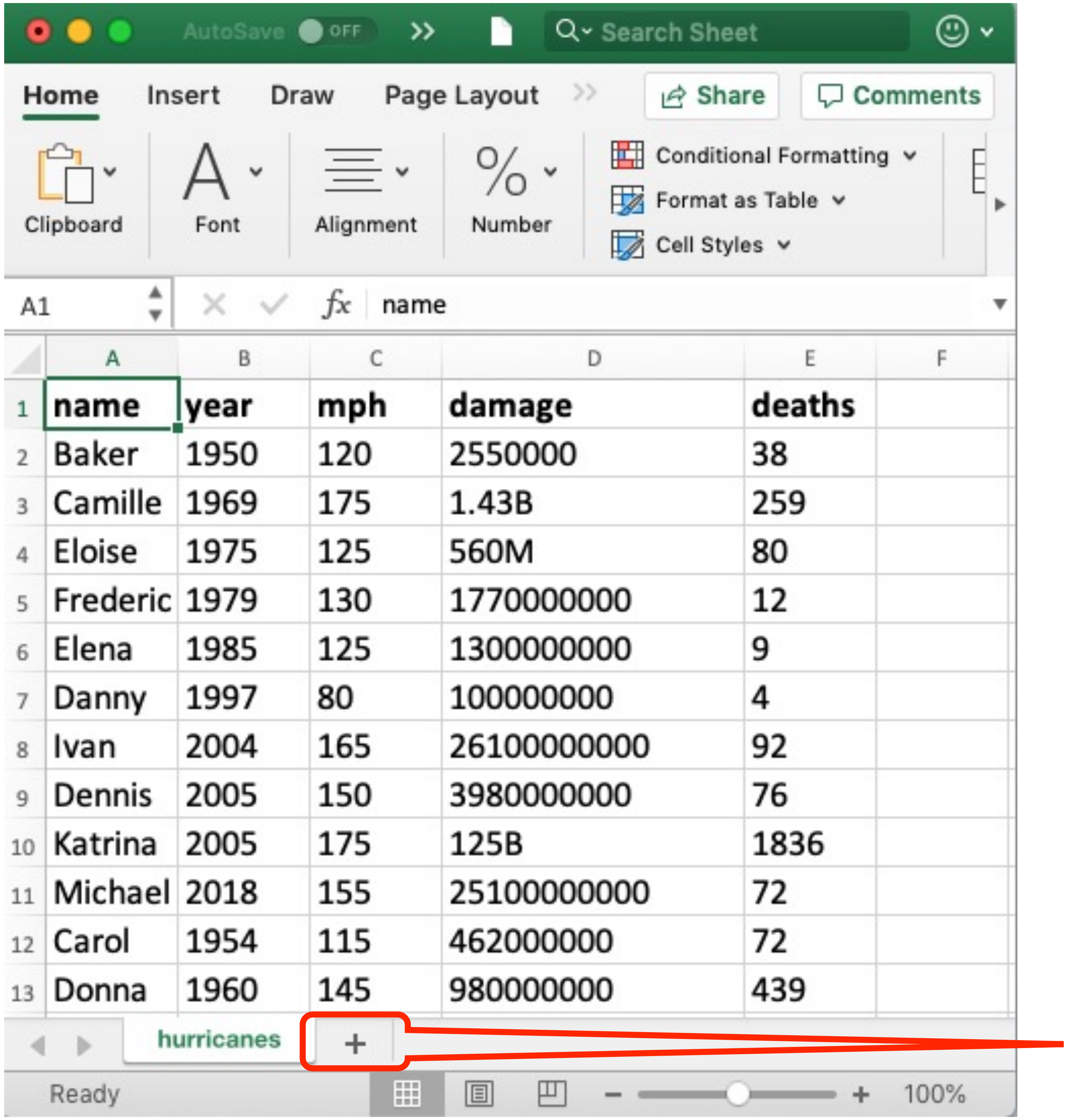


fx name

	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 often support multiple sheets



