Common Data Formats

Ling 250: Data Science for Linguistics

C.M. Downey

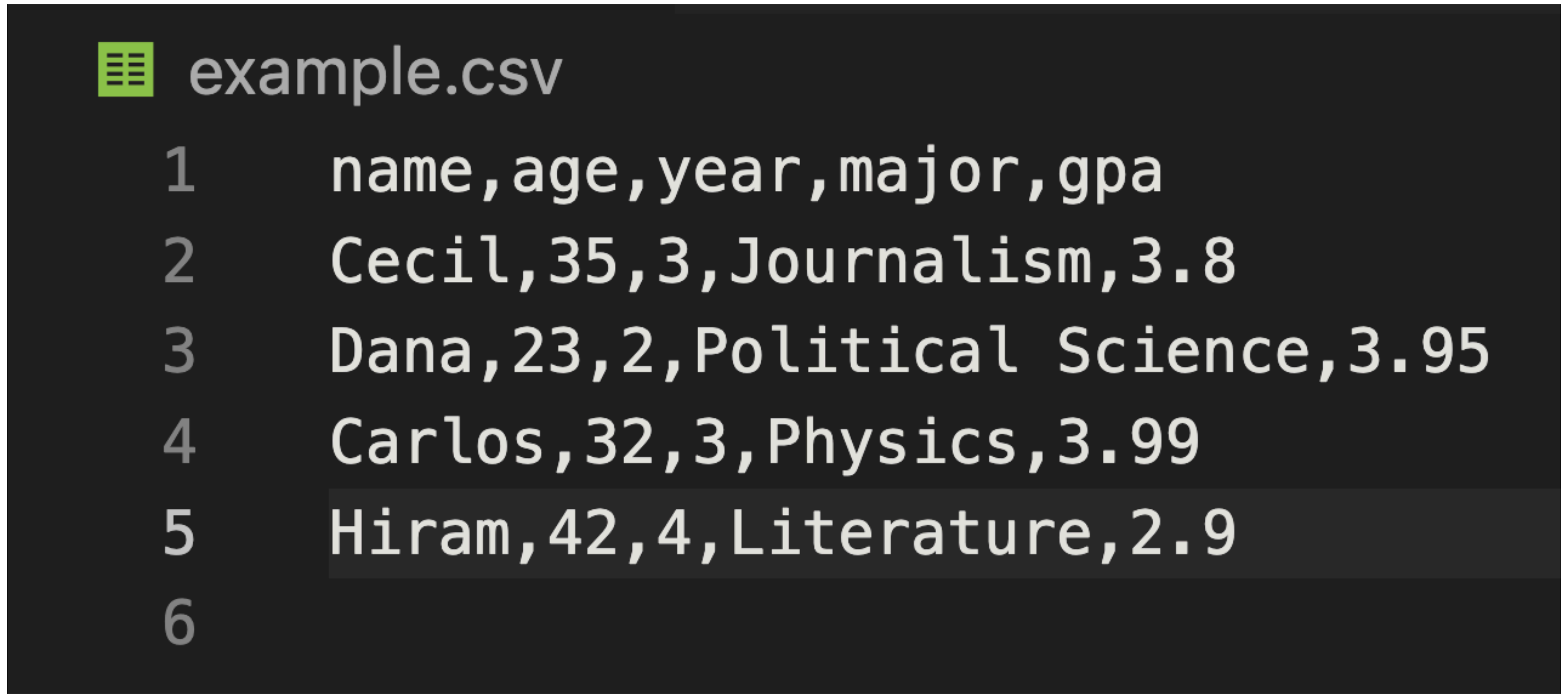
Spring 2025

Storing data in files

- So far we have mainly considered **text as data**
- We also want to work with all kinds of **general purpose data**
    - E.g. names, categories, measurements, statistics
    - These can be either the **input** or **output** of the data science process
- Like text, we need our data to be **standardized** and **machine readable**
    - i.e. we want to be able to read it with programs like **Python** and **R**
    - Formats like **Excel spreadsheets** store data, but are a **proprietary format** (only Microsoft Office can work with it)

