[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”

https://automatetheboringstuff.com/chapter14/

Today's Outline

Spreadsheets

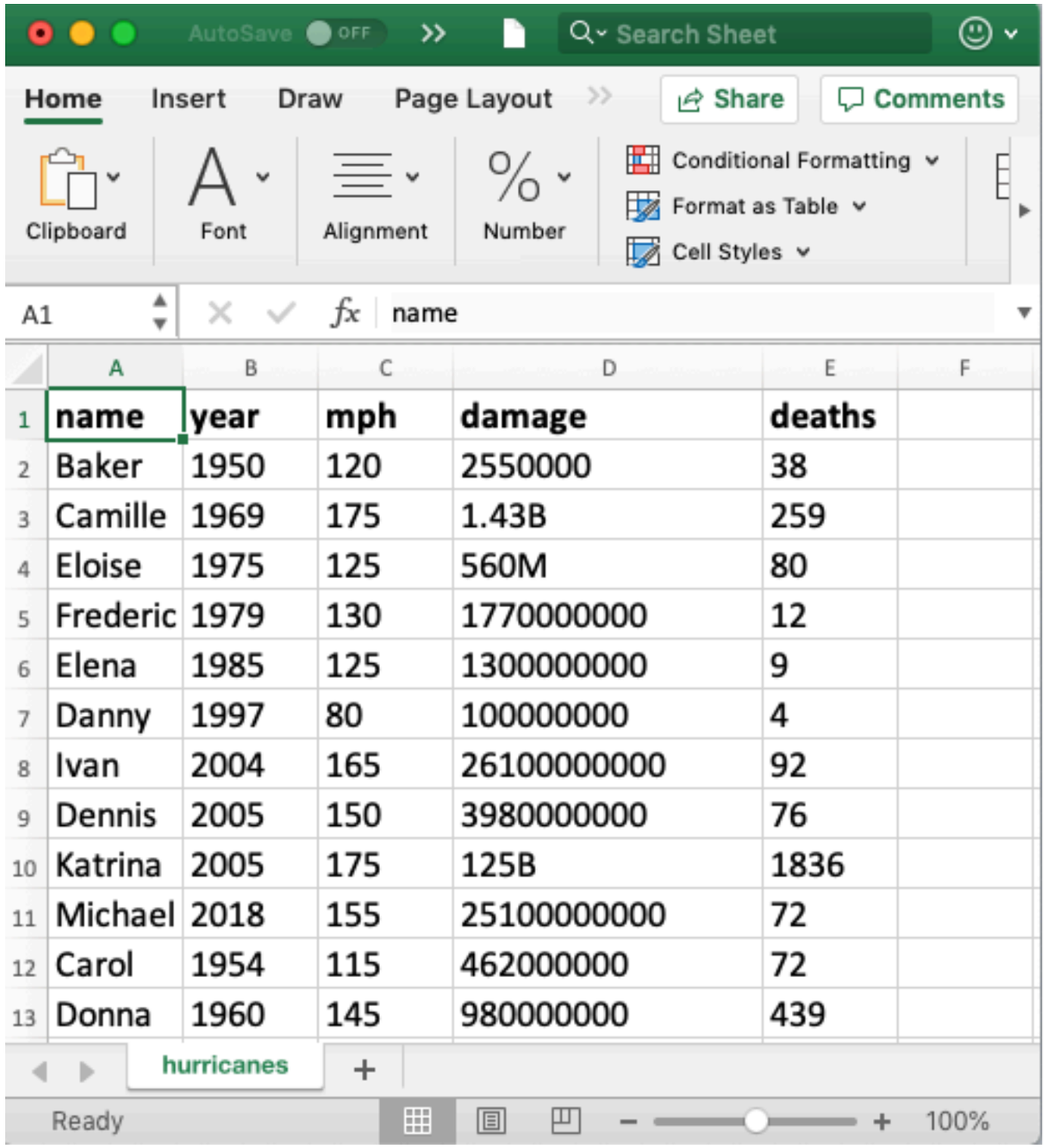
CSVs

Reading a CSV to a list of lists

Coding examples

Spreadsheets (e.g., Excel)

Spreadsheets are tables of cells, organized by rows and columns

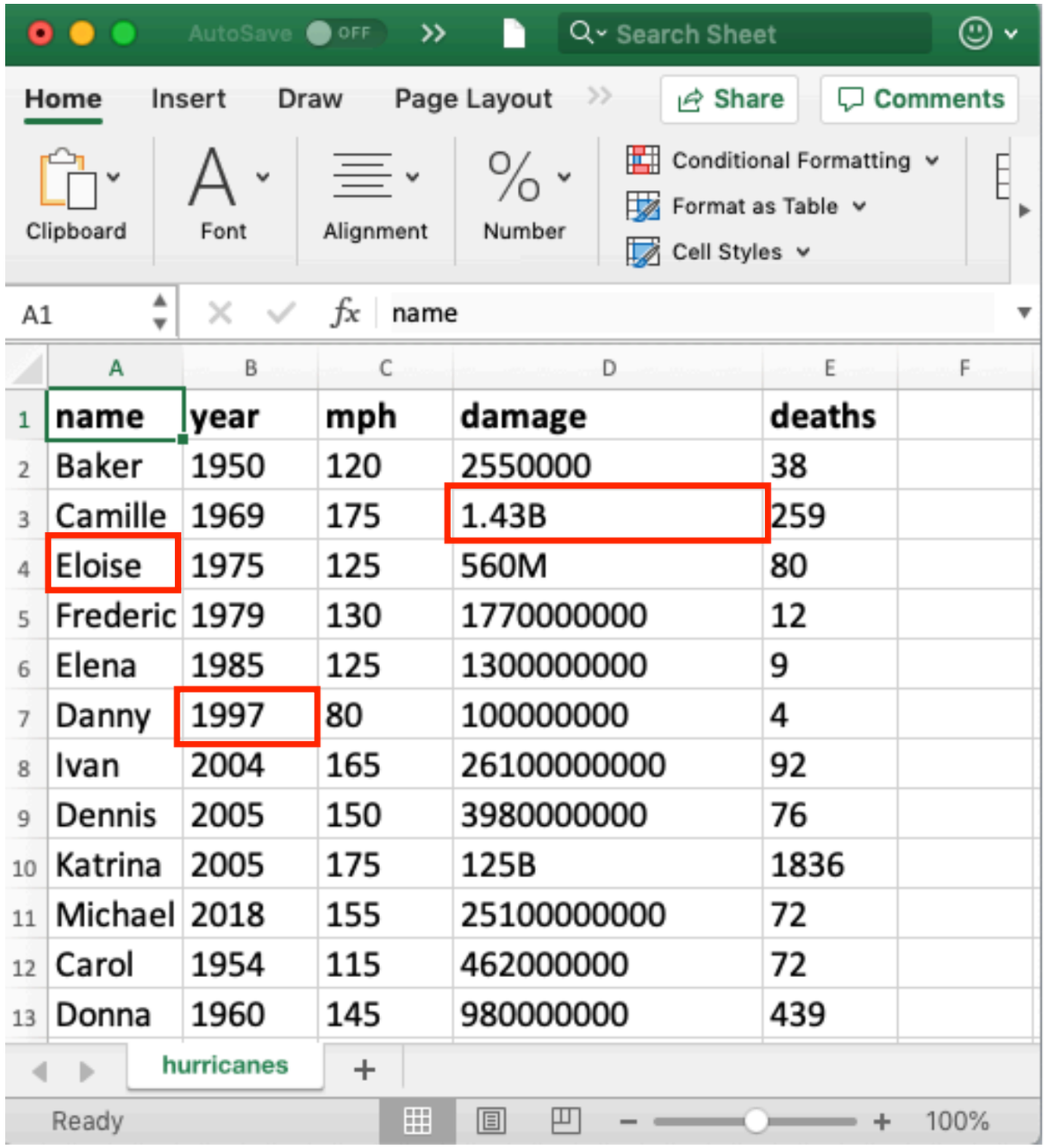


	A	B	C	D	E	F
1	**name**	**year**	**mph**	**damage**	**deaths**	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	
7	Danny	1997	80	100000000	4	
8	Ivan	2004	165	26100000000	92	
9	Dennis	2005	150	3980000000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

Spreadsheets (e.g., Excel)

