

CS 1: Intro to CS

From Files and Dictionaries to
CSV Processing!



L12 Exit Ticket Questions:

- Q1: Takeaways
- Q2: Questions you have
- Q3: What is a question you have for students and/or TAs that could be used for example CSV data this week? (e.g. a short answer question)

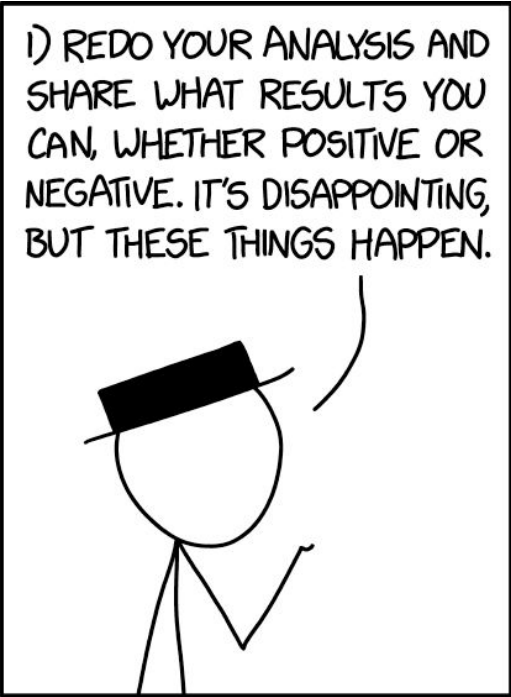
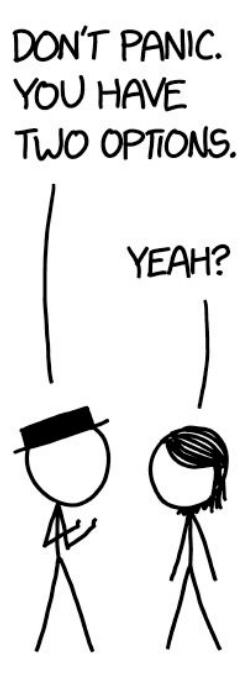
Download `lec12_csv_starter.py` and `lec12_datasets.zip` from
course website

Administrivia

Make sure you have been working on MP3!

MP4 will be out soon, and covers CSV processing/extensions to file processing and dictionaries

CSV File Processing



CSV Files

“Comma-Separated Values”

- Other delimiters may be used such as | or \t

Programs like Excel, Google Sheets, and Numbers all support CSV files and related tabular data and display them as tables

Real-world data is commonly stored in structured data formats like CSV since it's easier for parsing and analyzing data than plain text

	A	B	C
1	Col A	Col B	Col C
2	Row 1A	Row 1B	Row 1C
3	Row 2A	Row 2B	Row 2C
4	Row 3C	Row 3B	Row 3C

```
1 Col A ,Col B ,Col C
2 Row 1A,Row 1B,Row 1C
3 Row 2A,Row 2B,Row 2C
4 Row 3C,Row 3B,Row 3C
```

```
1 Col A |Col B |Col C
2 Row 1A|Row 1B|Row 1C
3 Row 2A|Row 2B|Row 2C
4 Row 3C|Row 3B|Row 3C
```

VSCode Extension: Rainbow CSV

The screenshot shows the VS Code interface. On the left, the Extensions Marketplace is open with the search term "rainbow csv". Three extensions are listed: "Rainbow CSV" (2.2M downloads), "indent-rainbow" (4M downloads), and "Rainbow Brackets" (1.8M downloads). The "Rainbow CSV" extension is highlighted. On the right, a file named "nobel_prizes.csv" is open in the editor. The file content is a CSV with columns: year, category, id, firstname, surname, and motivat. The first few rows of data are visible, with names like David J. Thouless, F. Duncan M. Haldane, J. Michael Kosterlitz, Jean-Pierre Sauvage, Sir J. Fraser Stoddart, Bernard L. Feringa, Yoshinori Ohsumi, Bob Dylan, and Juan Manuel Santos. The extension's functionality is demonstrated by the blue wavy underlines under the names in the CSV file.