	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	

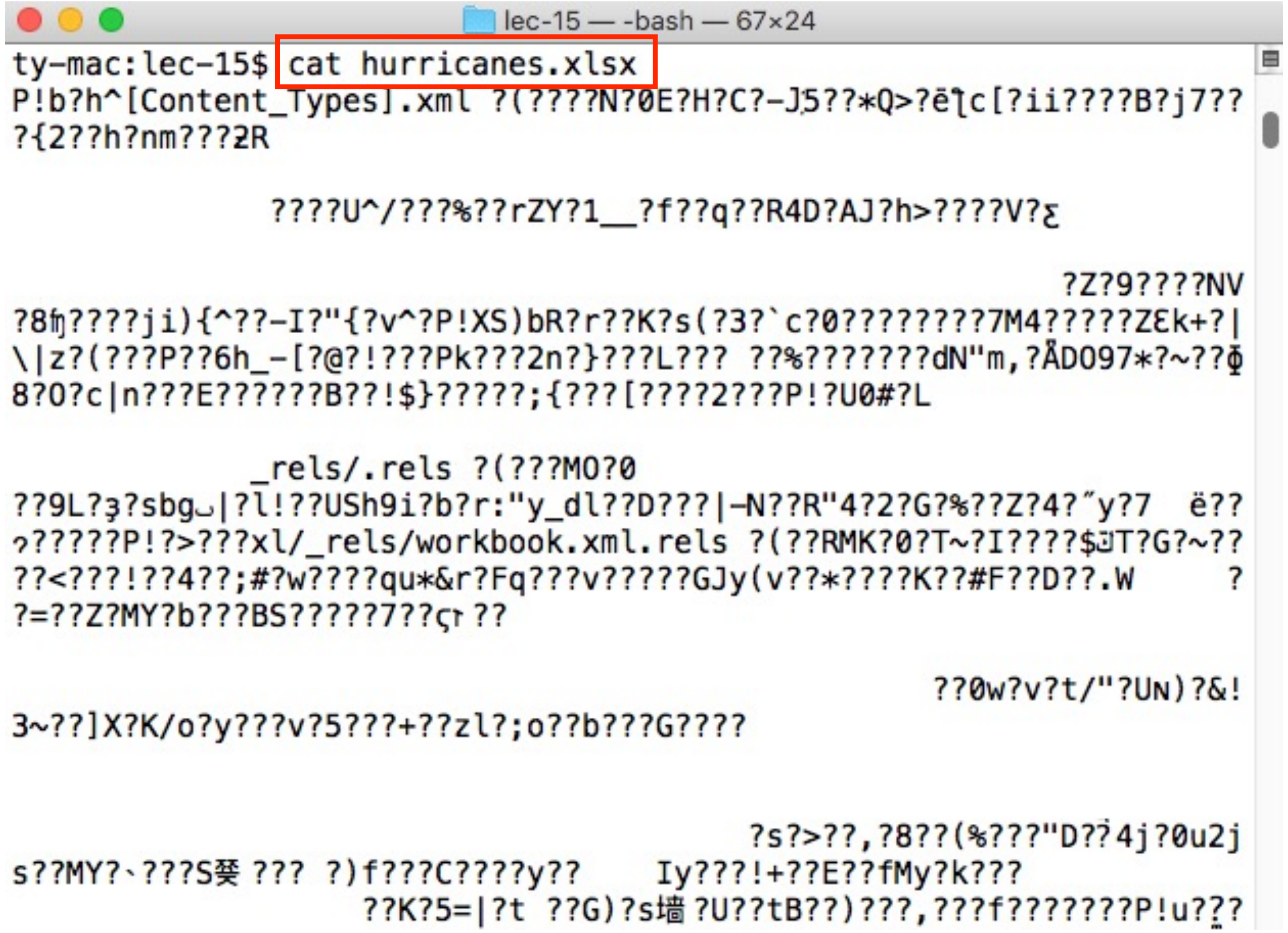
more tables of data

Excel Files

Extension: .xlsx

Format: binary

just 0's and 1's, not human-readable characters.
Need special software...



```
ty-mac:lec-15$ cat hurricanes.xlsx
P!b?h^[Content_Types].xml ?(????N?0E?H?C?-J5??*Q>?ē̃c[?ii????B?j7??
?{2??h?nm???ƻR

               ????U^/???%??rZY?1__?f??q??R4D?AJ?h>????V?ƹ

                                                           ?Z?9????NV
?8ŋ????ji){^??-I?"{?v^?P!XS)bR?r??K?s(?3?`c?0?????????7M4?????ZƐk+?|
\|z?(???P??6h_-[?@?!???Pk????2n?}???L??? ??%???????dN"m,?ÅDO97*?~??ɸ
8?O?c|n???E??????B??!$}?????;{???[????2???P!?U0#?L

              _rels/.rels ?(???MO?0
??9L?ȝ?sbgـ|?l!??USh9i?b?r:"y_dl??D???|-N??R"4?2?G?%??Z?4?˝y?7  ë??
ɂ?????P!?>???xl/_rels/workbook.xml.rels ?(??RMK?0?T~?I????$ܩT?G?~??
??<???!??4??;#?w????qu*&r?Fq???v?????GJy(v??*????K??#F??D??.W     ?
?==??Z?MY?b???BS?????7??ςɼ ??

                                                    ??0w?v?t/"?Uɴ)?&!
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墻 ?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-**S**eparated **V**alues

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

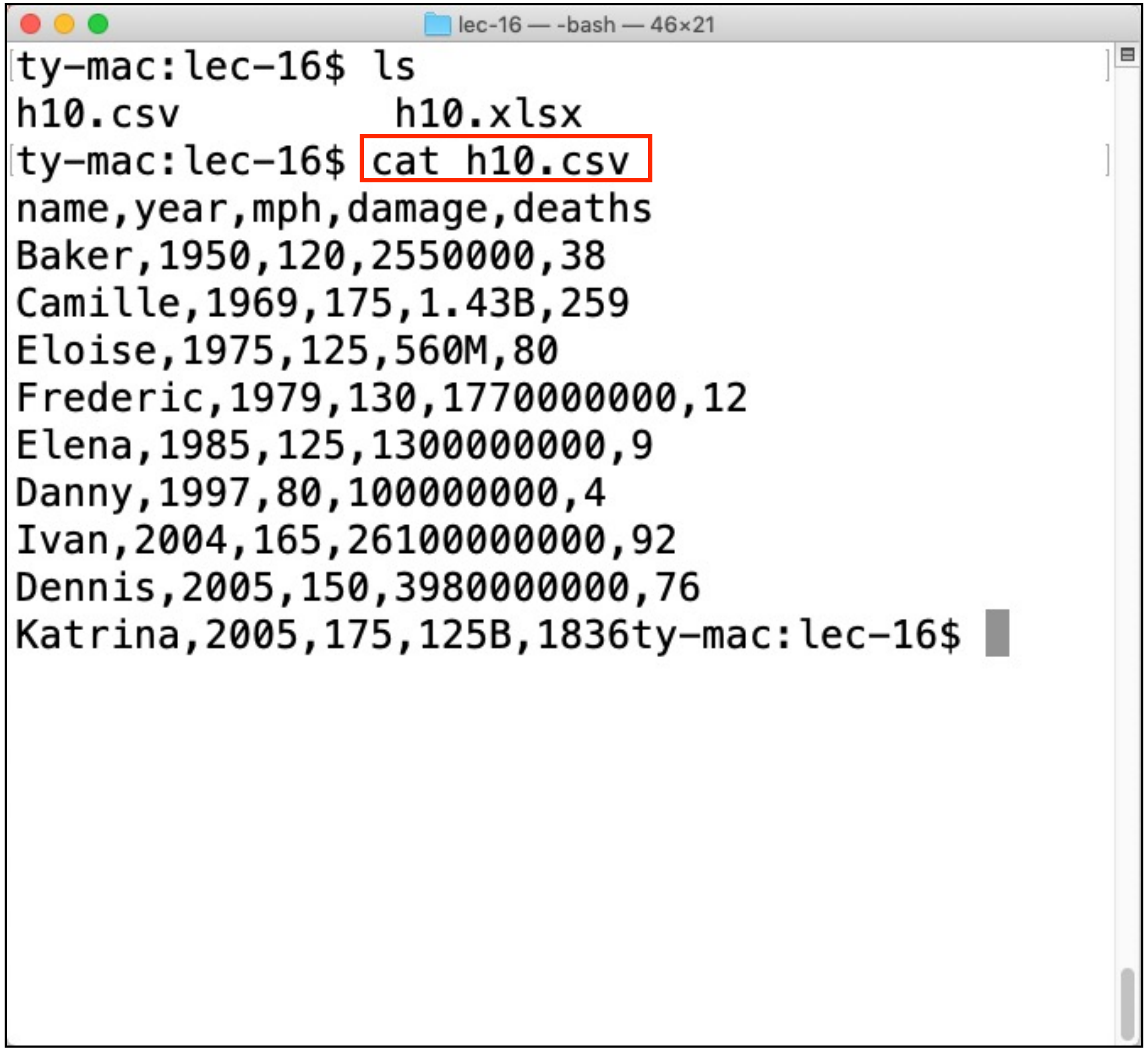
you'll do lots of type casting/conversion!

CSV Files

Extension: .csv

Format: plain text

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