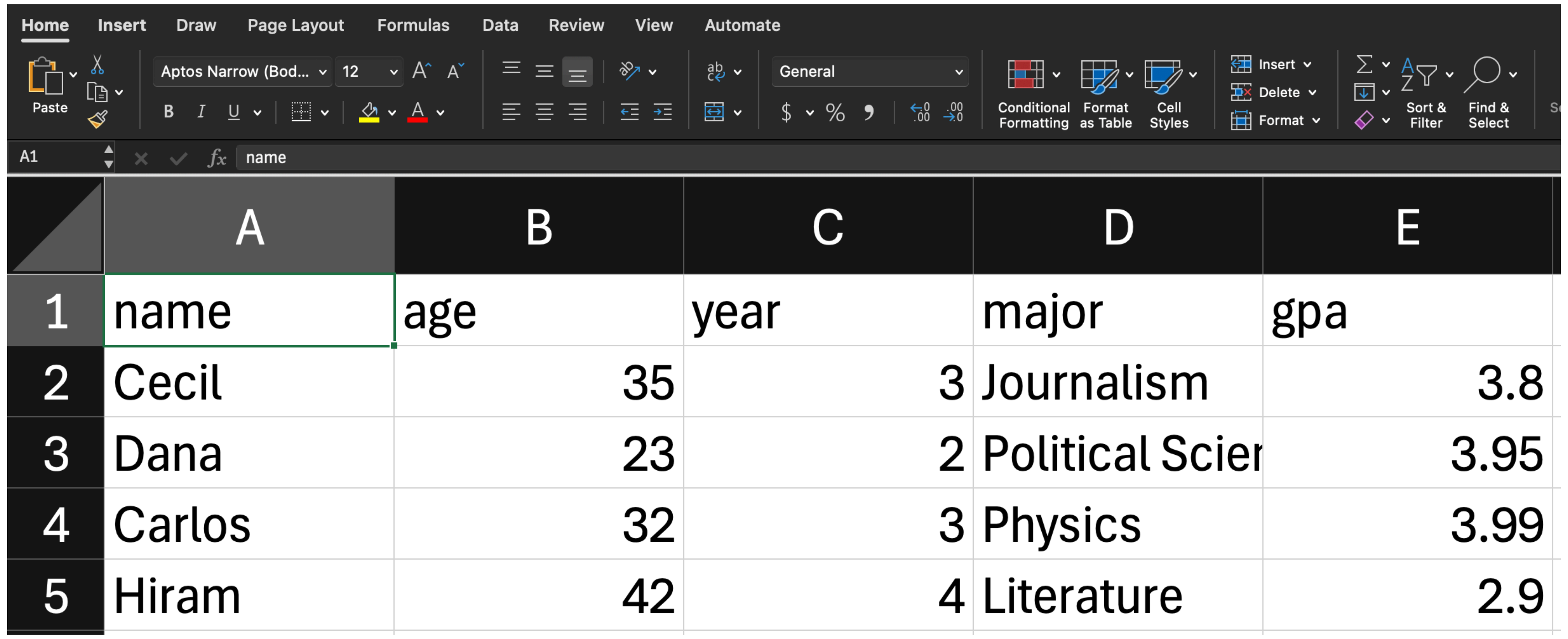
Comma Separated Values (CSV)

- CSV is one of the **most common** formats for data
- Essentially can be read as a **spreadsheet** with rows and columns
  - Each **line** is a separate row, and **commas** separate the columns
  - The first line represents the **column names**
- Can be opened with a **almost every data tool** (including Excel)