Spreadsheets are tables of cells, organized by rows and columns

cells

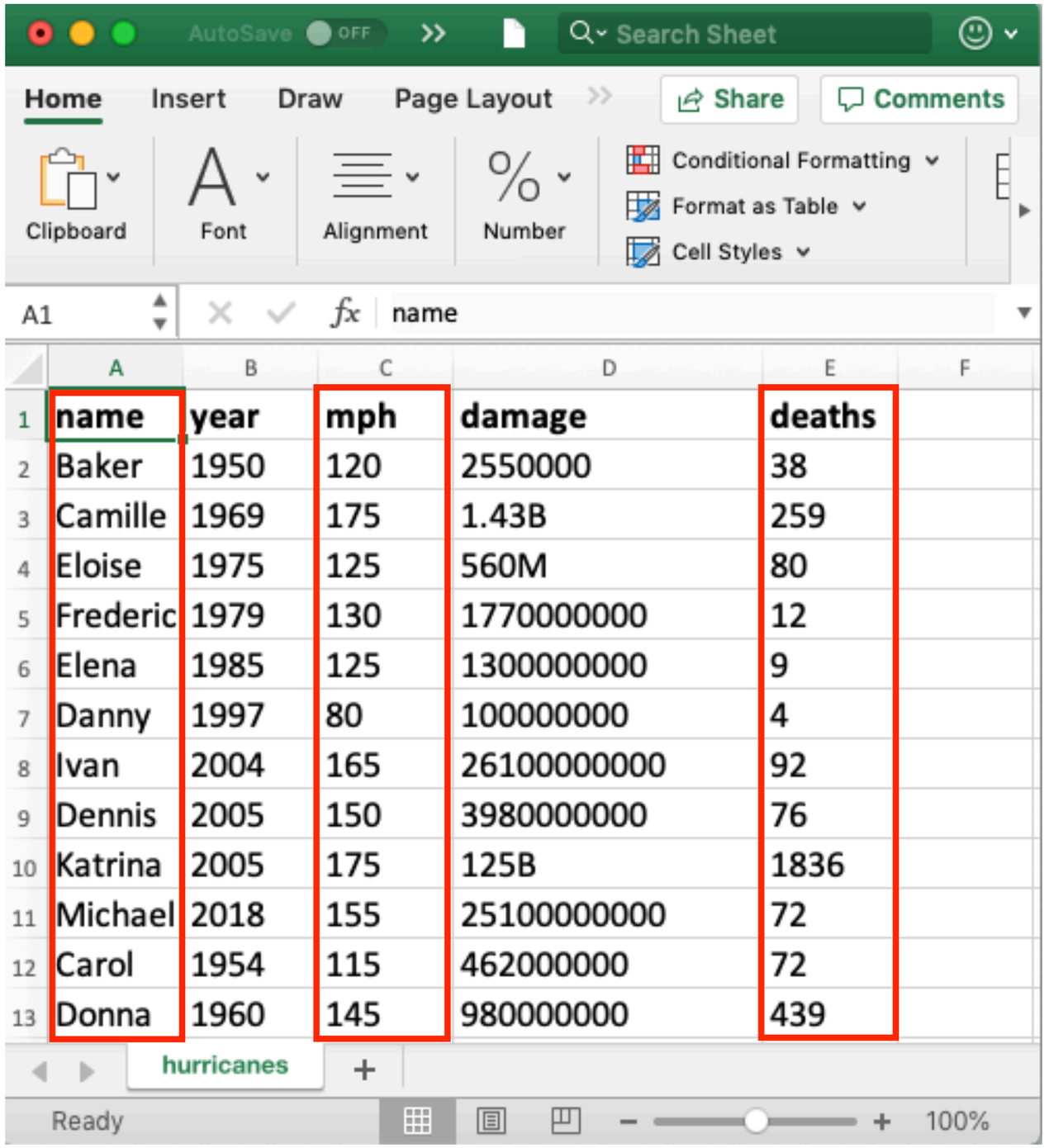


	A	B	C	D	E	F
1	**name**	**year**	**mph**	**damage**	**deaths**	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	
7	Danny	1997	80	100000000	4	
8	Ivan	2004	165	26100000000	92	
9	Dennis	2005	150	3980000000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



	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

AutoSave OFF | Search Sheet

Home | Insert | Draw | Page Layout | Share | Comments

Clipboard | Font | Alignment | Number | Conditional Formatting | Format as Table | Cell Styles

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	

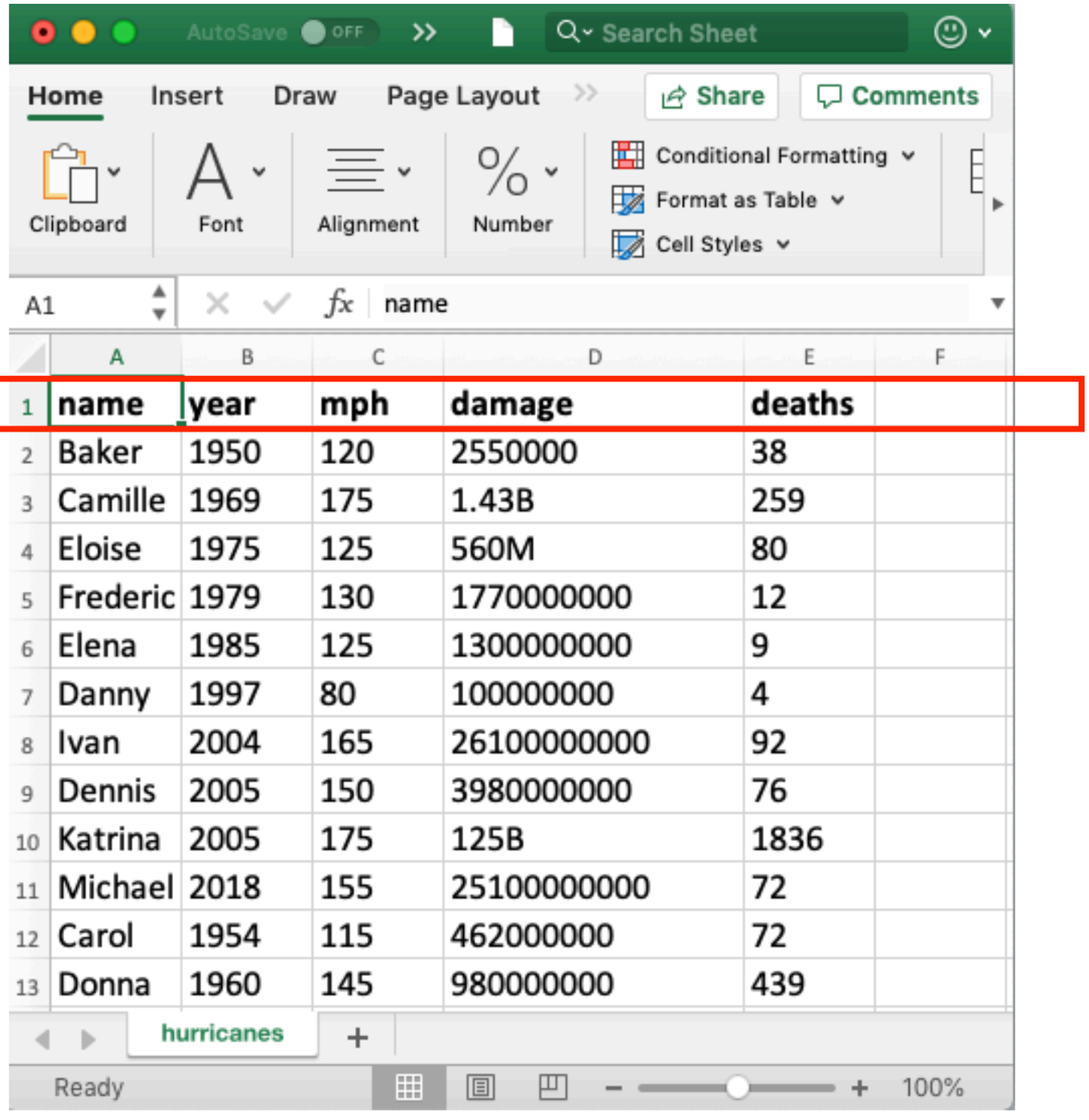
hurricanes | +

Ready | 100%

Spreadsheets (e.g., Excel)