```
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,2004,165,26100000000,92
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
```

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

Co

We call characters that act a separators “**delimiters**”

Na

Newlines delimit rows

HE

OL

TI

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

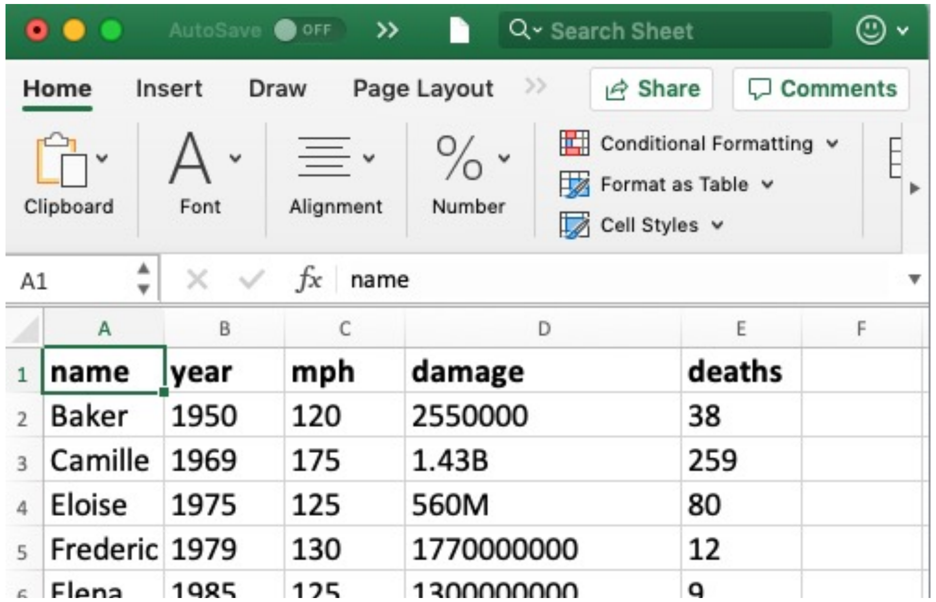
CSVs

Reading a CSV to a list of lists

Coding examples

Data Management

1. spreadsheet in Excel



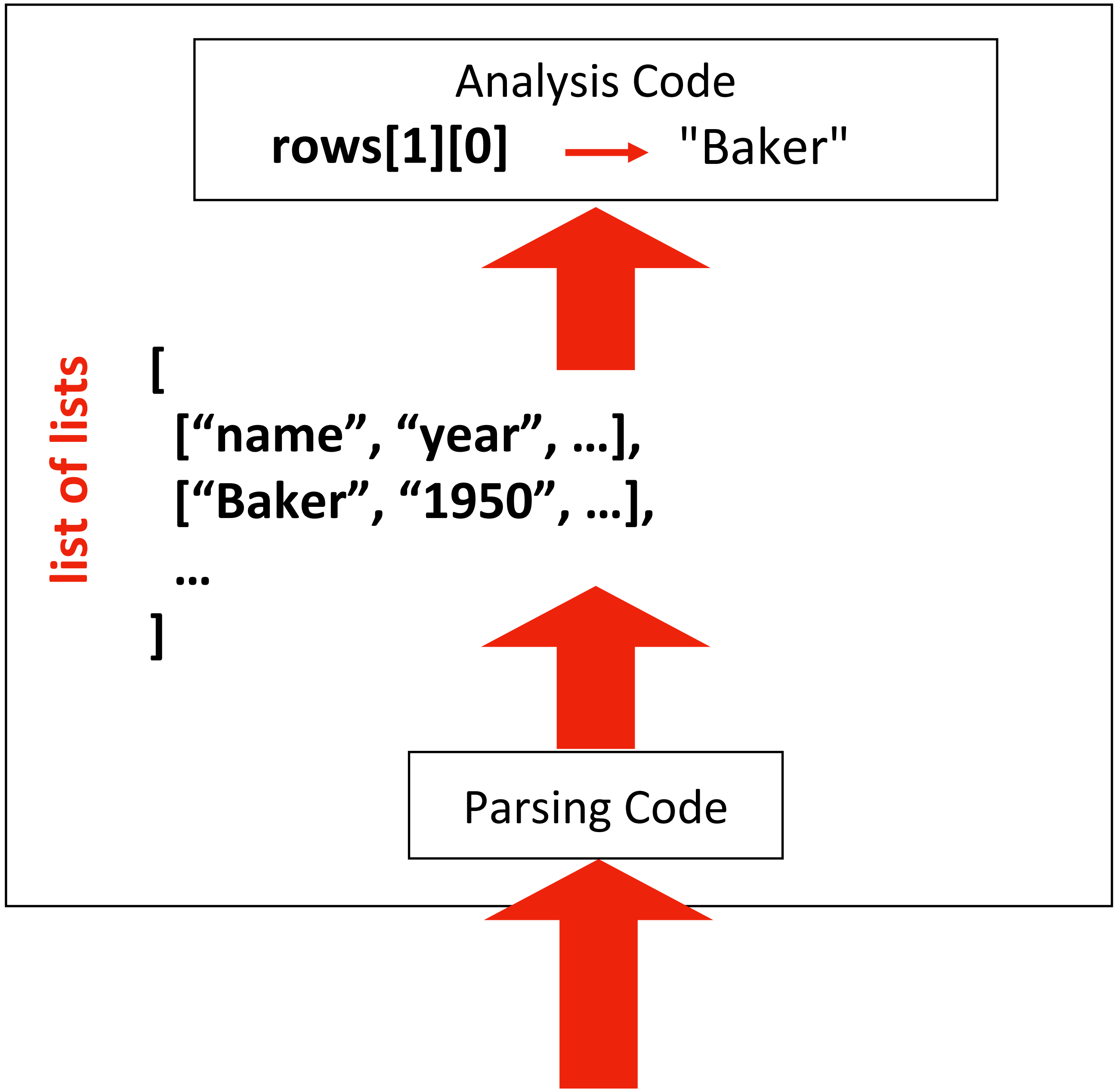
	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	