example.csv

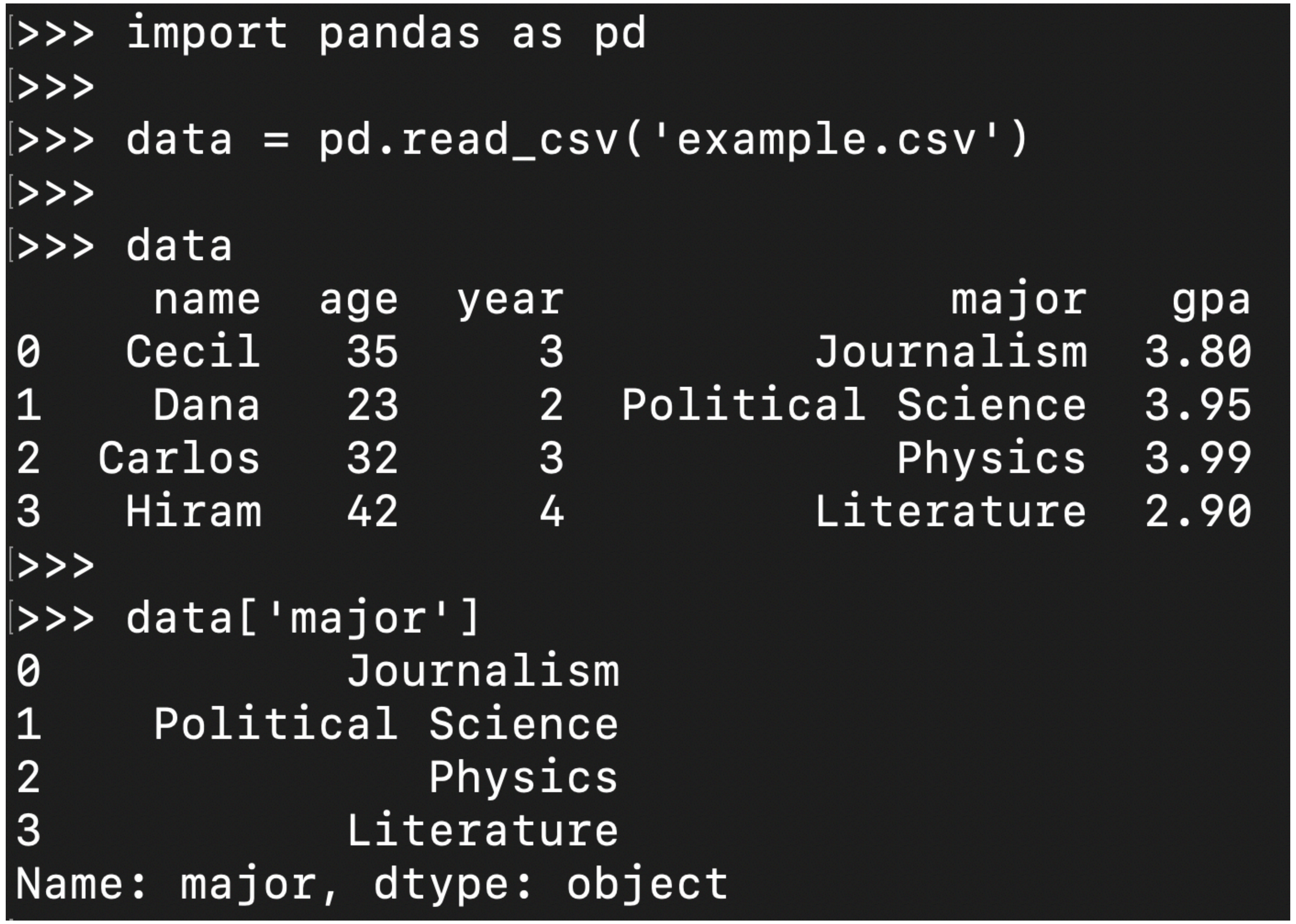
```
name,age,year,major,gpa
Cecil,35,3,Journalism,3.8
Dana,23,2,Political Science,3.95
Carlos,32,3,Physics,3.99
Hiram,42,4,Literature,2.9

```



	A	B	C	D	E
1	name	age	year	major	gpa
2	Cecil	35	3	Journalism	3.8
3	Dana	23	2	Political Scier	3.95
4	Carlos	32	3	Physics	3.99
5	Hiram	42	4	Literature	2.9