Spreadsheets are tables of cells, organized by rows and columns

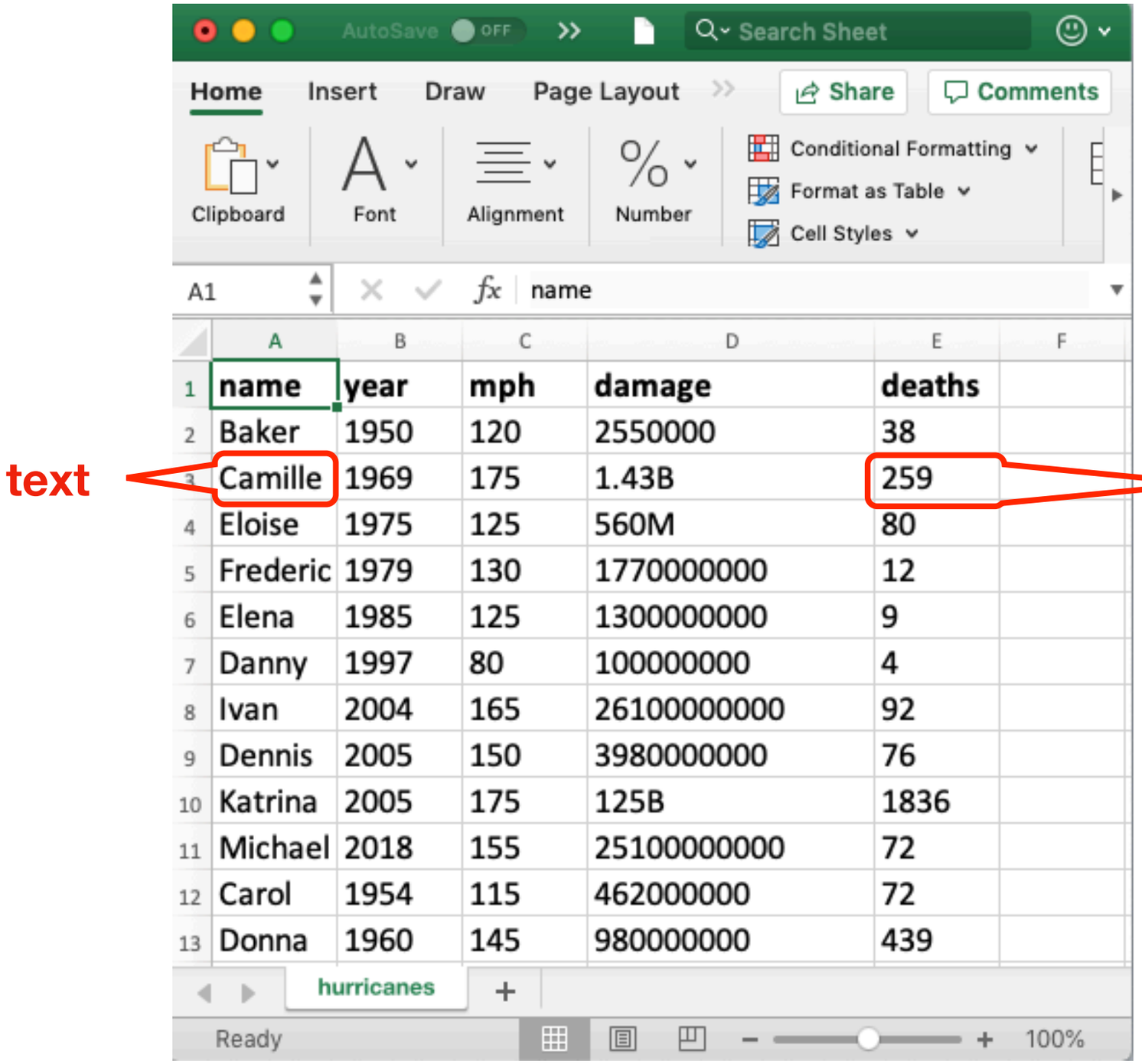
header



	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



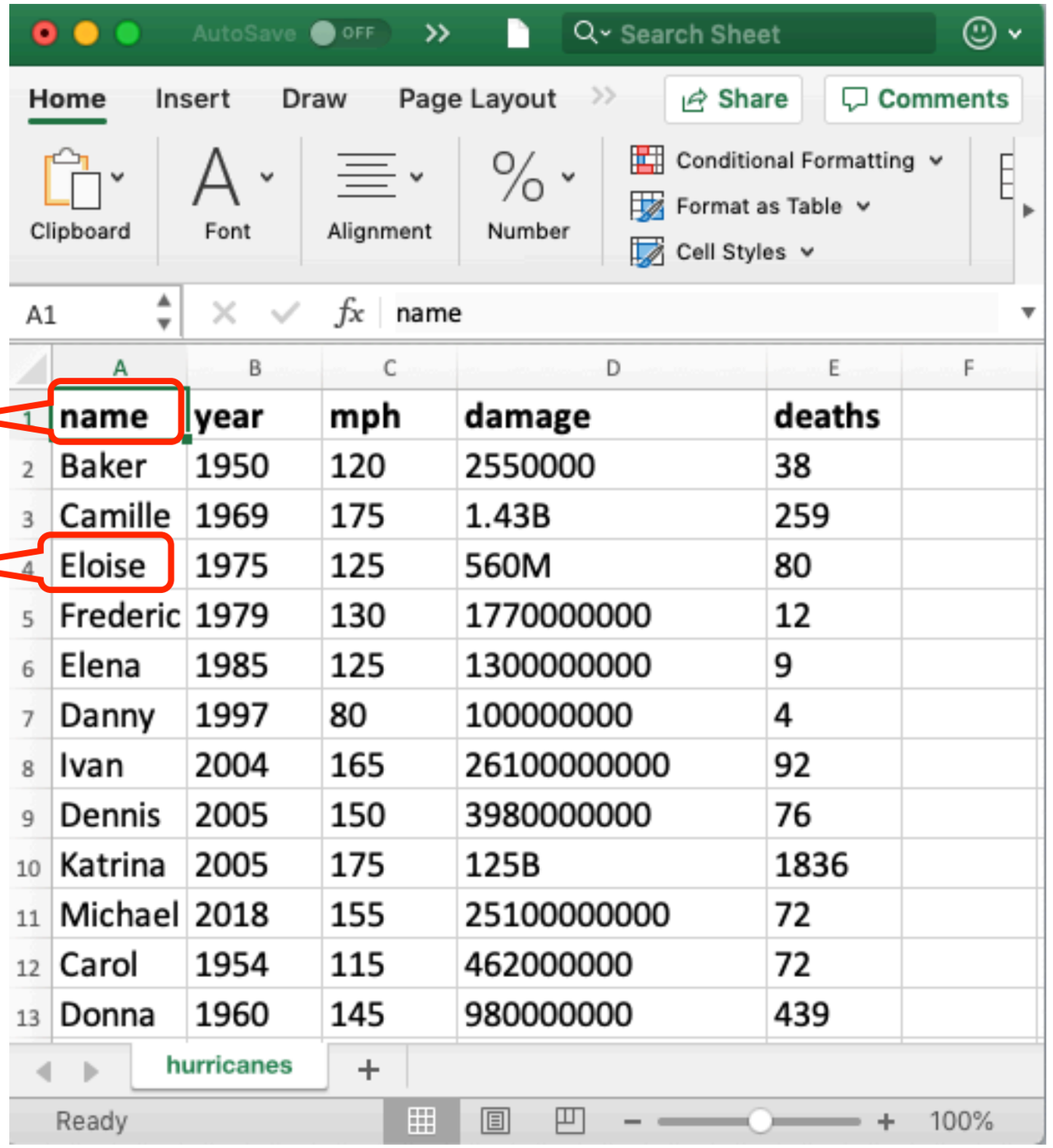
	A	B	C	D	E	F
1	name	year	mph	damage	deaths	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	
7	Danny	1997	80	100000000	4	
8	Ivan	2004	165	26100000000	92	
9	Dennis	2005	150	3980000000	76	
10	Katrina	2005	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 fonts

bold

regular



	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

AutoSave OFF | Search Sheet

Home | Insert | Draw | Page Layout | Share | Comments

Clipboard | Font | Alignment | Number | Conditional Formatting | Format as Table | Cell Styles