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"

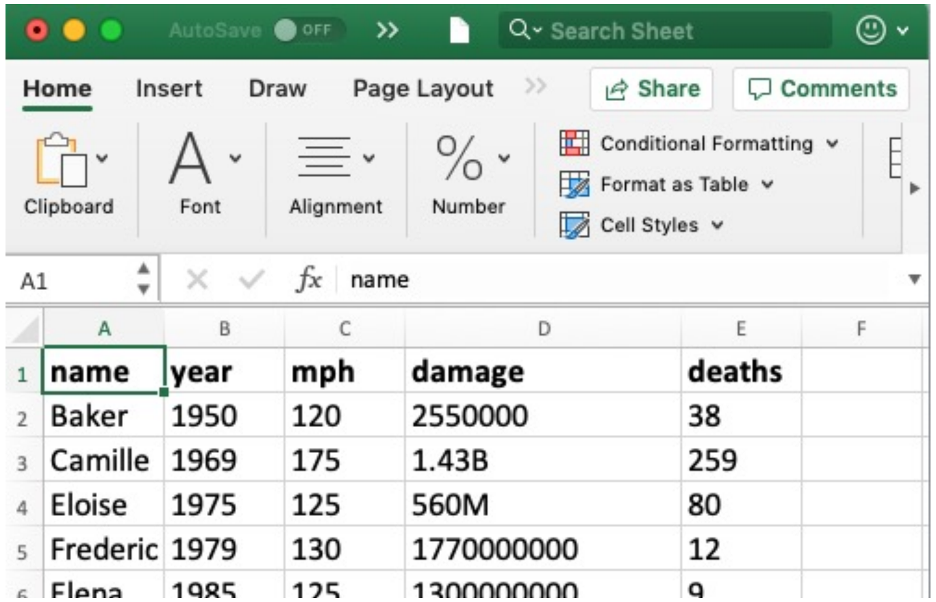
list of lists

[
[“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

Parsing Code

list of lists

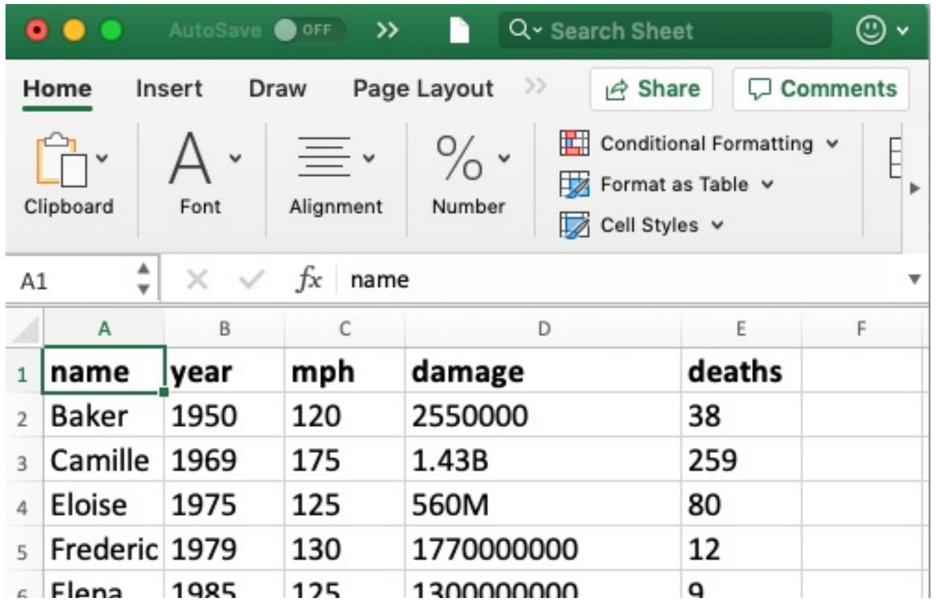
```
[
  ["name", "year", ...],
  ["Baker", "1950", ...],
  ...
]
```

Analysis Code

rows[3][1] → "1975"

Data Management

1. spreadsheet in Excel



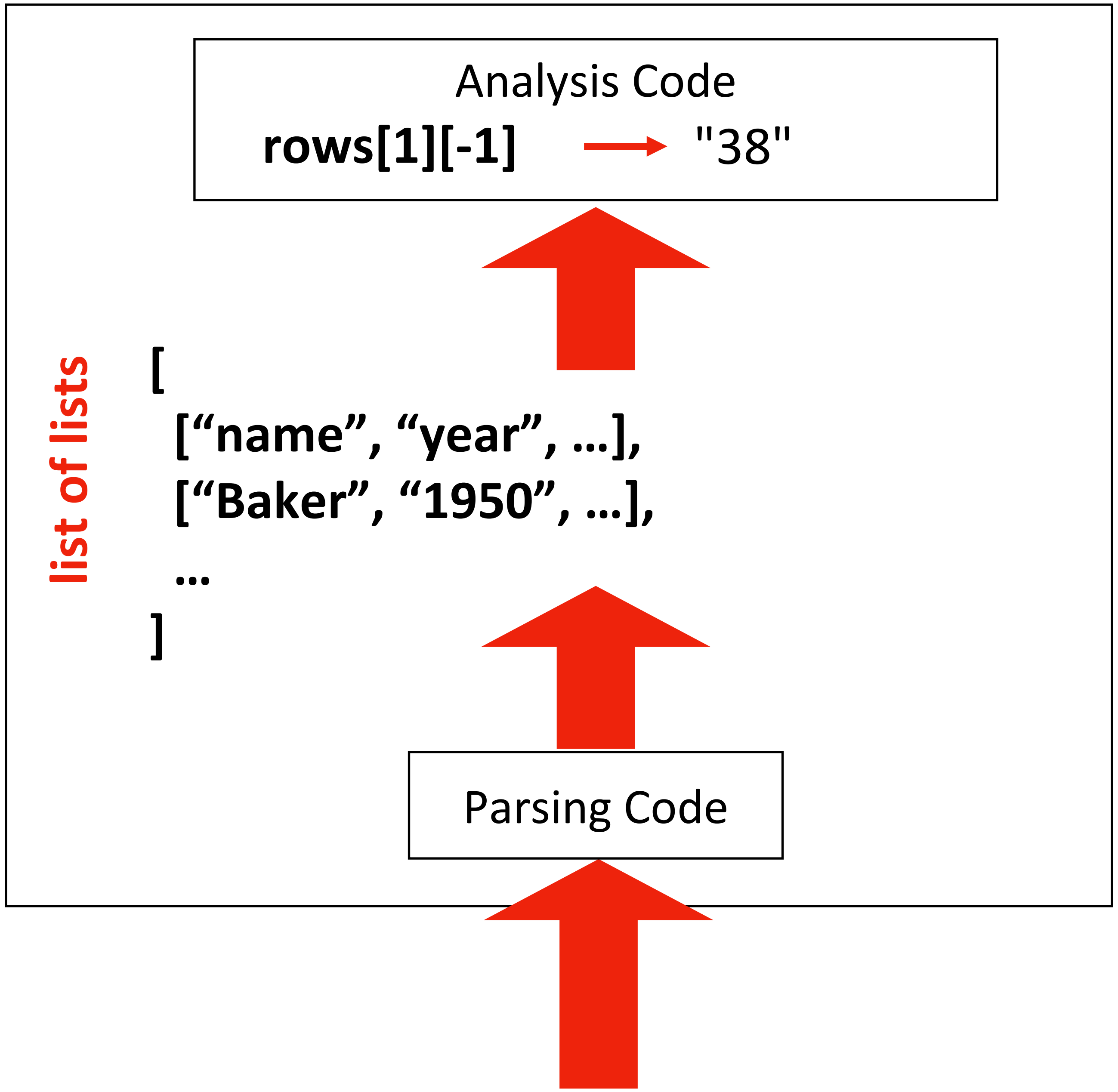
	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	

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] → "38"

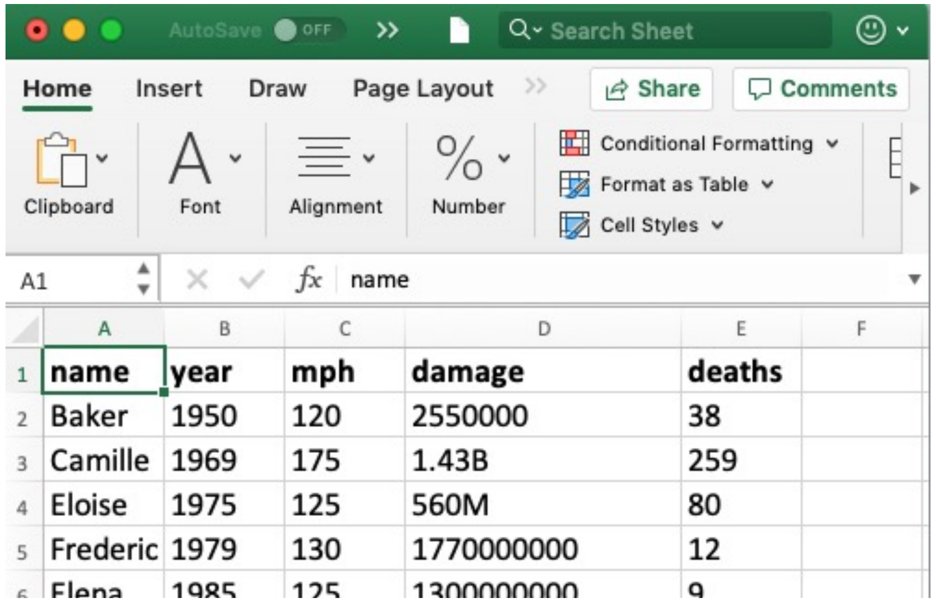
list of lists