CSV with Python

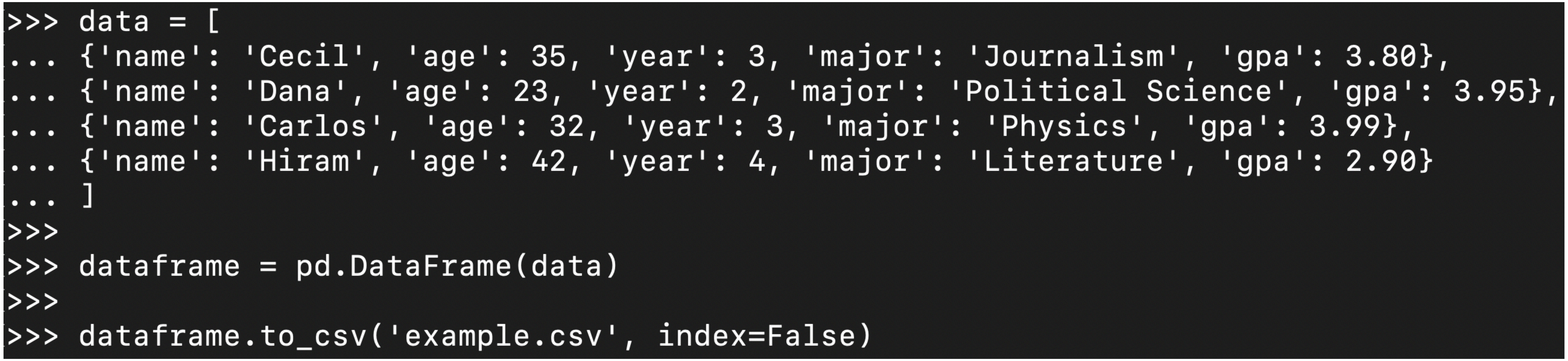
- The easiest way to work with CSV in Python is the **Pandas library**
  - Pandas has **tons of features**, but we won't go into them in this class
  - Could be useful for your **final project**
- `pandas.read_csv(filename)` opens the file as a Pandas dataframe
  - Supposed to be the Python answer to the **R dataframe** (which we'll see later)



```
>>> import pandas as pd
>>>
>>> data = pd.read_csv('example.csv')
>>>
>>> data
     name  age  year              major   gpa
0   Cecil   35     3         Journalism  3.80
1    Dana   23     2  Political Science  3.95
2  Carlos   32     3            Physics  3.99
3   Hiram   42     4         Literature  2.90
>>>
>>> data['major']
0           Journalism
1    Political Science
2              Physics
3           Literature
Name: major, dtype: object
```