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	

hurricanes | +

Ready | 100%

more tables of data

Excel Files

Extension: .xlsx

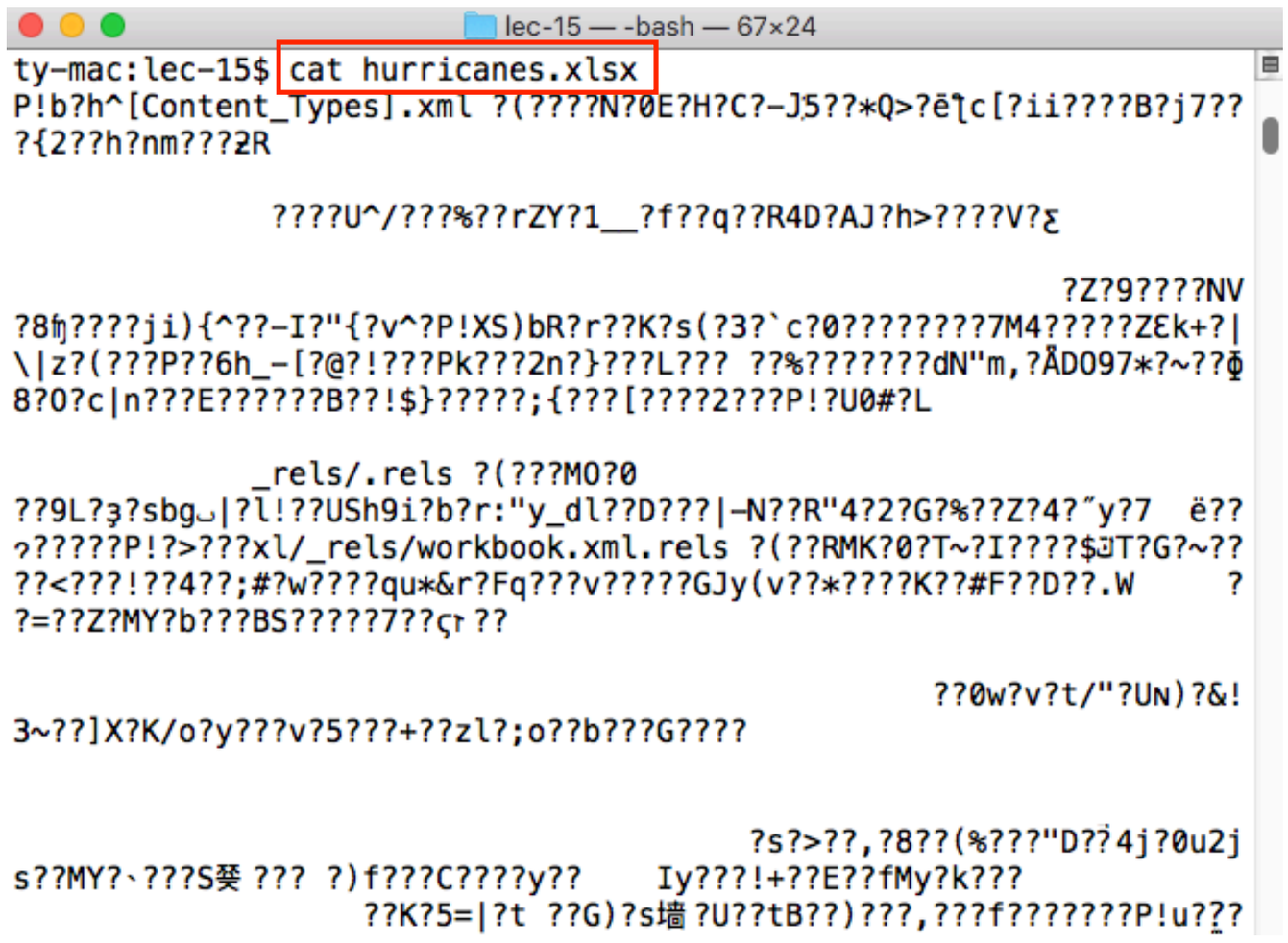
Format: binary

Excel Files

Extension: .xlsx

Format: binary

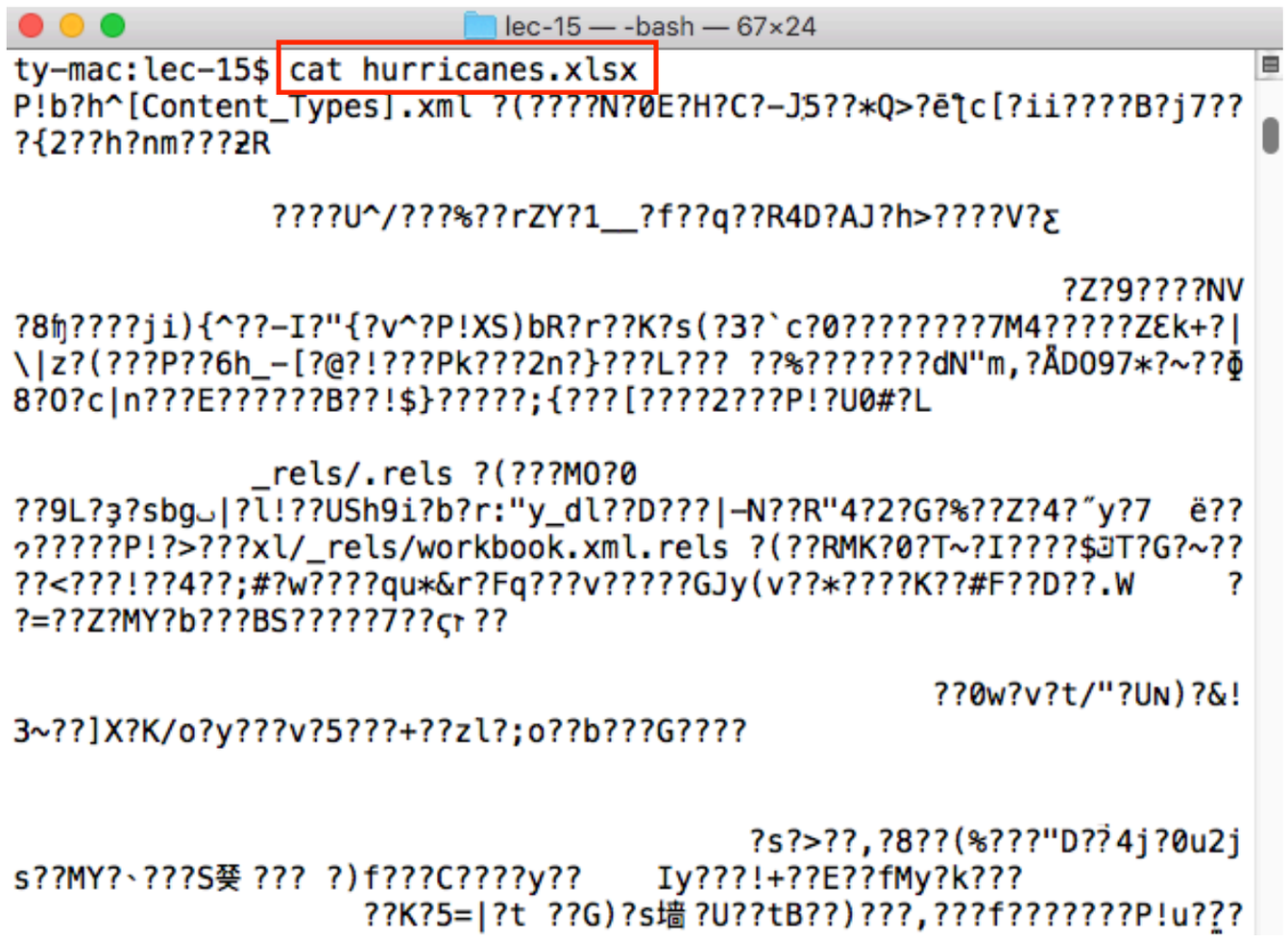
just 0's and 1's, not human-readable characters.
Need special software…



Excel Files

Extension: .xlsx

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



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 **C**omma-**S**eparated **V**alues

CSVs are like simple spreadsheets

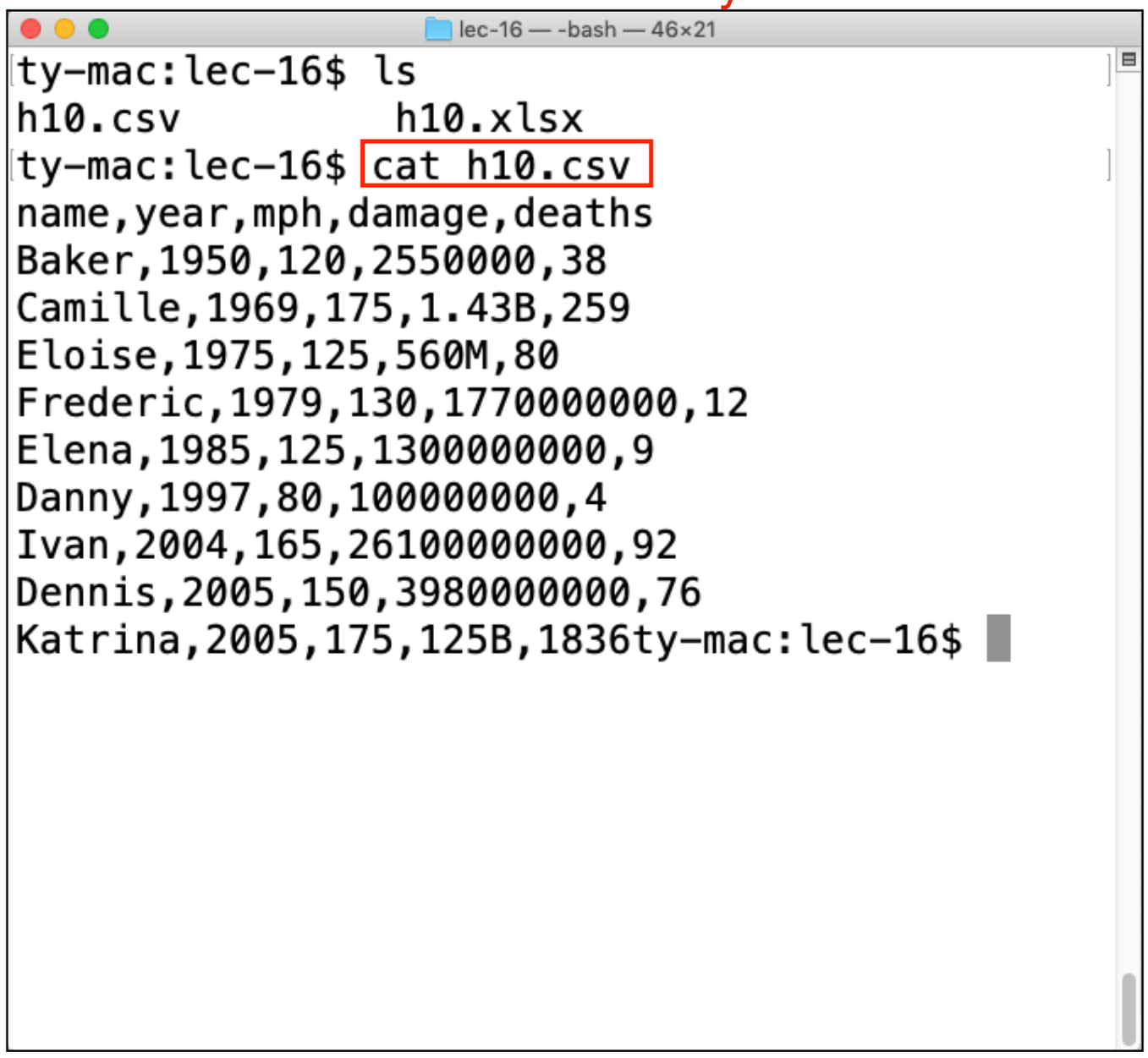
- organize cells of data into rows and columns
- only one sheet per file
- only holds strings
- no way to specify font, borders, cell size, etc