```
[
  [“name”, “year”, …],
  [“Baker”, “1950”, …],
  …
]
```

Parsing Code

Data Management

1. spreadsheet in Excel



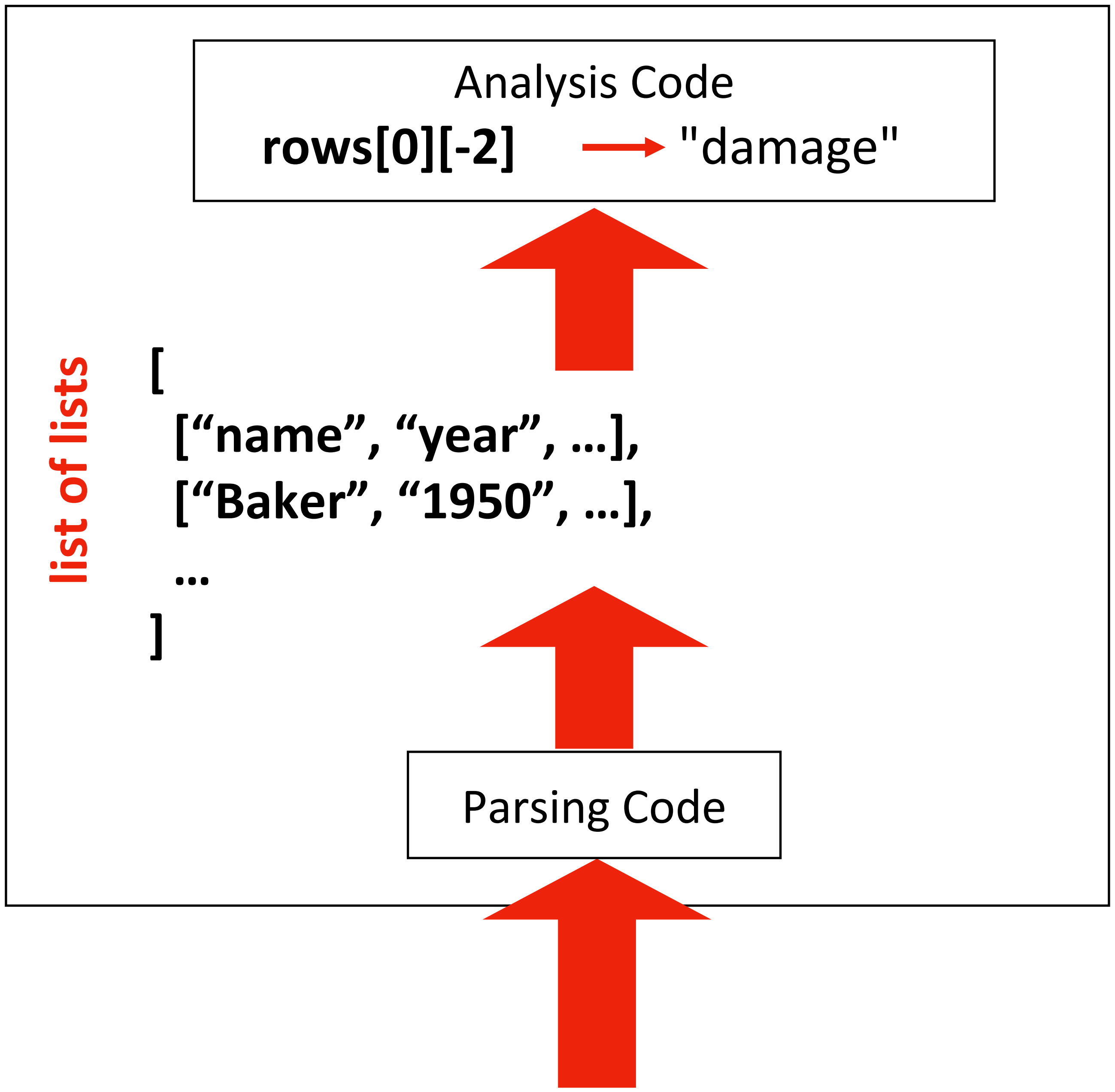
	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	

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"

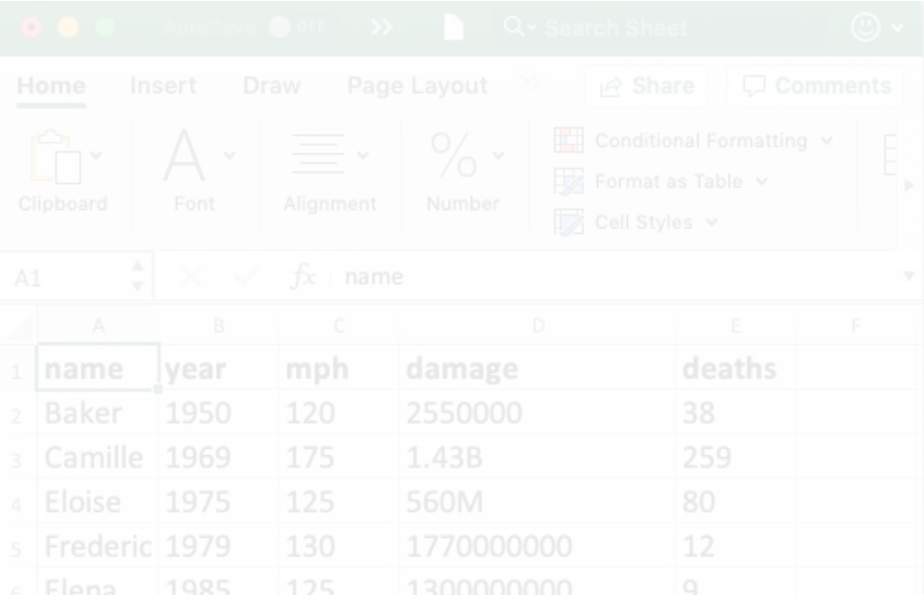
list of lists

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

Parsing Code

Data Management

1. spreadsheet in Excel



	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

What does this look like?

Example Copied From Sweigart Ch 16

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 16

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 16