CSV with Python

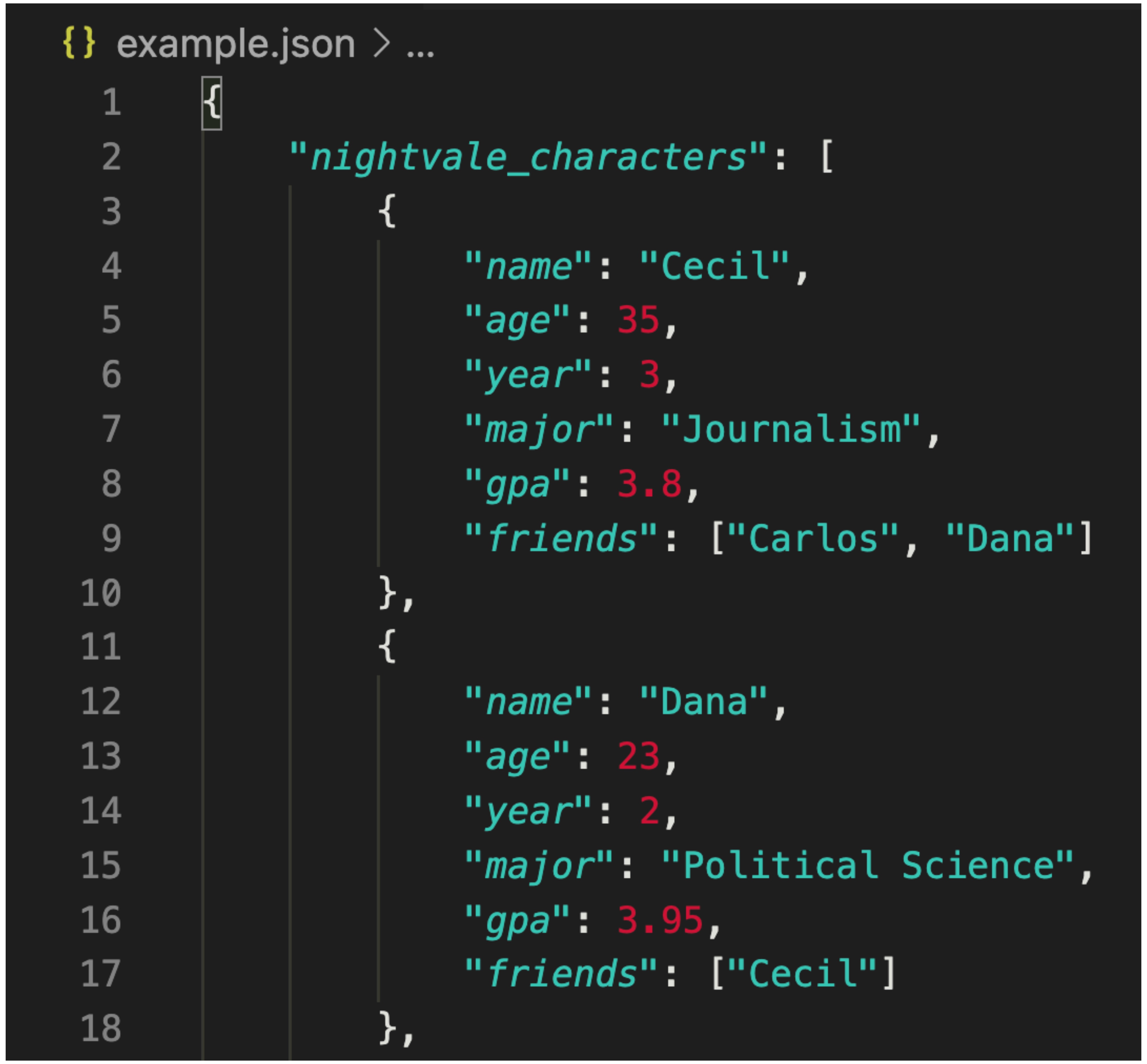
- A new CSV can be created in Python from a **list of dictionaries**
  - The keys in the dictionary are the **column names**
- The list of dictionaries can be made into a **Pandas DataFrame**
- The DataFrame can be saved to a CSV file is the `.to_csv()` method



```
>>> data = [
... {'name': 'Cecil', 'age': 35, 'year': 3, 'major': 'Journalism', 'gpa': 3.80},
... {'name': 'Dana', 'age': 23, 'year': 2, 'major': 'Political Science', 'gpa': 3.95},
... {'name': 'Carlos', 'age': 32, 'year': 3, 'major': 'Physics', 'gpa': 3.99},
... {'name': 'Hiram', 'age': 42, 'year': 4, 'major': 'Literature', 'gpa': 2.90}
... ]
>>>
>>> dataframe = pd.DataFrame(data)
>>>
>>> dataframe.to_csv('example.csv', index=False)
```

JSON

- Another extremely common data format
- Specialized for **structured data**
  - e.g. can represent **dictionaries within dictionaries**
- Similar **data formatting** to **Python**:
  - Dictionaries in {curly braces}
  - Lists in [square brackets]
  - Strings in "quotation marks"