CSV Files

Extension: .csv

Format: plain text

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



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 a 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

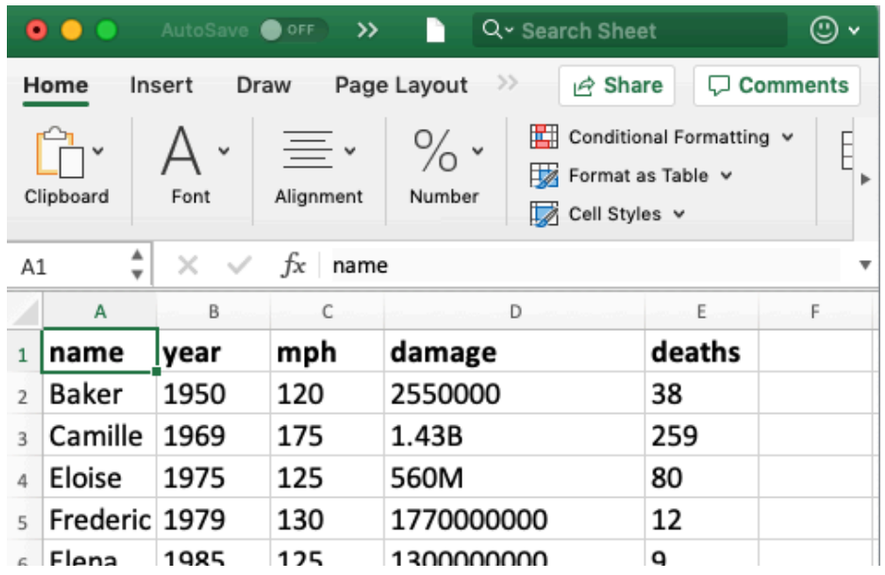
CSVs

Reading a CSV to a list of lists

Coding examples

Data Management

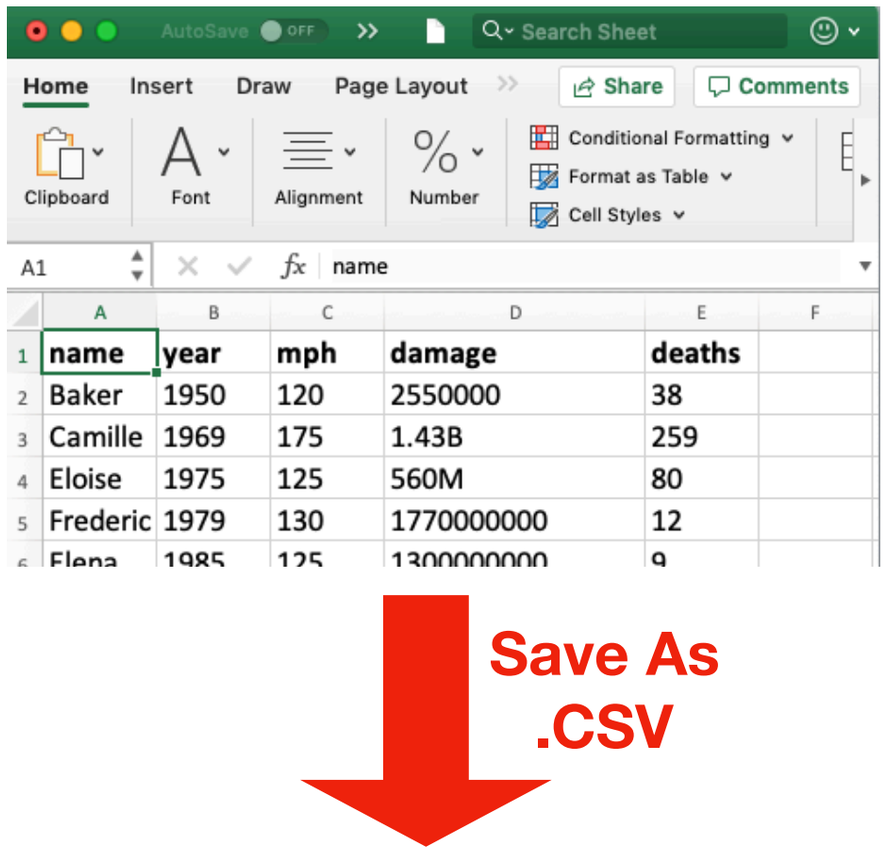
1. spreadsheet in Excel



	A	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	

Data Management

1. spreadsheet in Excel



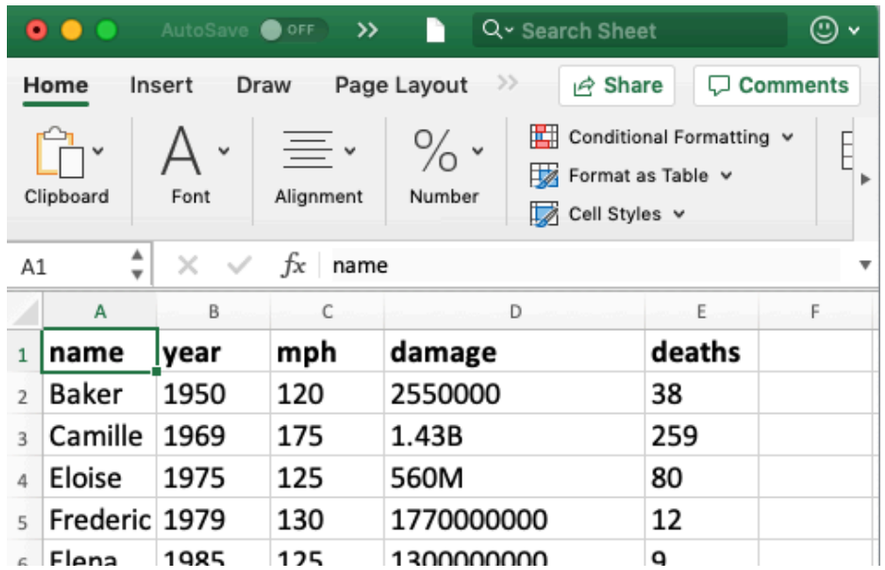
	A	B	C	D	E	F
1	**name**	**year**	**mph**	**damage**	**deaths**	
2	Baker	1950	120	2550000	38	
3	Camille	1969	175	1.43B	259	
4	Eloise	1975	125	560M	80	
5	Frederic	1979	130	1770000000	12	
6	Elena	1985	125	1300000000	9	

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

1. spreadsheet in Excel



	A	B	C	D	E
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

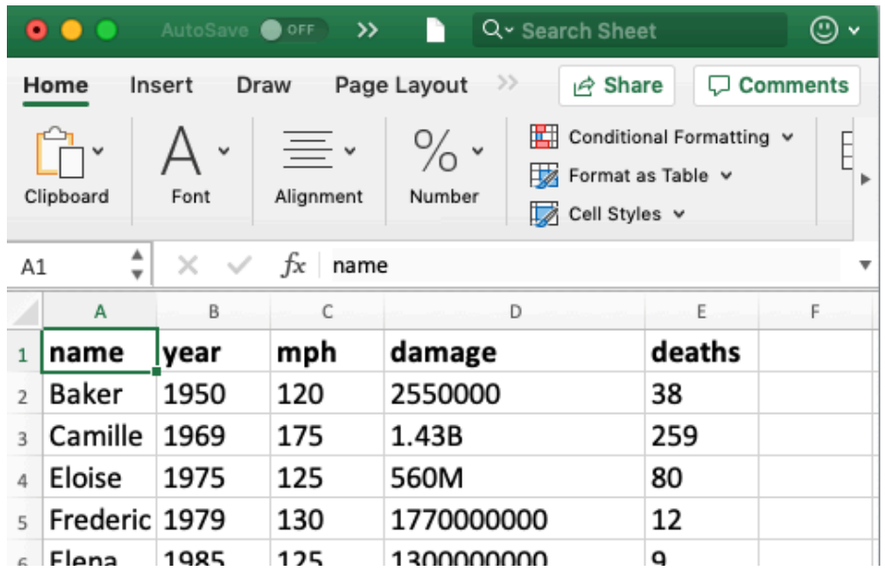
2. CSV file saved somewhere

```
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

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

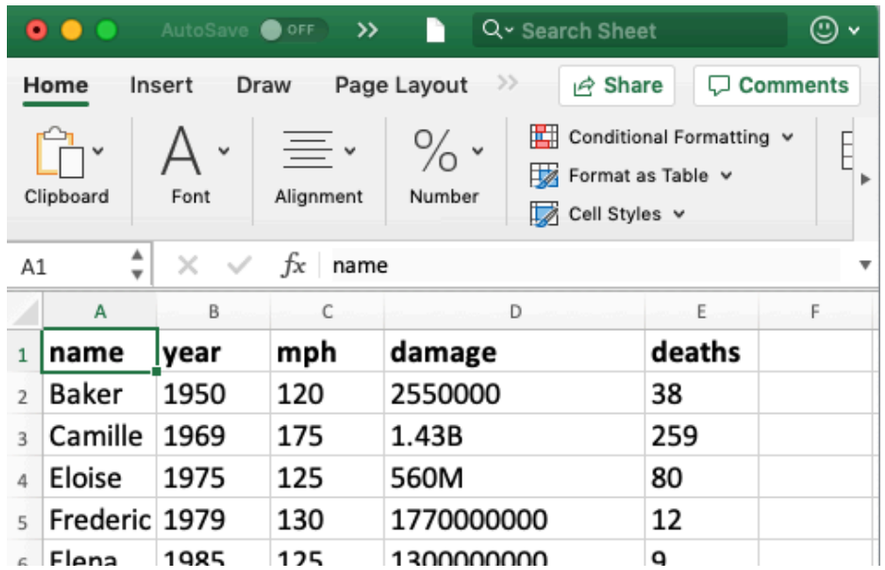
list of lists