```
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 16

```
import csv
```

input

```
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 16

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

1. move code to a function

Example Copied From Sweigart Ch 16

```
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 16

```
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 16

```
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 16

```
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 16

```
import csv

# copied from https://automatetheboringstuff.com/2e/chapter16/
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 16

```
import csv

# copied from https://automatetheboringstuff.com/2e/chapter16/
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

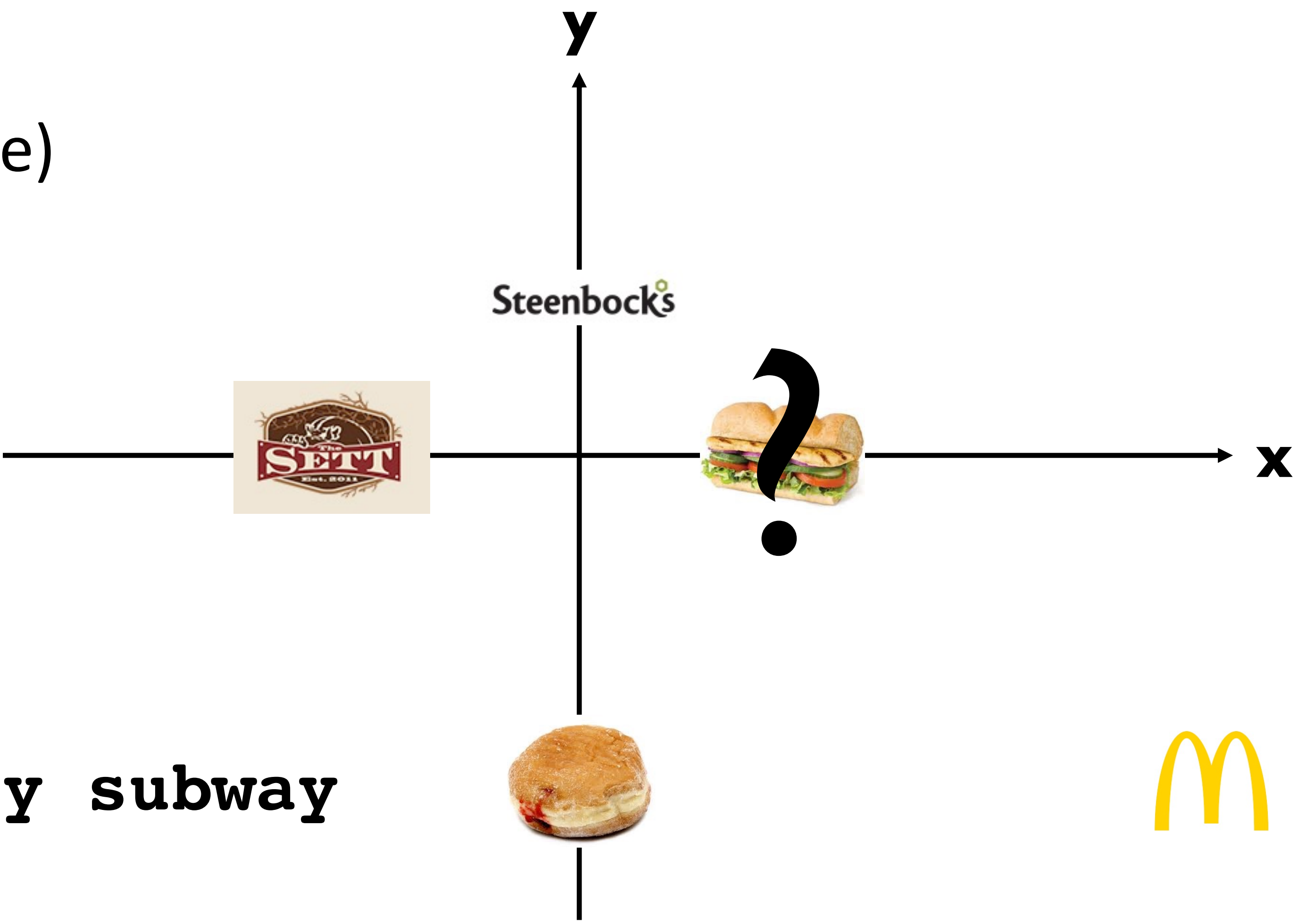
Coding examples

Example: 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
```