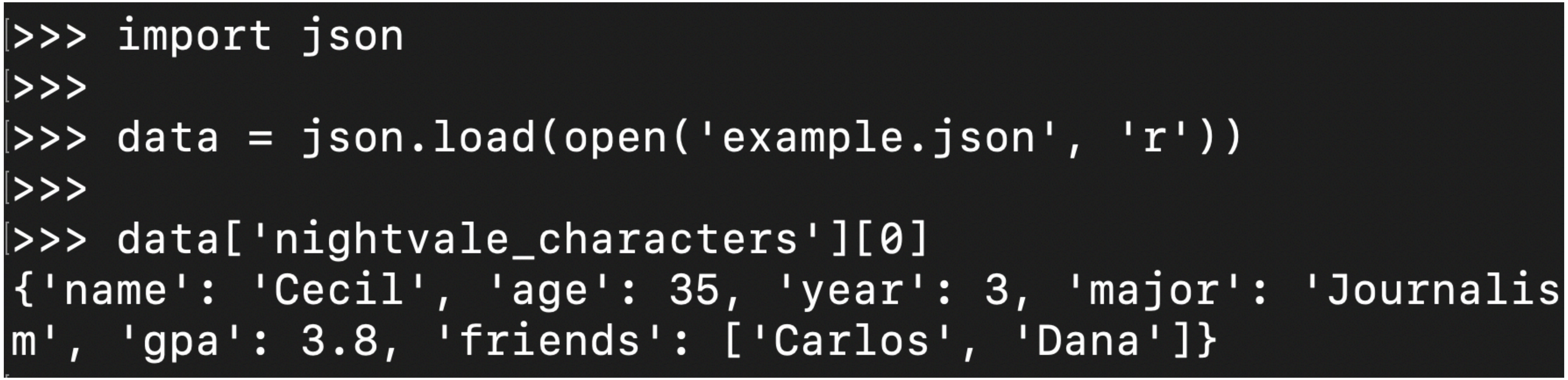


```json
{
    "nightvale_characters": [
        {
            "name": "Cecil",
            "age": 35,
            "year": 3,
            "major": "Journalism",
            "gpa": 3.8,
            "friends": ["Carlos", "Dana"]
        },
        {
            "name": "Dana",
            "age": 23,
            "year": 2,
            "major": "Political Science",
            "gpa": 3.95,
            "friends": ["Cecil"]
        },
```

JSON with Python

- JSON can be read and written with the **json library** (which comes with Python by default)

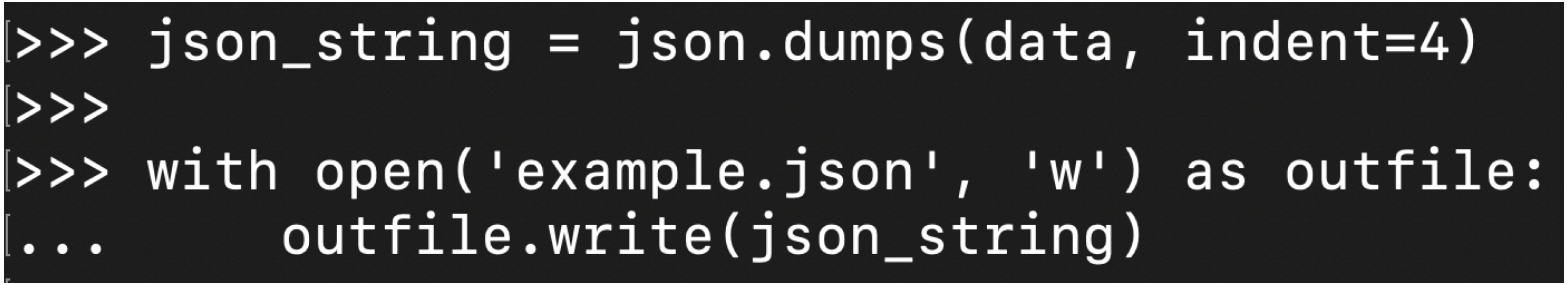
- `json.load(open(filename))` will read in the file as **equivalent Python objects**



```
>>> import json
>>>
>>> data = json.load(open('example.json', 'r'))
>>>
>>> data['nightvale_characters'][0]
{'name': 'Cecil', 'age': 35, 'year': 3, 'major': 'Journalis
m', 'gpa': 3.8, 'friends': ['Carlos', 'Dana']}
```