```
[
   ["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][0]  →  ????
```

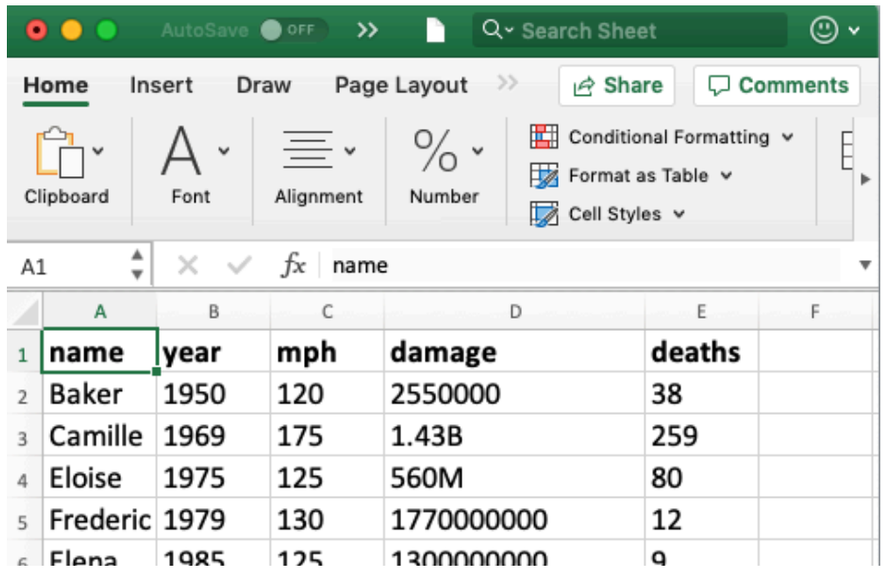
list of lists

```
[
   ["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][0]
```
→ "Baker"

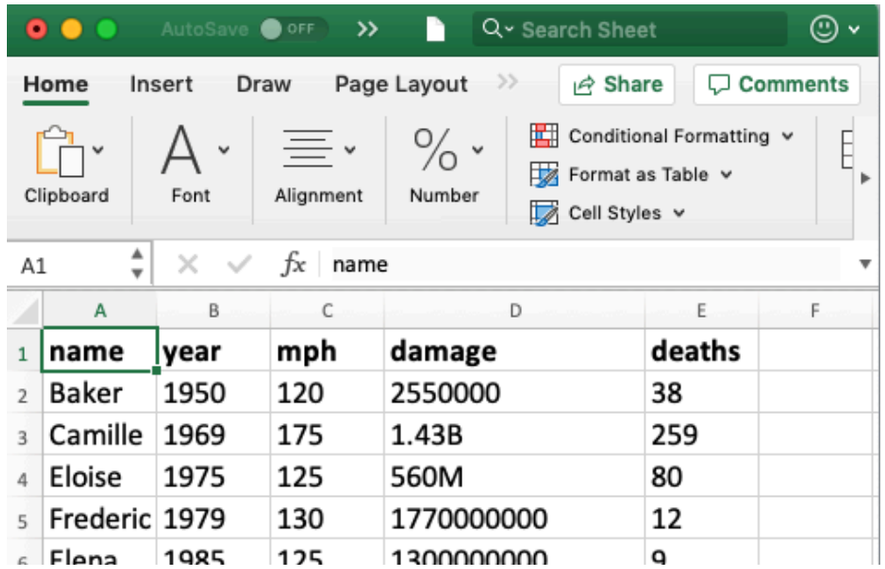
list of lists

```
[
  ["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]  →  ????
```

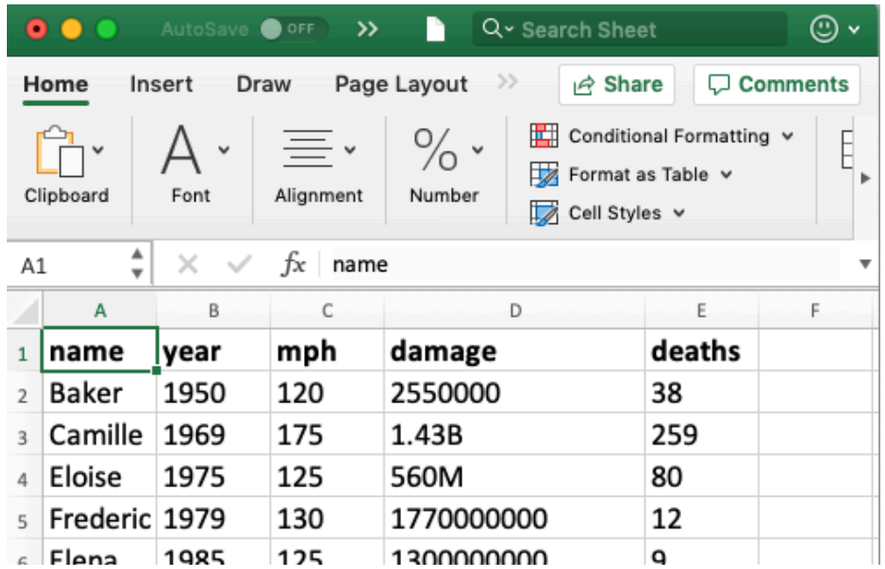
list of lists

```
[
   ["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]  →  "1975"
```

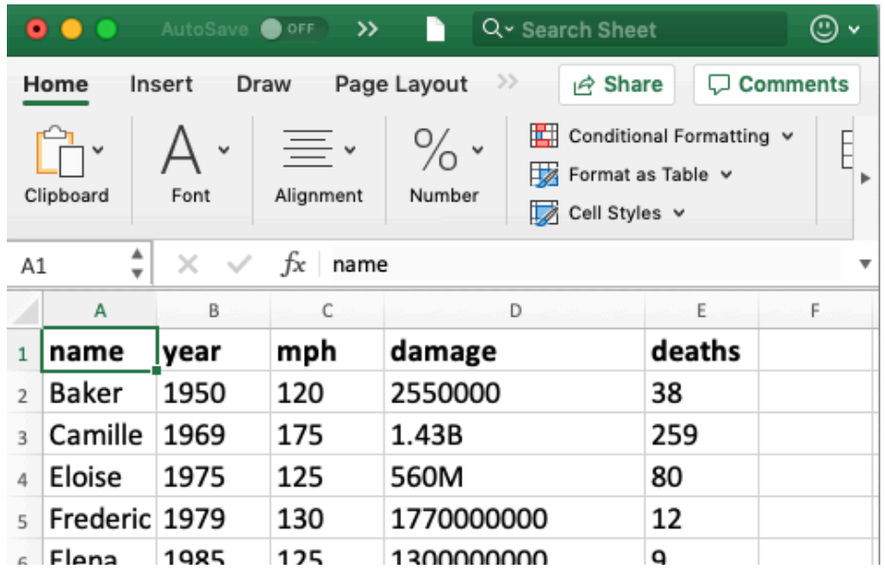
list of lists

```
[
   ["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]` → ????