Example: Nearest Restaurant Search – Next lecture…

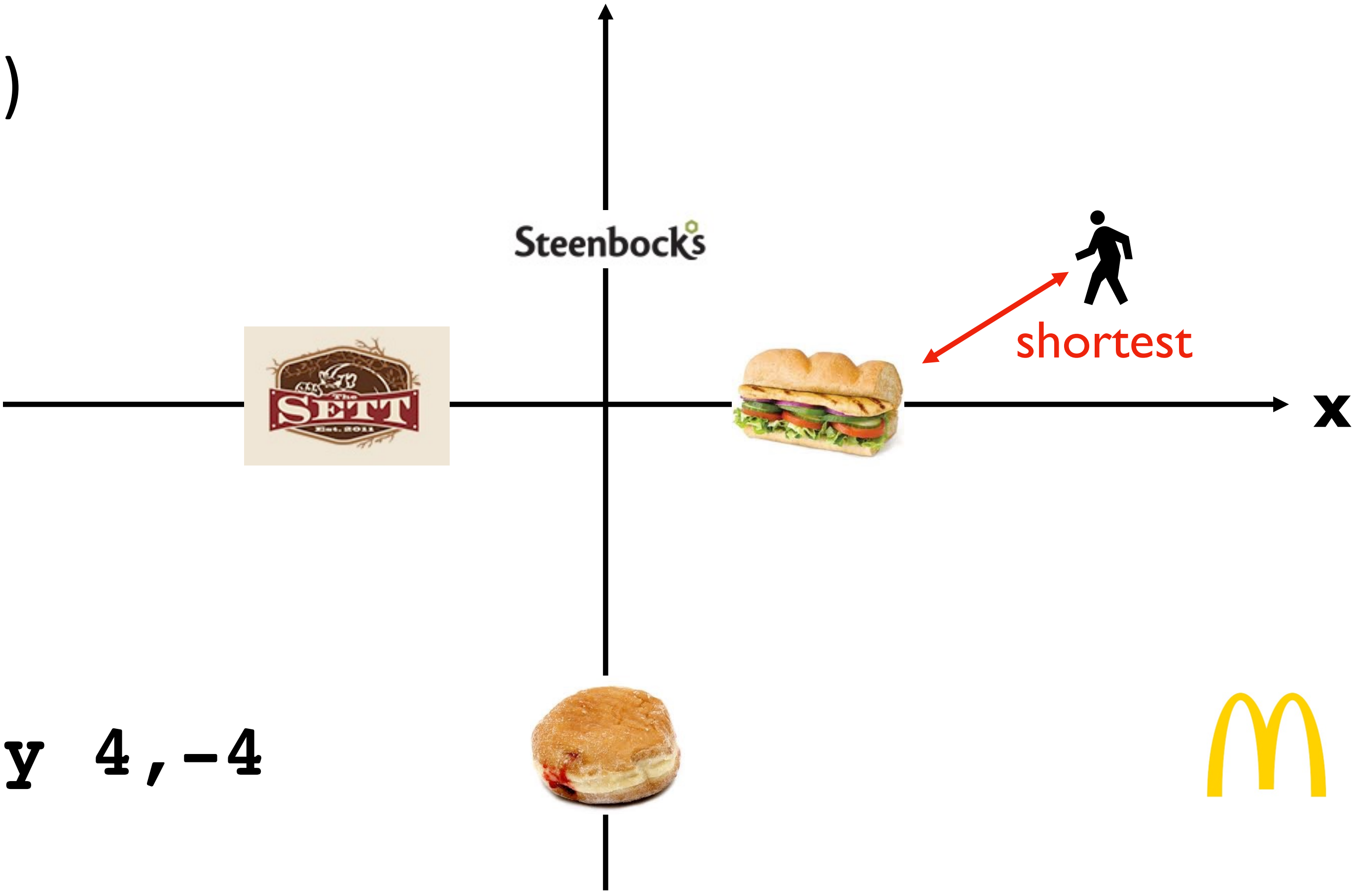
Goal: given a location, find the nearest restaurant

Input:
- X, Y coordinates (and a CSV file)

Output:
- nearest restaurant

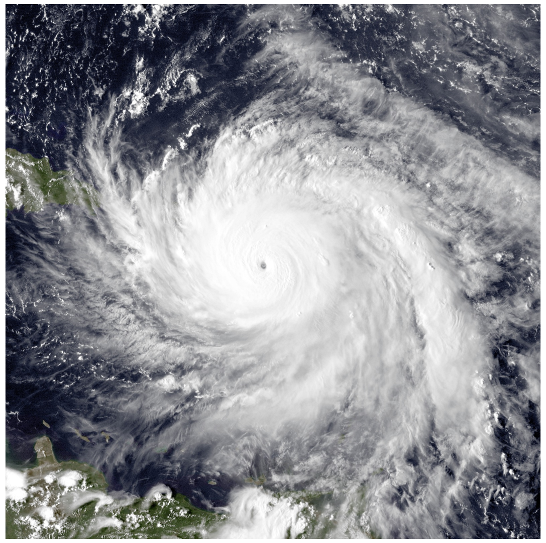
Example:

```
prompt> python nearest.py 4,-4
McDonalds
prompt> python nearest.py -2,0
The Sett
```