JSON with Python

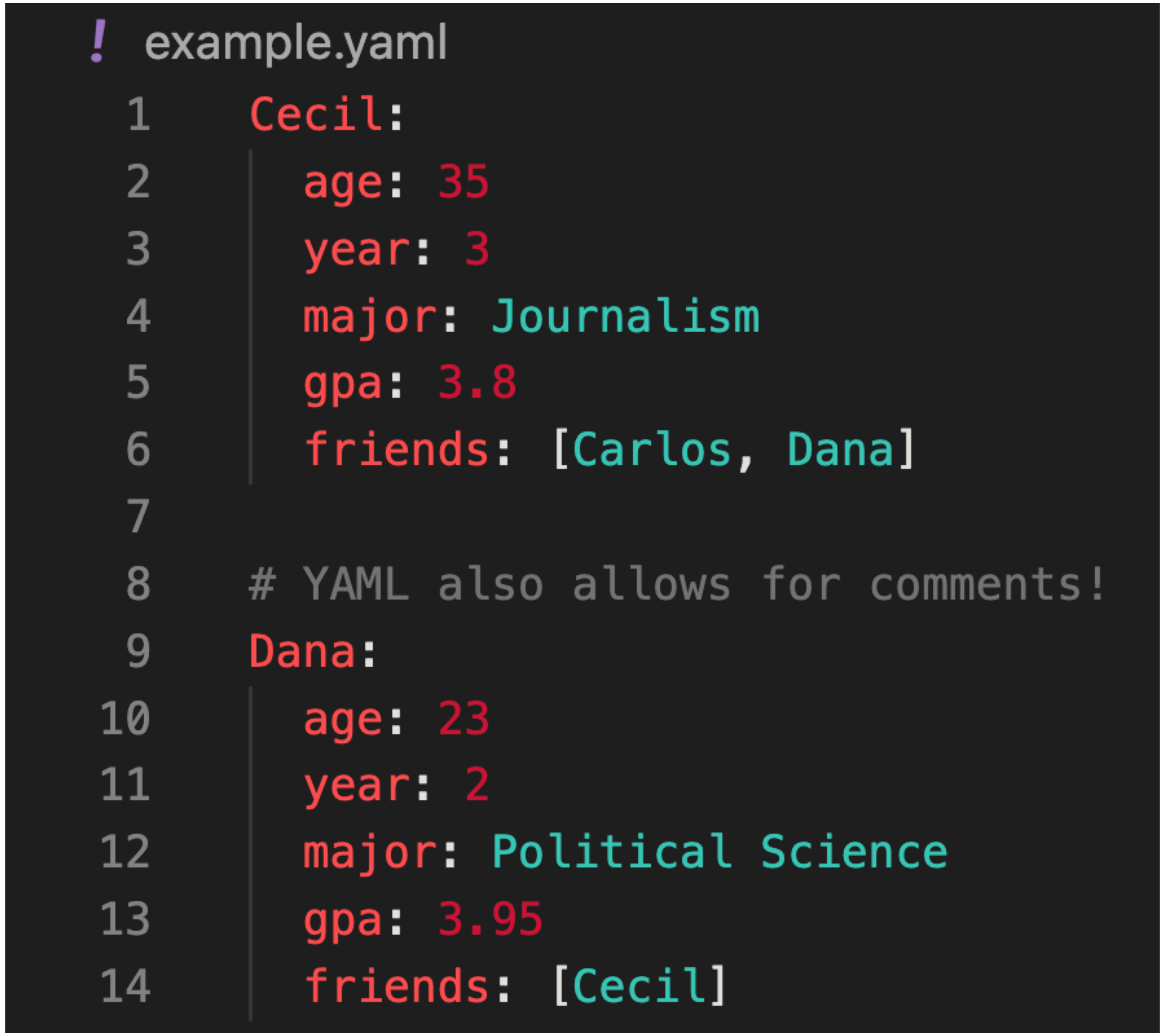
- Data can be written back to a file in two steps
  - Use `json.dumps(data)` to convert the data to a **nicely formatted JSON string**
  - Write the formatted string to a file with the `.write()` method



```
>>> json_string = json.dumps(data, indent=4)
>>>
>>> with open('example.json', 'w') as outfile:
...     outfile.write(json_string)
```