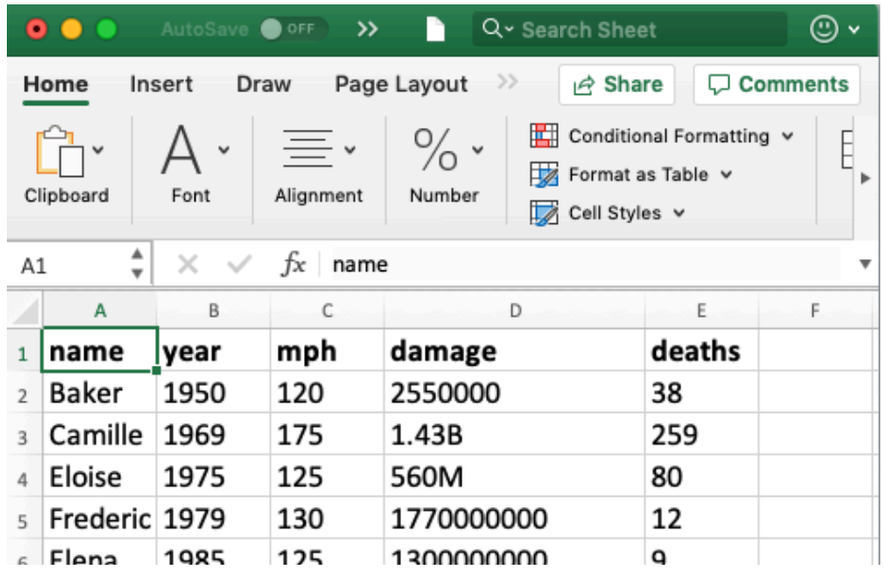
list of lists

```
[
  ["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	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"
```

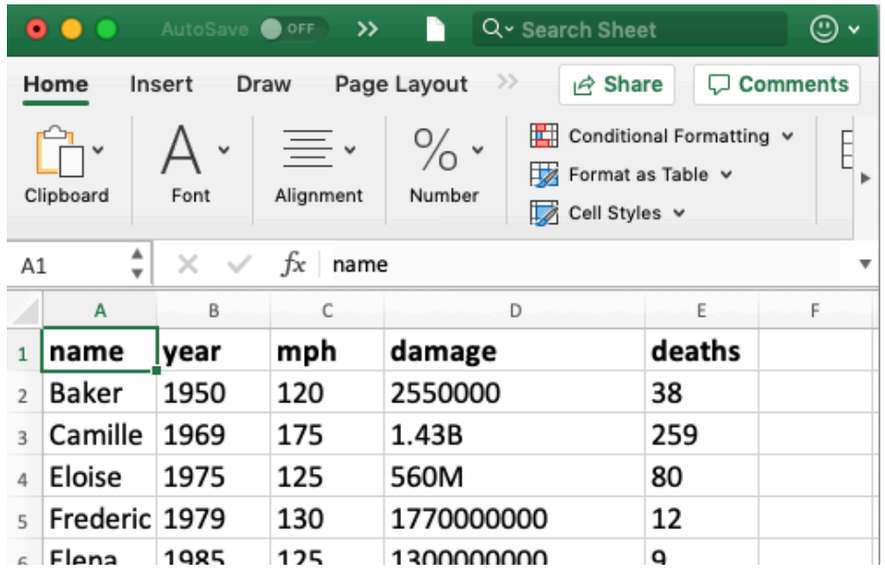
list of lists

```
[
  ["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]` → ????

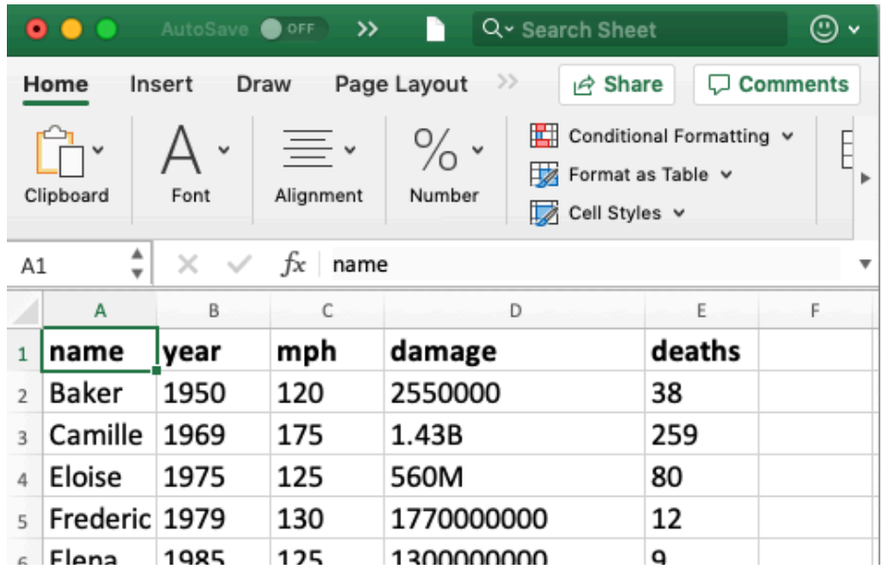
list of lists

```
[
   ["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
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
FOUR	20061007	1800	DB	11.3N	159.0W	25	Pacific
FLOSSIE	19780904	0	TD	12.0N	39.0W	25	Atlantic
GLORIA	19760926	1200	TD	23.0N	58.0W	20	Atlantic
MARIA	20050901	1200	TD	18.8N	45.5W	30	Atlantic
GILBERT	19880908	1800	TD	12.0N	54.0W	25	Atlantic
FELICE	19700912	0	TD	25.5N	77.5W	25	Atlantic

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

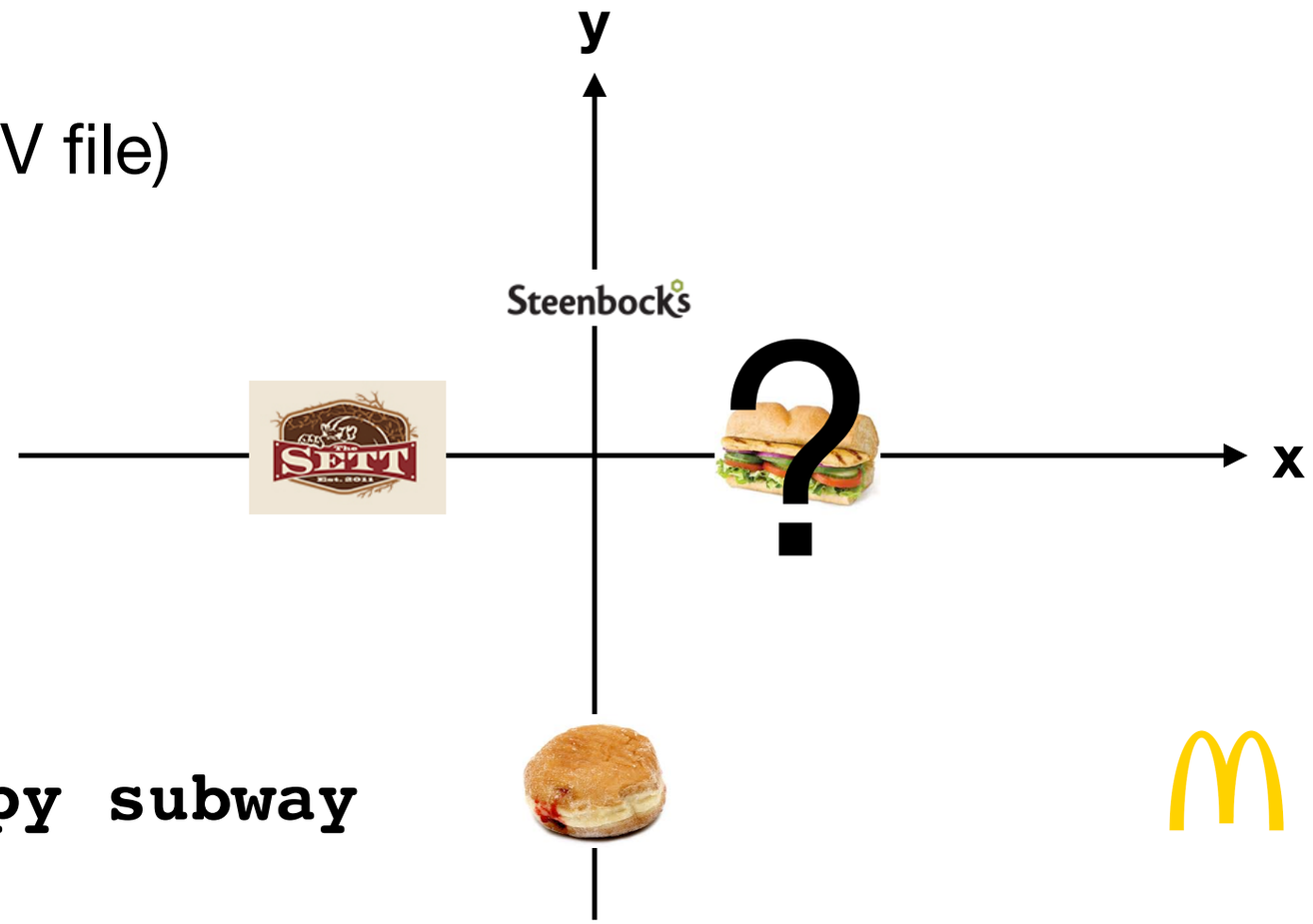
Goal: given a restaurant name, give x,y coordinates for it