Challenge: Hurricane Column Dump

Goal: column name, print that data for all hurricanes



Input:

- column name (and a CSV file)

Output:

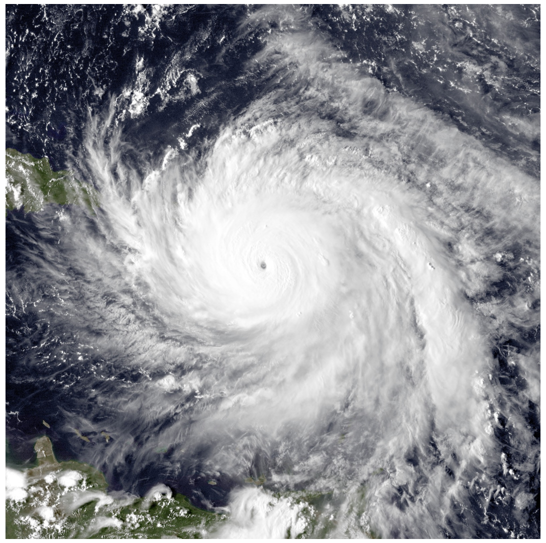
- data in given column, associated with name

Example:

```
prompt> python dump.py hurricanes.csv year
Baker: 1950
Camille: 1969
Eloise: 1975
…
```

Challenge: Hurricanes per Year

Goal: column name, print that data for all hurricanes



Input:

- none typed (only a CSV file)

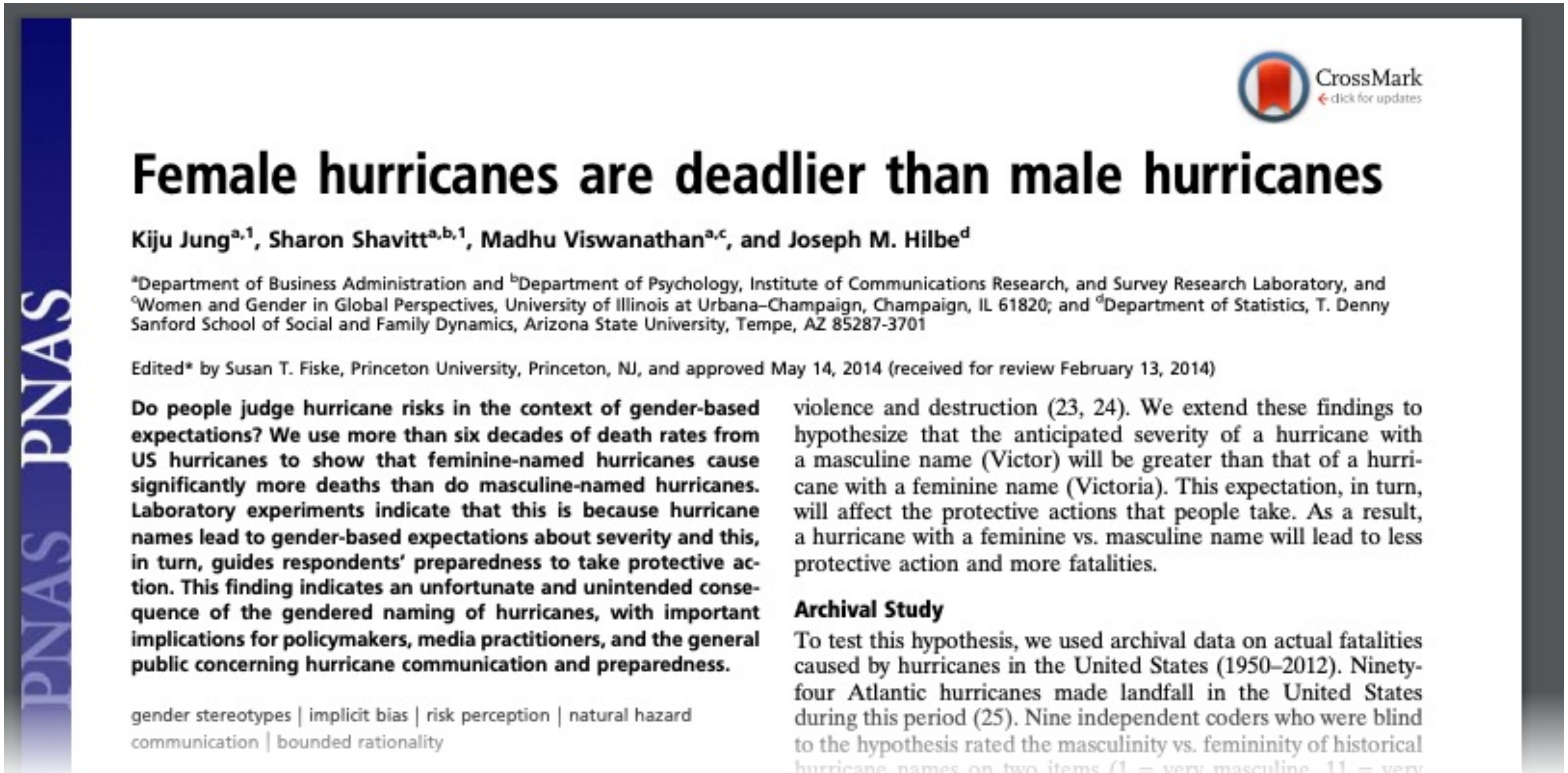
Output:

- the number of hurricanes in each year

Example:

```
prompt> python yearly.py
1967: 23
1968: 29
2969: 15
…
```

Challenge: Hurricane Names and Stereotypes



Female hurricanes are deadlier than male hurricanes

Kiju Jung[a,1], Sharon Shavitt[a,b,1], Madhu Viswanathan[a,c], and Joseph M. Hilbe[d]

[a]Department of Business Administration and [b]Department of Psychology, Institute of Communications Research, and Survey Research Laboratory, and [c]Women and Gender in Global Perspectives, University of Illinois at Urbana–Champaign, Champaign, IL 61820; and [d]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