Extensions: Marketplace (⇧%X) nobel_prizes.csv — Untitled (Workspace)

EXTEN... 🔍 ↻ ☰ ...

rainbow csv

Rainbow CSV 2.2M
Highlight CSV and TSV f...
mechatroner **Install** | ▾

indent-rainbow 4M
Makes indentation easie...
oderwat **Install** | ▾

Rainbow Brac... 1.8M
A rainbow brackets exte...
2gua **Install** | ▾

lec11 > data > nobel_prizes.csv

```
1 year,category,id,firstname,surname,motivat
2 2016,physics,928,David J.,Thouless,""for
3 2016,physics,929,F. Duncan M.,Haldane,""f
4 2016,physics,930,J. Michael,Kosterlitz,""
5 2016,chemistry,931,Jean-Pierre,Sauvage,""
6 2016,chemistry,932,Sir J. Fraser,Stoddart,
7 2016,chemistry,933,Bernard L.,Feringa,""f
8 2016,medicine,927,Yoshinori,Ohsumi,""for
9 2016,literature,937,Bob,Dylan,""for havin
10 2016,peace,934,Juan Manuel,Santos,""for h
11 2016,economics,935,Oliver,Hart,""for thei
12 2016,physics,936,David,Haldane,""f
```

VSCode Extension: Rainbow CSV

```
lec11 > data > nobel_prizes.csv
1  year,category,id,firstname,surname,motivation
2  2016,physics,928,David J.,Thouless,"""for theor
3  2016,physics,929,F. Duncan M.,Haldane,"""for th
4  2016,physics,930,J. Michael,Kosterlitz,"""for t
5  2016,chemistry,931,Jean-Pierre,Sauvage,"""for t
6  2016,chemistry,932,Sir J. Fraser,Stoddart,"""fo
7  2016,chemistry,933,Bernard L.,Feringa,"""for th
8  2016,medicine,927,Yoshinori,Ohsumi,"""for his d
9  2016,literature,937,Rob Dylis,"""for having cre
10 2016,peace,934,Juan Col 4: firstname ""for his re
11 2016,economics,935,Oliver,Hart,"""for their con
12 2016,economics,936,Bengt,Holmström,"""for their
13 2015,physics,919,Takaaki,Kajita,"""for the disc
14 2015,physics,920,Arthur B.,McDonald,"""for the
15 2015,chemistry,921,Tomas,Lindahl,"""for mechani
16 2015,chemistry,922,Paul,Modrich,"""for mechanis
```

CSV File Processing

Know that we know about dictionaries, we can use them to easily process CSV files

There are several ways to process CSV files in Python, but the easiest to get started with is with the built-in `csv` library

The [csv library](#) has objects and methods available to read, write, and process CSV data

- `csv.reader(file)`
- `csv.DictReader(file)`
- `csv.writer(file)` and `writerow`
- `csv.DictWriter(file, fieldnames)` with `writeheader`, `writerow`

We'll take a look at the basics today with some example datasets

Opening up a CSV File

We open a CSV file similar to how we open other files, but we use the `csv` library to process the data in a more structured way

csv_example.csv X

lec11 > data > csv_example.csv

```
1 Col A ,Col B ,Col C
2 Row 1A,Row 1B,Row 1C
3 Row 2A,Row 2B,Row 2C
4 Row 3C,Row 3B,Row 3C
```

DEBUG CONSOLE

JUPYTER

TERMINAL

...

Python

```
>>> import csv
>>> with open('data/csv_example.csv') as csvfile:
...     reader = csv.reader(csvfile)
...     for row in reader:
...         print(row) # first row printed is header
...
['Col A ', 'Col B ', 'Col C']
['Row 1A', 'Row 1B', 'Row 1C']
['Row 2A', 'Row 2B', 'Row 2C']
['Row 3C', 'Row 3B', 'Row 3C']
```


The `csv.reader` Object

The `csv.reader` object is what does most of the CSV processing for us

It takes a file object as input and an optional delimiter string (default is `,`)