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

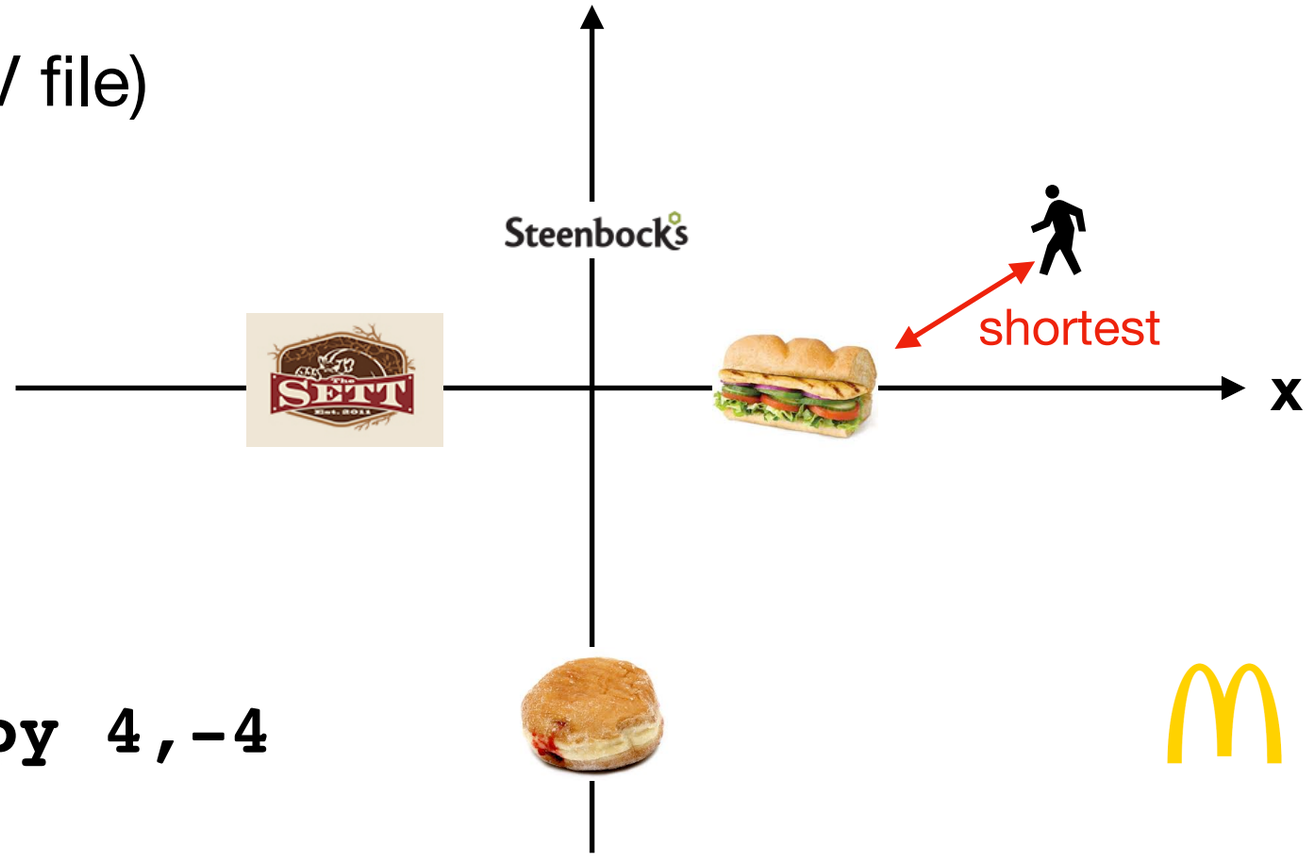
Goal: given a location, find the nearest restaurant

Input:

- X, Y coordinates (and a CSV file)

Output:

- nearest restaurant



Example:

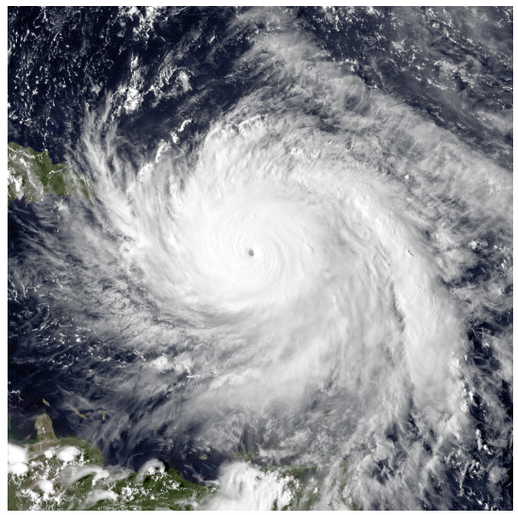
```
prompt> python nearest.py 4,-4
McDonalds
prompt> python nearest.py -2,0
The Sett
```

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:

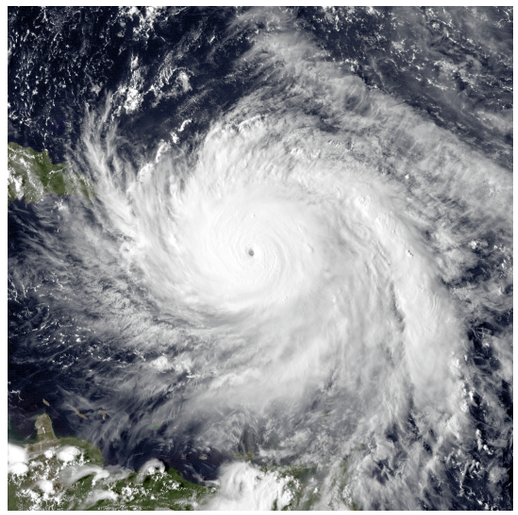
```
prompt> python dump.py hurricanes.csv year
Baker: 1950
Camille: 1969
Eloise: 1975
...
```

Demo 4: Hurricanes per Year

Goal: column name, print that data for all hurricanes

Input:

- none typed (only a CSV file)



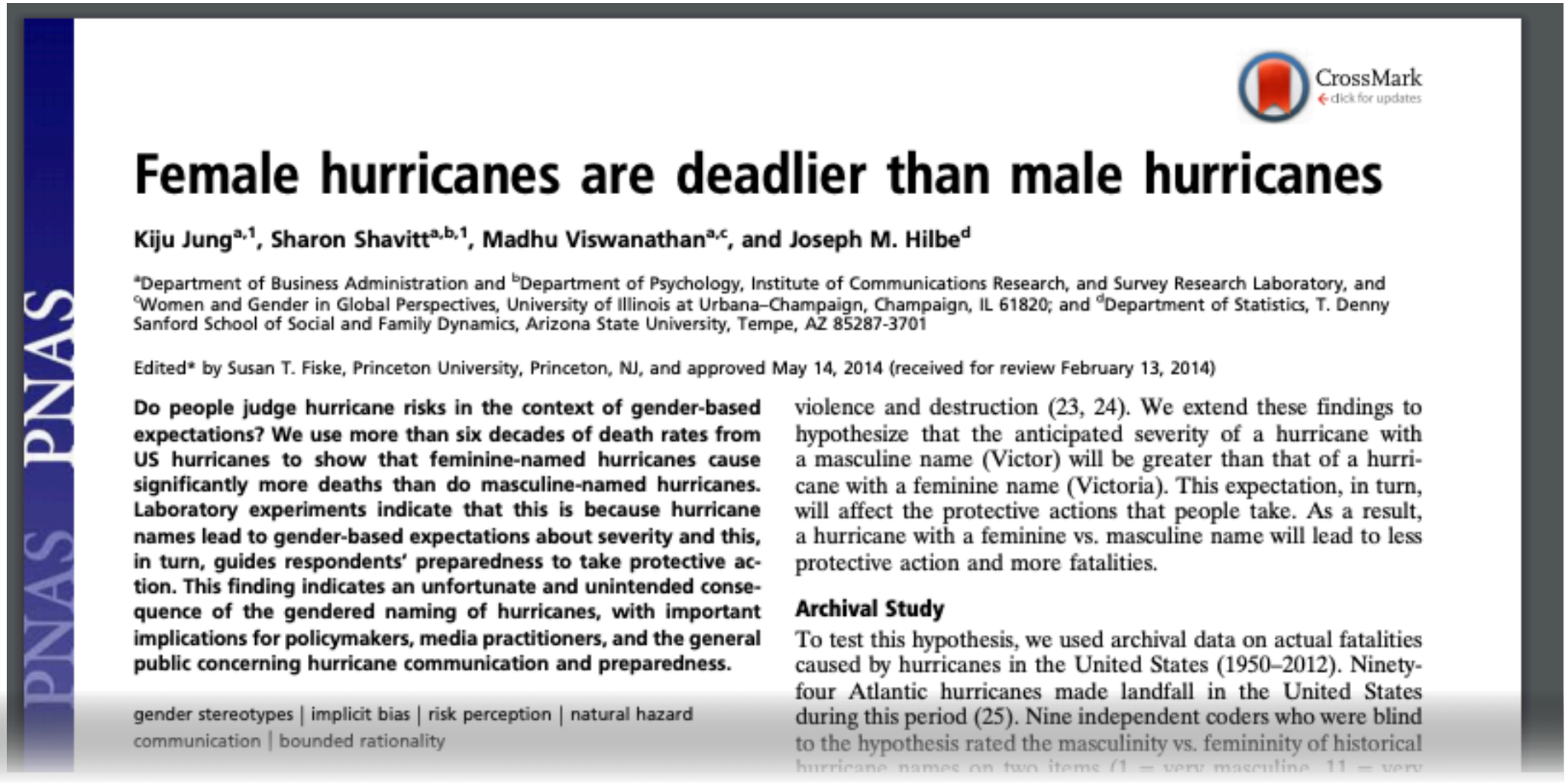
Output:

- the number of hurricanes in each year

Example:

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

Demo 5: Hurricane Names and Stereotypes



PNAS

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

what would it take to try to replicate this study?