YAML

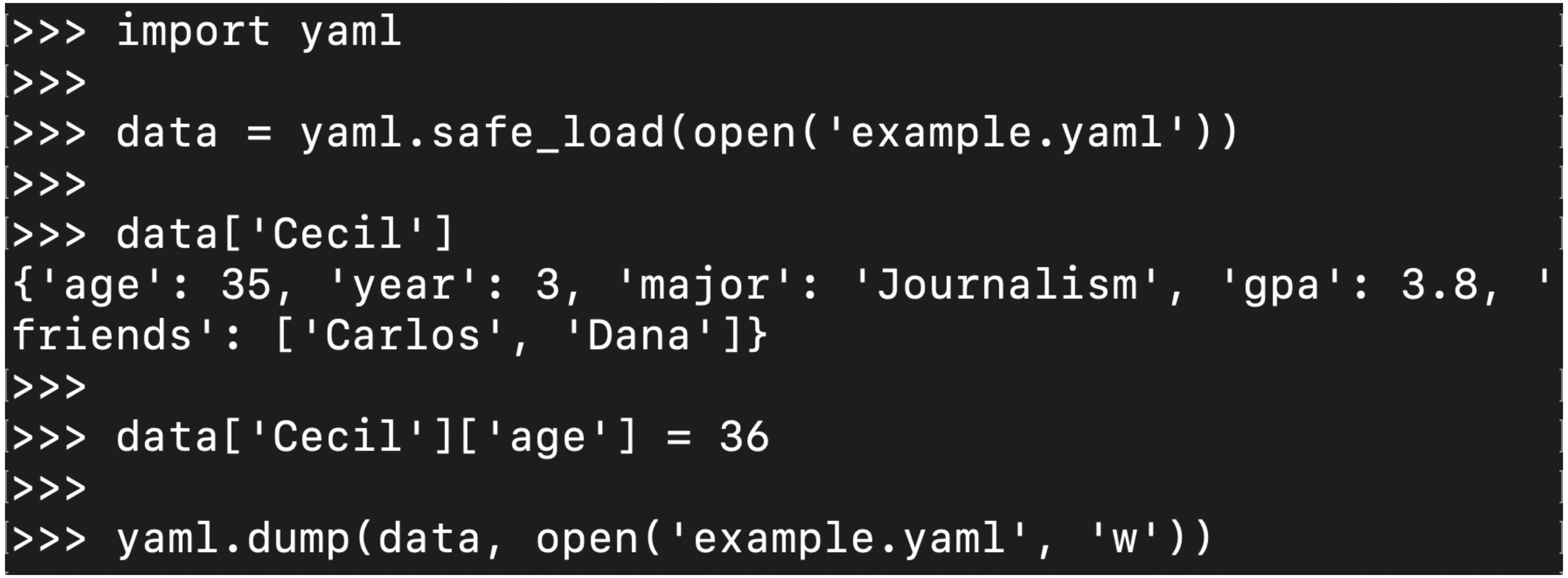
- YAML is similar to JSON in representing **structured data**, but is slightly **less common**
- Uses **newlines, indentation,** and **whitespace** to demarcate structure rather than braces and brackets like in JSON
  - Designed to be **human-friendly** in that way
- Each line assumed to be a **key-value pair** (dictionary entry) by default
  - The value can be a string, number, boolean, list, another dictionary, etc.



```yaml
Cecil:
  age: 35
  year: 3
  major: Journalism
  gpa: 3.8
  friends: [Carlos, Dana]

# YAML also allows for comments!
Dana:
  age: 23
  year: 2
  major: Political Science
  gpa: 3.95
  friends: [Cecil]
```

YAML with Python

- `yaml.safeload(open(filename))` will read the data into a **Python dictionary** (need to `import yaml` first)
- `yaml.dump(dictionary, open(filename, 'w'))` to **write to a YAML file**



```
>>> import yaml
>>>
>>> data = yaml.safe_load(open('example.yaml'))
>>>
>>> data['Cecil']
{'age': 35, 'year': 3, 'major': 'Journalism', 'gpa': 3.8, '
friends': ['Carlos', 'Dana']}
>>>
>>> data['Cecil']['age'] = 36
>>>
>>> yaml.dump(data, open('example.yaml', 'w'))
```

General Advice

- For any widespread data file type, there will be a **Python library** to parse it into Python data types
- CSV is most common for **tabular data** (spreadsheet-format)
  - We will talk much more about tabular data when we talk about R
  - Both R and Pandas define a **DataFrame** class for working with tabular data. We'll talk much more about **manipulating dataframes** in R
- Part of the **midterm project** will be writing some outputs of a data analysis to a data file