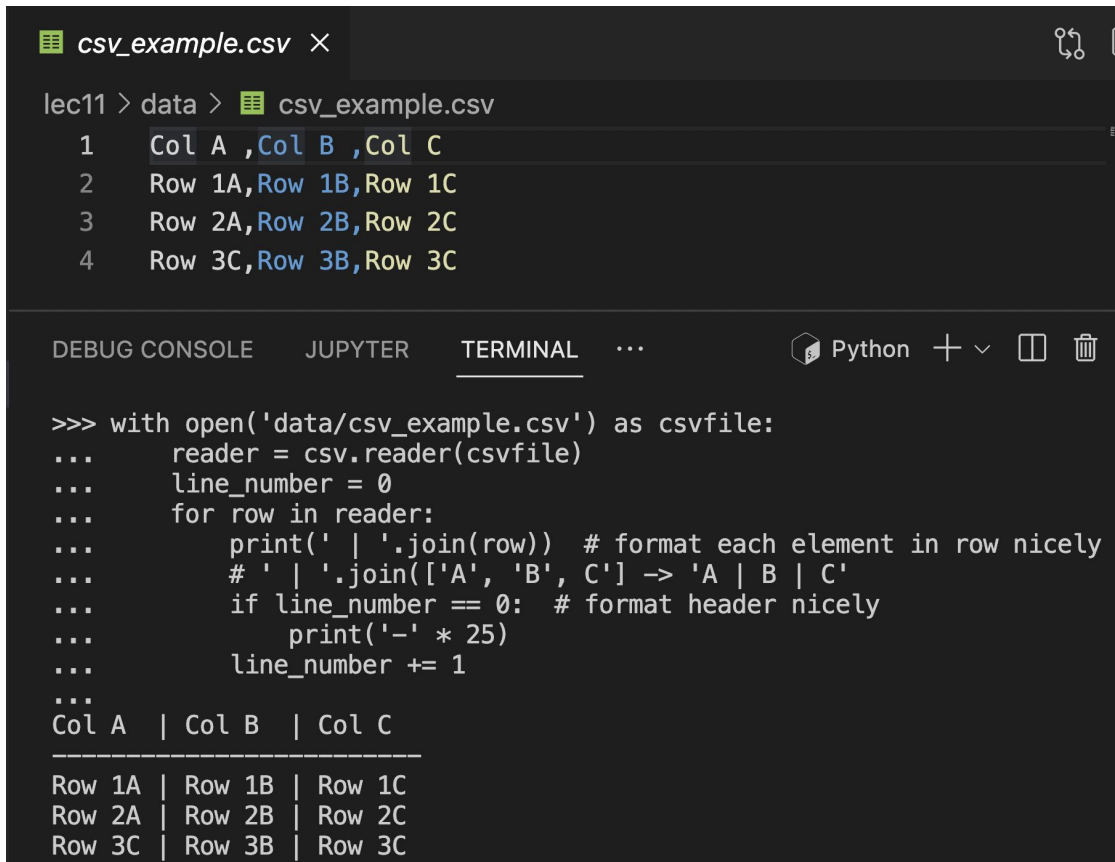
We can iterate over the `reader` object with a for loop just as we would with lists, files, etc.

Each row in the `reader` object is represented as a list of strings generated by splitting on the specified delimiter string (e.g. `['Row 1A', 'Row 1B', 'Row 1C']`)

Header Row vs. Data Rows

The first row in a CSV file usually contains the column names. We can use this to format our results nicely.

We will see this distinction when writing CSV files as well.



```
lec11 > data > csv_example.csv
1 Col A ,Col B ,Col C
2 Row 1A,Row 1B,Row 1C
3 Row 2A,Row 2B,Row 2C
4 Row 3C,Row 3B,Row 3C
```

DEBUG CONSOLE JUPYTER TERMINAL ... Python + v [] [X]

```
>>> with open('data/csv_example.csv') as csvfile:
...     reader = csv.reader(csvfile)
...     line_number = 0
...     for row in reader:
...         print(' | '.join(row)) # format each element in row nicely
...         # ' | '.join(['A', 'B', 'C']) -> 'A | B | C'
...         if line_number == 0: # format header nicely
...             print('-' * 25)
...         line_number += 1
...
Col A | Col B | Col C
-----
Row 1A | Row 1B | Row 1C
Row 2A | Row 2B | Row 2C
Row 3C | Row 3B | Row 3C
```

The `csv.DictReader` Object

Treating each row of data as a list of strings isn't usually ideal, and this is when dictionaries come in handy!

The `csv` library has another object called `csv.DictReader` which stores each row as a dictionary instead of a list. The keys are the columns for the CSV file and the values are that row's values for each column

The keys are determined by the header row of the CSV file; if your CSV file does not have a header row, you'll need to specify these manually with a second `fieldnames` argument to the `DictReader` constructor.

The csv.DictReader Object


```
lec11 > data >  csv_example.csv
```

```
1 Col A ,Col B ,Col C
2 Row 1A,Row 1B,Row 1C
3 Row 2A,Row 2B,Row 2C
4 Row 3C,Row 3B,Row 3C
```

DEBUG CONSOLE

JUPYTER

TERMINAL

 Python +

```
>>> with open('data/csv_example.csv') as csv_file:
...     reader = csv.DictReader(csv_file)
...     for row in reader:
...         print(row)
...
{'Col A ': 'Row 1A', 'Col B ': 'Row 1B', 'Col C': 'Row 1C'}
{'Col A ': 'Row 2A', 'Col B ': 'Row 2B', 'Col C': 'Row 2C'}
{'Col A ': 'Row 3C', 'Col B ': 'Row 3B', 'Col C': 'Row 3C'}
```

Preview to Monday

Writing CSV Files with the `csv.writer` Object

The process of reading/writing CSV files is similar to that of reading/writing other files

We can write files with `csv.writer` and `writerow` method

 `new_csv.csv` X

lec11 > data >  `new_csv.csv`

```
1 first,last,email
2 El,Hovik,hovik
3 Maddie,Ramos,maramos
4
```

DEBUG CONSOLE

JUPYTER

TERMINAL

 Python

```
>>> with open('data/new_csv.csv', 'w') as csvfile:
...     usernames = {('El', 'Hovik'): 'hovik',
...                   ('Maddie', 'Ramos'): 'maramos'}
...     columns = ['first', 'last', 'email']
...     writer = csv.writer(csvfile)
...     writer.writerow(columns)
...     for key in usernames:
...         first, last = key # tuple-unpacking!
...         username = usernames[key]
...         writer.writerow([first, last, username])
... 
```

Writing CSV Files with the `csv.DictWriter` Object

Similar to the motivation of `csv.DictReader`, we can use `csv.DictWriter` to more easily write a list of dictionaries to a CSV file:

General approach:

1. Open the file
2. Rows should be a list of dictionaries
3. Determine columns as list of string column names
4. Create `csv.DictWriter(file_obj, fieldnames=col_list)`
5. Write column header first with `writer.writeheader()`
6. Loop through list of row dictionaries, and write each one in order

```
with open('data/new_csv.csv', 'w') as csv_file:
    rows = <list of dicts> # [{'a': 1, 'b': 2}, {...}, ...]
    # key list for first dict, e.g. ['a', 'b']
    columns = rows[0].keys()
    writer = csv.DictWriter(csv_file, fieldnames=columns)
    writer.writeheader() # write column header, e.g. a,b
    for row in rows: # {'a': 1, 'b': 2}
        writer.writerow() # writes 1,2 on next line
```

Writing CSV Files with the `csv.DictWriter` Object

```
with open('data/new_csv.csv', 'w') as csv_file:
    rows = <list of dicts> # [{'a': 1, 'b': 2}, {...}, ...]
    # key list for first dict, e.g. ['a', 'b']
    columns = rows[0].keys()
    writer = csv.DictWriter(csv_file, fieldnames=columns)
    writer.writeheader() # write column header, e.g. a,b
    for row in rows: # {'a': 1, 'b': 2}
        writer.writerow() # writes 1,2 on next line
```

```
lec11 > data > new_csv.csv
```

```
1 a,b
2 1,2
3 2,4
4
```

DEBUG CONSOLE JUPYTER TERMINAL

```
>>> with open('data/new_csv.csv', 'w') as csv_file:
...     row1 = {'a': 1, 'b': 2}
...     row2 = {'a': 2, 'b': 4}
...     rows = [row1, row2]
...     columns = rows[0].keys() # key list for first dictionary
...     writer = csv.DictWriter(csv_file, fieldnames=columns)
...     writer.writeheader()
...     for row in rows: # {'a': 1, 'b': 2}
...         writer.writerow(row) # writes 1,2 on next line
```


csv.DictWriter: writer.writeheader()

When using `csv.DictWriter`, it's easy to forget to write the column headers. You specify the column names with the `fieldnames` keyword argument, and `writer.writeheader()` to "flush" (write) the column header to the new file.

```
lec11 > data > new_csv.csv
```

```
1 El,Hovik,hovik
2 Maddie,Ramos,maramos
3
```

DEBUG CONSOLE JUPYTER TERMINAL

```
>>> with open('data/new_csv.csv', 'w') as csv_file:
...     usernames = [{'first': 'El', 'last': 'Hovik', 'email': 'hovik@cs.cmu.edu'},
...                   {'first': 'Maddie', 'last': 'Ramos', 'email': 'mramos@cs.cmu.edu'}]
...     # columns = ['first', 'last', 'email']
...     columns = usernames[0].keys()
...     writer = csv.DictWriter(csv_file, fieldnames=columns)
...     for row in usernames:
...         writer.writerow(row)
... 
```

```
lec11 > data > new_csv.csv
```

```
1 first,last,email
2 El,Hovik,hovik
3 Maddie,Ramos,maramos
.
```

DEBUG CONSOLE JUPYTER TERMINAL

```
>>> with open('data/new_csv.csv', 'w') as csv_file:
...     usernames = [{'first': 'El', 'last': 'Hovik', 'email': 'hovik@cs.cmu.edu'},
...                   {'first': 'Maddie', 'last': 'Ramos', 'email': 'mramos@cs.cmu.edu'}]
...     # columns = ['first', 'last', 'email']
...     columns = usernames[0].keys()
...     writer = csv.DictWriter(csv_file, fieldnames=columns)
...     writer.writeheader()
...     for row in usernames:
...         writer.writerow(row)
... 
```

More Examples/Practice

You can find more examples in the `lec12_csv_examples.py` using `lec12_data.zip` CSV files posted on the course website (you can ask questions on Discord if you have any!)

- The assigned reading for CSV processing is also posted (an alternative to usual CS 1 readings)

We'll go over some more practical examples of CSV processing next week in Monday's lecture and Tuesday's lab; over the weekend, think of some examples of datasets you interact with and how you might extend what you've learned to analyze them in Python!

Summary

Dictionaries are a built-in Python data type that allows us to associate keys to values

- They have many useful methods
- They can be iterated over in **for** loops
- They are used often in Python code

CSV files are very common formats for storing data, and we can extend our standard file-processing with the built-in `csv` library to parse tabular data more efficiently

- `csv.reader(file)` to create a reader object (rows are lists of strings)
- `csv.DictReader(file)` to create a DictReader object (rows are dictionaries)
- `csv.writer(file)` to create a CSV writer object working with string list rows
- `csv.DictWriter(file)` to create a CSV writer object working with dictionary rows

Wed. 05/01 (Review, Live-Coding)

Agenda:

- Working through periodic table dataset (student request)

Administrivia:

- Thank you all who have filled out the midterm feedback survey! Please do so if you haven't :) A few things to be aware of...
- Lecture times: El has moved things around and will be in ANB 104 ~30 minutes before lectures on most days (except for the occasional conflict noted on Discord)
- We'll be holding regular review/worktime/morale sessions on Saturday afternoons; 2PM?

Midterm Feedback (Cont.)

- Engagement opportunities this week: Sharing practice problems for CSV processing/dictionaries with datasets of your choice!
- We are going to be stricter on assignment deadlines; if you do not successfully submit to CodePost **by 11:30PM on Monday** you will have to use a Rework submission (do not submit right before the deadline; we won't be extending due to 11:29PM submissions that have issues)
- More practice? Some problems on dictionaries [here](#)! El will also post more lecture checks for additional practice (bring to OH!)
- MP 5 will be data visualization with Matplotlib; utilize today's lecture to think about data science questions and what subsets of data (x and y lists) are needed to visualize
- El will be posting Exit Ticket responses regularly